Глава 21
Схемы идентификации

21.1 FEIGE-FIAT-SHAMIR

Схема цифровой подписи и проверки подлинности, разработанная Амосом Фиатом (Amos Fiat) и Ади Шамиром (Adi Shamir), рассматривается в [566, 567]. Уриель Фейге (Uriel Feige), Фиат и Шамир модифицировали алгоритм, превратив его в доказательство подлинности с нулевым знанием [544, 545]. Это лучшее доказательство подлинности с нулевым знанием.

9 июля 1986 года три автора подали заявку на получение патента США [1427]. Из-за возможного военного применения заявка была рассмотрена военными. Время от времени результатом работы Патентное бюро является не выдача патента, а нечто, называемое секретным распоряжением. 6 января 1987 года, за три дня до истечения шестимесячного периода, по просьбе армии Патентное бюро издало такое распоряжение. Заявило, что " . . . раскрытие или публикация предмета заявки . . . может причинить ущерб национальной безопасности . . ." Авторам было приказано уведомить всех граждан США, которые по тем или иным причинам узнали о проводимых исследованиях, что несанкционированное раскрытие информации может закончиться двумя годами тюремного заключения, штрафом $10,000 или тем и другим одновременно. Более того, авторы должны были сообщить Уполномоченному по патентам и торговым знакам обо всех иностранных гражданах, которые получили доступ к этой информации.

Это было нелепо. В течение второй половины 1986 года авторы представляли свою работу на конференциях в Израиле, Европе и Соединенных Штатах. Они даже не были американским гражданами, и вся работа была выполнена в Институте Вейцмана (Weizmann) в Израиле.

Слухи об этом стали распространяться в научном сообществе и прессе. В течение двух дней секретное распоряжение было аннулировано. Шамир и его коллеги считают, что за отменой секретного распоряжения стояло NSA, хотя никаких официальных комментариев не было. Дальнейшие подробности этой причудливой истории приведены в [936].

Упрощенная схема идентификации Feige-Fiat-Shamir

Перед выдачей любых закрытых ключей арбитр выбирает случайный модуль , n, который является произведением двух больших простых чисел. В реальной жизни длина n должна быть не меньше 512 битов и лучше как можно ближе к 1024 битам. n может общим для группы контролеров. (Использование чисел Блюма (Blum) облегчит вычисления, но не является обязательным для безопасности .)

Для генерации открытого и закрытого ключей Пегги доверенный арбитр выбирает число v, являющееся квадратичным остатком mod n. Другими словами выбирается v так, чтобы уравнение $x^2 \equiv v \pmod{n}$ имело решение, и существовало $v^{-1} \bmod n$. Это v и будет открытым ключом Пегги. Затем вычисляется наименьшее s, для которого $s \equiv \text{sqrt}\ (v^{-1}) \pmod{n}$. Это будет закрытый ключ Пегги. Используется следующий протокол идентификации.

(1) Пегги выбирает случайное r, меньшее n. Затем она вычисляет $x = -r^2 \bmod n$ и посылает x Виктору.

(2) Виктор посылает Пегги случайный бит b.

(3) Если $b = 0$, то Пегги посылает Виктору r. Если $b = 1$, то Пегги посылает Виктору $y = r*s \bmod n$.

(4) Если $b = 0$, Виктор проверяет, что $x = -r^2 \bmod n$, убеждаясь, что Пегги знает значение sqrt(x). Если $b = 1$, Виктор проверяет, что $x = y^2*v \bmod n$, убеждаясь, что Пегги знает значение sqrt(v^{-1}).

Это один этап протокола, называемый **аккредитацией**. Пегги и Виктор повторяют этот протокол t раз, пока Виктор не убедится, что Пегги знает s. Это протокол "разрезать и выбрать". Если Пегги не знает s, она может подобрать r так, что она сможет обмануть Виктора, если он пошлет ей 0, или она может подобрать r так, что она сможет обмануть Виктора, если он пошлет ей 1. Она не может сделать одновременно и то, и другое. Вероятность, что ей удастся обмануть Виктора один раз, равна 50 процентам. Вероятность, что ей удастся обмануть его t раз, равна $1/2^t$.

Виктор может попробовать вскрыть протокол, выдавая себя за Пегги. Он может начать выполнение протокола с с другим контролером, Валерией. На шаге (1) вместо выбора случайного r ему останется просто использовать значение r, которое Пегги использовало в прошлый раз. Однако, вероятность того, что Валерия на шаге (2) выберет то же значение b, которое Виктор использовал в протоколе с Пегги, равна 1/2. Следовательно, вероятность, что он обманет Валерию, равна 50 процентам. Вероятность, что ему удастся обмануть ее t раз, равна $1/2^t$.

Чтобы этот протокол работал, Пегги никогда не должна использовать r повторно. В противном случае, если Виктор на шаге (2) пошлет Пегги другой случайный бит, то он получит оба ответа Пегги. Тогда даже по одному из них он сможет вычислить s, и для Пегги все закончится.

Схема идентификации Feige-Fiat-Shamir

В своих работах [544, 545], Фейге, Фиат и Шамир показали, как параллельная схема может повысить число аккредитаций на этап и уменьшить взаимодействия Пегги и Виктора.

Сначала, как и в предыдущем примере, генерируется n, произведение двух больших простых чисел. Для генерации открытого и закрытого ключей Пегги сначала выбирается k различных чисел: $v_1, v_2, \ldots v_k$, где каждое v_i является квадратичным остатком mod n. Иными словами, v_i выбираются так, чтобы $x^2 \equiv v_i \pmod n$ имело решение, и существовало $v_i^{-1} \bmod n$. Строка, $v_1, v_2, \ldots v_k$, служит открытым ключом. Затем вычисляются наименьшие s_i, для которых $s_i \equiv \text{sqrt}\ (v_i^{-1}) \pmod n$. Строка $s_1, s_2, \ldots s_k$, служит закрытым ключом.

Выполняется следующий протокол:

(1) Пегги выбирает случайное r, меньшее n. Затем она вычисляет $x = -r^2 \bmod n$ и посылает x Виктору.

(2) Виктор посылает Пегги строку из k случайных битов: $b_1, b_2, \ldots b_k$.

(3) Пегги вычисляет $y = r * (s_1^{b_1} * s_2^{b_2} * \ldots * s_k^{b_k}) \bmod n$. (Она перемножает вместе значения s_i, соответствующие b_i=1. Если первым битом Виктора будет 1, то s_1 войдет в произведение, а если первым битом будет 0, то нет, и т.д.) Она посылает y Виктору.

(4) Виктор проверяет, что $x = y^2 * (v_1^{b_1} * v_2^{b_2} * \ldots * v_k^{b_k}) \bmod n$. (Он перемножает вместе значения v_i, основываясь га случайной двоичной строке. Если его первым битом является 1, то v_1 войдет в произведение, а если первым битом будет 0, то нет, и т.д.)

Пегги и Виктор повторяют этот протокол t раз, пока Виктор не убедится, что Пегги знает $s_1, s_2, \ldots s_k$.

Вероятность, что Пегги удастся обмануть Виктор t раз, равна $1/2^{kt}$. Авторы рекомендуют использовать вер оятность мошенничества $1/2^{20}$ и предлагают значения $k = 5$ и $t = 4$. Если у вас склонность к мании преследов ания, увеличьте эти значения.

Пример

Взглянем на работу этого протокола небольших числах. Если $n = 35$ (два простых числа - 5 и 7), то возможными квадратичными остатками являются:

1: $x^2 \equiv 1 \pmod{35}$ имеет решения: $x = 1, 6, 29, 34$.

4: $x^2 \equiv 4 \pmod{35}$ имеет решения: $x = 2, 12, 23, 33$.

9: $x^2 \equiv 9 \pmod{35}$ имеет решения: $x = 3, 17, 18, 32$.

11: $x^2 \equiv 11 \pmod{35}$ имеет решения: $x = 9, 16, 19, 26$.

14: $x^2 \equiv 14 \pmod{35}$ имеет решения: $x = 7, 28$.

15: $x^2 \equiv 15 \pmod{35}$ имеет решения: $x = 15, 20$.

16: $x^2 \equiv 16 \pmod{35}$ имеет решения: $x = 4, 11, 24, 31$.

21: $x^2 \equiv 21 \pmod{35}$ имеет решения: $x = 14, 21$.

25: $x^2 \equiv 25 \pmod{35}$ имеет решения: $x = 5, 30$.

29: $x^2 \equiv 29 \pmod{35}$ имеет решения: $x = 8, 13, 22, 27$.

30: $x^2 \equiv 30 \pmod{35}$ имеет решения: $x = 10, 25$.

Обратными значениями (mod 35) и их квадратными корнями являются:

v	v^{-1}	s=sqrt(v^{-1})
1	1	1
4	9	3
9	4	2
11	16	4

16	11	9
29	29	8

Обратите внимание, что у чисел 14, 15, 21, 25 и 30 нет обратных значений mod 35, так как они не взаимно просты с 35. Это имеет смысл, так как должно быть (5 - 1) * (7 - 1)/4 квадратичных остатков mod 35, взаимно простых с 35: НОД(x, 35) = 1 (см. раздел 11.3).

Итак, Пегги получает открытый ключ, состоящий из k = 4 значений: {4,11,16,29}. Соответствующим закрытым ключом является {3,4,9,8}. Вот один этап протокола.

(1) Пегги выбирает случайное r=16, вычисляет 16^2 mod 35 = 11 и посылает его Виктору.

(2) Виктор посылает Пегги строку случайных битов: {1, 1, 0, 1}

(3) Пегги вычисляет $16*(3^1*4^1*9^0*8^1)$ mod 35 = 31 и посылает его Виктору.

(4) Виктор проверяет, что $31^2*(4^1*11^1*16^0*29^1)$ mod 35 =11.

Пегги и Виктор повторяют этот протокол t раз, каждый раз с новым случайным r, пока Виктор будет убежден.

Небольшие числа, подобные использованным в примере, не обеспечивают реальной безопасности . Но когда длина n равна 512 и более битам, Виктор не сможет узнать о закрытом ключе Пегги ничего кроме того факта, что Пегги действительно знает его.

Улучшения

В протокол можно встроить идентификационные данные. Пусть I - это двоичная строка, представляющая идентификатор Пегги: имя, адрес, номер социального страхования, размер головного убора, любимый сорт прохладительного напитка и другая личная информация. Используем однонаправленную хэш-функцию $H(x)$ для вычисления $H(I,j)$, где j - небольшое число, добавленное к I. Найдем набор j, для которых $H(I,j)$ - это квадратичный остаток по модулю n. Эти значения $H(I,j)$ становятся $v_1, v_2, \ldots v_k$ (j не обязаны быть квадратичными остатками). Теперь открытым ключом Пегги служит I и перечень j. Пегги посылает I и перечень j Виктору перед шагом (1) протокола (или Виктор загружает эти значения с какой-то открытой доски объявлений), И Виктор генерирует $v_1, v_2, \ldots v_k$ из $H(I,j)$.

Теперь, после того, как Виктор успешно завершит протокол с Пегги, он будет убежден, что Трент, которому известно разложение модуля на множители, сертифицировал связь между I и Пегги, выдав ей квадратные корни из v_i, полученные из I. (См. раздел 5.2.) Фейге, Фиат и Шамир добавили следующие замечания [544, 545]:

> Для неидеальных хэш-функций можно посоветовать рандомизировать I, добавляя к нему длинную случайную строку R. Эта строка выбирается арбитром и открывается Виктору вместе с I.
>
> В типичных реализациях k должно быть от 1 до 18. Большие значения k могут уменьшить время и трудности связи, уменьшая количество этапов.
>
> Длина n должна быть не меньше 512 битов. (Конечно, с тех пор разложение на множители заметно продвинулось .)
>
> Если каждый пользователь выберет свое собственное n и опубликует его в файле открытых ключей, то можно обойтись и без арбитра. Однако такой RSA-подобный вариант делает схему заметно менее удобной.

Схема подписи Fiat-Shamir

Превращение этой схемы идентификации в схему подписи - это, по сути, вопрос превращения Виктора в хэш-функциию. Главным преимуществом схемы цифровой подписи Fiat-Shamir по сравнению с RSA является ее скорость: для Fiat-Shamir нужно всего лишь от 1 до 4 процентов модульных умножений, используемых в RSA. В этом протоколе снова вернемся к Алисе и Бобу.

Смысл переменных - такой же, как и в схеме идентификации. Выбирается n - произведение двух больших простых чисел. Генерируется открытый ключ, $v_1, v_2, \ldots v_k$, и закрытый ключ, $s_1, s_2, \ldots s_k$, где $s_i \equiv \text{sqrt}\ (v_i^{-1})$ (mod n).

(1) Алиса выбирает t случайных целых чисел в диапазоне от 1 до n - $r_1, r_2, \ldots, r_t$ - и вычисляет $x_1, x_2, \ldots x_t$, такие что $x_i = r_i^2$ mod n.

(2) Алиса хэширует объединение сообщения и строки x_i, создавая битовый поток: $H(m, x_1, x_2, \ldots x_t)$. Она использует первые $k*t$ битов этой строки в качестве значений b_{ij}, где i пробегает от1 до t, а j от 1 до k.

(3) Алиса вычисляет $y_1, y_2, \ldots y_t$,, где $y_i = r_i * (s_1^{b_{i1}} * s_2^{b_{i2}} * \ldots * s_k^{b_{ik}})$ mod n

(Для каждого i она перемножает вместе значения s_i, в зависимости от случайных значений b_{ij}. Если b_{ij}=1, то s_i участвует в вычислениях, если b_{ij}=0, то нет.)

(4) Алиса посылает Бобу m, все биты b_{ij}, и все значения y_i. У Боба уже есть открытый ключ Алисы: $v_1, v_2, \ldots v_k$.

(5) Боб вычисляет $z_1, z_2, \ldots z_t$, где $z_i = y^2 * (v_1^{b_{i1}} * v_2^{b_{i2}} * \ldots * v_k^{b_{ik}}) \bmod n$

(И снова Боб выполняет умножение в зависимости от значений b_{ij}.) Также обратите внимание, что z_i должно быть равно x_i.

(6) Боб проверяет, что первые $k*t$ битов $H(m, z_1, z_2, \ldots z_t)$ - это значения b_{ij}, которые прислала ему Алиса.

Как и в схеме идентификации безопасность схемы подписи пропорциональна $1/2^{kt}$. Она также зависит от сложности разложения n на множители. Фиат и Шамир показали, что подделка подписи облегчается, если сложность разложения n на множители заметно меньше 2^{kt}. Кроме того, из-за вскрытия методом дня рождения (см. раздел 18.1), они рекомендуют повысить $k*t$ от 20 по крайней мере до 72, предлагая $k = 9$ и $t = 8$.

Улучшенная схема подписи Fiat-Shamir

Сильвия Микали (Silvia Micali) и Ади Шамир улучшили протокол Fiat-Shamir [1088]. Они выбирали $v_1, v_2, \ldots v_k$ так, чтобы они были первыми k простыми числами. То есть

$v_1 = 1$, $v_2 = 3$, $v_3 = 5$, и т.д.

Это открытый ключ. Закрытым ключом, $s_1, s_2, \ldots s_k$, служат случайные квадратные корни, определяемые как

$s_i = \text{sqrt}\ (v_i^{-1}) \pmod n$

В этой версии у каждого участника должен быть свой n. Такая модификация облегчает проверку подписей, не влияя на время генерации подписей и их безопасность.

Другие улучшения

На основе алгоритма Fiat-Shamir существует и N-сторонняя схема идентификации [264]. Два других улучшения схемы Fiat-Shamir в [1218]. Еще один вариант - в [1368].

Схема идентификации Ohta-Okamoto

Этот протокол является вариантом схемы идентификации Feige-Fiat-Shamir, его безопасность основана на трудности разложения на множители [1198, 1199]. Эти же авторы разработали схему с несколькими подписями (см. раздел 23.1), с помощью которой различные люди могут последовательно подписывать [1200]. Эта схема была предложена для реализации на интеллектуальных карточках [850].

Патенты

Fiat-Shamir запатентован [1427]. При желании получить лицензию на алгоритм свяжитесь с Yeda Research and Development, The Weizmann Institute of Science, Rehovot 76100, Israel.

21.2 GUILLOU-QUISQUATER

Feige-Fiat-Shamir был первым практическим протоколом идентификации. Он минимизировал вычисления, увеличивая число итераций и аккредитаций на итерацию. Для ряда реализаций, например, для интеллектуальных карточек, это не слишком подходит. Обмены с внешним миром требуют времени, а хранение данных для каждой аккредитации может быстро исчерпать ограниченные возможности карточки.

Луи Гиллу (Louis Guillou) и Жан-Жак Кискатр (Jean-Jacques Quisquater) разработали алгоритм идентификации с нулевым знанием, который больше подходит для подобных приложений [670, 1280]. Обмены между Пегги и Виктором, а также параллельные аккредитации в каждом обмене сведены к абсолютному минимуму : для каждого доказательства существует только один обмен, в котором - только одна аккредитация. Для достижения того же уровня безопасности при использовании схемы Guillou-Quisquater потребуется выполнить в три раза больше вычислений, чем при Feige-Fiat-Shamir. И, как и Feige-Fiat-Shamir, этот алгоритм идентификации можно превратить в алгоритм цифровой подписи.

Схема идентификации Guillou-Quisquater

Пегги - это интеллектуальная карточка, которая собирается доказать свою подлинность Виктору. Идентификация Пегги проводится по ряду атрибутов, представляющих собой строку данных содержащих название карточки, период действия, номер банковского счета и другие, подтверждаемые ее применимость, данные. Эта битовая строка называется J. (В реальности строка атрибутов может быть очень длинной, и в качестве J используется ее хэш-значение. Это усложнение никак не влияет на протокол.) Эта строка аналогична открытому ключу. Другой открытой информацией, общей для всех "Пегги", которые могут использовать это приложение, является показатель степени v и модуль n, где n - это произведение двух хранящихся в секрете простых чисел. Закрытым

ключом служит B, рассчитываемое так, чтобы $JB^v \equiv 1 \pmod{n}$.

Пегги посылает Виктору свои атрибуты J. Теперь она хочет доказать Виктору, что это именно ее атрибуты. Для этого она должна убедить Виктора, что ей известно B. Вот этот протокол:

(1) Пегги выбирает случайное целое r, находящееся в диапазоне от 1 до n-1. Она вычисляет $T = r^v \bmod n$ и отправляет его Виктору.

(2) Виктор выбирает случайное целое d, находящееся в диапазоне от 0 до v-1. Он посылает d Пегги.

(3) Пегги вычисляет $D = rB^d \bmod n$ и посылает его Виктору.

(4) Виктор вычисляет $T' = D^v J^d \bmod n$. Если $T \equiv T' \pmod{n}$, то подлинность Пегги доказана.

Математика не слишком сложна:

$T' = D^v J^d = (rB^d)^v J^d = r^v B^{dv} J^d = r^v (B^v J)^d = r^v = r' \equiv T \pmod{n}$, так как $JB^v \equiv 1 \pmod{n}$

Схема подписи Guillou-Quisquater

Эту схему идентификации можно превратить в схему подписи, также пригодную для реализации в интелле ктуальных карточках [671, 672]. Открытый и закрытый ключи не меняются. Вот как выглядит протокол:

(1) Алиса выбирает случайное целое r, находящееся в диапазоне от 1 до n-1. Она вычисляет $T = r^v \bmod n$.

(2) Алиса вычисляет $d = H(M,T)$, где M - подписываемое сообщение, а $H(x)$ - однонаправленная хэш-функция. Значение d, полученное с помощью хэш-функции, должно быть в диапазоне от 0 до v-1 [1280]. Если выход хэш-функции выходит за этот диапазон, он должен быть приведен по модулю v.

(3) Алиса вычисляет $D = rB^d \bmod n$. Подпись состоит из сообщения M, двух вычисленных значений, d and D, и ее атрибутов J. Она посылает подпись Бобу.

(4) Боб вычисляет $T' = D^v J^d \bmod n$. Затем он вычисляет $d' = H(M,T')$. Если $d \equiv d'$, то Алиса знает B, и ее подпись действительна.

Несколько подписей

Что если несколько человек захотят подписать один и тот же документ? Проще всего, чтобы они подписали его порознь, но рассматриваемая схема подписи делает это лучше. Пусть Алиса и Боб подписывают документ, а Кэрол проверяет подписи, но в процесс подписания может быть вовлечено произвольное количество людей. Как и раньше, Алиса и Боб обладают уникальными значениями J и B: (J_A,B_A) и (J_B,B_B). Значения n и v являются общими для всей системы.

(1) Алиса выбирает случайное целое r_A, находящееся в диапазоне от 1 до n-1. Она вычисляет $T_A = r_A{}^v \bmod n$ и посылает T_A Бобу.

(2) Боб выбирает случайное целое r_B, находящееся в диапазоне от 1 до n-1. Он вычисляет $T_B = r_B{}^v \bmod n$ и посылает T_B Алисе.

(3) Алиса и Боб, каждый вычисляет $T = (T_A * T_B) \bmod n$.

(4) Алиса и Боб, каждый вычисляет $d = H(M,T)$, где M - подписываемое сообщение, а $H(x)$ - однонаправленная хэш-функция. Значение d, полученное с помощью хэш-функции, должно быть в диапазоне от 0 до v-1 [1280]. Если выход хэш-функции выходит за этот диапазон, он должен быть приведен по модулю v.

(5) Алиса вычисляет $D_A = r_A B_A{}^d \bmod n$ и посылает D_A Бобу.

(6) Боб вычисляет $D_B = r_B B_B{}^d \bmod n$ и посылает D_B Алисе.

(7) Алиса и Боб, каждый вычисляет $D = D_A\, D_B \bmod n$. Подпись состоит из сообщения M, двух вычисленных значений, d and D, и атрибутов обоих подписывающих: J_A и J_B.

(8) Кэрол вычисляет $J = J_A\, J_B \bmod n$.

(9) Кэрол вычисляет $T' = D^v J^d \bmod n$. Затем она вычисляет $d' = H(M,T')$. Если $d \equiv d'$, то множественная подпись действительна.

Этот протокол может быть расширен на любое количество людей. Для этого подписывающие сообщение люди должны перемножить свои значения T_i на этапе (3), и свои значения D_i на этапе (7). Чтобы проверить множественную подпись, нужно на этапе (8) перемножить значения J_i подписывающих (8). Либо все подписи правильны, либо существует по крайней мере одна неправильная подпись.

21.3 SCHNORR

Безопасность схемы проверки подлинности и подписи Клауса Шнорра [1396,1397] опирается на трудность вычисления дискретных логарифмов. Для генерации пары ключей сначала выбираются два простых числа, p и q так, чтобы q было сомножителем p-1. Затем выбирается a, не равное 1, такое что $a^q \equiv 1 \pmod{p}$. Все эти числа могут быть свободно опубликованы и использоваться группой пользователей.

Для генерации конкретной пары ключей выбирается случайное число, меньшее q. Оно служит закрытым ключом, s. Затем вычисляется открытый ключ $v = a^{-s} \bmod p$.

Протокол проверки подлинности

(1) Пегги выбирает случайное число r, меньшее q, и вычисляет $x = a^r \bmod p$. Эти вычисления являются предварительными и могут быть выполнены задолго до появления Виктора.

(2) Пегги посылает x Виктору.

(3) Виктор посылает Пегги случайное число e, из диапазона от 0 до 2^{t-1}. (Что такое t, я объясню чуть позже.)

(4) Пегги вычисляет $y = (r + se) \bmod q$ и посылает y to Виктору.

(5) Виктор проверяет, что $x = a^y v^e \bmod p$.

Безопасность алгоритма зависит от параметра t. Сложность вскрытия алгоритма примерно равна 2^t. Шнорр советует использовать p около 512 битов, q - около 140 битов и t - 72.

Протокол цифровой подписи

Алгоритм Schnorr также можно использовать и в качестве протокола цифровой подписи сообщения M. Пара ключей используется та же самая, но добавляется однонаправленная хэш-функция $H(M)$.

(1) Алиса выбирает случайное число r, меньшее q, и вычисляет $x = a^r \bmod p$. Это стадия предварительных вычислений.

(2) Алиса объединяет M и x и хэширует результат:

$e = H(M,x)$

(3) Алиса вычисляет $y = (r + se) \bmod q$. Подписью являются значения e и y, она посылает их Бобу.

(4) Боб вычисляет $x' = a^y v^e \bmod p$. Затем он проверяет, что хэш-значение для объединения M и x' равно e.

$e = H(M,x')$

Если это так, то он считает подпись верной.

В своей работе Шнорр приводит следующие новые свойства своего алгоритма:

> Большая часть вычислений, нужных для генерации подписи и независящих от подписываемого сообщения, может быть выполнена на стадии предварительных вычислений. Следовательно, эти вычисления могут быть выполнены во время простоя и не влияют на скорость подписания. Вскрытие, направленное против стадии предварительных вычислений, рассматривается в [475], я не думаю, что оно имеет практическую ценность.
>
> При одинаковом уровне безопасности длина подписей для Schnorr короче, чем для RSA. Например, при 140-битовом q длина подписей равна всего лишь 212 битам, меньше половины длины подписей RSA. Подписи Schnorr также намного короче подписей EIGamal.

Конечно, из практических соображений количество битов, используемых в этой схеме, может быть уменьшено: например, для схемы идентификации, в которой мошенник должен выполнить диалоговое вскрытие всего лишь за несколько секунд (сравните со схемой подписи, когда мошенник может годами вести расчеты, чтобы выполнить подлог).

Модификация, выполненная Эрни Брикеллом (Ernie Brickell) и Кевином МакКерли (Kevin McCurley), повысила безопасность этого алгоритма [265].

Патенты

Schnorr запатентован в Соединенных Штатах [1398] и многих других странах. В 1993 году PKP приобрело обще мировые права на этот патент (см. раздел 25.5). Срок действия патента США истекает 19 февраля 2008 года.

21.4 Преобразование схем идентификации в схемы подписи

Вот стандартный метод преобразования схемы идентификации в схему подписи: Виктор заменяется однонаправленной хэш-функцией. Перед подписанием сообщение не хэшируется, вместо этого хэширование встраив а-

ется в алгоритм подписи. В принципе, такую манипуляцию можно проделать с любой схемой идентификации .

Глава 22 Алгоритмы обмена ключами

22.1 DIFFIE-HELLMAN

Diffie-Hellman, первый в истории алгоритм с открытым ключом, был изобретен 1976 году [496]. Его безопасность опирается на трудность вычисления дискретных логарифмов в конечном поле (в сравнении с легкостью возведения в степень в том же самом поле. Diffie-Hellman может быть использован для распределения ключей - Алиса и Боб могут воспользоваться этим алгоритмом для генерации секретного ключа - но его нельзя использовать для шифрования и дешифрирования сообщений.

Математика несложна. Сначала Алиса и Боб вместе выбирают большие простые числа n и g так, чтобы g было примитивом mod n. Эти два целых числа хранить в секрете необязательно, Алиса и Боб могут договориться об из использовании по несекретному каналу. Эти числа даже могут совместно использоваться группой пользователей. Без разницы. Затем выполняется следующий протокол:

(1) Алиса выбирает случайное большое целое число x и посылает Бобу

$X = g^x \bmod n$

(2) Боб выбирает случайное большое целое число y и посылает Алисе

$Y = g^y \bmod n$

(3) Алиса вычисляет

$k = Y^x \bmod n$

(4) Боб вычисляет

$k' = X^y \bmod n$

И k, и k' равны $g^{xy} \bmod n$. Никто из подслушивающих этот канал не сможет вычислить это значение, им и известно только n, g, X и Y. Пока они не смогут вычислить дискретный логарифм и раскрыть x или y, они не смогут решить проблему. Поэтому, k - это секретный ключ, который Алиса и Боб вычисляют независимо.

Выбор g и n может заметно влиять на безопасность системы. Число $(n-1)/2$ также должно быть простым [1253]. И, самое главное, n должно быть большим: безопасность системы основана на сложности разложения на множители чисел того же размера, что и n. Можно выбирать любое g, которое является примитивом mod n; нет причин, по которым нельзя было бы выбрать наименьшее возможное g - обычно одноразрядное число. (К тому же, на самом деле, g не должно даже быть примитивом, оно только должно генерировать достаточно большую подгруппу мультипликативной группы mod n.)

Diffie-Hellman с тремя и более участниками *

Протокол обмена ключами Diffie-Hellman легко можно расширить на случай с тремя и более участниками. В приводимом примере Алиса, Боб и Кэрол вместе генерируют секретный ключ.

(1) Алиса выбирает случайное большое целое число x и вычисляет

$X = g^x \bmod n$

(2) Боб выбирает случайное большое целое число y и посылает Кэрол

$Y = g^y \bmod n$

(3) Кэрол выбирает случайное большое целое число z и посылает Алисе

$Z = g^z \bmod n$

(4) Алиса посылает Бобу

$Z' = Z^x \bmod n$

(5) Боб посылает Кэрол *

$X' = X^y \bmod n$

(6) Кэрол посылает Алисе

$Y' = Y^z \bmod n$

(7) Алиса вычисляет

$k = Y'^x \bmod n$

(8) Боб вычисляет

$k = Z^{y} \bmod n$

(9) Кэрол вычисляет

$k = X^{e} \bmod n$

Секретный ключ k равен $g^{xyz} \bmod n$, и никто из подслушивающих каналы связи не сможет вычислить это значение. Протокол можно легко расширить для четверых и более участников, просто добавляются участники и этапы вычислений.

Расширенный Diffie-Hellman

Diffie-Hellman также работает в коммутативных кольцах [1253]. З. Шмули (Z. Shmuley) и Кевин МакКерли (Kevin McCurley) изучили вариант алгоритма, в котором модуль является составным числом [1441, 1038]. В.С. Миллер (V. S. Miller) и Нил Коблиц (Neal Koblitz) расширили этот алгоритм, используя эллиптические кривые [1095, 867]. Тахер ЭльДжамаль (Taher ElGamal) использовал основополагающую идею для разработки алгоритма шифрования и цифровой подписи (см. раздел 19.6).

Этот алгоритм также работает в поле Галуа $GF(2^k)$ [1442, 1038]. В ряде реализаций используется именно этот подход [884, 1631, 1632], так как вычисления выполняются намного быстрее. Но и криптоаналитические вычисления выполняются намного быстрее, поэтому важно тщательно выбирать поле, достаточно большое, чтобы обеспечить нужную безопасность.

Hughes

Этот вариант алгоритма Diffie-Hellman позволяет Алисе генерировать ключ и послать его Бобу [745].

(1) Алиса выбирает случайное большое целое число x и генерирует

$k = g^{x} \bmod n$

(2) Боб выбирает случайное большое целое число y и посылает Алисе

$Y = g^{y} \bmod n$

(3) Алиса посылает Бобу

$X = Y^{x} \bmod n$

(4) Боб вычисляет

$z = y^{-1}$

$k' = X^{z} \bmod n$

Если все выполнено правильно, $k = k'$.

Преимуществом этого протокола над Diffie-Hellman состоит в том, что k можно вычислить заранее, до взаимодействия, и Алиса может шифровать сообщения с помощью k задолго до установления соединения с Бобом. Она может послать сообщение сразу множеству людей, а передать ключ позднее каждому по отдельности.

Обмен ключом без обмена ключом

Если у вас сообщество пользователей, каждый может опубликовать открытый ключ, $X = g^{x} \bmod n$, в общей базе данных. Если Алиса захочет установить связь с Бобом, ей понадобится только получить открытый ключ Боба и генерировать их общий секретный ключ. Она может зашифровать сообщение этим ключом и послать его Бобу. Боб извлечет открытый ключ Алисы и вычислит общий секретный ключ.

Каждая пара пользователей может использовать уникальный секретный ключ, не требуется никаких предварительных обменов данными между пользователями. Открытые ключи должны пройти сертификацию, чтобы предотвратить мошеннические вскрытия, и должны регулярно меняться, но в любом случае это очень умная идея

Патенты

Алгоритм обмена ключами Diffie-Hellman запатентован в Соединенных Штатах [718] и Канаде [719]. Группа, называющаяся Public Key Partners (PKP, Партнеры по открытым ключам), получила вместе с другими патентами в области криптографии с открытыми ключами получила лицензию на этот патент (см. раздел 25.5). Срок действия патента США истекает 29 апреля 1997 года.

22.2 Протокол "точка-точка"

Обмен ключами Diffie-Hellman чувствителен к вскрытию "человек в середине". Одним из способов предотвратить это, является необходимость для Алисы и Боба подписывать сообщения, которые они посылают друг другу [500].

Этот протокол предполагает, что у Алисы есть сертифицированный открытый ключ Боба, а у Боба есть сертифицированный открытый ключ Алисы. Эти сертификаты подписаны некоторым заслуживающим доверия органом власти, непосредственно не участвующим в протоколе. Вот как Алиса и Боб генерируют секретный ключ k.

(1) Алиса генерирует случайное число x и посылает его Бобу.

(2) Боб генерирует случайное число y. Используя протокол Diffie-Hellman, он вычисляет общий ключ k на базе x и y. Он подписывает x и y и шифрует подпись ключом k. Затем он посылает получившееся вместе с y Алисе.

$y, E_k(S_B(x,y))$

(3) Алиса также вычисляет k. Она расшифровывает оставшуюся часть сообщения Боба и проверяет его подпись. Затем она посылает Бобу подписанное сообщение, состоящее из x и y, зашифрованных общим ключом k.

$E_k(S_A(x,y))$

(4) Боб расшифровывает сообщение и проверяет подпись Алисы.

22.3 Трехпроходный протокол Шамира

Этот изобретенный Ади Шамиром но никогда не опубликованный протокол позволяет Алисе и Бобу безопасно обмениваться информацией, не используя предварительного обмена ни секретными, ни открытыми ключами [1008]. Он предполагает использование коммутативного симметричного шифра, для которого:

$E_A(E_B(P)) = E_B(E_A(P))$

Секретный ключ Алисы - A, а Боба - B. Алиса хочет послать сообщение M Бобу. Вто этот протокол.

(1) Алиса шифрует M своим ключом и посылает его Бобу

$C_1 = E_A(M)$

(2) Боб шифрует C_1 своим ключом и посылает Алисе

$C_2 = E_B(E_A(M))$

(3) Алиса расшифровывает C_2 своим ключом и посылает Бобу

$C_3 = D_A(E_B(E_A(M))) = D_A(E_A(E_B(M))) = E_B(M)$

(4) Боб расшифровывает C_3 своим ключом, получая M.

Коммутативны и обладают совершенной безопасностью одноразовые блокноты, но с этим протоколом они работать не будут. При использовании одноразового блокнота три шифротекста будут выглядеть следующим образом be:

$C_1 = M \oplus A$

$C_2 = M \oplus A \oplus B$

$C_3 = M \oplus B$

Ева, записав эти три сообщения, которыми обмениваются Алиса и Боб, просто выполнит XOR всех этих шифротекстов и восстановит сообщение:

$C_1 \oplus C_2 \oplus C_3 = (M \oplus A) \oplus (M \oplus A \oplus B) \oplus (M \oplus B) = M$

Очевидно, что такой способ работать не будет.

Шамир (и независимо Джим Омура (Jim Omura)) описал похожий на RSA алгоритм шифрования, который будет работать с этим протоколом. Пусть p будет большим большим простым числом, причем множитель p-1 является большим простым. Выберем ключ шифрования e, взаимно простой с p-1. Вычислим d, для которого выполняется $de = 1 \pmod{p - 1}$. Для шифрования сообщения вычисляем

$C = M^e \bmod p$

Для дешифрирования сообщения вычисляем

$M = C^d \bmod p$

По видимому, у Евы нет способа получить M, не решив проблему дискретного логарифма, но это никогда не было доказано.

Как и Diffie-Hellman, этот протокол позволяет Алисе начать секретный обмен информацией с Бобом, не зная ни одного из его ключей. При использовании алгоритма с открытым ключом Алиса должна знать открытый ключ Боба. Применяя трехпроходный алгоритм Шамира, она просто посылает Бобу шифротекст сообщения . То же действие с помощью алгоритма с открытым ключом выглядит следующим образом :

(1) Алиса запрашивает у Боба (или у KDC) его открытый ключ.

(2) Боб (или KDC) посылает Алисе свой открытый ключ.

(3) Алиса шифрует M открытым ключом Боба и посылает его Бобу.

Трехпроходный алгоритм Шамира не может устоять перед вскрытием "человек в середине" .

22.4 COMSET

COMSET (COMmunications SETup, установление связи) это протокол одновременной идентификации и обмена ключом, разработанный для проекта RIPE [1305] (см. раздел 25.7). С помощью криптографии с открытыми ключами он позволяет Алисе и Бобу идентифицировать друг друга, при этом обмениваясь секретным ключом.

Математической основой COMSET служит схема Rabin [1283] (см. раздел 19.5). Сама схема впервые была предложена в [224]. См. подробности в [1305].

22.5 Обмен зашифрованными ключами

Протокол обмена зашифрованными ключами (Encrypted Key Exchange, EKE) был разработан Стивом Белловином (Steve Bellovin) и Майклом Мерриттом (Michael Merritt) [109]. Он обеспечивает безопасность и проверку подлинности в компьютерных сетях, по новому используя и симметричную криптографию, и криптографию с открытыми ключами: общий секретный ключ используется для шифрования генерированного случайным образом открытого ключа.

Базовый протокол EKE

Алиса и Боб (два пользователя, клиент и сервер, или кто угодно) имеют общий пароль P. Используя следующий протокол, они могут проверить подлинность друг друга и генерировать общий сеансовый ключ K.

(1) Алиса Случайным образом генерирует пару "открытый ключ/закрытый ключ" . Она шифрует открытый ключ K' с помощью симметричного алгоритма, используя P в качестве ключа: $E_P(K')$. Она посылает Бобу

$A, E_P(K')$

(2) Боб знает P. Он расшифровывает сообщение, получая K'. Затем он генерирует случайный сеансовый ключ K шифрует его открытым ключом, который он получил от Алисы, а затем используя P качестве ключа. Он посылает Алисе

$E_P(E_{K'}(K)$

(3) Алиса расшифровывает сообщение, получая K. Она генерирует случайную строку R_A, шифрует ее с помощью K и посылает Бобу

$E_K(R_A)$

(4) Боб расшифровывает сообщение, получая R_A. Он генерирует другую случайную строку, R_B, шифрует обе строки ключом K и посылает Алисе результат.

$E_K(R_A,R_B)$

(5) Алиса расшифровывает сообщение, получая R_A и R_B. Если строка R_A, полученная от Боба, - это та самая строка, которую она послала Бобу на этапе (3), она, используя K, шифрует R_B и посылает ее Бобу.

$E_K(R_B)$

(6) Боб расшифровывает сообщение, получая R_B. Если строка R_B, полученная от Алисы, - это та самая строка, которую он послал ей на этапе (4), завершен. Теперь обе стороны могут обмениваться информацией, используя K в качестве сеансового ключа.

На этапе (3) и Алиса, и Боб знают K' и K. K - это сеансовый ключ, он может быть использован для шифрования всех других сообщений, которыми обмениваются Алиса и Боб. Ева, сидя между Алисой и Бобом, знает только $E_P(K')$, $E_P(E_{K'}(K))$ и несколько сообщений, зашифрованных K. В других протоколах Ева могла бы попробовать угадать P (люди все время любят выбирать плохие пароли, и если Ева достаточно умна, она может этот пароль) и затем проверить свои предположения. В рассматриваемом протоколе Ева не может проверять свои предположения, не вскрыв при этом и алгоритм с открытым ключом. И, если K' и K выбираются случайным образом, то эта проблема будет непреодолимой.

Ответная часть протокола, этапы (3) - (6), обеспечивает подтверждение. Этапы (3) - (5) доказывают Алисе, что Боб знает K, этапы (4) - (6) доказывают Бобу, что Алиса знает K. Обмен метками времени, используемый в протоколе Kerberos, решает ту же задачу.

EKE может быть реализован с множеством алгоритмов с открытыми ключами: RSA, ElGamal, Diffie-Hellman. Проблемы с безопасностью возникают при реализации EKE с алгоритмом рюкзака (даже без учета проблем безопасности, присущих самим алгоритмам рюкзака): нормальное распределение шифротекста сообщений сводит на нет преимущества EKE.

Реализация EKE с помощью RSA

Алгоритм RSA кажется идеальным для такого использования, но есть ряд тонких проблем. Авторы рекомендуют шифровать на этапе (1) только показатель степени, посылая модуль. Объяснение этого совета и другие тонкости, связанные с использованием RSA, можно найти [109].

Реализация EKE с помощью ElGamal

Реализация EKE на базе алгоритма ElGamal проста, можно даже упростить основной протокол. Используя обозначения из раздела 19.6, g и p служат частями открытого ключа, общими для всех пользователей. Закрытым ключом является случайное число r. Открытым - $g^r \bmod p$. На этапе (1) Алиса посылает Бобу следующее сообщение

Алиса, $g^r \bmod p$

Обратите внимание, что этот открытый ключ не нужно шифровать с помощью P. В общем случае это неверно, но это так для алгоритма ElGamal algorithm. Подробности в [109].

Боб выбирает случайное число R (для алгоритма ElGamal, независимо от других случайных чисел, выбираемых для EKE), и сообщение, которое он посылает Алисе на этапе (2), выглядит так

$E_P(g^R \bmod p, Kg^{rR} \bmod p)$

Существующие ограничения на выбор переменных для ElGamal были приведены в разделе 19.6.

Реализация EKE с помощью Diffie-Hellman

При использовании протокола Diffie-Hellman K генерируется автоматически. Окончательный протокол еще проще. Значения g и n определяются для всех пользователей сети.

(1) Алиса выбирает случайное число r_A и посылает Бобу

$A,\ g^{r_A} \bmod n$

При использовании Diffie-Hellman Алисе не нужно шифровать с помощью P свое первое сообщение.

(2) Боб выбирает случайное число r_B и вычисляет

$K = g^{r_A * r_B} \bmod n$

Он генерирует случайную строку R_B, затем вычисляет и посылает Алисе:

$E_P(g^{r_B} \bmod n), E_K(R_B)$

(3) Алиса расшифровывает первую половину сообщения Боба, получая $g^{r_B} \bmod n$. Затем она вычисляет K и использует его для шифрования R_B. Она генерирует другую случайную строку R_A,, шифрует обе строки ключом K и посылает результат Бобу.

$E_K(R_A,,R_B)$

(4) Боб расшифровывает сообщение, получая R_A, и R_B. Если полученная от Алисы строка R_B совпадает с той, которую он посылал ей на этапе (2), он шифрует R_A ключом K и посылает результат Алисе.

$E_K(R_A)$

(5) Алиса расшифровывает сообщение, получая R_A. Если полученная от Боба строка R_A совпадает с той, которую она посылала Бобу на этапе (3), протокол завершается. Теперь стороны могут обмениваться сообщениями, используя K в качестве сеансового ключа .

Усиление EKE

Белловин (Bellovin) и Мерритт (Merritt) предложили улучшение запросно-ответной части алгоритма , которое позволяет избежать возможного вскрытия при обнаружении криптоаналитиком ста рого значения K.

На базовый протокол EKE. На этапе (3) Алиса генерирует другое случайное число S_A и посылает Бобу

$E_K(R_A, S_A)$

На этапе (4), Боб генерирует другое случайное число S_B и посылает Алисе

$E_K(R_A,,R_B,S_B)$

Теперь Алиса и Боб могут вычислить истинный сеансовый ключ , $S_A \oplus S_B$. Этот ключ в дальнейшем используется для сообщений, которыми обмениваются Алиса и Боб, K используется в качестве ключа обмена ключами.

Посмотрим на уровни защиты, предоставляемые EKE. Восстановленное значение S не дает Еве никакой информации о P, так как P никогда не используется для шифрования чего-то такого, что ведет непосредственно к S. Криптоаналитическое вскрытие K также невозможно, K используется только для шифрования случайных данных, а S никогда не шифруется отдельно.

Расширенный EKE

Протокол EKE страдает одним серьезным недостатком: он требует, чтобы обе стороны знали P. В большинстве систем авторизации доступа хранятся значения однонаправленной хэш-функции паролей пользователей, а не сами пароли (см. раздел 3.2). Протокол Расширенный EKE (Augmented EKE, A-EKE) использует в варианте EKE на базе Diffie-Hellman значение однонаправленной хэш-функции пароля пользователя в качестве ключа сверхшифрования. Затем пользователь посылает дополнительное сообщение, основанное на реальном пароле, это сообщение удостоверяет заново выбранный сеансовый ключ .

Вот как это работает. Как и обычно, Алиса и Боб хотят проверить подлинность друг друга и генерировать общий ключ. Они выбирают какую-нибудь схему цифровой подписи, в которой в качестве закрытого ключа может использоваться любое число, а открытый ключ получается из закрытого, а не генерируется отдельно . Прекрасно подходят алгоритмы ElGamal и DSA. Пароль Алисы P (или, может быть, какое-нибудь простое хэш-значение этого пароля) будет использоваться в качестве закрытого ключа и как P'.

(1) Алиса выбирает случайный показатель степени R_a и отправляет

$E_{P'}(g^{r_B} \bmod n)$

(2) Боб, который знает только P' и не может получить из него P, выбирает R_b и посылает

$E_P(g^{r_B} \bmod n)$

(3) Алиса и Боб вычисляют общий сеансовый ключ $K= g^{r_A * r_B} \bmod n$. Наконец Алиса доказывает, что она сама знает P, а не только P', посылая

$E_K(S_P(K))$

Боб, который знает K и P', может расшифровать и проверить подпись. Только Алиса могла прислать это сообщение, так как только она знает P. Самозванец, добывший копию файла паролей Боба, может попытаться P, но он не сможет подписать сеансовый ключ.

Схема A-EKE не работает с вариантом EKE, использующим открытые ключи, так как в этом протоколе одна сторона выбирает сеансовый ключ и навязывает его другой. Это позволяет взломщику, заполучившему P', выполнить вскрытие "человек в середине".

Применения EKE

Белловин и Мерритт предлагают использовать этот протокол для безопасной телефонной связи [109]:

> Предположим, что развернута сеть шифрующих телефонных аппаратов. Если кто-нибудь хочет воспользоваться таким телефоном, то понадобится определенная ключевая информация. Общепринятые решения. . . требуют, чтобы у звонящего был физический ключ. Во многих ситуациях это нежелательно. EKE позволяет использовать короткий, вводимый с клавиатуры пароль, обеспечивая гораздо более длинный сеансовый ключ .
>
> EKE мог бы быть полезен и для сотовой связи. Мошенничество представляет собой большую проблему сотовой телефонии, EKE может помочь защититься от него (и обеспечить закрытость звонка) за счет продажи телефонов, бесполезных без

введения PIN-кода. Так как PIN-код не хранится в телефоне, его невозможно извлечь из украденного экземпляра.

Главная сила EKE состоит в том, что криптография с открытыми ключами и симметричная криптография объединяются и усиливают друг друга:

> В общей перспективе EKE работает как *усилитель секретности*. То есть, его можно использовать для усиления сравнительно слабых симметричных и асимметричных систем, используемых вместе. Рассмотрим, например, размер ключа, необходимый для обеспечения безопасности при использовании обмена ключом - показателем степени. Как показали ЛаМачча (LaMacchia) и Одлыжко (Odlyzko) [934], даже модули с размерами, считавшимися безопасными, (а именно, 192 бита) чувствительны к вскрытию, занимающему несколько минут компьютерного времени. Но их вскрытие становится невозможным, если необходимо перед применением вскрытия угадать пароль.
>
> С другой стороны, сложность вскрытия обмена ключами - показателями степени может быть использована для срыва попыток угадать пароль. Возможность вскрытия угадыванием пароля зависит от скорости проверки каждого предположения. Если для выполнения такой проверки необходимо выполнить обмен ключами - показателями степени, то общее время эффектно возрастает.

EKE запатентован [111].

22.6 Защищенные переговоры о ключе

Эта схема также защищает переговоры о ключе от плохого выбора паролей и вскрытий "человек в середине" [47, 983]. В ней используется хэш-функция двух переменных, обладающая особенным свойством: она часто приводит к столкновениям по первой переменной, и практически никогда - по второй.

$H'(x,y) = H(H(k,x) \bmod 2^m, x)$, где $H(k,x)$ - обычная функция k и x

Вот как выглядит этот протокол. Алиса и Боб используют общий секретный пароль P и уже обменялись секретным ключом K, используя обмен ключом Dime-Hellman. Они используют P для проверки, что их сеансовые ключи одинаковы (и что Ева не предприняла вскрытие "человек в середине"), не позволяя Еве получить P.

(1) Алиса посылает Бобу

$H'(P,K)$

(2) Боб вычисляет $H'(P,K)$ и сравнивает результат со значением, присланным Алисой. Если они совпадают, он посылает Алисе

$H'(H(P,K))$

(3) Алиса вычисляет $H'(H(P,K))$ и сравнивает результат со значением, полученным от Боба.

Если Ева пытается выполнить вскрытие "человек в середине", она использует один ключ, K_1, общий с Алисой, и другой, K_2, общий с Бобом. Чтобы обмануть Боба на этапе (2), ей придется вычислить общий пароль и затем послать Бобу $H'(P,K_2)$. При использовании обычной хэш-функции она может перебирать часто встречающиеся пароли, пока не угадает правильный, и затем успешно проникнуть в протокол. Но при использовании предлагаемой хэш-функции, многие пароли дают одно и то же значение при хэшировании с ключом K_1. Поэтому, когда она находит совпадение, то скорее всего это неправильный пароль, и в этом случае Боба обмануть не удастся.

22.7 Распределение ключа для конференции и секретная широковещательная передача

Алиса хочет передать сообщение M сразу нескольким получателям. Однако она совсем не хочет, чтобы кто угодно смог прочесть его. В действительности, ей нужно, чтобы только получатели из определенного подмножества могли правильно раскрыть M. У всех остальных должна получиться чепуха.

Алиса может использовать для каждого получателя отличный ключ (секретный или открытый). Она шифрует сообщение каким-нибудь случайным ключом K. Затем она шифрует копию K каждым из ключей выбранных получателей сообщения. Наконец она широковещательно посылает зашифрованное сообщение, а затем все зашифрованные K. Слушающий передачу Боб либо пытается расшифровать все K своим секретным ключом, пытаясь найти правильный, либо, если Алиса не забыла перечислить получателей своего сообщения, он ищет свое имя, сопровождаемое зашифрованным ключом. Также будет работать и ранее рассмотренная криптография с несколькими ключами.

Другой способ предлагается в [352]. Сначала каждый из получателей договаривается с Алисой об общем для них двоих ключе, который длиннее любого возможного шифрованного сообщения. Все эти ключи должны быть взаимно простыми. Она шифрует сообщение случайным ключом K. Затем она вычисляет одно целое число R, которое по модулю секретного ключа конгруэнтно K, если этот секретный ключ предполагается использовать для расшифровки сообщения, и конгруэнтно нулю в противном случае.

Например, если Алиса хочет, чтобы секрет получили Боб, Кэрол и Эллен, но не Дэйв и Фрэнк, она шифрует

сообщение ключом K и затем вычисляет такое R, что

$R \equiv K \pmod{K_B}$

$R \equiv K \pmod{K_C}$

$R \equiv 0 \pmod{K_D}$

$R \equiv K \pmod{K_E}$

$R \equiv 0 \pmod{K_F}$

Это простая алгебраическая проблема, которая легко может быть решена Алисой . Когда это сообщение будет принято получателями, они вычислят значение полученного ключа по модулю их секретного ключа . Те, кому предназначалось это сообщение, в результате вычисления получат нужный ключ. В противном случае р езультатом будет 0.

Еще один, третий, путь, использующий пороговую схему (см. раздел 3.7), предлагается в [141]. Как и в других способах каждый потенциальный получатель получает секретный ключ . Этот ключ является тенью в еще не созданной пороговой схеме. Алиса сохраняет ряд секретных ключей для себя, внося некоторую непредсказу емость в систему. Пусть всего существует k возможных получателей. Тогда для широковещательной передачи M Алиса шифрует M ключом K и делает следующее.

(1) Алиса выбирает случайное число j. Это число призвано замаскировать количество получателей сообщения. Оно не должно быть слишком большим и даже может равняться нулю .

(2) Алиса создает пороговую схему $(k + j + 1, 2k + j + 1)$, в которой:

K - это секрет.

Секретные ключи адресатов сообщения служат тенями .

Секретные ключи пользователей, которых нет среди получателей сообщения, не являются тенями .

j теней выбираются случайным образом, не совпадая ни с одним секретным ключом.

(3) Алиса широковещательно передает $k + j$ случайно выбранных теней, ни одна из которых не совпадает с тенями этапа (2).

(4) Каждый из слушателей, принявших широковещательное сообщение, добавляет свою тень к полученным $k + j$ теням. Если добавление своей тени позволяет пользователю вычислить секрет, то ему удалось открыть ключ. В противном случае - не удалось .

Другой подход можно найти в [885, 886, 1194]. И еще один - в [1000].

Распределение ключей для конференции

Этот протокол позволяет группе из n пользователей договориться о секретном ключе, используя только н есекретные каналы. Группа использует два общих больших простых числа p и q, а также генератор g той же длины, что и q.

(1) Пользователь i, где i от 1 до n, выбирает случайное число r_i, меньшее q, и широковещательно отправляет

$z_i = g^{r_i} \bmod p$

(2) Каждый пользователь проверяет, что $z_i^q \equiv 1 \pmod{p}$ для всех i от 1 до n.

(3) i-ый пользователь широковещательно передает

$xi = (z_{i+1}/z_{i-1})^{r_i} \bmod p$

(4) i-ый пользователь вычисляет

$K = (z_{i-1})^{nr_i} * x_i^{n-1} * x_{i+1}^{n-2} * \ldots * x_{i-2} \bmod p$

Все вычисления индексов в приведенном протоколе - i-1, i-2 и i+1 - проводятся mod n. По окончании протокола у всех честных пользователей окажется один и тот же K. А все остальные ничего не получат. Однако этот протокол не может устоять перед вскрытием "человек в середине" . Другой протокол, не такой хороший, приведен в [757].

Tateboyashi-Matsuzaki-Newman

Этот протокол распределения ключей подходит для использования в сетях [1521]. Алиса хочет с помощью Трента, KDC, генерировать ключ для сеанса связи с Бобом. Всем участникам известен открытый ключ Трента

n. Тренту известны два простых множителя n, и, следовательно, он может легко вычислять квадратные корни по модулю n. Следующий протокол не содержит некоторых деталей, но позволяет получить общее представление.

(1) Алиса выбирает случайное число r_A и посылает Тренту

$r_A^3 \bmod n$

(2) Трент сообщает Бобу, что кто-то хочет обменяться с ним ключом.

(3) Боб выбирает случайное число r_B и посылает Тренту

$r_B^3 \bmod n$

(4) Трент, используя свой закрытый ключ, расшифровывает r_A и r_B. Он посылает Алисе

$r_A \oplus r_B$

(5) Алиса вычисляет

$(r_A \oplus r_B) \oplus r_A = r_B$

Она использует r_B для безопасного сеанса связи с Бобом.

Протокол выглядит хорошо, но содержит заметный изъян. Кэрол может подслушать этап(3) и использовать эту информацию, воспользовавшись помощью доверчивого Трента и своего сообщника Дэйва, чтобы раскрыть [1472].

(1) Кэрол выбирает случайное число r_C и посылает Тренту

$r_B^3 r_C^3 \bmod n$

(2) Трент сообщает Дэйву, что кто-то хочет обменяться с ним ключом.

(3) Дэйв выбирает случайное число r_D и посылает Тренту

$r_D^3 \bmod n$

(4) Трент, используя свой закрытый ключ, расшифровывает r_C и r_D. Он посылает Кэрол

$(r_B r_C \bmod n) \oplus r_D$

(5) Дэйв посылает r_D Кэрол.

(6) Кэрол использует r_C и r_D для получения r_B. Она использует r_B для расшифровывания переговоров Алисы и Боба.

Это плохо.

Глава 23
Специальные алгоритмы для протоколов

23.1 Криптография с несколькими открытыми ключами

Это обобщение RSA (см. раздел 19.3) [217, 212]. Модуль n является произведением двух простых чисел p и q. Однако вместо e и d, для которых $ed \equiv 1 \bmod ((p\text{-}1)(q\text{-}1))$, выбирается t ключей K_i, для которых выполняется

$K_1 * K_2 * \ldots * K_t \equiv 1 \bmod ((p\text{-}1)(q\text{-}1))$

Так как

$$M^{K_1 * K_2 * \ldots * K_t} = M$$

то эта схема оказывается схемой с несколькими ключами, описанная в разделе 3.5.

Если, например, используется пять ключей, то сообщение, зашифрованное ключами K_3 и K_5, может быть расшифровано с помощью K_1, K_2 и K_4.

$$C = M^{K_3 * K_5} \bmod n$$

$$M = C^{K_1 * K_2 * K_4} \bmod n$$

Одним из применений этой схемы является подписание документа несколькими людьми. Представим ситуацию, когда для того, чтобы документ был действителен, он должен быть подписан и Алисой, и Бобом. Используются три ключа: K_1, K_2 и K_3. Алиса и Боб получают по одному ключу из первых двух, а третий опубликовывается.

(1) Сначала Алиса подписывает M и посылает его Бобу.

$$M' = M^{K_1} \bmod n$$

(2) Боб может восстановить M по M'.

$$M = M'^{K_3 * K_5} \bmod n$$

(3) Он может также добавить свою подпись.

$$M'' = M'^{K_2} \bmod n$$

(4) Проверить подписи можно при помощи открытого ключа K_3.

$$M = M''^{K_3} \bmod n$$

Обратите внимание, что для работоспособности этой системы нужна заслуживающая доверия сторона, которая установила бы систему и выдала ключи Алисе и Бобу. Та же проблема существует и в схеме [484]. Более тонкая схема описана в [695, 830, 700], Но усилия, предпринимаемые для проверки, пропорциональны количеству подписывающих. Новые схемы [220, 1200], основанные на схемах идентификации с нулевым знанием, преодолевают эти недостатки предшествующих систем.

23.2 Алгоритмы разделения секрета

В разделе 3.7 я рассматривал идею, используемую в схемах разделения секрета. Четыре приведенных ниже различных алгоритма представляют собой частные случаи общего теоретического подхода [883].

Схема интерполяционных многочленов Лагранжа

Для создания пороговой схема Ади Шамир воспользовался уравнениями для многочленов в конечном поле [1414]. Выберем простое число p, которое больше количества возможных теней и больше самого большого из возможных секретов. Чтобы сделать секрет общим, сгенерируем произвольный многочлен степени m-1. Например, если нужно создать пороговую схему (3,n) (для восстановления M потребуется три тени), генерируется квадратичный многочлен

$(ax^2 + bx + M) \bmod p$

где p - это случайное простое число, большее любого из коэффициентов. Коэффициенты a и b выбираются случайным образом, они хранятся в тайне и отбрасываются после того, как распределяются тени. M - это сообщение. Простое число должно быть опубликовано. Тени получаются с помощью вычисления многочлена в n различных точках:

$k_i = F(x_i)$

Другими словами, первой тенью может быть значение многочлена при $x = 1$, второй тенью - значение многочлена при $x = 2$, и т.д.

Так как в квадратичных многочленах три неизвестных коэффициента , a, b и M, для создания трех уравнений можно использовать любые три цели. Одной или двух теней не хватит, а четырех или пяти теней будет много .

Например, пусть M равно 11. Чтобы создать пороговую схему (3, 5), в которой любые трое из пяти человек могут восстановить M, сначала получим квадратичное уравнение (7 и 8 - случайно выбранные числа chosen randomly):

$F(x) = (7x^2 + 5x + 11) \bmod 13$

Пятью тенями являются:

$k_1 = F(1) = 7 + 8 + 11 \equiv 0 \pmod{13}$

$k_2 = F(2) = 28 + 16 + 11 \equiv 3 \pmod{13}$

$k_3 = F(3) = 63 + 24 + 11 \equiv 7 \pmod{13}$

$k_4 = F(4) = 112 + 32 + 11 \equiv 12 \pmod{13}$

$k_5 = F(5) = 175 + 40 + 11 \equiv 5 \pmod{13}$

Чтобы восстановить M по трем теням, например, k_2, k_3 и k_5, решается система линейных уравнений:

$a*2^2 + b*2 + M = 3 \pmod{13}$

$a*3^2 + b*3 + M = 7 \pmod{13}$

$a*5^2 + b*5 + M = 5 \pmod{13}$

Решением будут $a = 7$, $b = 8$ и $M = 11$. Итак, M получено.

Эту схему разделения можно легко реализовать для больших чисел. Если вы хотите разбить сообщение на 30 равных частей так, чтобы восстановить сообщение можно было, объединив любые шесть из них , выдайте каждому из 30 человек значения многочлена пятой степени .

$F(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + M \pmod{p}$

Шесть человек могут шесть неизвестных (включая M), но пятерым не удастся узнать ничего об M.

Наиболее впечатляющим моментом совместного использования секрета является то, что, если коэффициенты выбраны случайным образом, пять человек даже при помощи бесконечных вычислительных мощностей не смогут узнать ничего, кроме длины сообщения (которая и так им известна) . Это также безопасно, как одноразовый блокнот, попытка выполнить исчерпывающий поиск (то есть, перебор всех возможных шестых теней) покажет, что любое возможное сообщение останется секретным . Это справедливо для всех представленных в этой книге схем разделения секрета.

Векторная схема

Джордж Блэкли (George Blakley) изобрел схему, использующую понятие точек в пространстве [182]. Сообщение определяется как точка в m-мерном пространстве. Каждая тень - это уравнение (m-1)-мерной гиперплоскости, содержащей эту точку.

Например, если для восстановления сообщения нужны три тени, то оно является точкой в трехмерном пространстве. Каждая тень представляет собой иную плоскость. Зная одну тень, можно утверждать, что точка находится где-то на плоскости. Зная две тени - что она находится где-то на линии пересечения двух плоскостей . Зная три тени, можно точно определить, что точка находится на пересечении трех плоскостей .

Asmuth-Bloom

В этой схеме используются простые числа [65]. Для (m, n)-пороговой схемы выбирается большое простое число p, большее M. Затем выбираются числа, меньшие p - d_1, d_2, . . . d_n, для которых:

1. Значения d_i упорядочены по возрастанию, $d_i < d_{i+1}$
2. Каждое d_i взаимно просто с любым другим d_i
3. $d_1*d_2* \ldots *d_m > p*d_{n-m+2}*d_{n-m+3}* \ldots *d_n$

Чтобы распределить тени, сначала выбирается случайное число r и вычисляется

$M' = M + rp$

Тенями, k_i, являются

$k_i = M' \bmod d_i$

Объединив любые m теней, можно восстановить M, используя китайскую теорему об остатках, но это невозможно с помощью любых m-1 теней. Подробности приведены в [65].

Karnin-Greene-Hellman

В этой схеме используется матричное умножение [818]. Выбирается n+1 m-мерных векторов, V_0, V_1, . . . V_n, так, что ранг любой матрицы размером m*m, образованной из этих векторов, равен m. Вектор U - это вектор размерности m+1.

M - это матричное произведение $U \cdot V_0$. Тенями являются произведения $U \cdot V_i$, где i меняется от 1 до n.

Любые m теней можно использовать для решения системы линейных уравнений размерности m*m, неизвестными являются коэффициенты U. $U \cdot V_0$ можно вычислить по U. Используя любые m-1 теней, решить систему уравнений и, таким образом, восстановить секрет невозможно.

Более сложные пороговые схемы

В предыдущих примерах показаны только простейшие пороговые схемы : секрет делится на n теней так, чтобы, объединив любые m из них, можно было раскрыть секрет. На базе этих алгоритмов можно создать намного более сложные схемы. В следующих примерах будет использоваться алгоритм Шамира, хотя будут работать и все остальные.

Чтобы создать схему, в которой один из участников важнее других, ему выдается больше теней . Если для восстановления секрета нужно пять теней, и у кого-то есть три тени, а у всех остальных - по одной , этот человек вместе с любыми двумя другими может восстановить секрет . Без его участия для восстановления секрета потребуется пять человек.

По несколько теней могут получить два человека и более . Каждому человеку может быть выдано отличное число теней. Независимо от того, сколько теней было роздано, для восстановления секрета потребуется любые m из них. Ни один человек, ни целая группа не смогут восстановить секрет, обладая только m-1 тенями.

Для других схем представим сценарий с двумя враждебными делегациями . Можно распределить секрет так, чтобы для его восстановления потребовалось двое из 7 участников делегации A и трое из 12 участников делегации B. Создается многочлен степени 3, который является произведением линейного и квадратного выражений . Каждому участнику делегации A выдается тень, которая является значением линейного выражения, а участникам делегации B выдаются значения квадратичного выражения .

Для восстановления линейного выражения достаточны любые две тени участников делегации A, но независимо от того, сколько других теней есть у делегации , ее участники не смогут ничего узнать о секрете . Аналогично для делегации B: ее участники могут сложить три тени, восстанавливая квадратное выражение, но другую информацию, необходимую для восстановления секрета в целом, они получить не смогут . Только перемножив свои выражения, участники двух делегаций смогут восстановить секрет .

В общем случае, может быть реализована любая мыслимая схема разделения секрета . Потребуется только написать систему уравнений, соответствующих конкретной системе. Вот несколько прекрасных статей на тему обобщенных схем разделения секрета [1462, 1463, 1464].

Разделение секрета с мошенниками

Этот алгоритм изменяет стандартную пороговую схему (m, n) для обнаружения мошенников [1529]. Я покажу его использование на базе схемы Лагранжа , но алгоритм работает и с другими схемами . Выбирается простое число p, большее n и большее

$(s - 1)(m - 1)/e + m$

где s - это самый большой возможный секрет, а e - вероятность успеха мошенничества . e можно сделать настолько малым, насколько это необходимо, это просто усложнит вычисления . Постройте тени как раньше, но вместо использования 1, 2, 3, . . . , n для x_i, выберите случайным образом числа из диапазона от 1 до p-1.

Теперь, если Мэллори при восстановлении секрета заменит свою часть подделкой , его тень с высокой вероятностью окажется невозможной . Невозможный секрет, конечно же, окажется подделанным секретом . Математика этой схемы приведена в [1529].

К сожалению, хотя мошенничество Мэллори и будет открыто, ему удастся узнать секрет (при условии, что все остальные нужные тени правильны). От этого защищает другой протокол, описанный в [1529, 975]. Основ-

ной идеей является использование набора из k секретов, так чтобы никто из участников заранее не знал, какой из них правильный. Каждый секрет, за исключением настоящего, больше предыдущего. Участники объединяют свои тени, получая один секрет за другим, пока они не получат наименьшее значение секрета. Этот секрет и будет правильным.

В этой схеме мошенники легко выявляются еще до получения конечного секрета. Существует определенные сложности, если участники предъявляют свои тени по очереди, подробности можно найти в литературе. В следующих работах также рассматриваются обнаружение и предотвращение мошенничества в пороговых схемах [355, 114, 270].

23.3 Подсознательный канал

Ong-Schnorr-Shamir

Этот подсознательный канал (см. раздел 4.2), разработанный Густавусом Симмонсом (Gustavus Simmons) [1458, 1459, 1460], использует схему идентификации Ong-Schnorr-Shamir (см. раздел 20.5). Как и в оригинальной схеме отправитель (Алиса) выбирает общедоступный модуль n и закрытый ключ k так, чтобы n и k были взаимно простыми числами. В отличии от оригинальной схемы k используется совместно Алисой и Бобом, получателем в подсознательном канале. Открытый ключ вычисляется следующим образом:

$h = -k^2 \bmod n$

Если Алисе нужно отправить подсознательное сообщение M в безобидном сообщении M', она сначала проверяет, что пары M' и n, а также M и n являются взаимно простыми числами. Алиса вычисляет

$S_1 = 1/2*((M'/M + M)) \bmod n$

$S_2 = 1/2*((M'/M - M)) \bmod n$

Пара чисел S_1 и S_2 представляет собой подпись в традиционной схеме Ong-Schnortr-Shamir и одновременно является носителем подсознательного сообщения.

Тюремщик Уолтер (помните такого?) может проверить подлинность сообщения, как это принято в Ong-Schnorr-Shamir, но Боб может сделать еще кое-что. Он может проверить подлинность сообщения (Всегда возможно, что Уолтер попытается ему подсунуть поддельное сообщение). Он проверяет, что

$S_1^2 - S_2^2 \equiv M' \pmod n$

Если подлинность сообщения доказана, получатель может извлечь и подсознательное сообщение, используя следующую формулу:

$M=M'/(S_1 + S_2 k^{-1}) \bmod n$

Это работает, но не забывайте, что сама схема Ong-Schnorr-Shamir была взломана.

ElGamal

Другой предложенный Симмонсом подсознательный канал [1459], описанный в [1407, 1473], основан на схеме подписи ElGamal см. раздел 19.6).

Генерация ключа выполняется также, как и в основной схеме подписи ElGamal. Сначала выбирается простое число p и два случайных числа, g и r, меньшие p. Затем вычисляется

$K = g^r \bmod p$

Открытым ключом служат K, g и p. Закрытым ключом является r. Помимо Алисы r известно и Бобу, это число используется не только для подписи безобидного сообщения, но и в качестве ключа для отправки и чт ения подсознательного сообщения.

Чтобы послать подсознательное сообщение M в безобидном сообщении, M', M и p должны быть попарно взаимно простыми, кроме того, взаимно простыми должны быть M и p-1. Алиса вычисляет

$X = g^M \bmod p$

и решает следующее уравнение для Y (с помощью расширенного алгоритма Эвклида):

$M' = rX + MY \bmod (p\text{-}1)$

Как и в базовой схеме ElGamal, подписью является пара чисел: X и Y. Уолтер может проверить подпись ElGamal. Он убеждается, что

$K^X X^Y \equiv g^{M'} \pmod p$

Боб может восстановить подсознательное сообщение. Сначала он убеждается, что

$(g^r)^X X^Y \equiv g^{M'} \pmod p$

Если это так, он считает сообщение подлинным (не подделанным Уолтером). Затем для восстановления M он вычисляет

$M = (Y^{-1} (M' - rX)) \bmod (p - 1)$

Например, пусть $p = 11$, а $g = 2$. Закрытый ключ r выбирается равным 8. Это означает, что открытым ключом, который Уолтер может использовать для проверки подписи, будет $g^r \bmod p = 2^8 \bmod 11 = 3$.

Чтобы отправить подсознательное сообщение $M = 9$, используя безобидное сообщение $M' = 5$, Алиса проверяет, что 9 и 11, а также 5 и 11 попарно взаимно просты. Она также убеждается, что взаимно просты 9 и 11-1=10. Это так, поэтому она вычисляет

$X = g^{M'} \pmod p = 2^9 \bmod 11 = 6$

Затем она решает следующее уравнение для Y:

$5 = 8\ 6 + 9\ Y \bmod 10$

$Y = 3$, поэтому подписью служит пара чисел 6 и 3 (X и Y). Боб убеждается, что

$(g^r)^X X^Y \equiv g^{M'} \pmod p$

$(2^8)^6 6^3 \equiv 2^5 \pmod{11}$

Это так (выполните арифметические действия самостоятельно, если вы мне не верите), поэтому он может раскрыть подсознательное сообщение, вычисляя

$M = (Y^{-1} (M' - rX)) \bmod (p - 1) = 3^{-1}(5 - 8*6) \bmod 10 = 7(7) \bmod 10 = 49 \bmod 10 = 9$

ESIGN

Подсознательный канал можно добавить и к ESIGN [1460] (см. раздел 20.6). В ESIGN секретный ключ является парой больших простых чисел p и q, а открытым ключом служит $n = p^2q$. Использовании подсознательного канала закрытым ключом являются три простых числа p, q и r, а открытым ключом - n, такое что

$n = p^2qr$

Переменная r - это дополнительные данные, нужные Бобу для прочтения подсознательного сообщения .

Чтобы подписать обычное сообщение, Алиса сначала выбирает случайное число x, меньшее pqr, и вычисляет:

w, наименьшее целое, которое больше или равно $(H(m) - x^k \bmod n)/pq$

$s = x + ((w/kx^{k-1} \bmod p)\ pq$

$H(m)$ - это хэш-значение сообщения, а k - параметр безопасности. Подписью является значение s.

Для проверки подписи Боб вычисляет $s^k \bmod n$. Кроме этого, он вычисляет a, наименьшее целое, которое больше или равно удвоенному числу битов n, деленному на 3. Если $H(m)$ меньше или равна $s^k \bmod n$, и если $s^k \bmod n$ меньше $H(m)+2^a$, то подпись считается правильной.

Для отправки подсознательного сообщения M с помощью безобидного сообщения M' Алиса вычисляет s, используя M вместо of $H(m)$. Это означает, что сообщение должно быть меньше, чем p^2qr. Затем она выбирает случайное число u и вычисляет

$x' = M' + ur$

Затем это значение x' используется в качестве "случайного числа" x при подписи M'. Соответствующее значение s посылается в качестве подписи.

Уолтер может проверить, что s (второе s) является правильной подписью M' Точно также проверить подлинность сообщения может и Боб. Но, так как ему известно и r, он может вычислить

$s = x' + ypqr = M + ur + ypqr \equiv M \pmod r$

Эта реализация подсознательного канала намного лучше двух предыдущих. В вариантах Ong-Schnorr-Shamir и ElGamal у Боба должен быть закрытый ключ Алисы. Боб сможет не только читать подсознательные сообщения Алисы, но и выдавать себя за Алису, подписывая обычные документы . Алиса ничего с этим не сможет поделать, устанавливая такой подсознательный канал, ей придется довериться Бобу .

Схема ESICN страдает от этой проблемы. Закрытым ключом Алисы служит набор трех простых чисел: p, q

и r. Секретным ключом Боба является только r. Он знает $n = p^2qr$, но, чтобы раскрыть p и q, ему понадобится разложить на множители это число. Если простые числа достаточно велики, Бобу будет так же трудно выдать себя за Алису, как и Уолтеру или кому-нибудь еще.

DSA

Подсознательный канал существует и в DSA (см. раздел 20.1) [1468, 1469, 1473]. На самом деле их даже может быть несколько. Простейший подсознательный канал включает выбор k. Предполагается, что это будет 160-битовое число. Однако, если Алиса выбирает конкретное k, то Боб, зная закрытый ключ Алисы, сможет раскрыть это k. Алиса посылать Бобу 160-битовое подсознательное сообщение в каждой подписи DSA, а все остальные будут только проверять подпись Алисы. Дополнительное усложнение: Так как k должно быть случайным, Алиса и Боб должны использовать общий одноразовый блокнот и шифровать подсознательное сообщение с помощью этого блокнота, генерируя k.

В DSA есть подсознательные каналы, не требующие передавать Бобу закрытый ключ Алисы. Они также подразумевают выбор конкретных значений k, но не могут передавать по 160 битов информации. Следующая схема, представленная в [1468, 1469], позволяет Алисе и Бобу обмениваться в каждой подписи одним битом подсознательной информации.

(1) Алиса и Боб выбирают случайное простое число P (отличающееся от параметра p в схеме подписи). Это секретный ключ для подсознательного канала.

(2) Алиса подписывает безобидное сообщение M. Если она хочет отправить Бобу подсознательный бит 1, она убеждается, что параметр r подписи является квадратичным остатком по модулю P. Если она хочет отправить ему 0, она проверяет, что параметр r подписи не является квадратичным остатком по модулю P. Она добивается этого, подписывая сообщение с помощью случайных значений k, пока она не получит подпись с нужным ей свойством для r. Так как числа, являющиеся квадратичными остатками и не являющиеся ими, равновероятны, то это не должно быть слишком сложно.

(3) Алиса посылает Бобу подписанное сообщение.

(4) Боб проверяет подпись, убеждаясь в подлинности сообщения. Затем он проверяет, является ли r квадратичным остатком по модулю P и восстанавливает подсознательный бит.

Передача таким образом нескольких битов подразумевает подбор такого r, которое является или не является квадратичным остатком по нескольким модулям. Подробности приведены в [1468, 1469].

Эта схема может быть легко расширена для передачи нескольких подсознательных битов на подпись. Если Алиса и Боб выбирают два случайных числа P и Q, то Алиса может посылать два бита, выбирая случайное k так, чтобы r являлось или не являлось квадратичным остатком $\bmod P$, а также являлось или не являлось квадратичным остатком $\bmod Q$. Случайное значение k с вероятностью 25 процентов позволит получить r с нужными свойствами.

Вот как Мэллори, нечестный реализатор DSA, может создать алгоритм, извлекающий по 10 битов закрытого ключа Алисы из каждой ее подписи.

(1) Мэллори строит свою реализацию DSA базе устойчивой к взлому СБИС, чтобы никто не смог проверить, как она работает. Он создает 14 подсознательных каналов в своей реализации DSA. То есть, он выбирает 14 случайных простых чисел и использует микросхему, которая выбирает значение k так, чтобы r являлось или не являлось квадратичным остатком по модулю каждого из этих 14 простых чисел, в зависимости от подсознательного сообщения.

(2) Мэллори выдает микросхемы Алисе, Бобу и остальным желающим.

(3) Алиса обычным образом подписывает сообщение, используя свой закрытый 160-битовый ключ x.

(4) Микросхема случайным образом выбирает 10-битовый блок x: первые 10 битов, вторые 10 битов, и т.д. Так как существует 16 возможных 10-битовых блоков, то номер блока выражается 4-битовым числом. Этот 4-битовый идентификатор и 10 битов ключа и будут 14-битовым подсознательным сообщением.

(5) Микросхема перебирает случайные значения k, пока не удастся найти то, которое обладает правильными квадратичными остатками, нужными для передачи подсознательного. Вероятность случайного k обладать правильной формой равна 1/16384. Если микросхема может проверить 10000 значений k в секунду, нужное значение будет найдено меньше, чем за пару секунд. Эти вычисления не зависят от сообщения и могут быть вычислены заранее, до того, как Алиса захочет подписать сообщение.

(6) Микросхема обычным образом подписывает сообщение, используя выбранное на этапе (5) значение k.

(7) Алиса посылает цифровую подпись Бобу, или опубликовывает ее в сети, или еще что-нибудь делает.

(8) Мэллори раскрывает r и, так как он знает 14 простых чисел, расшифровывает подсознательное

сообщение.

Страшнее всего, что, даже если Алиса знает, что происходит, она ничего не сможет доказать. Пока 14 простых чисел хранятся в секрете, Мэллори в безопасности.

Уничтожение подсознательного канала в DSA

Подсознательный канал опирается на то, что Алиса может выбирать k для передачи подсознательной информации. Чтобы сделать подсознательный канал невозможным, Алисе не должно быть позволено выбирать k. Однако, выбор k должен быть запрещен и для всех других. Если кому-то другому будет позволено выбирать k, то этот человек получит возможность подделать подпись Алисы. Единственным решением для Алисы является проведение генерации k вместе с другой стороной, Бобом, так, чтобы Алиса не могла контролировать ни один бит k, а Боб не мог определить ни один бит k. На другой стороне протокола у Боба должна быть возможность проверить, что Алиса использовала именно совместно созданное k.

Вот этот протокол [1470, 1472, 1473]

(1) Алиса выбирает k' и посылает Бобу

$u = g^{k'} \bmod p$

(2) Боб выбирает k'' и посылает его Алисе.

(3) Алиса вычисляет $k = k'k'' \bmod (p - 1)$. Она использует k, чтобы подписать свое сообщение M, используя DSA, и посылает Бобу свою подпись: r и s.

(4) Боб проверяет, что $((u = g^{k'} \bmod p) \bmod q) = r$

Если это так, то он знает, что для подписи M использовалось k. После этапа (4) Боб знает, что в r не было включено никакой подсознательной информации. Если он является доверенной стороной, он может проверить, что в подписи Алисы нет подсознательной информации. Другим придется поверить его заявлению, Боб не сможет доказать этот факт третьей стороне, воспроизведя протокол.

Удивительно то, что Боб, если захочет, может использовать этот протокол для создания собственного подсознательного канала. Боб может включить подсознательную информацию в одну из подписей Алисы, выбрав k'' с определенными характеристиками. Когда Симмонс открыл такую возможность, он назвал ее "Каналом кукушки". Подробности работы Канала кукушки, и мешающий этому трехпроходный протокол генерации k, рассматриваются в [1471, 1473].

Другие схемы

Подсознательный канал можно организовать для любой схемы подписи [1458, 1460, 1406]. Описание протокола встраивания подсознательного канала в схемы Fiat-Shamir и Feige-Fiat-Shamir вместе с возможными злоупотреблениями можно найти в [485].

23.4 Неотрицаемые цифровые подписи

Автором этого алгоритма неотрицаемой подписи (см. раздел 4.3) является Дэвид Чаум (David Chaum) [343,327]. Сначала опубликовываются большое простое число p и примитивный элемент g, которые будут совместно использоваться группой подписывающих. У Алисы есть закрытый ключ x и открытый ключ $g^x \bmod p$.

Чтобы подписать сообщение, Алиса вычисляет $z = m^x \bmod p$. Это все, что ей нужно сделать. Проверка подписи немного сложнее.

(1) Боб выбирает два случайных числа, a и b, меньшие p, и отправляет Алисе:

$c = z^a (g^x)^b \bmod p$

(2) Алиса вычисляет $t = x^{-1} \bmod (p-1)$, и отправляет Бобу:

$d = c^t \bmod p$

(3) Боб проверяет, что

$d \equiv m^a g^b \pmod p$

Если это так, он считает подпись истинной.

Представим, что Алиса и Боб выполнили этот протокол, и Боб теперь считает, что Алиса подписала сообщение. Боб хочет убедить в этом Кэрол, поэтому он показывает ей запись протокола. Дэйв, однако, хочет убедить Кэрол, что документ подписан кем-то другим. Он создает поддельную запись протокола. Сначала он генерирует сообщение на этапе (1). Затем на этапе (3) он генерирует d и ложную передачу от другого человека на этапе (2).

Наконец, он создает сообщение этапа (2). Для Кэрол записи Боба и Дэйва одинаковы. Ее невозможно убедить в правильности подписи, пока она не выполнит протокол самостоятельно .

Конечно, если бы она следила из-за плеча Боба за тем, как он выполняет протокол, она была бы убеждена . Кэрол нужно увидеть выполнение этапов по порядку, так, как это делал Боб .

Используя эту схему подписи, можно столкнуться с проблемой, но я не знаю подробностей . Прежде, чем воспользоваться этой схемой, просмотрите литературу .

Другой протокол включает не только протокол подтверждения - Алиса может убедить Боба в правильности своей подписи - но и протокол отрицания. Алиса может с помощью интерактивного протокола с нулевым знанием убедить Боба, что ее подпись неправильна, если это так [329].

Как и предыдущий протокол группа подписывающих использует общедоступное большое простое число p и примитивный элемент g. У Алисы есть закрытый ключ x и открытый ключ $g^x \bmod p$. Чтобы подписать сообщение, Алиса вычисляет $z = m^x \bmod p$. Чтобы проверить подпись:

(1) Боб выбирает два случайных числа, a и b, меньшие p, и отправляет Алисе:

$c = m^a g^b \bmod p$

(2) Алиса выбирает случайное число q, меньшее p, а затем вычисляет и отправляет Бобу:

$s_1 = cg^q \bmod p$, $s_2 = (cg^q)^x \bmod p$

(3) Боб посылает Алисе a и b, чтобы Алиса могла убедиться, что Боб не мошенничал на этапе (1).

(4) Алиса посылает Бобу q, чтобы он мо воспользоваться m^x и восстановить s_1 и s_2. Если

$s_1 \equiv cg^q \bmod p$

$s_2 \equiv (g^x)^{b+q} z^a \pmod p$

то подпись правильна.

Алиса может также отказаться от подписи z под сообщением m. Подробности приведены в [329]. Дополнительные протоколы для неотрицаемых подписей можно найти в [584, 344]. Лейн Харн (Lein Harn) и Шубао Янг (Shoubao Yang) предложили схему групповых неотрицаемых подписей [700].

Преобразуемые неотрицаемые подписи

Алгоритм для **преобразуемых неотрицаемых подписей**, которые можно проверять, отменять и преобразовывать в обычные неотрицаемые подписи, приведен в [213]. Он основан на алгоритме цифровых подписей ElGamal.

Как и в ElGamal, сначала выбираются два простых числа, p и q, так, чтобы q было делителем p-1. Теперь нужно создать число g, меньшее q. В диапазоне от 2 до p-1 выбирается случайное число h и вычисляется

$g=h^{(p-1)/q} \bmod p$

Если g равно 1, выбирается другое случайное h. Если нет, используется полученное значение g.

Закрытыми ключами служат два различных случайных числа, x и z, меньшие q. Открытыми ключами являются p, q, g, y и u, где

$y = g^x \bmod p$

$u=g^я \bmod p$

Для вычисления преобразуемой неотрицаемой подписи сообщения m (которое в действительности является хэш-значением сообщения), сначала диапазоне от 1 до q-1 выбирается случайное число t. Затем вычисляется

$T = g^r \bmod p$

и

$m' = Ttzm \bmod q$.

Теперь вычисляется обычная подпись ElGamal для m'. Выбирается случайное число R, меньшее p-1 и взаимно простое с ним. Затем вычисляется $r = g^R \bmod p$ и, с помощью расширенного алгоритма Эвклида, вычисляется s, для которого

$m' \equiv rx + Rs \pmod q$

Подписью служат подпись ElGamal (r, s) и T. Вот как Алиса подтверждает свою подпись Бобу:

(1) Боб генерирует два случайных числа, a и b, и вычисляет $c = T^{Tma}g^b \mod p$ и посылает результат Алисе.

(2) Алиса генерирует случайное число k и вычисляет $h_1 = cg^k \mod p$ и $h_2 = h_1^z \mod p$, а затем посылает оба числа Бобу.

(3) Боб посылает Алисе a и b.

(4) Алиса проверяет, что $c = T^{Tma}g^b \mod p$. Она посылает k Бобу.

(5) Боб проверяет, что $h_1 = T^{Tma}g^{b+k} \mod p$, и что $h_2 = y^{ra}r^{sa}u^{b+k} \mod p$.

Алиса может преобразовать все свои неотрицаемые подписи в обычные, опубликовав z. Теперь любой может проверить ее подпись без ее помощи.

Схемы неотрицаемых подписей можно объединить со схемами разделения секрета, создав **распределенные преобразуемые неотрицаемые подписи** [1235]. Кто-нибудь может подписать сообщение, а затем распределить возможность подтверждения правильности подписи. Он может, например, потребовать, чтобы в протоколе убеждения Боба в правильности подписи участвовали трое из пяти обладателей возможность подтверждения правильности. В [700, 1369] предложены улучшения, позволяющие отказаться от необходимости доверенного лица - распределителя.

23.5 Подписи, подтверждаемые доверенным лицом

Вот как Алиса может подписать сообщение, а Боб проверить его так, чтобы и Кэрол немного позже могла доказать Дэйву правильность подписи Алисы (см. раздел 4.4) [333].

Сначала опубликовываются большое простое число p и примитивный элемент g, которые будут совместно использоваться группой пользователей. Также опубликовывается n, произведение двух простых чисел. У Кэрол есть закрытый ключ z и открытый ключ $h = g^x \mod p$.

В этом протоколе Алиса может подписать m так, чтобы Боб мог проверить правильность ее подписи, но не мог убедить в этом третью сторону.

(1) Алиса выбирает случайное x и вычисляет

$a = g^x \mod p$

$b = h^x \mod p$

Она вычисляет хэш-значение m, $H(m)$, и хэш-значение объединения a и b, $H(a,b)$, а затем

$j = (H(m) \oplus H(a,b))^{1/3} \mod n$

и посылает a, b и j Бобу.

(2) Боб выбирает два случайных числа, s и t, меньших p, и посылает Алисе

$c = g^s h^t \mod p$

(3) Алиса выбирает случайное q, меньшее p, и посылает Бобу

$d = g^q \mod p$

$e = (cd)^x \mod p$

(4) Боб посылает Алисе s и t.

(5) Алиса проверяет, что

$g^s h^t \equiv c \pmod p$

затем она посылает Бобу q.

(6) Боб проверяет

$d \equiv g^q \mod p$

$e/a^q \equiv a^s b^t \pmod p$

$(H(m) \oplus H(a,b)) = j^{1/3} \mod n$

Если все тождества выполняются, то Боб считает подпись истинной.

Боб не может использовать запись этого доказательства для убеждения Дэйва в истинности подписи, но Дэйв может выполнить протокол с доверенным лицом Алисы, Кэрол. Вот как Кэрол убеждает Дэйва в том, что a и b образуют правильную подпись.

(1) Дэйв выбирает случайные u и v, меньшие p, и посылает Кэрол

$k = g^u a^v \bmod p$

(2) Кэрол выбирает случайное w, , меньшее p, и посылает Дэйву

$l = g^w \bmod p$

$y = (kl)^z \bmod p$

(3) Дэйв посылает Кэрол u и v.

(4) Кэрол проверяет, что

$g^u a^v \equiv k \pmod p$

Затем она посылает Дэйву w.

(5) Дэйв проверяет, что

$g^w \equiv l \pmod p$

$y/h^w \equiv h^u b^v \pmod p$

Если все тождества выполняются, то Дэйв считает подпись истинной.

В другом протоколе Кэрол может преобразовать протокол доверенного лица в обычную цифровую подпись. Подробности в [333].

23.6 Вычисления с зашифрованными данными

Проблема дискретного логарифма

Существует большое простое число p и генератор g. Алиса хочет для конкретного x найти такое e, для которого

$g^e \equiv x \pmod p$

Это трудная проблема, и Алисе не хватает вычислительных мощностей для вычисления результата. У Боба есть такие возможности - он представляет правительство, или мощный вычислительный центр, или еще какую-нибудь влиятельную организацию. Вот как Алиса может получить помощь Боба, не раскрыв ему x [547, 4]:

(1) Алиса выбирает случайное число r, меньшее p.

(2) Алиса вычисляет

$x' = xg^r \bmod p$

(3) Алиса просит Боба решить

$g^{e'} \equiv x' \pmod p$

(4) Боб вычисляет e' и посылает его Алисе.

(5) Алиса восстанавливает e, вычисляя

$e = (e' - r) \bmod (p - 1)$

Аналогичные протоколы для проблем квадратичных остатков и примитивных корней приведены в [3, 4]. (См. также раздел 4.8.)

23.7 Бросание "честной" монеты

Следующие протоколы позволяют Алисе и Бобу бросать честную монету в сети передачи данных (см. раздел 4.9) [194]. Это пример бросания монеты в колодец (см. раздел 4.10). Сначала только Боб узнает результат броска и сообщает его Алисе. Затем Алиса может проверить, что Боб сообщил правильный результат броска.

Бросание "честной" монеты с помощью квадратных корней

Подпротокол бросания честной монеты:

(1) Алиса выбирает два больших простых числа, p и q, и посылает их произведение n Бобу.

(2) Боб выбирает случайное положительное целое число r, меньшее $n/2$. Боб вычисляет

$z = r^2 \bmod n$

и посылает z Алисе.

(3) Алиса вычисляет четыре квадратных корня z (mod n). Она может сделать это, так как она знает разложение n на множители. Назовем их $+x$, $-x$, $+y$ и $-y$. Обозначим как x' меньшее из следующих двух чисел:

x mod n

$-x$ mod n

Аналогично, обозначим как y' меньшее из следующих двух чисел:

y mod n

$-y$ mod n

Обратите внимание, что r равно либо x', либо y'.

(4) Алиса делает пытается угадать, какое из значений равно $r - x'$ или y', и посылает свою догадку Бобу.

(5) Если догадка Алисы правильна, результатом броска монеты является "орел", а если неправильна - "решка". Боб объявляет результат броска монеты.

Подпротокол проверки:

(6) Алиса посылает p и q Бобу.

(7) Боб вычисляет x' и y' и посылает их Алисе.

(8) Алиса вычисляет r.

У Алисы нет возможности узнать r, поэтому она действительно угадывает. Она на этапе (4) сообщает Бобу только один бит своей догадки, не давая Бобу получить и x', и y'. Если Боб получит оба этих числа, он сможет изменить r после этапа (4).

Бросание "честной" монеты с помощью возведения в степень по модулю p

В этом протоколе в качестве однонаправленной функции используется возведение в степень по модулю простого числа p [1306]:

Подпротокол бросания честной монеты:

(1) Алиса выбирает простое число p так, чтобы множители p-1 были известны, и среди них было по крайней мере одно большое простое число.

(2) Боб выбирает два примитивных элемента, h и t, в GF(p). Он посылает их Алисе.

(3) Алиса убеждается, что h и t являются примитивными элементами, и затем выбирает случайное число x, взаимно простое с p-1. Затем она вычисляет одно из двух значений:

$y = h^x \bmod p$, или $y = t^x \bmod p$

Она посылает y Бобу.

(4) Боб пытается угадать, вычислила Алиса y как функцию h или как функцию t, и посылает свое предположение Алисе.

(5) Если догадка Боба правильна, результатом бросания монеты является "орел", в противном случае - "решка". Алиса объявляет результат броска монеты.

Подпротокол проверки:

(6) Алиса раскрывает Бобу значение x. Боб вычисляет $h^x \bmod p$ и $t^x \bmod p$, убеждаясь, что Алиса играла честно и проверяя результат броска. Он также проверяет, что x и p-1 - взаимно простые числа.

Чтобы Алиса могла смошенничать, она должна знать два целых числа, x и x', для которых выполняется $h^x \equiv t^{x'} \bmod p$. Для того, чтобы узнать эти значения, ей нужно вычислить:

$\log_t h = x'x^{-1} \bmod p\text{-}1$ и $\log_t h = xx'^{-1} \bmod p\text{-}1$.

Это трудные проблемы.

Алиса смогла бы сделать это, если бы она знала $\log_t h$, но Боб выбирает h и t на этапе (2). У Алисы нет другого способа кроме, как попытаться вычислить дискретный логарифм. Алиса может также попытаться смошенничать, выбрав x, которое не является взаимно простым с p-1, но Боб обнаружит это на этапе (6).

Боб может смошенничать, если h и t не являются примитивными элементами в поле in GF(p), но Алиса сможет легко проверить это после этапа (2), так как ей известно разложение p-1 на простые множители.

Удачным в этом протоколе является то, что если Алиса и Боб захотят бросить несколько монет, он7и смогут использовать одни и те же значения p, h и t. Алиса просто генерирует новое x, и протокол продолжается с этапа (3).

Бросание "честной" монеты с помощью целых чисел Блюма

В протоколе бросания монеты можно использовать челые числа Блюма .

(1) Алиса генерирует целое число Блюма n, случайное x, взаимно простое с n, $x_0 = x^2 \bmod n$ и $x_1 = x_0^2 \bmod n$. Она посылает Бобу n и x_1.

(2) Боб угадывает, четным или нечетным является x_0.

(3) Алиса посылает x Бобу.

(4) Боб проверяет, что n является целым числом Блюма (Алиса нужно передать Бобу множители n и доказательства того, что они являются простыми, или выполнить некоторый протокол с нулевым знанием, убеждающий Боба, что n - это целое число Блюма), и что $x_0 = x^2 \bmod n$ и $x_1 = x_0^2 \bmod n$. Если все проверки выполняются, и Боб угадал правильно, он выигрывает бросок .

Это важно, чтобы n было числом Блюма. Иначе Алиса сможет найти такое x'_0, что $x'^2_0 \bmod n = x_0^2 \bmod n = x_1$, где x'_0 также является квадратичным остатком. Если бы x_0 был четным, а x'_0 - нечетным (или наоборот), Алиса могла бы мошенничать.

23.8 Однонаправленные сумматоры

Существует простая функция однонаправленного сумматоры [116] (см. раздел 4.12.):

$A(x_i, y) = x_{i-1}^y \bmod n$

Числа n (являющееся произведением двух простых чисел) и x_0 должны быть заранее согласованы. Тогда суммированием y_1, y_2 и y_3 будет

$$((x_0^{y_q} \bmod n)^{y_2} \bmod n)^{y_3} \bmod n$$

Это вычисление не зависит от порядка y_1, y_2 и y_3.

23.9 Раскрытие секретов "все или ничего"

Этот протокол позволяет нескольким сторонам (для работы протокола нужно не меньше двух участников) покупать различные секреты у одного продавца (см. раздел 4.13) [1374, 1175]. Начнем с определения. Возьмем две строки битов, x и y. Фиксированным битовым индексом (fixed bit index, **FBI**) x и y называется последовательность номеров совпадающих битов этих строк.

Например:

$x = 110101001011$

$y = 101010000110$

$FBI(x, y) = \{1, 4, 5, 11\}$

(Мы читаем биты справа налево, считая нулевым крайний правый бит .)

Теперь вот как выглядит протокол. Алиса будет продавцом. Боб и Кэрол - покупателями. У Алисы есть k n-битовых секретов: $S_1, S_2, \ldots S_k$. Боб хочет купить секрет S_b, Кэрол - секрет S_c.

(1) Алиса генерирует пару "открытый ключ/закрытый ключ"и сообщает Бобу (но не Кэрол) открытый ключ. Она генерирует другую пару "открытый ключ/закрытый ключ"и сообщает Кэрол (но не Бобу) открытый ключ.

(2) Боб генерирует k n-битовых случайных чисел, $B_1, B_2, \ldots B_k$, и сообщает их Кэрол. Кэрол генерирует k n-битовых случайных чисел, $C_1, C_2, \ldots C_k$, и сообщает их Бобу.

(3) Боб шифрует C_b (напомним, он хочет купить секрет S_b) открытым ключом, полученным от Алисы. Он вычисляет FBI для C_b и только что зашифрованного результата. Он посылает этот FBI Кэрол.

Кэрол шифрует B_c (напомним, она хочет купить секрет S_c) открытым ключом, полученным от Алисы. Она вычисляет FBI для B_c и только что зашифрованного результата. Она посылает этот FBI Бобу.

(4) Боб берет каждое из n-битовых чисел $B_1, B_2, \ldots B_k$ и заменяет каждый бит, номера которого нет в FBI, полученном от Кэрол, его дополнением. Он посылает этот новый список n-битовых чисел $B'_1, B'_2, \ldots B'_k$

Алисе.

Кэрол берет каждое из n-битовых чисел $C_1, C_2, \ldots C_k$ и заменяет каждый бит, номера которого нет в FBI, полученном от Боба, его дополнением. Она посылает этот новый список n-битовых чисел $C'_1, C'_2, \ldots C'_k$ Алисе.

(5) Алиса расшифровывает все C'_i закрытым ключом Боба, получая k n-битовых чисел $C''_1, C''_2, \ldots C''_k$. Она вычисляет $S_i \oplus C''_i$ для $i = 1, \ldots k$, и посылает результаты Бобу.

Алиса расшифровывает все B'_i закрытым ключом Кэрол, получая k n-битовых чисел $B''_1, B''_2, \ldots B''_k$. Она вычисляет $S_i \oplus B''_i$ для $i = 1, \ldots k$, и посылает результаты Кэрол.

(6) Боб вычисляет S_b, выполняя XOR C_b и b-го числа, полученного от Алисы.

Кэрол вычисляет S_c, выполняя XOR B_c и c-го числа, полученного от Алисы..

Все так сложно. Поясним эти долгие действия на примере.

У Алисы есть для продажи восемь 12-битовых секретов: $S_1 = 1990$, $S_2 = 471$, $S_3 = 3860$, $S_4 = 1487$, $S_5 = 2235$, $S_6 = 3751$, $S_7 = 2546$ и $S_8 = 4043$. Боб хочет купить S_7, а Кэрол - S_2.

(1) Алиса использует алгоритм RSA. В диалоге с Бобом она использует следующую пару ключей: $n = 7387$, $e = 5145$ и $d = 777$, а в диалоге с Кэрол - $n = 2747$, $e = 1421$ и $d = 2261$. Она сообщает Бобу и Кэрол их открытые ключи.

(2) Боб генерирует восемь 12-битовых чисел, $B_1 = 743$, $B_2 = 1988$, $B_3 = 4001$, $B_4 = 2942$, $B_5 = 3421$, $B_6 = 2210$, $B_7 = 2306$ и $B_8 = 222$, и сообщает их Кэрол. Кэрол генерирует восемь 12-битовых чисел, $C_1 = 1708$, $C_2 = 711$, $C_3 = 1969$, $C_4 = 3112$, $C_5 = 4014$, $C_6 = 2308$, $C_7 = 2212$ и $C_8 = 222$, и сообщает их Бобу.

(3) Боб хочет купить S_7, поэтому он открытым ключом, выданным Алисой, шифрует C_7.

$2212^{5145} \bmod 7387 = 5928$

Теперь:

2212 = 0100010100100

5928 = 1011100101000

Следовательно, FBI этих двух чисел равен {0, 1, 4, 5, 6}. Он посылает его Кэрол.

Кэрол хочет купить S_2, поэтому она открытым ключом, выданным Алисой, шифрует B_2 и вычисляет FBI B_2 и результата шифрования. Она посылает Бобу {0, 1, 2, 6, 9, 10}.

(4) Боб берет $B_1, B_2, \ldots B_8$ и заменяет каждый бит, индекс которого отсутствует в наборе {0, 1, 2, 6, 9, 10} его дополнением. Например:

B_2 = 111111000100 = 1988

B'_2 = 011001111100 = 1660

Он посылает $B'_1, B'_2, \ldots B'_8$ Алисе.

Кэрол берет $C_1, C_2, \ldots C_8$ и заменяет каждый бит, индекс которого отсутствует в наборе {0, 1, 4, 5, 6} его дополнением. Например:

C_7 = 0100010100100 = 2212

C'_7 = 1011100101000 = 5928

Она посылает $C'_1, C'_2, \ldots C'_8$ Алисе.

(5) Алиса расшифровывает все C'_i закрытым ключом Боба и выполняет XOR результатов с S_i. Например, для $i = 7$:

$5928^{777} \bmod 7387 = 2212$; $2546 \oplus 2212 = 342$

Она посылает результат Бобу.

Алиса расшифровывает все B'_i закрытым ключом Кэрол и выполняет XOR результатов с S_i. Например, для $i = 2$:

$1660^{2261} \pmod{2747} = 1988$; $471 \oplus 1988 = 1555$

Она посылает результат Кэрол.

(6) Боб вычисляет S_7, выполняя XOR C_7 и седьмого числа, полученного им от Алисы:

2212 ⊕ 342=2546

Кэрол вычисляет S_2, выполняя XOR B_2 и второго числа, полученного ей от Алисы.

1988 ⊕ 1555 = 471

Протокол работает для любого количества покупателей. Если Боб, Кэрол и Дэйв хотят купить секреты, Алиса выдает каждому покупателю два открытых ключа, по одному на каждого другого покупателя. Каждый покупатель получает набор чисел от каждого другого покупателя. Затем они выполняют протокол с Алисой для каждого из своих наборов номеров и выполняют XOR всех полученных от Алисы результатов, получая свои секреты. Более подробно это описано в [1374, 1175].

К сожалению, пара нечестных участников могут смошенничать. Алиса и Кэрол, действуя на пару, могут легко понять, какой секрет получил Боб: если они знают FBI C_b и алгоритм шифрования Боба, они могут подыскать такое b, что у C_b будет правильный FBI. А Боб и Кэрол, действуя вместе, могут легко заполучить все секреты Алисы.

Если вы считаете, что участники честны, можно использовать протокол попроще [389].

(1) Алиса шифрует все секреты RSA и посылает их Бобу:

$C_i = S_i^e \bmod n$

(2) Боб выбирает свой секрет C_b, генерирует случайное число r и посылает Алисе.

$C' = C_b r^e \bmod n$

(3) Алиса посылает Бобу^

$P' = C'^d \bmod n$

(4) Боб вычисляет P'

$S_b = P' r^{-1} \bmod n$

Если участники могут жульничать, Боб может доказать с нулевым знанием, что он знает некоторое r, такое что $C' = C_b r^e \bmod n$, и хранить в b секрете, пока Алиса не передаст ему на этапе (3) P' [246].

23.10 Честные и отказоустойчивые криптосистемы

Честная схема Diffie-Hellman

Честные криптосистемы представляют собой программный способ условного вручения документов (см. раздел 4.14). Этот пример взят из работ Сильвии Микали (Silvia Micali) [1084, 1085]. Он запатентован [1086, 1087].

В базовой схеме Diffie-Hellman группа пользователей использует общее простое число p и генератор g. Закрытым ключом Алисы является s, а ее открытым ключом $t = g^s \bmod p$. Вот как сделать схему Diffie-Hellman честной (в этом примере используется пять доверенных лиц).

(1) Алиса выбирает пять целых чисел, s_1, s_2, s_3, s_4, s_5, меньших p-1. Закрытым ключом Алисы является

$s = (s_1 + s_2 + s_3 + s_4 + s_5) \bmod p\text{-}1$

а ее открытым ключом

$t = g^s \bmod p$

Алиса также вычисляет

$t_i = g^{s_i} \bmod p$, для $i = 1, \ldots 5$.

Открытыми частями Алисы являются t_i, а закрытыми - s_i.

(2) Алиса посылает закрытую и соответствующую открытую части каждому доверенному лицу. Например, она посылает s_1 и t_2 доверенному лицу 1. Она посылает t в KDC.

(3) Каждое доверенное лицо проверяет, что

$t_i = g^{s_i} \bmod p$

Если это так, доверенное лицо подписывает t_i и посылает его в KDC. Доверенное лицо сохраняет s_i в безопасном месте.

(4) Получив все пять открытых частей, KDC проверяет, что

$t=(t_1 * t_2 * t_3 * t_4 * t_5) \bmod p$

Если это так, KDC признает открытый ключ.

В этот момент KDC знает, что у каждого доверенного лица есть правильная часть, и что они при необходимости смогут восстановить закрытый ключ. Однако ни KDC, ни любые четыре доверенных лица не могут восстановить закрытый ключ Алисы.

Работы Микали [1084, 1085] также содержат последовательность действия для создания честного RSA и для объединения пороговой схемы с честной криптосистемой, позволяющей m доверенным лицам из n восстановить закрытый ключ.

Отказоустойчивая схема Diffie-Hellman

Как и в предыдущем протоколе у группы пользователей есть общие простое число p и генератор g. Закрытым ключом Алисы является s, а ее открытым ключом $t = g^s \bmod p$.

(1) KDC выбирает случайное число B из диапазона от 0 до p-2 и вручает B с помощью протокола вручения битов (см. раздел 4.9).

Алиса выбирает случайное число A из диапазона от 0 до p-2. Она посылает KDC $g^A \bmod p$.

(2) Пользователь "разделяет" A с каждым доверенным лицом, используя схему подтверждаемого совместного использования секрета (см. раздел 3.7).

(3) KDC раскрывает B Алисе.

(4) Алиса проверяет вручение этапа (1). Затем она устанавливает свой открытый ключ равным

$t = g^A g^B \bmod p$

а закрытый ключ равным

$s = (A + B) \bmod (p\text{-}1)$

Доверенные лица могут восстановить A. Так как KDC знает B, этого достаточно для восстановления s. И Алиса не сможет использовать никаких подсознательных каналов для передачи несанкционированной информации. Этот протокол, рассмотренный в [946, 833] в настоящее время патентуется.

23.11 ZERO-KNOWLEDGE PROOFS OF KNOWLEDGE

Доказательство с нулевым знанием для дискретного логарифма

Пегги хочет доказать Виктору, что ей известно x, являющееся решением

$A^x \equiv B \pmod p$

где p - простое число, а x - произвольное число, взаимно простое с p-1. Числа A, B и p общедоступны, а x хранится в секрете. Вот как Пегги, не раскрывая значения x, может доказать, что оно ей известно (см. раздел 5.1) [338, 337].

(1) Пегги генерирует t случайных чисел, $r_1, r_2, \ldots r_t$, причем все r_i меньше p-1.

(2) Пегги вычисляет $h_i = A^{r_i} \bmod p$ для всех значений i и посылает их Виктору.

(3) Пегги и Виктор, воспользовавшись протоколом бросания монеты генерируют t битов: $b_1, b_2, \ldots b_t$.

(4) Для всех t битов Пегги выполняет одну из следующих операций:

a) Если $b_i = 0$, она посылает Виктору r_i

b) Если $b_i = 1$, она посылает Виктору $s_i = (r_i - r_j) \bmod (p\text{-}1)$, где j - наименьшее значение индекса, при котором $b_j = 1$

(5) Для всех t битов Виктор проверяет одно из следующих условий:

a) При $b_i = 0$ что $A^{r_i} \equiv h_i \pmod p$

b) При $b_i = 1$ что $A^{s_i} \equiv h_i h_j^{-1} \pmod p$

(6) Пегги посылает Виктору Z, где

$Z = (x - r_j) \bmod (p\text{-}1)$

(7) Виктор проверяет, что $A^Z \equiv B h_j^{-1} \pmod p$

Вероятность удачного мошенничества Пегги равна $1/2^t$.

Доказательство с нулевым знанием для возможности вскрыть RSA

Алиса знает закрытый ключ Кэрол. Может быть она взломала RSA, а может она взломала дверь квартиры Кэрол и выкрала ключ. Алиса хочет убедить Боба, что ей известен ключ Кэрол. Однако она не хочет ни сообщать Бобу ключ, ни даже расшифровать для Боба одно из сообщений Кэрол. Далее приведен протокол с нулевым знанием, с помощью которого Алиса убеждает Боба, что она знает закрытый ключ Кэрол [888]. Пусть открытый ключ Кэрол - e, ее закрытый ключ - d, а модуль RSA - n.

(1) Алиса и Боб выбирают случайное k и m, для которых

$km \equiv e \pmod{n}$

Числа они должны выбирать случайным образом, используя для генерации k протокол бросания монеты, а затем вычисляя m. Если и k, и m больше 3, протокол продолжается. В противном случае числа выбираются заново.

(2) Алиса и Боб генерируют случайный шифротекст C. И снова они должны воспользоваться протоколом бросания монеты.

(3) Алиса, используя закрытый ключ Кэрол, вычисляет

$M = C^d \bmod n$

Затем она вычисляет

$X = M^k \bmod n$

и посылает X Бобу.

(4) Боб проверяет, что $X^m \bmod n = C$. Если это так, то он убеждается в правильности заявления Алисы.

Аналогичный протокол можно использовать для демонстрации возможности вскрытия проблемы дискретного логарифма [888].

Доказательство с нулевым знанием того, что n является числом Блюма

Пока неизвестно никаких действительно практичных доказательств того, что $n = pq$, где p и q - простые числа, конгруэнтные 3 по модулю 4. Однако если n имеет форму $p^r q^s$, где r и s нечетны, то у числа n сохраняются свойства, которые делают числа Блюма полезными для криптографии. И тогда существует доказательство с нулевым знанием того, что n имеет такую форму.

Предположим, что Алисе известно разложение на множители числа Блюма n, где n обладает рассмотренной выше формой. Вот как она может доказать Бобу, что n имеет такую форму [660].

(1) Алиса посылает Бобу число u, чей символ Якоби равен -1 по модулю n.

(2) Алиса и Боб совместно выбирают случайные биты: $b_1, b_2, \ldots b_k$.

(3) Алиса и Боб совместно выбирают случайные числа: $x_1, x_2, \ldots x_k$.

(4) Для каждого $i = 1, 2, \ldots k$ Алиса посылает Бобу квадратный корень по модулю n для одного из четырех чисел: x_i, $-x_i$, ux_i, $-ux_i$. Символ Якоби квадратного корня должен быть равен b_i.

Вероятность удачного мошенничества Алисы равна $1/2^k$.

23.12 Слепые подписи

Понятие слепых подписей (см. раздел 5.3) было придумано Дэвидом Чаумом (David Chaum) [317, 323], который также предложил и первую реализацию этого понятия [318]. Она использует алгоритм RSA.

У Боба есть открытый ключ e, закрытый ключ d и открытый модуль n. Алиса хочет, чтобы Боб вслепую, не читая, подписал сообщение m.

(1) Алиса выбирает случайное число k из диапазона от 1 до n. Затем она маскирует m, вычисляя

$t = mk^e \bmod n$

(2) Боб подписывает t

$t^d = (mk^e)^d \bmod n$

(3) Алиса снимает маскировку с t^d, вычисляя

$s = t^d/k \bmod n$

(4) Результатом является

$s = m^d \bmod n$

Это можно легко показать

$t^d \equiv (mk^e)^d \equiv m^d k \pmod n$, поэтому $t^d/k = m^d k/k \equiv m^d \pmod n$.

Чаум придумал целое семейство более сложных алгоритмов слепой подписи [320, 324], называемых неожиданными слепыми подписями. Схемы этих подписей сложнее, но они дают больше возможностей .

23.13 Передача с забыванием

В этом протоколе, предложенном Майклом Рабином (Michael Rabin) [1286], Алиса с вероятностью 50 процентов удается передать Бобу два простых числа, p и q. Алиса не знает, успешно ли прошла передача (См. раздел 5.5.) (Этот протокол можно использовать для передачи Бобу любого сообщения с 50-процентной вероятностью успешной передачи, если p и q раскрывают закрытый ключ RSA.)

(1) Алиса посылает Бобу произведение двух простых чисел: $n = pq$.

(2) Боб выбирает случайное число x, меньшее n и взаимно простое с n. Он посылает Алисе:

$a = x^2 \bmod n$

(3) Алиса, зная p и q, вычисляет четыре квадратных корня a: x, n-x, y и n-y. Она случайным образом выбирает любой из этих корней и посылает его Бобу.

(4) Если Боб получает y или n-y, он может вычислит наибольший общий делитель x+y и n, которым будет либо p, либо q. Затем, конечно же, $n/p = q$. Если Боб получает x или n-x, он не может ничего вычислить.

У этого протокола может быть слабое место: возможна ситуация, когда Боб может вычислить такое число a, что при известном квадратном корне a он сможет все время раскладывать n на множители.

23.14 Безопасные вычисления с несколькими участниками

Этот протокол взят из [1373]. Алиса знает целое число i, а Боб - целое число j. Алиса и Боб вместе хотят узнать, что правильно - $i \leq j$ или $i > j$, но ни Алиса, ни Боб не хочет раскрыть свое число партнеру. Этот особый случай безопасных вычислений с несколькими участниками (см. раздел 6.2) иногда называют **проблемой миллионера Яо** [162, 7].

В приводимом примере предполагается, что i и j выбираются из диапазона от 1 до 100. У Боба есть открытый и закрытый ключи.

(1) Алиса выбирает большое случайное число x и шифрует его открытым ключом Боба.

$c = E_B(x)$

(2) Алиса вычисляет c-j и посылает результат Бобу.

(3) Боб вычисляет следующие 100 чисел:

$y_u = D_B(c\text{-}i\text{+}u)$, для $1 \leq u \leq 100$

D_B обозначает дешифрирование закрытым ключом Боба.

Он выбирает большое случайное число p. (Размер p должен быть немного меньше x. Боб не знает x, но Алиса может легко сообщить ему размер x.) он вычисляет следующие 100 чисел:

$z_u = (y_u \bmod p)$, для $1 \leq u \leq 100$

Далее он проверяет, что для всех $u \neq v$

$|z_u - z_м| \geq 2$

и что для всех u

$0 < z_u < p\text{-}1$

Если это не так, то Боб выбирает другое простое число и пробует снова.

(4) Боб посылает Алисе эту последовательность чисел, соблюдая их точный порядок:

$z_1, z_2, \ldots z_j, z_{j+1}+1, z_{j+2}+1, \ldots z_{100}+1, p$

(5) Алиса проверяет, конгруэнтен ли i-ый член последовательности $x \bmod p$. Если это так, она делает вывод, что $i \leq j$. В противном случае она решает, что $i > j$.

(6) Алиса сообщает Бобу свои выводы.

Проверка, которую Боб выполняет на этапе (3), должна гарантировать, что ни одно число не появится дважды в последовательности, генерированной на этапе (4). В противном случае, если $z_a = z_b$, Алиса узнает, что $a \leq j < b$.

Недостатком этого протокола является то, что Алиса узнает результаты вычислений раньше Боба. Ничто не помешает ей завершить протокол на этапе (5), отказавшись сообщать Бобу результаты. Она даже может солгать Бобу на этапе (6).

Пример протокола

Пусть они используют RSA. Открытым ключом Боба является 7, а закрытым - 23. $n = 55$. Секретное число Алисы, i, равно 4, секретное число Боба, j - 2. (Предположим, что числа i и j могут принимать только значения 1, 2, 3 и 4.)

(1) Алиса выбирает $x = 39$ и $c = E_B(39) = 19$.

(2) Алиса вычисляет c-i=19-4=15. Она посылает 15 Бобу.

(3) Боб вычисляет следующие четыре числа:

$y_1 = D_B\{15+1) = 26$

$y_2 = D_B\{15+2) = 18$

$y_3 = D_B\{15+3) = 2$

$y_4 = D_B\{15+4) = 39$

Он выбирает $p = 31$ и вычисляет:

$z_1 = (26 \bmod 31) = 26$

$z_2 = (18 \bmod 31) = 18$

$z_3 = (2 \bmod 31) = 2$

$z_4 = (39 \bmod 31) = 8$

Он выполняет все проверки и убеждается, что последовательность правильна.

(4) Боб посылает Алисе эту последовательность чисел, соблюдая их порядок:

26, 18, 2+1, 8+1, 31, т.е., 26, 18, 3, 9, 31

(5) Алиса проверяет, конгруэнтно ли четвертое число $X \bmod p$. Так как $9 \neq 39 \pmod{31}$, то $i > j$.

(6) Алиса сообщает об этом Бобу.

Этот протокол можно использовать для создания намного более сложных протоколов. Группа людей может проводить секретный аукцион по сети. Они логически упорядочивают себя по кругу и, с помощью попарных сравнений, определяют, кто предложил большую цену. Чтобы помешать людям уже изменять сделанные предложения в середине аукциона должен использоваться какой-то протокол вручения битов. Если аукцион проводится по голландской системе, то предложивший наивысшую цену получает предмет за предложенную цену. Если аукцион проводится по английской системе, то он получает предмет за вторую высшую цену. (Это может быть выяснено во время второго круга попарных сравнений.) Аналогичные идеи применимы при заключении сделок, переговорах и арбитраже.

23.15 Вероятностное шифрование

Понятие **вероятностного шифрования** было изобретено Шафи Голдвассером (Shafi Goldwasser) и Сильвией Микали [624]. Хотя их теория позволяет создать самую безопасную из изобретенных криптосистем, ранняя реализации была неэффективной [625]. Но более поздние реализации все изменили.

Идеей вероятностного шифрования является устранение утечки информации в криптографии с открытыми ключами. Так как криптоаналитик всегда может расшифровать случайные сообщения открытым ключом, он может получить некоторую информацию. При условии, что у него есть шифротекст $C = E_K(M)$, и он пытается получить открытый текст M, он может выбрать случайное сообщение M' и зашифровать его: $C' = E_K(M')$. Если $C' = C$, то он угадал правильный открытый текст. В противном случае он делает следующую попытку.

Кроме того, вероятностное шифрование позволяет избежать даже частичной утечки информации об оригинальном сообщении. При использовании криптографии с открытыми ключами криптоаналитик иногда может узнать кое-что о битах: XOR 5-го, 17-го и 39-го битов равно 1, и т.п.. При вероятностном шифровании остается скрытой и такая информация.

Таким способом можно извлечь не много информации, но потенциально возможность криптоаналитика расшифровывать случайные сообщения вашим открытым ключом может создать определенные проблемы. Каждый раз, шифруя сообщение, криптоаналитик может извлечь немного информации. Никто не знает, насколько значительна эта информация.

Вероятностное шифрование пытается устранить эту утечку. Цель этого метода состоит в том, чтобы ни вычисления, проводимые над шифротекстом, ни проверка любых других открытых текстов не смогли дать криптоаналитику никакой информации о соответствующем открытом тексте.

При вероятностном шифровании алгоритм шифромания является вероятностным, а не детерминированным. Другими словами, многие шифротексты при расшифровке дают данный открытый текст, и конкретный шифротекст, используемый в любом конкретном шифровании, выбирается случайным образом.

$C_1 = E_K(M),\ C_2 = E_K(M),\ C_3 = E_K(M), \ldots C_i = E_K(M)$

$M = D_K(C_1) = D_K(C_2) = D_K(C_3) = \ldots = D_K(C_i)$

При вероятностном шифровании криптоаналитику больше не удастся шифровать произвольные открытые тексты в поисках правильного шифротекста. Для иллюстрации пусть у криптоаналитика есть шифротекст $C_i = E_K(M)$. Даже если он праильно угадает M, полученный при шифровании $E_K(M)$ результат будет совершенно другим шифротекстом C: C_j. Сравнивая C_i и C_j, он не может по их совпадению определить правильность своей догадки.

Это поразительно. Даже если у криптоаналитика есть открытый ключ шифрования, открытый текст и шифротекст, он не может без закрытого ключа дешифрирования доказать, что шифротекст является результатом шифрования конкретного открытого текста. Даже выполнив исчерпывающий поиск, он может доказать только, что каждый возможный открытый текст является возможным открытым текстом.

В этой схеме шифротекст всегда будет больше открытого текста. Этого невозможно избежать, это является результатом того, что многие шифротексты расшифровываются в один и тот же открытый текст. В первой схеме вероятностного шифрования [625] шифротекст получался настолько больше открытого текста, что он был бесполезным.

Однако Мануэль Блюм (Manual Blum) и Голдвассер (Goldwasser) получили эффективную реализацию вероятностного шифрования с помощью генератора псевдослучайных битов Blum Blum Shub (BBS), описанного в разделе 17.9 [199].

Генератор BBS основан на теории квадратичных остатков. Существуют два простых числа, p и q, конгруэнтных 3 по модулю 4. Это закрытый ключ. Их произведение, $pq = n$, является открытым ключом. (Запомните свои p и q, безопасность схемы опирается на сложность разложения n на множители.)

Для шифрования сообщения M сначала выбирается случайное x, взаимно простое с n. Затем вычисляется

$x_0 = x^2 \bmod n$

x_0 служит стартовой последовательностью для генератора псевдослучайных битов BBS, а выход генератора используется в качестве потокового шифра. Побитно выполняется XOR M с выходом генератора. Генератор выдает биты b_i (младший значащий бит x_i, где $x_i = x_{i-1}^2 \bmod n$), поэтому

$M=M_1, M_2, M_3, \ldots M_t$

$c = M_1 \oplus b_1, M_2 \oplus b_2, M_3 \oplus b_3, \ldots\ M_t \oplus b_t$

где t - это длина открытого текста

Добавьте последнее вычисленное значение, x_t, к концу сообщения, и дело сделано.

Расшифровать это сообщение можно только одним способом - получить x_0 и с этой стартовой последовательностью запустить генератор BBS, выполняя XOR выхода с шифротекстом. Так как генератор BBS безопасен влево, значение x_t бесполезно для криптоаналитика. Только тот, кому известны p и q, может расшифровать сообщение. Вот как на языке C выглядит алгоритм получения x_0 из x_t:

```
int xO (int p, int q, int n, int t, int xt) {
     int a, b, u, v, w, z;
     /* мы уже знаем, что НОД(p,q) == 1 */
     (void)extended_euclidian(p, q, &a, &b);
```

```
    u = modexp ((p+1)/4, t, p-1);
    v = modexp ((q+1)/4, t, q-1);
    w = modexp (xt%p, u, p);
    z = modexp (xt%p, v, q);
    return (b*q*w + a*p*z) % n;
}
```

При наличии x_0 дешифрирование несложно. Просто задайте стартовую последовательность генератора BBS и выполните XOR результата с шифротекстом.

Эту схему можно сделать еще быстрее, используя все известные безопасные биты x_i, а не только младший значащий бит. С таким улучшением вероятностное шифрование Blum-Goldwasser оказывается быстрее RSA и не допускает утечки информации об открытом тексте. Кроме того, можно доказать, что сложность вскрытия этой схемы равна сложности разложения n на множители.

С другой стороны, эта схема совершенно небезопасна по отношению к вскрытию с выбранным шифротекстом. По младшим значащим битам правильных квадратичных остатков можно вычислить квадратный корень любого квадратичного остатка. Если это удастся, то удастся и разложение на множители. Подробности можно найти в [1570, 1571, 35, 36].

23.16 Квантовая криптография

Квантовая криптография вводит естественную неопределенность квантового мира. С ее помощью можно создавать линии связи, которые невозможно послушать, не внося помех в передачу. Законы физики надежно защищают такой квантовый канал, даже если подслушивающий может предпринимать любые действия, даже если он имеет доступ к неограниченной вычислительной мощности, даже если **P = NP**. Шарль Бенне (Charles Bennett), Жиль Брассар (Gilles Brassard), Клод Крепо (Claude Crepeau) и другие расширили эту идею, описав квантовое распределение ключей, квантовое бросание монеты, квантовое вручение бита, квантовую передачу с забыванием и квантовые вычисления с несколькими участниками. Описание их результатов можно найти в [128, 129, 123, 124, 125, 133, 126, 394, 134, 392, 243, 517, 132, 130, 244, 393, 396]. Лучшим обзором по квантовой криптографии является [131]. Другим хорошим нетехническим обзором может служить [1651]. Полную библиографию по квантовой криптографии можно найти в [237].

Эти идеи так и остались бы предметом обсуждения фанатиков криптографии, но Бенне и Брассар разработали действующую модель [127, 121, 122]. Теперь у нас есть *экспериментальная* квантовая криптография.

Итак устройтесь поудобнее, налейте себе чего-нибудь выпить и расслабьтесь. Я попробую объяснить вам, что это такое.

В соответствии с законами квантовой механики частицы на самом деле не находятся в одном месте, а с определенной вероятностью существуют сразу во многих местах. Однако это так только до тех пор, пока не приходит ученый и не обмеряет частицу, "оказавшуюся" в данном конкретном месте. Но измерить все параметры частицы (например, координаты и скорость) одновременно невозможно. Если измерить одну из этих двух величин, сам акт измерения уничтожает всякую возможность измерить другую величину. Неопределенность является фундаментальным свойством квантового мира, и никуда от этого не денешься.

Эту неопределенность можно использовать для генерации секретного ключа. Путешествуя, фотоны колеблются в определенном направлении, вверх-вниз, влево-вправо, или, что более вероятно, под каким-то углом. Обычный солнечный свет неполяризован, фотоны колеблются во всех возможных направлениях. Когда направление колебаний многих фотонов совпадает, они являются **поляризованными**. Поляризационные фильтры пропускают только те фотоны, которые поляризованы в определенном направлении, а остальные блокируются. Например, горизонтальный поляризационный фильтр пропускает только фотоны с горизонтальной поляризацией. Повернем этот фильтр на 90 градусов, и теперь сквозь него будут проходить только вертикально поляризованные фотоны.

Пусть у вас есть импульс горизонтально поляризованных фотонов. Если они попробуют пройти через горизонтальный фильтр, то у них у всех прекрасно получится. Если медленно поворачивать фильтр на 90 градусов, количество пропускаемых фотонов будет становиться все меньше и меньше, и наконец ни один фотон не пройдет через фильтр. Это противоречит здравому смыслу. Кажется, что даже незначительный поворот фильтра должен остановить все фотоны, так как они горизонтально поляризованы. Но в квантовой механике каждая частица с определенной вероятностью может изменить свою поляризацию и проскочить через фильтр. Если угол отклонения фильтра невелик, эта вероятность высока, а если он равен 90 градусам, то вероятность равна нулю. А если угол поворота фильтра равен 45 градусам, вероятность фотона пройти фильтр равна 50 процентам.

Поляризацию можно измерить в любой **системе координат**: двух направлениях, расходящихся под прямым углом. Примерами систем координат являются прямоугольная - горизонтальное и вертикальное направления - и

диагональная - левая и правая диагонали. Если импульс фотонов поляризован в заданной системе координат, то при измерении в той же системе координат вы узнаете поляризацию. При измерении в неправильной системе координат, вы получите случайный результат. Мы собираемся использовать это свойство для генерации секретного ключа:

(1) Алиса посылает Бобу последовательность фотонных импульсов. Каждый из импульсов случайным образом поляризован в одном из четырех направлений: горизонтальном, вертикальном, лево- и праводиагональном.

Например, Алиса посылает Бобу:

| | / — \ — | — /

(2) У Боба есть детектор поляризации. Он может настроить свой детектор на измерение прямоугольной или диагональной поляризации. Одновременно мерить и ту, и другую у него не получится, ему не позволит квантовая механика. Измерение одной поляризации не даст измерить другую. Итак, он устанавливает свои детекторы произвольным образом:

X + + X X X + X + +

Теперь, если Боб правильно настроит свой детектор, он зарегистрирует правильную поляризацию. Если он настроит детектор на измерение прямоугольной поляризации, и импульс будет поляризован прямоугольно, он узнает, какую поляризацию фотонов выбрала Алиса. Если он настроит детектор на измерение диагональной поляризации, а импульс будет поляризован прямоугольно, то результат измерения будет случайным. Боб не сможет определить разницу. В приведенном примере он может получить следующий результат:

/ | — \ / \ — / — |

(3) Боб сообщает Алисе по незащищенному каналу, какие настройки он использовал.

(4) Алиса сообщает Бобу, какие настройки были правильными. В нашем примере детектор был правильно установлен для импульсов 2, 6, 7 и 9.

(5) Алиса и Боб оставляют только правильно измеренные поляризации. В нашем примере они оставляют:

* | * * * \ — * — *

С помощью заранее приготовленного кода Алиса и Боб преобразуют в биты эти результаты измерений поляризации. Например, горизонтальная и леводиагональная могут означать единицу, а вертикальная и праводиагональная - ноль. В нашем примере они оба получат:

0 0 1 1

Итак, Алиса и Боб получили четыре бита. С помощью этой системы они могут генерировать столько битов, сколько им нужно. В среднем Боб правильно угадывает в 50 процентах случаев, поэтому для генерации n битов Алисе придется послать $2n$ фотонных импульсов. Они могут использовать эти биты как секретный ключ симметричного алгоритма или обеспечить абсолютную безопасность, получив достаточно битов для использования в качестве одноразового блокнота.

Замечательным является то, что Ева не сможет подслушать. Как и Бобу, ей нужно угадать тип измеряемой поляризации, и, как и у Боба, половина ее догадок будет неправильной. Так как неправильные измерения изменяют поляризацию фотонов, то при подслушивании она неминуемо вносит ошибки в передачу. Если это так, Алиса и Боб получат различные битовые последовательности. Итак, Алиса и Боб заканчивают протокол подобными действиями:

(6) Алиса и Боб сравнивают несколько битов своих строк. По наличию расхождений они узнают о подслушивании. Если строки не отличаются, то они отбрасывают использованные для сравнения биты и используют оставшиеся.

Улучшения этого протокола позволяют Алисе и Боб использовать свои биты даже в присутствии Евы [133, 134, 192]. Они могут сравнивать только четность битовых подмножеств. Тогда, если не обнаружено расхождений, им придется отбросить только один бит подмножества. Это обнаруживает подслушивание с вероятностью 50 процентов, но если они сверят таким образом n различных битовых подмножеств, вероятность Евы подслушать и остаться незамеченной будет равна $1/2^n$.

В квантовом мире не бывает пассивного подслушивания. Если Ева попытается раскрыть все биты, она обязательно разрушит канал связи.

Бенне и Брассар построили работающую модель квантового распределения ключей и обменялись безопасными битами на оптической скамье. Последнее, что я слышал, было сообщение о том, что в British Telecom по-

сылали биты по 10-километровому оптоволокну [276, 1245, 1533]. Они считают, что достижимо и расстояние в 50 километров. Это поражает воображение.

Часть IV

Реальный мир

Глава 24
Примеры реализаций

Одно дело разрабатывать протоколы и алгоритмы, и совсем другое дело встраивать их в операционные системы. В теории практика и теория не отличимы, но на практике между ними огромные различия. Часто идеи замечательно выглядят на бумаге, но не работают в реальной жизни. Может быть слишком велики требования к скорости канала, может быть протокол слишком медлителен. Некоторые из вопросов использования криптографии рассматриваются в главе 10, в этой главе обсуждаются примеры того, как криптографические алгоритмы реализуются на практике.

24.1 Протокол управления секретными ключами компании IBM

В конце 70-х годов IBM, используя только симметричную криптографию, разработала законченную систему управления ключами для передачи данных и безопасности файлов в компьютерных сетях [515, 1027]. Не так важны реальные механизмы протокола, как его общая философия: за счет автоматизации генерации, распределения, установки, хранения, изменения и разрушения ключей этот протокол далеко продвинулся, обеспечивая безопасность лежащих в его основе криптографических алгоритмов.

Этот протокол обеспечивает три вещи: безопасную связь между сервером и различными терминалами, безопасное хранение файлов на сервере и безопасную связь между серверами. Протокол не обеспечивает настоящего прямого соединения терминал-терминал, хотя его модификация может реализовать такую возможность.

Каждый сервер сети подключен к криптографической аппаратуре, которая выполняет все шифрование и дешифрирование. У каждого сервера есть **Главный ключ** (Master Key), KM_0, и два варианта, KM_1 и KM_2, которые являются упрощенными вариантами KM_0. Эти ключи используются для шифрования других ключей и для генерации новых ключей. У каждого терминала есть **Главный ключ терминала** (Master Terminal Key), *KMT*, который используется для обмена ключами с другими терминалами.

KMT хранятся на серверах, зашифрованные ключом KM_1. Все остальные ключи, например, используемые для шифрования файлов ключей (они называются *KNF*), хранятся в зашифрованной форме, закрытые ключом KM_2. Главный ключ KM_0 хранится в энергонезависимом модуле безопасности. Сегодня это может быть либо ключ в ПЗУ, либо магнитная карточка, или ключ может вводиться пользователем с клавиатуры (возможно как текстовая строка, преобразуемая в ключ). KM_1 и KM_2 не хранятся где-нибудь в системе, а, когда понадобится, вычисляются по KM_0. Сеансовые ключи для связи между серверами генерируются на сервере с помощью псевдослучайного процесса. Аналогичным образом генерируются ключи для шифрования хранимых файлов (*KNF*).

Сердцем протокола служит устойчивый к вскрытию модуль, называемый **криптографической аппаратурой** (cryptographic facility). И на сервере, и на терминале все шифрование и дешифрирование происходит именно в этом модуле. В этом модуле хранятся самые важные ключи, используемые для генерации действительных ключей шифрования. После того, как эти ключи записаны, считать их становится невозможным. Кроме того, они помечены для конкретного использования: ключ, предназначенный для решения одной задачи, не может случайно быть использован для решения другой. Эта концепция **векторов управления ключами** возможно является самым значительным достижением этой системы. Дональд Дэвис (Donald Davies) Вильям Прайс (William Price) подробно рассматривают этот протокол управления ключами в [435].

Модификация

Модификацию этой схемы главного и сеансовых ключей можно найти в [1478]. Она построена на базе сетевых узлов с аппаратурой проверки подлинности ключей, которая обслуживает локальные терминалы. Эта модификация была разработана, чтобы:

— Обезопасить дуплексный канал между двумя пользовательскими терминалами.

— Обезопасить связь с помощью шифрованной почты.

— Обеспечить защиту личных файлов.

— Обеспечить возможность цифровой подписи.

Для связи и передачи файлов между пользователями в этой схеме используются ключи, генерированные в аппаратуре проверки подлинности ключей, отправляемые пользователям после шифрования с помощью главного ключа. Информация о личности пользователя встраивается в ключ, предоставляя доказательство того, что сеансовый ключ используется конкретной парой пользователей. Возможность **проверки подлинности ключей** является главной в этой системе. Хотя в системе не используется криптография с открытыми ключами, она поддерживает возможность, похожую на цифровую подпись: ключ может быть прислан только из конкретного источника и прочитан только в конкретном месте назначения.

24.2 MITRENET

Одной из самых ранних реализаций криптографии с открытыми ключами была экспериментальная система MEMO (MITRE Encrypted Mail Office, Шифрованное почтовое отделение). MITRE - это была команда умных парней, работающая по заказу Министерства обороны. MEMO служила системой безопасной электронной почты для пользователей сети MITRENET и использовала криптографию с открытыми ключами для обмена ключами и DES для шифрования файлов.

В системе MEMO все открытые ключи хранятся в Центре распределения открытых ключей (Public Key Distribution Center), который является отдельным узлом сети. Ключи хранятся в стираемом перепрограммируемом ПЗУ, чтобы не дать изменить их. Закрытые ключи генерируются пользователями системы.

Чтобы пользователь мог отправлять безопасные сообщения, система сначала устанавливает безопасное соединение с Центром распределения открытых ключей. Пользователь запрашивает в Центре файл всех открытых ключей. Если пользователь проходит идентификацию с использованием его закрытого ключа, Центр пересылает запрошенный список на рабочую станцию пользователя. Для обеспечения целостности список шифруется с помощью DES.

Для шифрования сообщений используется DES. Для шифрования файлов система генерирует случайный ключ DES, пользователь шифрует файл ключом DES, а ключ DES - открытым ключом получателя. Зашифрованный файл и ключ отправляются получателю.

MEMO не предусматривает мер предосторожности против потерь ключей. Существуют некоторые средства проверки целостности сообщений с использованием контрольных сумм. В систему не встроены средства проверки подлинности.

Прежде, чем система была реализована, была доказана небезопасность конкретной реализации системы открытых ключей в MEMO - обмена ключами по схеме Diffie-Hellman над $GF(2^{127})$ (см. раздел 11.6), хотя нетрудно изменить систему, чтобы можно было использовать большие числа. MEMO была изобретена главным образом для экспериментальных целей и никогда не использовалась в реальной системе MITRENET.

24.3 ISDN

Bell-Northern Research разработала прототип безопасного телефонного терминала ISDN (Integrated Services Digital Network, Цифровая сеть с интегрированием услуг) [499, 1192, 493, 500]. Как телефонный аппарат, терминал остался на уровне прототипа. В результате появился Уровень безопасности пакетов данных (Packet Data Security Overlay). Терминал использует схему обмена ключами Diffie-Hellman, цифровые подписи RSA и DES для шифрования данных. Он может передавать и принимать речь и данные со скоростью 64 Кбит/с.

Ключи

В телефон встроена пара "открытый ключ/закрытый ключ" для длительного использования. Закрытый ключ хранится в устойчивом от вскрытия модуле телефона. Открытый ключ служит для идентификации телефона. Эти ключи являются частью самого телефонного аппарата и не могут быть изменены.

Кроме того, в телефоне хранятся еще два открытых ключа. Одним из них является открытый ключ владельца аппарата. Этот ключ используется для проверки подлинности команд владельца, он может быть изменен по команде, подписанной владельцем. Так пользователь может передать кому-то другому право владения аппаратом.

В телефоне также хранится открытый ключ сети. Он используется для проверки подлинности команд аппаратуры управления сетью и проверки подлинности вызовов от других пользователей сети. Этот ключ также можно изменить командой, подписанной владельцем. Это позволяет владельцу менять сеть, к которой подключен его аппарат.

Эти ключи рассматриваются как ключи длительного пользования - они меняются редко, если вообще меняются. В телефоне также хранится пара "открытый ключ/закрытый ключ" для краткосрочного использования. Они встроены в сертификат, подписанный центром управления ключами. Два телефона обмениваются сертификатами при установлении соединения. Подлинность этих сертификатов удостоверяется открытым ключом сети.

Обмен сертификатами и их проверка выполняются только при установлении безопасного соединения между аппаратами. Для установления безопасного соединения между людьми протокол содержит дополнительный компонент. В аппаратном **ключе зажигания**, который вставляется в телефон владельцем, хранится закрытый ключ владельца, зашифрованный секретным паролем, известным только владельцу (его не знает ни телефонный аппарат, ни центр управления сетью, ни еще кто-нибудь). Ключ зажигания также содержит сертификат, подписанный центром управления сетью, в который включены открытый ключ владельца и некоторая идентификационная информация (имя, компания, специальность, степень допуска, любимые сорта пиццы, сексуальная ориентация и прочее). Все это также зашифровано. Для дешифрирования этой информации и ввода ее в телефон

пользователь вводит свой секретный пароль с клавиатуры аппарата. Телефонный аппарат использует эту информацию для соединения, но она удаляется после того, как пользователь извлечет свой ключ зажигания.

В телефоне также хранится набор сертификатов, выданных центром управления сетью. Эти сертификаты удостоверяют право конкретных пользователей пользоваться конкретными телефонными аппаратами.

Вызов

Вызов Боба Алисой происходит следующим образом.

(1) Алиса вставляет в телефон свой ключ зажигания и вводит свой пароль.

(2) Телефон опрашивает ключ зажигания, чтобы определить личность Алисы и выдать ей сигнал "линия свободна".

(3) Телефон проверяет свой набор сертификатов, проверяя, что Алиса имеет право использовать этот аппарат.

(4) Алиса набирает номер, телефон определяет адресата звонка.

(5) Два телефона используют протокол обмена ключами на базе криптографии с открытыми ключами, чтобы генерировать уникальный и случайный сеансовый ключ. Все последующие этапы протокола шифруются с помощью этого ключа.

(6) Телефон Алисы передает свой сертификат и идентификатор пользователя.

(7) Телефон Боба проверяет подписи сертификата и идентификатора пользователя, используя открытый ключ сети.

(8) Телефон Боба инициирует последовательность запросов/ответов. Для этого необходимо в реальном времени (не позднее заданной задержки) отправлять подписанные ответы на запросы. (Это помешает злоумышленнику использовать сертификаты, скопированные из предыдущего обмена.) Один ответ должен быть подписан закрытым ключом телефона Алисы, а другой - закрытым ключом Алисы.

(9) Если Боба нет у телефона, то его телефон звонит.

(10) Если Боб дома, он вставляет в телефон свой ключ зажигания. Его телефон опрашивает ключ зажигания и проверяет сертификат Боба, как на этапах (2) и (3).

(11) Боб передает свой сертификат и идентификатор пользователя.

(12) Телефон Алисы проверяет подписи Боба, как на этапе (7) и инициирует последовательность запросов/ответов, как на этапе (8).

(13) Оба телефона выводят на свои экраны личность и номер телефона другого пользователя.

(14) Начинается безопасный разговор.

(15) Когда одна из сторон вешает трубку, удаляются сеансовый ключ, а также сертификаты, которые телефон Боба получил от телефона Алисы, и сертификаты, которые телефон Алисы получил от телефона Боба.

Каждый ключ DES уникален для каждого звонка. Он существует только внутри двух телефонных аппаратов и только в течение разговора, а после его окончания немедленно уничтожается. Если злоумышленник добудет один или оба участвовавших в разговоре аппарата, он не сможет расшифровать ни один предшествующий разговор, в котором участвовали эти два аппарата.

24.4 STU-III

STU обозначает "Secure Telephone Unit" (Безопасный телефонный модуль), разработанный в NSA безопасный телефон. По размерам и форме этот модуль почти такой же, как и обычный телефон, и может быть использован также, как и обычный телефон. Аппараты устойчивы к взлому, без ключа они работают как несекретные. Они также включают порт передачи данных и помимо передачи речи могут быть использованы для безопасной передачи данных по модемному каналу [1133].

Уитфилд Диффи описал STU-III в [494]:

> Чтобы позвонить, используя STU-III, звонящий сначала обычным образом звонит на другой STU-III, затем вставляет похожее на ключ устройство, содержащее криптографическую переменную, и нажимает кнопку "секретные переговоры" ("go secure"). Спустя примерно 15 секунд задержки, нужной для криптографической настройки, каждый телефон выводит на экран информацию о личности и допуске другой стороны, и разговор может начинаться.
>
> Беспрецедентным шагом был объявление Уолтера Дили (Walter Deeley), заместителя директора NSA по безопасности коммуникаций, о STU-III или будущей системе безопасной голосовой связи в эксклюзивном интервью, данном *The New York Times* [282]. Главной целью новой системы было предоставить Министерству обороны США и его подрядчикам средства безопасной передачи речи и безопасной низкоскоростной передачи данных. В интервью не было много сказано о работе системы, но постепенно информация начала появляться. В новой системе используются открытые ключи.

О новом подходе к распределению ключей было рассказано в [68], в одной статье говорилось о телефонах, "перепрограммируемых раз в год по безопасному телефонному каналу", что весьма вероятно предполагает использование протокола проверки сертификатов, аналогичного описанному [в разделе 24.3], который минимизирует для телефонов необходимость общаться с центром управления ключами. Последние известия были более информативными, в них рассказывалось о системе управления ключами, названной FIREFLY, которая [1341] "разработана на базе технологии открытых ключей и используется для распределения ключей шифрования попарного трафика". И это описание, и свидетельские показания, данные Конгрессу США Ли Ньювиртом (Lee Neuwirth) из Cylink [1164] предполагают использование комбинации обмена ключами и сертификатами, аналогичного используемому в безопасных телефонах ISDN. Весьма вероятно, что FIREFLY также основана на возведении в степень.

STU-III производятся AT&T и GE. За 1994 год было выпущено 300000-400000 штук. Новая версия, Secure Terminal Equipment (STE, Безопасный терминал), будет работать по линиям ISDN.

24.5 KERBEROS

Kerberos представляет собой разработанный для сетей TCP/IP протокол проверки подлинности с доверенной третьей стороной. Служба Kerberos, работающая в сети, действует как доверенный посредник, обеспечивая безопасную сетевую проверку подлинности, дающую пользователю возможность работать на нескольких машинах сети. Kerberos на симметричной криптографии (реализован DES, но вместо него можно использовать и другие алгоритмы). При общении с каждым объектом сети Kerberos использует отличный общий секретный ключ, и знание этого секретного ключа равносильно идентификации объекта.

Kerberos был первоначально разработан в МТИ для проекта Афина. Модель Kerberos основана на протоколе Needham-Schroeder с доверенной третьей стороной (см. раздел 3.3) [1159]. Оригинальная версия Kerberos, Версия 4, определена в [1094, 1499]. (Версии с 1 по 3 были внутренними рабочими версиями.) Версия 5, модификация Версии 4, определена в [876, 877, 878]. Лучшим обзором по Kerberos является [1163]. Другие обзорные статьи - [1384, 1493], использование Kerberos в реальном мире хорошо описано в [781, 782].

Модель Kerberos

Базовый протокол Kerberos был схематично описан в разделе 3.3. В модели Kerberos существуют расположенные в сети объекты - клиенты и серверы. Клиентами могут быть пользователи, но могут и независимые программы, выполняющие следующие действия: загрузку файлов, передачу сообщений, доступ к базам данных, доступ к принерам, получение административных привилегий, и т.п.

Kerberos хранит базу данных клиентов и их секретных ключей. Для пользователей-людей секретный ключ является зашифрованным паролем. Сетевые службы, требующие проверки подлинности, и клиенты, которые хотят использовать эти службы, регистрируют в Kerberos свои секретные ключи.

Так как Kerberos знает все секретные ключи, он может создавать сообщения, убеждающие один объект в подлинности другого. Kerberos также создает сеансовые ключи, которые выдаются клиенту и серверу (или двум клиентам) и никому больше. Сеансовый ключ используется для шифрования сообщений, которыми обменив аются две стороны, и уничтожается после окончания сеанса.

Для шифрования Kerberos использует DES. Kerberos версии 4 обеспечивал нестандартный, слабый режим проверки подлинности - он не мог определить определенный изменения шифротекста (см. раздел 9.10). Kerberos версии 5 использует режим CBC.

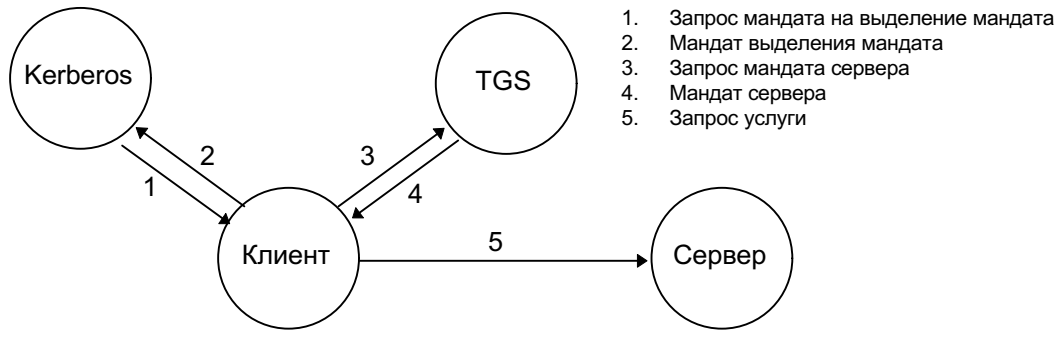

Рис. 24-1. Этапы проверки подлинности Kerberos

Как работает Kerberos

В этом разделе рассматривается Kerberos версии 5. Ниже я обрисую различия между версиями 4 и 5. Протокол Kerberos прост (см. 23rd). Клиент запрашивает у Kerberos мандат на обращение к **Службе выделения мандатов** (Ticket-Granting Service, TGS). Этот мандат, зашифрованный секретным ключом клиента, посылается

клиенту. Для использования конкретного сервера клиент запрашивает у TGS мандат на обращение к серверу. Если все в порядке, TGS посылает мандат клиенту. Затем клиент предъявляет серверу этот мандат вместе с удостоверением. И снова, если атрибуты клиента правильны, сервер предоставляет клиенту доступ к услуге.

Табл. 24-1.
Таблица сокращений Kerberos

c	= клиент
s	= сервер
a	= сетевой адрес клиента
v	= начало и окончание времени действия мандата
t	= метка времени
K_x	= секретный ключ x
$K_{x,y}$	= сеансовый ключ для x и y
$(m)K_x$	= m, шифрованное секретным ключом x
$T_{x,y}$	= мандат x на использование y
$A_{x,y}$	= удостоверение x для y

Атрибуты

Kerberos использует два типа атрибутов: **мандаты** и **удостоверения**. (В дальнейшем в этом разделе будет использоваться нотация, используемая в документах Kerberos - см. 23-й.) Мандат используется для безопасной передачи серверу личности клиента, которому выдан этот мандат. В нем также содержится информация, которую сервер может использовать для проверки того, что клиент, использующий мандат, - это именно тот клиент, которому этот мандат был выдан. Удостоверение - это дополнительный атрибут, предъявляемый вместе с мандатом. Мандат Kerberos имеет следующую форму:

$T_{c,s} = s, \{c, a, v, K_{c,s}\}K_s.$

Мандат хорош для одного сервера и одного клиента. Он содержит имя клиента, его сетевой адрес, имя сервера, метку времени и сеансовый ключ. Эта информация шифруется секретным ключом сервера. Если клиент получил мандат, он может использовать его для доступа к серверу много раз - пока не истечет срок действия мандата. Не может расшифровать мандат (он не знает секретного ключа сервера), но он может предъявить его серверу в зашифрованной форме. Прочитать или изменить мандат при передаче его по сети невозможно. Удостоверение Kerberos имеет следующую форму:

$A_{c,s} = \{c, t, ключ\}K_{c,s}$

Клиент создает его каждый раз, когда ему нужно воспользоваться услугами сервера. Удостоверение содержит имя клиента, метку времени и необязательный дополнительный сеансовый ключ, все эти данные шифруются сеансовым ключом, общим для клиента и сервера. В отличие от мандата удостоверение используется только один раз. Однако это не проблема, так как клиент может генерировать удостоверения по мере надобности (ему известен общий секретные ключ).

Использование удостоверения преследует две цели. Во первых, оно содержит некоторый открытый текст, зашифрованный сеансовым ключом. Это доказывает, что клиенту известен ключ. Что не менее важно, зашифрованный открытый текст включает метку времени. Злоумышленник, которому удалось записать и мандат, и удостоверение, не сможет использовать их спустя два дня.

Сообщения Kerberos версии 5

В Kerberos версии 5 используется пять сообщений (см. 23-й):

1. Клиент-Kerberos: c,tgs
2. Kerberos-клиент: $\{K_{c,tgs}\}K_c$, $\{T_{c,tgs}\}K_{tgs}$
3. Клиент-TGS: $\{A_{c,s}\}K_{c,tgs}\{T_{c,tgs}\}\ K_{tgs,s}$
4. TGS-клиент: $\{K_{c,s}\}K_{c,tgs}\{T_{c,s}\}K_s$
5. Клиент-сервер: $\{A_{c,s}\}K_{c,s}\ \{T_{c,s}\}K_s$

Теперь рассмотрим использование этих сообщений подробно.

Получение первоначального мандата

У клиента есть часть информации, доказывающей его личность - его пароль. Понятно, что не хочется заставлять клиента передавать пароль по сети. Протокол Kerberos минимизирует вероятность компрометации пароля, но при этом не позволяет пользователю правильно идентифицировать себя, если он не знает пароля.

Клиент посылает сообщение, содержащее его имя и имя его сервера TGS на сервер проверки подлинности Kerberos. (может быть несколько серверов TGS.) На практике пользователь, скорее всего, просто вводит свое имя и программа входа в систему посылает запрос.

Сервер проверки подлинности Kerberos ищет данные о клиенте в своей базе данных. Если информация о клиенте есть в базе данных, Kerberos генерирует сеансовый ключ, который будет использоваться для обмена данными между клиентом и TGS. Он называется **Мандатом на выделение мандата** (Ticket Granting Ticket, TGT). Kerberos шифрует этот сеансовый ключ секретным ключом клиента. Затем он создает для клиента TGT, доказывающий подлинность клиента TGS, и шифрует его секретным ключом TGS. Сервер проверки подлинности посылает эти два зашифрованных сообщения клиенту.

Теперь клиент расшифровывает первое сообщение и получает сеансовый ключ. Секретный ключ является однонаправленной хэш-функцией клиентского пароля, поэтому у законного пользователя не будет никаких проблем. Самозванец не знает правильного пароля и, следовательно, не может расшифровать ответ сервера проверки подлинности. Доступ запрещается, и самозванный клиент не может получить мандат или сеансовый ключ.

Клиент сохраняет TGT и сеансовый ключ, стирая пароль и хэш-значение. Эта информация уничтожается для уменьшения вероятности компрометации. Если враг попытается скопировать память клиента, он получит только TGT и сеансовый ключ. Эти данные важны, но только на время жизни TGT. Когда срок действия TGT истечет, эти сведения станут бессмысленными. Теперь в течение времени жизни TGT клиент может доказывать TGS свою подлинность.

Получение серверных мандатов

Клиенту требуется получить отдельный мандат для каждой нужной ему услуги. TGS выделяет мандаты для отдельных серверов.

Когда клиенту нужен мандат, которого у него пока нет, он посылает запрос к TGS. (На практике программа, скорее всего, делает это автоматически и незаметно для пользователя.)

TGS, получив запрос, расшифровывает TGT своим секретным ключом. Затем TGS использует включенный в TGT сеансовый ключ, чтобы расшифровать удостоверение. Наконец TGS сравнивает информацию удостоверения с информацией мандата, сетевой адрес клиента с адресом отправителя запроса и метку времени с текущим временем. Если все совпадает, TGS разрешает выполнение запроса.

Проверка меток времени предполагает, что часы всех компьютеров синхронизированы, по крайней мере с точностью до нескольких минут. Если время, указанное в запросе, отстоит от текущего момента слишком далеко в прошлое или в будущее, TGS считает запрос попыткой повторения предыдущего запроса. TGS должна также отслеживать правильность сроков действия удостоверений, так как услуги сервера могут запрашиваться несколько раз последовательно с одним мандатом, но разными удостоверениями. Другой запрос с тем же мандатом и уже использованной меткой времени удостоверения будет отвергнут.

В ответ на правильный запрос TGS возвращает правильный мандат, который клиент может предъявить серверу. TGS также создает новый сеансовый ключ для клиента и сервера, зашифрованный сеансовым ключом, общим для клиента и TGS. Оба этих сообщения отправляются клиенту. Клиент расшифровывает сообщение и извлекает сеансовый ключ.

Запрос услуги

Теперь клиент может доказать свою подлинность серверу. Он создает сообщение, очень похожее на то, которое посылалось TGS (и это понятно, так как TGS - тоже услуга).

Клиент создает удостоверение, состоящее из его имени, сетевого адреса и метки времени, зашифрованное сеансовым ключом, который был генерирован TGS для сеанса клиента и сервера. Запрос состоит из мандата, полученного от Kerberos (уже зашифрованного секретным ключом сервера) и зашифрованного идентификатора.

Сервер расшифровывает и проверяет мандат и удостоверение, как уже обсуждалось, а также проверяет адрес клиента и метку времени. Если все в порядке, то сервер уверен, что, согласно Kerberos, клиент - именно тот, за кого он себя выдает.

Если приложение требует взаимной проверки подлинности, сервер посылает клиенту сообщение, состоящее из метки времени, зашифрованной сеансовым ключом. Это доказывает, что серверу известен правильный секретный ключ, и он может расшифровать мандат и удостоверение.

При необходимости клиент и сервер могут шифровать дальнейшие сообщения общим ключом. Так как этот ключ известен только им, они оба могут быть уверены, что последнее сообщение, зашифрованное этим ключом, отправлено другой стороной.

Kerberos версии 4

В предыдущих разделах рассматривался Kerberos версии 5. Версия 4 немного отличается сообщениями и конструкцией мандатов и удостоверений. В Kerberos версии 4 используются следующие пять сообщений:

1. Клиент-Kerberos: *c*,*tgs*
2. Kerberos-клиент: $\{K_{c,tgs}\{T_{c,tgs}\}K_{tgs}\}K_c$,
3. Клиент-TGS: $\{A_{c,s}\}K_{c,tgs}\{T_{c,tgs}\}\ K_{tgs,s}$
4. TGS-клиент: $\{K_{c,s}\{T_{c,s}\}K_s\}K_{c,tgs}$
5. Клиент-сервер: $\{A_{c,s}\}K_{c,s}\ \{T_{c,s}\}K_s$

$$T_{c,s} = \{s, c, a, v, l, K_{c,s}\}K_s$$

$$A_{c,s} = \{c, a, t\}\ K_{c,s}$$

Сообщения 1,3 и 5 не изменились. Двойное шифрование мандата на этапах 2 и 4 в версии 5 было устранено. Мандаты версии 5 дополнительно включают возможность использовать несколько адресов, а поле "время жи зни", *l*, заменено временем начала и окончания. В удостоверение версии пять добавлена возможность включения дополнительного ключа.

Безопасность Kerberos

Стив Белловин (Steve Bellovin) и Майкл Мерритт (Michael Merritt) проанализировали некоторые потенциальные уязвимые места Kerberos [108]. Хотя эта работа была написана про протоколы версии 4, многие ее замечания применимы и к версии 5.

Возможно кэширование и повторное использование старых удостоверений. Хотя метки должны предотвратить иакую возможность, удостоверения могут использоваться повторно в течение времени жизни мандата. Предполагается, что серверы хранят все правильные мандаты, чтобы обнаружить повторы, но это не всегда возможно. Кроме того, время жизни бывает достаточно большим, часто до восьми часов.

Использование удостоверений основаны на том, что все часы сети более или менее синхронизированы. Если время компьютера будет установлено неправильно, то старое удостоверение может быть использовано без проблем. Большинство сетевых протоколов поддержки единого времени небезопасны, поэтому такая возможность представляет собой серьезную проблему.

Kerberos также чувствителен к вскрытиям с угадыванием пароля. Злоумышленник может записать мандаты и затем попытаться их расшифровать. Не забудем, что средний пользователь редко выбирает хороший пароль. Если Мэллори добудет достаточно мандатов, у него появятся неплохие шансы раскрыть пароль.

Возможно самым опасным является вскрытие, использующее специальное программное обеспечение. Протоколы Kerberos подразумевают, что программному обеспечению можно доверять. Нет способа помешать Мэллори исподтишка заменить все клиентское программное обеспечение Kerberos такой версией, которая помимо выполнения протоколов Kerberos записывает пароли. Это является проблемой для любого криптографического программного пакета, работающего на небезопасном компьютере, но широко распространенное использование Kerberos в подобных средах делает его особенно привлекательной мишенью.

Ведутся работы над улучшением Kerberos, включая модернизацию управления ключами с помощью криптографии с открытыми ключами и интерфейса интеллектуальных карточек.

Лицензии

Kerberos не является общедоступным, но код МТИ доступен свободно. Действительная реализация в работающих системах UNIX - это совсем другая история. Ряд компаний продает версии Kerberos, но можно получить хорошую версию бесплатно от Cygnus Support, 814 University Ave., Pale Alto, CA, 94301; (415) 32,2.-3811; fax: (415) 32.2.-3270.

24.6 KRYPTOKNIGHT

KryptoKnight (КриптоРыцарь) является системой проверки подлинности и распределения ключей, разработанной в IBM. Это протокол с секретным ключом, использующий либо DES в режиме CBC (см. раздел 9.3) или модифицированную версию MD5 (см. раздел 18.5). KryptoKnight поддерживает четыре сервиса безопасности:

— Проверка подлинности пользователя (называемая единственной подписью - single sign-on)

— Двусторонняя проверка подлинности

— Распределение ключей

— Проверка подлинности содержания и происхождения данных

С точки зрения пользователя, KryptoKnight похож на Kerberos. Вот некоторые отличия:

— Для проверки подлинности и шифрования мандатов KryptoKnight использует хэш-функцию.

— KryptoKnight не использует синхронизированных часов, используются только текущие запросы (см. раздел 3.3).

— Если Алисе нужно связаться с Бобом, одна из опций KryptoKnight позволяет Алисе послать сообщение Бобу, а затем позволяет Бобу начать протокол обмена ключами.

KryptoKnight, как и Kerberos, использует мандаты и удостоверения. Он содержит и TGS, но в KryptoKnight называются серверами проверки подлинности. Разработчики KryptoKnight потратили немало усилий, минимизируя количество сообщений, их размер и объем шифрования. О KryptoKnight читайте в [1110, 173, 174, 175].

24.7 SESAME

SESAME означает Secure European System for Applications in a Multivendor Environment - Безопасная европейская система для приложений в неоднородных средах. Это проект Европейского сообщества, на 50 процентов финансируемый RACE (см. раздел 25.7), главной целью которой является разработка технологии для проверки подлинности пользователя при распределенном контроле доступа. Эту систему можно рассматривать как европейский вариант Kerberos. Проект состоит из двух частей: на первой стадии разрабатывается базовая архитектура, а вторая стадия представляет собой ряд коммерческих проектов. Следующие три компании принимают наибольшее участие в разработке системы - ICL в Великобритании, Siemens в Германии и Bull во Франции.

SESAME представляет собой систему проверки подлинности и обмена ключами [361, 1248, 797, 1043]. Она использует протокол Needham-Schroeder, применяя криптографию с открытыми ключами для связи между различными безопасными доменами. В системе есть ряд серьезных изъянов. Вместо использования настоящего алгоритма шифрования в этой системе применяется XOR с 64-битовым ключом. Что еще хуже, в SESAME используется XOR в режиме CBC, который оставляет незашифрованным половину открытого текста. В защиту разработчиков надо сказать, что они собирались использовать DES, но французское правительство выразило неудовольствие по этому поводу. Они утвердили код с DES, но затем убрали его. Эта система меня не впечатлила.

Отождествление в SESAME является функцией первого блока, а не всего сообщения. В результате этого тождественность сообщений будет проверена по словам "Dear Sir", а не по всему содержанию сообщений. Генерация ключей состоит из двух вызовов функции rand операционной системы UNIX, которая совсем не случайна. В качестве однонаправленных хэш-функций SESAME использует crc32 и MD5. И конечно, SESAME подобно Kerberos чувствительна к угадыванию паролей.

24.8 Общая криптографическая архитектура IBM

Общая криптографическая архитектура (Common Cryptographic Architecture, CCA) была разработана компанией IBM, чтобы обеспечить криптографические примитивы для конфиденциальности, целостности, управления ключами и обработки персонального идентификационного кода (PIN) [751, 784, 1025, 1026, 940, 752]. Управление ключами происходит с помощью векторов управления (control vector, CV) (см. раздел 8.5). Каждому ключу соответствует CV, с которым ключ объединен операцией XOR. Ключ и CV разделяются только в безопасном аппаратном модуле. CV представляет собой структуру данных, обеспечивающую интуитивное понимание привилегий, связанных с конкретным ключом.

Отдельные биты CV обладают конкретным смыслом при использовании каждого ключа, применяемого в CGA. CV передаются вместе с зашифрованным ключом в структурах данных, называемых ключевыми маркерами (key token). Внутренние ключевые маркеры используются локально и содержат ключи, шифрованные локальным главным ключом (master key, MK). Внешние ключевые маркеры используются для шифрованными ключами между системами. Ключи во внешних ключевых маркерах зашифрованы ключами шифрования ключей (key-encrypting key, KEK). Управление KEK осуществляется с помощью внутренних ключевых маркеров. Ключи разделяются на группы в соответствии с их использованием.

Длина ключа также задается при помощи битов CV. Ключи одинарной длины - 56-битовые - используются для таких функций, как обеспечение конфиденциальности и сообщений. Ключи двойной длины - 112-битовые - применяются для управления ключами, функций PIN и других специальных целей. Ключи могут быть DOUBLE-ONLY (только двойные), правые и левые половины которых должны быть различны, DOUBLE (двойные) половины которых могут случайно совпасть, SINGLE-REPLICATED (одинарные-повторенные), в которых правые и левые половины равны, или SINGLE (одинарные), содержащие только 56 битов. CGA определяет аппаратную реализацию определенных типов ключей, используемых для некоторых операций.

CV проверяется в безопасном аппаратном модуле: для каждой функции CGA вектор должен соответствовать определенным правилам. Если CV успешно проходит проверку, то при помощи XOR KEK или MK с CV получается вариант KEK или MK, и извлеченный ключ для дешифрирования открытого текста сообщения использ уется только при выполнении функции CGA. При генерации новых ключей CV задает способ использования созданного ключа. Комбинации типов ключей, которые могут быть использованы для вскрытия системы, не со здаются в CGA-совместимых системах и не импортир уются в них.

Для распределения ключей CGA применяет комбинацию криптографии с открытыми ключами и криптогр афии с секретными ключами. KDC шифрует сеансовый ключ для пользователя секретным главным ключом, ра зделяемым с этим пользователем. Распределение главных ключей происходит с помощью криптографии с о ткрытыми ключами.

Разработчики системы выбрали такой гибридный подход по двум причинам. Первой из них является эффе ктивность. Криптография с открытыми ключами требует больших вычислительных ресурсов, если сеансовые ключи распределяются с помощью криптографии с открытыми ключами, система может повиснуть. Второй причиной является обратная совместимость, система может быть с минимальными последствиями установлена поверх существующих схем с секретными ключами.

CGA-системы проектировались так, чтобы они могли взаимодействовать с различными другими системами. При контакте с несовместимыми системами функция трансляции вектора управления (Control Vector Translate, CVXLT) позволяет системам обмениваться ключами. Инициализация функции CVXLT требует контроля с обе их сторон. Каждая из них должна независимо установить нужные таблицы трансляции. Такой двойной контроль обеспечивает высокую степень надежности, касающейся целостности и происхождения ключей, импортируемых в систему.

Тип ключа DATA поддерживается для совместимости с другими системами. Ключ типа DATA хранится вместе с соответствующим CV, указывающим, что это ключ типа DATA. Ключи типа DATA могут использоваться достаточно широко, и поэтому к ним нужно относиться с подозрением и использовать их с осторожн остью. Ключи типа DATA нельзя использовать ни для каких функций управления ключ ами.

Аппаратура закрытия коммерческих данных (Commercial Data Masking Facility, CDMF) представляет собой экспортируемую версию CGA. Ее особенностью является уменьшение эффективной длины ключей DES до разрешенных к экспорту 40 битов (см. раздел 15.5) [785].

24.9 Схема проверки подлинности ISO

Для использования в схеме проверки подлинности ISO, также известной как протоколы X.509, рекомендуется криптография с открытыми ключами [304]. Эта схема обеспечивает проверку подлинности по сети. Хотя конкретный алгоритм не определен ни для обеспечения безопасности, ни для проверки подлинности, специф икация рекомендует использовать RSA. Однако возможно использование нескольких алгоритмов и хэш-функций. Первоначальный вариант X.509 был выпущен в 1988 г. После открытого изучения и комментирования он был пересмотрен в 1993 году, чтобы исправить некоторые изъяны в безопасности [1100, 750].

Версия
Последовательный номер
Идентификатор алгоритма - Алгоритм - Параметры
Выдавшая организация
Время действия - начало действия - гонец действия
Субъект
Открытый ключ субъекта - Алгоритм - Параметры - Открытый ключ

Подпись

Рис. 24-2. Сертификат X.509.

Сертификаты

Наиболее важной частью X.509 используемая им структура сертификатов открытых ключей. Имена всех пользователей различны. Доверенный Орган сертификации (Certification Authority, CA) присваивает каждому пользователю уникальное имя и выдает подписанный сертификат, содержащий имя и открытый ключ пользов ателя. Структура сертификата X.509 показана на 22-й [304].

Поле версии определяет формат сертификата. Последовательный номер уникален для конкретного CA. Следующее поле определяет алгоритм, использованный для подписи сертификата, вместе со всеми необходимыми параметрами. Выдавшей организацией является CA. Срок действия представляет собой пару дат, сертификат действителен в промежутке между этими двумя датами. Субъект - это имя пользователя. Информация об открытом ключе включает название алгоритма, все необходимые параметры и открытый ключ. Последним полем является подпись CA.

Если Алиса хочет связаться с Бобом, она сначала извлекает из базы данных его сертификат и проверяет его достоверность. Если у них общий CA, то все просто. Алиса проверяет подпись CA на сертификате Боба.

Если они пользуются различными CA, то все гораздо сложнее. Представьте себе древовидную структуру, в которой одни CA сертифицируют другие CA и пользователей. На самом верху находится главный CA. У каждого CA есть сертификаты, подписанные вышестоящим CA и нижестоящим CA. При проверке сертификата Боба Алиса использует эти сертификаты.

Такая схема продемонстрирована на 21-й. Сертификат Алисы заверен CA_A, сертификат Боба заверен CA_B. Алиса знает открытый ключ CA_A. У CA_C есть сертификат, подписанный CA_A, поэтому Алиса может проверить это. У CA_C есть сертификат, подписанный CA_D. И сертификат Боба подписан CA_D. Подымаясь по дереву сертификации до общей точки, в данном случае CA_D, Алиса может проверить сертификат Боба.

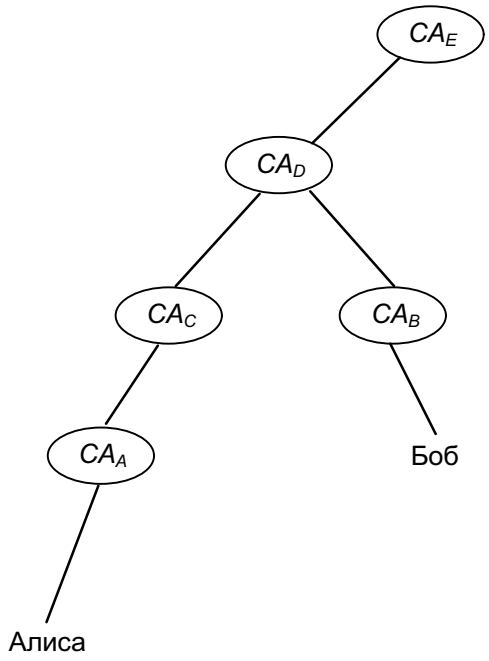

Рис. 24-3. Пример иерархии сертификации.

Сертификаты могут храниться в базах данных на различных узлах сети. Пользователи могут посылать их друг другу. Истечении срока действия сертификата он должен быть удален из всех общедоступных каталогов. Однако CA, выдавший сертификат, должен продолжать хранить его копию, которая может потребоваться при разрешении возможных споров.

Сертификаты также могут быть отозваны, либо из-за компрометации ключа пользователя, либо из-за того, что CA больше не хочет подтверждать сертификат данного пользователя. Каждый CA должен поддерживать список всех отозванных сертификатов, срок действия которых еще не закончился. Когда Алиса получает новый сертификат, она должна проверить, не был ли он отозван. Она может проверить базу данных отозванных ключей по сети, но скорей всего она проверит локально кэшируемый перечень отозванных сертификатов. В такой системе определенно вероятны злоупотребления, отзыв сертификатов возможно является самой слабой частью

этой схемы.

Протоколы проверки подлинности

Алисе нужно связаться с Бобом. Сначала она извлекает из базы данных **последовательность сертификации** от Алисы до Боба и открытый ключ Боба. В этот момент Алиса может инициировать однопроходный, двухпроходный или трехпроходный протокол проверки подлинности.

Однопроходный протокол представляет собой простую передачу данных Бобу Алисой. Протокол устанавливает личности и Алисы, и Боба, а также целостность информации, передаваемой Бобу Алисой. Кроме того, он обеспечивает защиту от вскрытия линии связи с помощью повтора.

В двухпроходном протоколе добавлен ответ Боба. Протокол устанавливает, что именно Боб, а не какой-то самозванец, посылает ответ. Он также обеспечивает безопасность обеих передач и защищает от вскрытия повтором.

И в однопроходных, и в двухпроходных алгоритмах используются метки времени. В трехпроходном протоколе добавляется еще одно сообщение Алисы Бобу и позволяет избежать меток времени (и, следовательно, правильного единого времени).

Однопроходный протокол:

(1) Алиса генерирует случайное число R_A.

(2) Алиса создает сообщение, $M = (T_A, R_A, I_B, d)$, где T_A - метка времени Алисы, I_B - идентификатор Боба, d - произвольные данные. Для безопасности данные могут быть зашифрованы открытым ключом Боба E_B.

(3) Алиса посылает Бобу $(C_A, D_A(M))$. (C_A - это сертификат Алисы, D_A - это общий узел дерева сертификации.)

(4) Боб проверяет C_A и получает E_A. Он проверяет, что срок действия этих ключей еще не истек. (E_A - это открытый ключ Алисы.)

(5) Боб использует E_A для дешифрирования $D_A(M)$. Этим действием он проверяет и подпись Алисы, и целостность подписанной информации.

(6) Боб для точности проверяет I_B в M.

(7) Боб проверяет T_A в M и убеждается, что сообщение является текущим.

(8) Дополнительно Боб может проверить R_A в M по базе данных старых номеров, чтобы убедиться, что сообщение не является повторяемым старым сообщением.

Двухпроходный протокол состоит из однопроходного протокола и последующего аналогичного однопроходного протокола от Боба к Алисе. После выполнения этапов (1)-(8) однопроходного протокола двухпроходный протокол продолжается следующим образом:

(9) Боб генерирует случайное число R_B.

(10) Боб создает сообщение $M' = (T_B, R_B, I_A, R_A, d)$, где T_B - метка времени Боба, I_A- идентификатор Алисы, а d - произвольные данные. Для безопасности данные могут быть зашифрованы открытым ключом Алисы E_A. R_A - случайное число Алисы, созданное на этапе (1).

(11) Боб посылает Алисе sends $D_B(M')$.

(12) Алиса использует E_B, чтобы расшифровать $D_B(M')$. Таким образом одновременно проверяются подпись Боба и целостность подписанной информации.

(13) Алиса для точности проверяет I_A в M'.

(14) Алиса проверяет T_B в M' и убеждается, что сообщение является текущим.

(15) Дополнительно Алиса может проверить R_B в M', чтобы убедиться, что сообщение не является повторяемым старым сообщением.

Трехпроходный протокол решает ту же самую задачу, но без меток времени. Этапы (1) - (15) такие же, как в двухпроходном алгоритме, но $T_A = T_B = 0$.

(16) Алиса сверяет полученную версию R_A с R_A, которое было отправлено Бобу на этапе (3).

(17) Алиса посылает Бобу $D_A(R_B)$.

(18) Боб использует E_A, чтобы расшифровать $D_A(R_B)$. Таким образом одновременно проверяются подпись Алисы и целостность подписанной информации.

(19) Алиса сверяет полученную версию R_B с R_B, которое было отправлено Алисе на этапе (10).

24.10 Почта с повышенной секретностью PRIVACY-ENHANCED MAIL (PEM)

Почта с повышенной секретностью (Privacy-Enhanced Mail, PEM) представляет собой стандарт Internet для почты с повышенной секретностью, одобренный Советом по архитектуре Internet (Internet Architecture Board, IAB) для обеспечения безопасности электронной почты в Internet. Первоначальный вариант был разработан Группой секретности и безопасности (Privacy and Security Research Group, PSRG) Internet Resources Task Force (IRTF), а затем их разработка была передана в Рабочую группу PEM Internet Engineering Task Force (IETF) PEM Working Group. Протоколы PEM предназначены для шифрования, проверки подлинности, проверки цел остности сообщения и управления ключами.

Полностью протоколы PEM сначала были детально описаны в ряде RFC (Requests for Comment, Запросы комментариев) в [977] и затем пересмотрены в [978]. Третья итерация протоколов [979, 827, 980] сведена в [177, 178]. Протоколы были изменены и улучшены, и окончательные протоколы детально описываются в др угом наборе RFC [981, 825, 76, 802]. В другой статье Мэтью Бишопа (Matthew Bishop) [179] подробно описаны все изменения. Попытки реализации PEM рассматриваются в [602, 1505, 1522, 74, 351, 1366, 1367]. См. также [1394].

PEM является расширяемым стандартом. Процедуры и протоколы PEM разработаны так, чтобы быть совместимыми со множеством подходов к управлению ключами, включая симметричную схему и использование открытых ключей для шифрования ключей шифрования данных. Симметричная криптография применяется для шифрования текста сообщений. Для контроля целостности сообщения используются криптографические спос обы хэширования. Другие документы поддерживают механизмы управления ключами с помощью сертификатов открытых ключей, алгоритмов, режимов и связанных идентификаторов, а также и электронные подробности, инфраструктуру и процедуры управления ключами.

PEM поддерживает только определенные алгоритмы, но позволяет добавлять и более поздние алгоритмы. Сообщения шифруются алгоритмом DES в режиме CBC. Проверка подлинности, обеспечиваемая средством **Проверки целостности сообщения** (Message Integrity Check, MIC), использует MD2 или MD5. Симметричное управление ключами может применять либо DES в режиме , либо тройной DES с двумя ключами (так называемый режим EDE). Для управления ключами PEM также поддерживает сертификаты открытых ключей, испол ьзуя RSA (длина ключа до 1024 битов) и стандарт X.509 для структуры сертификатов.

PEM обеспечивает три сервиса повышения секретности: конфиденциальность, проверка подлинности и ко нтроль целостности сообщений. К электронной постовой системе не предъявляется никаких специальных треб ований. PEM может быть встроены выборочно, в определенные узлы или у определенных пользователей, не вл ияя на работу остальной сети.

Документы PEM

PEM определяется в следующих четырех документах:

— RFC 1421: Часть I, Процедуры шифрования и проверки подлинности сообщений. В этом документе опре-
  деляются процедуры шифрования и проверки подлинности сообщений, которые должны обеспечить
  функции почты с повышенной секретностью для передачи электронной почты в Internet.

— RFC 1422: Часть II, Управление ключами с помощью сертификатов. В этом документе определяется ар-
  хитектура и инфраструктура управления ключами, которые основаны на методе сертификатов открытых
  ключей, предоставляющих информацию о ключах отправителям и получателям сообщений.

— RFC 1423: Часть III, Алгоритмы, режимы и идентификаторы. Этот документ содержит определения,
  форматы, ссылки и цитаты для криптографических алгоритмов, режимов использования и связанных
  идентификаторов и параметров.

— RFC 1424: Часть IV, Сертификация ключей и родственные функции. В этом документе описываются три
  типа функций, поддерживаемых PEM: сертификация ключей, хранение и извлечение списка отозванных
  сертификатов (certificate revocation list, CRL).

Сертификаты

PEM совместим со схемой проверки подлинности, описанной в [304], см. также [826]. PEM представляет собой надмножество X.509, определяя процедуры и соглашения для инфраструктуры управления ключами, и спользуемой с PEM и в будущем другими протоколами (включая стеки TCP/IP и OSI).

Инфраструктура управления ключами использует общий корень для всей сертификации Internet. Центр регистрационной политики (Internet Policy Registration Authority, IPRA) определяет глобальную стратегию, приме нимую ко всей иерархии. Ниже корня - IPRA - находятся Центры сертификационной политики (Policy Certification Authorities, PCA), каждый из которых определяет и опубликовывает свою стратегию регистрации пользов ателей и организаций. Каждый PCA сертифицирован IPRA. Следом за PCA идут CA, сертифицирующие пользо-

вателей и и управляющие организационными подразделениями (департаментами, офисами, дочерними компаниями). Первоначально предполагалось, что большинство пользователей будет регистрироваться в качестве членов организаций.

Как ожидается, ряд PCA будет обеспечивать сертификацию пользователей, не входящих ни в одну организацию. Предполагается выделить один или несколько PCA для регистрации пользователей, желающих воспользоваться преимуществами секретности PEM и сохранить анонимность. Стратегия этих PCA будет позволять регистрировать пользователей, не желающих раскрывать свои личности.

Сообщения PEM

Сердцем PEM является формат сообщений. На 20-й показано зашифрованное сообщение при симметричном управлении ключами. На 19-й показано подписанное и зашифрованное сообщение при управлении ключами на базе открытых ключей, и на Figure 24.6 показано подписанное (но незашифрованное) сообщение при управлении ключами на базе открытых ключей.

```
   -----BEGIN PRIVACY-ENHANCED MESSAGE-----
   Proc-Type: 4,ENCRYPTED
   Content-Domain: RFC822
   DEK-Info: DES-CBC,F8143EDE5960C597
   Originator-ID-Symmetric: schneler@counterpane.com,,
   Recipient-ID-Symmetric: schneler@chinet.com,ptf-kmc,3
   Key-Info                                                                :
DES-ECB,RSA-MD2,9FD3AAD2F2691B9A,B70665BB9BF7CBCDA60195DB94F727D3
   Recipient-lD-Symmetric: penl-dev@tis.com,ptf-kmc,4
   Key-Info :
DES-ECB,RSA-MD2,161A3F75DC82EF26,E2EF532C65CBCFF79F83A2658132DB47
LLrHBOeJzyhP+/fSStdH8okeEnv47jxe7SJ/iN72ohNcUk2jHEUSoHlnvNSIHE9M
8tEjmF/zxB+bATMtPjCUHbz8Er9wloxIkjHUlBEpvXROUrUzYbkNpkOagV2IzUpk
J6UiRRGcDSvzrsoK+oNvqu6z7Xs5Xfz5rDqUcMlKlZ6720dcBHGGsDLpTpSCnpot
   dXd/H5LMDHnonNvPCmQUHt==
   -----END PRIVACY ENHANCED MESSAGE-----
```

Рис. 24-4. Пример встроенного сообщения (симметричный случай)

```
   -----BEGIN PRIVACY-ENHANCED MESSAGE-----
   Proc-Type: 4,ENCRYPTED
   ContentDomain: RFC822
   DEK-Info: DESCBC,BFF968AA74691AC1
   0riginator - Certificate :
MIIBlTCCAScCAWuwDQYJKoZIhvcNAQECBQAwUTELMAkGAlUEBhMCVVMxIDAeBgNV
BAoTFIJTQSBEYXRhIFNIY3VyaXR5LCBJbmMuMQ8wDQYDVQQLEmZCZXRhIDExDzAN
BgNVBAsTBk5PVEESWTAeFw05MTA5MDQxODM4MTdaFmO5MzA5MDMxODM4MTZaMEUx
CzAJBgNVBAYTAlVTMSAmHgYDVQQKExdSUOEgRGFOYSBTZWNlcmlOeSwgSW5jLjEU
MBIGAIUEAxMLVGVzdCBVc2VyIDEmHTAKBgRVCAEBAglCAANLADBIAkEAwHZHI7i+
yJcqDtjJComzTdBJrdAiLAnSC+CnnjOJEEyuQiBgkGrgIh3j8/xOfM+YrsyFlu3F
LZPVtzlrldhYFJQI DAQABMAOGCSqGSIb3DQEBAgUAAIkACKrOPqphJYwlj+YPtc
iWlFPuN5jJ79Khfg7ASFxskYkEMjRNZV/HZDZQEhtVaU7Jxfzs2mfX5byMp2X3U/
   5XIJXGx7qlJsDgHQGs7Jk9H8CHlfuSHUgN4w==
   Key-Info: RSA,
I3rRIGXUGWAF8js5wCzRTkdh034PTHdRZY9Tuvm03M+NM7fx6qc5uIxps2LrlgO+
   wGrtiUm/ovtKdlnzeZQ/aQ==
   Issuer-Certificate:
MIIB3DCCAUgCAQowDQYJKoZIhvcNAQECBQAwTzELMAkGAIUEBhMCVVMxIDAeBgNV
BAoTFIJTQSBEYXRhIFNIY3VyaXR5LCBJbmMuMQ8mDQYDVQQEEwZCZXRhIDExDTAE
BgNVBAsTBFRMQOEwHhcNOTEmOTAxMDgwMDAwWhcNOTlwOTAxMDclOTU5HjBRMQsw
CQYDVQQGEwJVUzEgMB4GAIUEChMXUINBIERhdGEgU2VjdXJpdHksIEIuYy4xDzAN
BgNVBAsTBkJIdGEgMTEPMAOGAIUECxMGTk9UQVJZMHAmCgYEVQgBAQICArmDYgAm
XwJYCsnp61QCxYykNIODwutF7jMJ3kE+3PjYyHOwk+79rLg6X65B/ED4bJHt05XH
cqAz/7R7XhjYCmOPcqbdzoACZtIIETrKrcJiDYoP+DkZ8klgCk7hQHpbImIDAQAB
MAOGCSqGSIb3DQEBAgUAA38AAICPv4f9Gx/tY4+p+4DB7MV+tKZnvBoy8zgoMGOx
dD2jMZ/3HsyWKHgSFOeH7AJB3qr9zosG47pyMnTf3aSy2nBO7CMxpUHRBcXUpE+x
EREZd9++32ofGBIXaIaInOgVUnOOzSYgugiQO77nJLDUjOhQehCizEs5wUJ35a5h
   MIC-Info: RSA-MD5,RSA,
UdFJR8u/TIGhfH651eeme210H4tooa3vZCvVNGBZirf/7nrgzHDABz8m9NsXSexv
   AjRFbHoNPzBuxwnlOAFeAOHJszE4yBvhG
   Recipient-ID-Asymmetric :
MFExCzAJBgNVBAYTAIVTMSAmHgYDVQQKExdSUOEgRGFOYSBTZHNlcmIOeSwgSH5j
   LjEPMAOGAIUECxMGQftiVOYSAxMQ81tfDQYDVQQLEwZOTIRBUIk=,
   66
   Key-Info: RSA,
06BSIww9CTyHPtS3bMLD+EOhejdvX6QvlHK2ds2sQPEaXhX8EhvVphHYTjmekdHv
   7xOZ3Jx2vTAhOYHMcqqCjA==
qeWlj/YJ2Uf5ng9yznPbtDOmYloSwIuV9FRYx+gzY+81Xd/NQrXHfi6/MhPfPF3d
   jIqCJAxvld2xgqQimUzoSla4r7kQQ5c/Iua4LqKeq3clFzEv7MbZhA==
   -----END PRIVACY ENHANCED MESSAGE-----
```

Рис. 24-5. Пример встроенного шифрованного (ENCRYPTED) сообщения (асимметричный случай).

Первым полем является "Proc-Type", идентификатор типа обработки, которой подверглось сообщение. Существует три возможных типа сообщений. Спецификатор "ENCRYPTED" обозначает, что сообщение зашифро-

вано и подписано. Спецификатор "MIC-ONLY" и "MIC-CLEAR" указывают, что сообщение подписано, но не зашифровано. Сообщения MIC-CLEAR не кодируются и могут быть прочитаны с помощью другого, не входящего в PEM программного обеспечения. Для преобразования сообщений MIC-ONLY в удобочитаемую форму необходимо программное обеспечение PEM. Сообщение PEM подписывается всегда, а шифрование не является обязательным.

Следующее поле, "Content-Domain", задает тип почтового сообщения. Оно не влияет на безопасность. Поле "DEK-Info" содержит информацию о **ключе обмена данными** (Data Exchange Key, DEK), алгоритме, используемом для шифрования текста, и параметрах, связанных с алгоритмом шифрования . В настоящее время определен единственный алгоритм - DES в режиме CBC, "DES-CBC" Второе подполе содержит IV. В будущем для PEM могут быть определены и другие алгоритмы, их использование будет запротоколировано в поле DEK-Info и других полях, определяющих алгоритм.

В сообщениях с симметричным управлением ключами (см. 20th) следующим полем будет "Originator-ID-Symmetric" с тремя подполями. Первое подполе с помощью уникального адреса электронной почты определяет отправителя. Второе поле не является обязательным и определяет орган, выдавший заменяемый ключ . Третьим является необязательное подполе Версия/Окончание срока .

Далее, при использовании симметричного управления ключами, у каждого получателя есть два поля : "Recipient-ID-Symmetric" и "Key-Info." Поле "Recipient-ID-Symmetric" содержит три подполя, которые определяют получателя также, как подполя поля "Originator- ID-Symmetric" определяют отправителя.

Поле "Key-Info" задает параметры управления ключами. У этого поля четыре подполя. Первое определяет алгоритм, использованный для шифрования DEK. Так как в рассматриваемом сообщении применяется симметричное управление ключами, то отправитель и получатель используют общий ключ . Он называется **заменяемым ключом** (Interchange Key, IK) и используется для шифрования DEK. DEK может быть зашифрован либо с помощью DES в режиме ECB (этот способ обозначается "DES-ECB"), либо тройным DES ("DES-EDE"). Второе подполе определяет алгоритм MIC. Может использоваться MD2 (обозначается "RSA-MD2") или MD5 ("RSA-MD5"). Третье подполе, DEK, и четвертое подполе, MIC, шифруются с помощью IK.

На 19-й и 18-й показаны сообщения, в которых используется управление ключами с помощью открытых ключей (в перечне PEM такой способ называется асимметричным). Заголовки изменяются. В сообщениях ENCRYPTED после поля "DEK-Info" идет поле "Originator-Certificate". Форма сертификата соответствует стандарту X.509 (см. раздел 24.9). Следующим полем является "Key-Info" с двумя подполями. Первое подполе определяет алгоритм с открытым ключом, использованный для шифрования DEK, в настоящее время поддерживается только RSA. Следующее подполе - DEK, зашифрованный открытым ключом отправителя. Это необязательное поле, которое позволяет отправителю расшифровать свое собственное сообщение, возвращенное почтовой системой. Следующим полем является "Issuer-Certificate", сертификат организации, подписавшей сертификат отправителя ("Originator-Certificate").

Далее при асимметричном управлении ключами следует поле "MIC-Info". Первое подполе задает алгоритм вычисления MIC, а второе - алгоритм, использованный для подписи MIC. Третье подполе содержит MIC, подписанный закрытым ключом отправителя.

```
   -----BEGIN PRIVACY-ENHANCED MESSAGE-----
   Proc-Type: 4,MIC-ONLY
   Content-Domain: RFC822
   0riginator - Certificate :
MIIBITCCAScCAHUwDQYJKoZIhvcNAQECBQAwTzELMAkGAIUEBhMCVVMxlDAeBgNV
BAoTFIJTQSBEYXRhIFNIY3VyaXR5LCBJbmMuMQ8wDQYDVQQLEwZCZXRhIDExDzAN
BgNVBAsTBk5PVEFSHTAeEw05MTA5MDQxODM4MTdaFw05MzA5MDMxODM4MTZaMEUx
CzAJBgNVBAYTAIVTMSAwHgYDVQQKExdSUOEgRGFOYSBTZMNlcmlOeSwgSW5jLjEU
MEIGAIUEAxMLVGVzdCBVc2VylDEwHTAKBgRVCAEBAgICAANLADBIAkEAmHZHI71+
yJcqDtjJCowzTdBJrdAiLAnSC+CnnjOJELyuQiBgkGrgIh3j8/xOfM+YrsyFlu3F
LZPVtzlndhYFJQIDAQABMAOGCSqGSIb3DQEBAgUAAIkACKrOPqphJYwlj+YPtcIq
ItJIFPuN5jJ79Khfg7ASFxskYkEMjRNZV/HZDZQEhtVaU7Jxfzs2mfX5bMp2X3U/
   5XUXGx7qusDgHQGs7Jk9W8CW1fuSI4UgN4w==
   Issuer-Certificate:
MIIB3DCCAUgCAQowDQYJKoZIhvcNAQECBQAkTzELMAkGAIUEBhMCVVMxlDAeBgNV
BAoTFIJTQSBEYXRhIFNIY3VyaXR5LCBJbmMuMQ8wDQYDVQQLEIZCZXRhIDExDTAE
BgNVBAsTBFRMQOEwHhcNOTEwOTAxMDgwMDAmkmcNOTImOTAxMDclOTU5HjBRMQsw
CQYDVQQGEwJVUzEgMB4GAIUEChMXUINBIERhdGEgU2VjdXJpdHksIEIuYywxDzAN
BgNVBAsTBkJIdGEgMTEPMAOGAIUECxMGTk9UQVJZMHAwCgYEVQgBAQICArwDYgAw
XwJYCsnpelQCxYkNIODwlltF/jMJ3kE+3PjYjHOwk+/9rEg6X65B7LD4bJHt05XH
cqAz77R7XhjYCmOPcqbdzoACZIlETrKrcJIDYoP+DkZ8klgCk7hQHpbltflDAQAB
MAOGCSqGSIb3DQEBAgUAA38AAICPv4f9Gx7tY4+p+4DB7MV+tKZrlvBo8zgoMGOx
dD2jMZ73Hs*kl4gSFOeH7AJB3qr9zosG47pyMnTf3aS2nBO7CMxpUkfRBcXUpE+x
EREZd9++32otGBIXaIalnOgVUnOOzSYgljglQO77nJEDUOhQehCizEs5mUJ35a5h
   MIC-Info: RSA-MD5,RSA,
jV20fH+nnXHU8bnE8kPAad7mSQITDZIbVuxvZAOVRZ5q5+EjI5bQvqNeqOUNQjr6
   EtE7K2QDeVMCj/XsdJIA8fA==
LSBBIGIIc3NhZ2UgZrti9lHVzZSBpbBOZXNOaH5nLgOKESBGb2xsb3dpbmcgaXMg
   YSBibGFuaj/Bsakf51OgOKDQpUaGIzIGIzIHRoZSBIbrnQuDQo=
   -----END PRIVACY-ENHANCED MESSAGE-----
```

Рис. 24-6. Пример встроенного MIC-ONLY сообщения (асимметричный случай).

Следующие поля связаны с получателями. Каждому получателю соответствуют два поля : "Recipient-ID-Asymmetric" и "Key-Info". У поля"Recipient-ID-Asymmetric" два подполя. Первое определяет орган, выдавший открытый ключ получателя, а вторым является необязательное подполе Версия/Окончание срока . Поле "Key-Info" задает параметры управления ключами: первое подполе определяет алгоритм, использованный для ши фрования сообщения, а вторым подполем служит DEK, зашифрованный открытым ключом получателя.

Безопасность PEM

Длина ключей RSA, используемых в PEM, может меняться в диапазоне от 508 до 1024 битов. Этого достаточно практически для любого уровня безопасности. Более вероятно, что вскрытие будет направлено против протоколов управления ключами. Мэллори может украсть ваш закрытый ключ - не записывайте его нигде - или попытаться подсунуть вам фальшивый открытый ключ. Процедуры сертификации ключей в PEM делают это невозможным, если все пользователи строго следуют соответствующим процедурам , но, как известно, люди часто неаккуратны.

Мэллори может поступить хитрее и модифицировать реализацию PEM, работающую в вашей системе. Эта измененная версия может тайком пересылать Мэллори всю вашу почту, зашифровав ее его открытым ключом. Ему может быть послана даже копия вашего закрытого ключа . Если измененная реализация будет работать х орошо, то вы никогда не узнаете, что случилось.

Реального способа предотвратить такое вскрытие не существует. Вы можете использовать однонаправле нную хэш-функцию и получить контрольную сумму исполняемого кода PEM. Затем, при каждом запуске программного обеспечения вы можете проверять контрольную сумму, чтобы вовремя обнаружить изменения . Но Мэллори точно также может изменить и код контрольной суммы при изменении кода PEM. Можно сохранить контрольную сумму контрольной суммы, но Мэллори может изменить и ее. Если у Мэллори есть доступ к вашему компьютеру, он может разрушить безопасность PEM.

Мораль в том, что вы не должны доверять никакому элементу программного обеспечения, если вы не мож ете доверять аппаратуре, на которой работает это программное обеспечение . Для большинства такие опасения покажутся необоснованными. Но для некоторых людей они вполне реальны .

TIS/PEM

Доверенные информационные системы (TIS, Trusted Information Systems), частично поддерживаемые Управлением по передовым научным проектам правительства Соединенных Штатов , включают реализацию PEM (TIS/PEM). Разработанные для платформ UNIX, они были также перенесены на VMS, DOS и Windows.

Хотя спецификации PEM определяют для Internet один главный сертификационный центр, TIS/PEM поддерживает существование нескольких иерархий сертификации. Узлы могут определить набор сертификатов, которые будут считаться действительными, включая все сертификаты, выданные узлами. Для того, чтобы пользоваться TIS/PEM узлу не нужно присоединяться к иерархии Internet.

Все организации и граждане США и Канады при желании могут получить доступ к TIS/PEM, которая распространяется в виде исходного кода. Заинтересованные лица должны обращаться по следующему адресу: Privacy-Enhanced Mail, Trusted Information Systems, Inc., 3060 Washington Road IRte. 97), Glenwood, MD 2,1738; (301) 854-6889; fax: (301) 854-5363; Internet: pern-info@tis.com.

RIPEM

RIPEM - это программа, написанная Марком Риорданом (Mark Riordan) и реализующая протоколы PEM. Хотя эта программа не является свободно доступной, ей можно воспользоваться бесплатно для частного, некоммерческого использования. Лицензия на ее использование входит в документацию.

Код не может быть экспортирован. Конечно, законы правительства США не действуют за пределами Соединенных Штатов, и ряд людей игнорирует экспортные ограничения. Код RIPEM доступен по всему миру на электронных досках объявлений. Разрешена для экспорта версия, называемая RIPEM/SIC, реализующая только цифровые подписи.

К моменту написания этих строк RIPEM не полностью реализовала протоколы PEM, в ней нет возможности использовать сертификаты проверки подлинности ключей.

До RIPEM Риордан написал похожую программу RPEM. Подразумевалось, что это будет общедоступная программа электронной почты. Пытаясь обойти патентные проблемы, Риордан использовал алгоритм Rabin (см. раздел 19.5). Public Key Partners заявила, что их патенты распространяются на всю криптографию с открытыми ключами. Под угрозой судебного процесса Риордан прекратил распространение программы.

Сейчас RPEM не используется. Она не совместима с RIPEM. Так как можно использовать RIPEM, не встречая препятствий со стороны Public Key Partners, нет повода возвращаться к RPEM.

24.11 Протокол безопасности сообщений

Протокол безопасности сообщений (Message Security Protocol, MSP) - это военный эквивалент PEM. Он был разработан NSA в конце 80-х годов при работе по программе создания Безопасной системы передачи данных по сети (Secure Data Network System, SDNS) program. Это совместимый с X.400 протокол уровня приложения для закрытия электронной почты. MSP планируется использовать в разрабатываемой сети оборонных сообщений (Defense Message System, DMS) Министерства обороны.

Предварительный протокол безопасности сообщений (Preliminary Message Security Protocol, PMSP), который предполагается использовать для "несекретных, но важных" сообщений, представляет собой адаптированную для использования с X.400 и TCP/IP версию MSP. Этот протокол также называют Mosaic.

Как и PEM, программные реализации MSP и PMSP достаточно гибки, их конструкция позволяет подстроиться под использование различных алгоритмов для осуществления функций безопасности, таких как подпись, хэширование и шифрование. PSMP будет работать с микросхемой Capstone (см. раздел 24.17).

24.12 PRETTY GOOD PRIVACY (PGP)

Pretty Good Privacy (PGP, весьма хорошая секретность) - это свободно распространяемая программа безопасной электронной почты, разработанная Филипом Циммерманном (Philip Zimmermann) [1652]. Для шифрования данных она использует IDEA, для управления ключами и цифровой подписи - RSA (длина ключа до 2047 битов), а для однонаправленного хэширования - MD5.

Для получения случайных открытых ключей PGP использует вероятностную проверку чисел на простоту, используя для получения стартовых последовательностей интервалы между нажатиями пользователем клавиш на клавиатуре. PGP генерирует случайные ключи IDEA с помощью метода, в ANSI X9.17, Appendix C (см. раздел 8.1) [55], используя вместо DES в качестве симметричного алгоритма IDEA. PGP также шифрует закрытый ключ пользователя с помощью хэшированной парольной фразы, а не пароля непосредственно.

Сообщения, зашифрованные PGP, имеют несколько уровней безопасности. Единственная вещь, известная криптоаналитику о зашифрованном сообщении, - это получатель сообщения при условии, что криптоаналитику известен ID ключа получателя. Только расшифровав сообщение, получатель узнает, кем оно подписано, если оно подписано. Это резко отличается от сообщения PEM, в заголовке которого немало информации об отправителе, получателе и самом сообщении хранится в незашифрованном виде.

Самой интересной особенностью PGP является распределенный подход к управлению ключами (см. раздел 8.12). Центров сертификации ключей нет, вместо этого в PGP поддерживается "сеть доверия". Каждый пользователь сам создает и распространяет свой открытый ключ. Пользователи подписывают ключи друг друга, создавая взаимосвязанное сообщество пользователей PGP.

Например, Алиса может физически передать Бобу свои открытый ключ. Боб лично знает Алису, поэтому он

подписывает ее открытый ключ. Одну подписанную копию он возвращает Алисе, а другую оставляет. Когда Алисе нужно связаться с Кэрол, она посылает Кэрол подписанную Бом копию ключа. Кэрол, у которой каким-то образом уже есть ключ Боба (она получила его раньше), и которая доверяет Бобу заверить ключ другого человека, проверяет его подпись под ключом Алисы и убеждается, что она правильна. Таким образом, Боб знакомит Алису и Кэрол.

PGP не определяет стратегию установки доверительных связей, пользователи сами решают, кому верить, а кому нет. PGP обеспечивает механизмы для поддержки ассоциативного доверия открытым ключам и для использования доверия. Каждый пользователь хранит набор подписанных открытых ключей в виде файла **кольца открытых ключей** (public-key ring). Каждый ключ кольца обладает полем законности ключа, определяющим уровень доверия к ключу конкретного пользователя. Чем больше уровень доверия, тем больше пользователь уверен в законности ключа. Поле доверия к подписи измеряет, насколько пользователь верит тому, кто подписал открытые ключи других пользователей. И наконец поле доверия к владельцу ключа задает уровень, определяющий, насколько конкретный пользователь верит владельцу ключа, подписавшему другие открытые ключи. Это поле вручную устанавливается пользователем. PGP непрерывно обновляет эти поля по мере появления новой информации.

На 17-й показано, как выглядит эта модель для конкретного пользователя, Алисы. Ключ Алисы находится в самом верху иерархии, владелец ключа абсолютно надежен. Алиса подписывает ключи Боба, Кэрол, Дэйва, Элен и Фрэнка. Она доверяет Бобу и Кэрол подписывать открытые ключи других людей, кроме того, она частично доверяет Дэйву и Элен подписывать открытые ключи других людей. И она доверяет Гейл подписывать открытые ключи других людей, хотя сама не подписывала ключ Гейл.

Двух частично доверяемых подписей может оказаться достаточным для сертификации ключа. Алиса считает, что ключ Курта законен, так как Дэйв и Элен подписали его. Уровень доверия устанавливается в PGP вручную, Алиса может выбрать устраивающую ее степень паранойи.

Алиса не должна автоматически доверять ключам других людей только потому, что они подписаны ключом, который она считает правильным. Алиса Она не доверяет Фрэнку Она подписывать другие ключи, хотя она собственноручно подписывала его ключ. Кроме того, она не доверяет подписи Ивана под ключом Мартина или подписи Курта под ключом.

Ключ Оуэна вообще на входит в сеть, может быть, Алиса получила его от сервера. PGP не считает ключ автоматически правильным, Алиса должна либо объявить о правильности ключа, либо решиться поверить одному из тех, кто подписал ключ.

Конечно, ничто не мешает Алисе использовать ключи, которым она не доверяет. Задача PGP - предупредить Алису о подозрительности ключа, а не помешать ей устанавливать соединения.

Самым слабым звеном этой системы является отзыв ключей: гарантировать, что кто-нибудь не воспользуется скомпрометированным ключом, невозможно. Если закрытый ключ Алисы украден, она может послать некий **сертификат отзыва ключа** (key revocation certificate), но, так как некое распределение ключей уже произошло, нельзя гарантировать, что это сообщение будет получено всеми, использующими ее открытый ключ в своем кольце ключей. И так как Алиса должна будет подписать свой сертификат отзыва ключа своим закрытым ключом, то если она потеряет ключ, она не сможет и отозвать его.

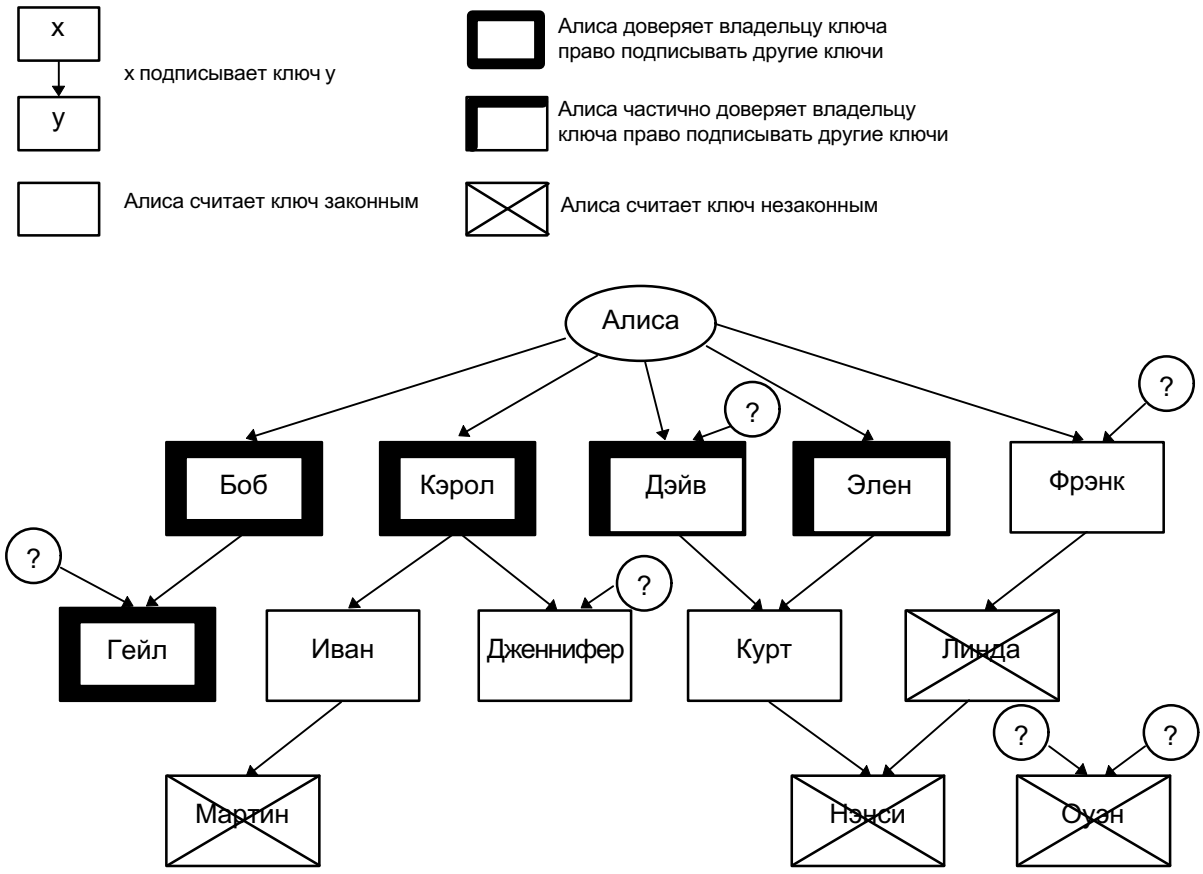

Рис. 24-7. Модель доверия в PGP.

Текущей версией PGP является 2.6.2. Появление новой версии, PGP 3.0, ожидается к концу 1995 года. В 3.0 включены опции тройного DES, SHA, другие алгоритмы с открытыми ключами, разделение пар "открытый ключ/закрытый ключ" для шифрования и для подписи, расширенные процедуры отзыва ключей, улучшенные функции управления кольцом ключей, API для интегрирования PGP в другие программы и полностью переписанные исполняемые модули.

PGP доступна для MS-DOS, UNIX, Macintosh, Amiga и Atari. В личных, некоммерческих целях ее можно использовать свободно, скачав со многих узлов ftp в Internet. Чтобы скопировать PGP с узла MIT с помощью telnet подключитесь к net-dist.mit.edu, войдите в систему как getpgp, ответьте на вопросы, затем используйте ftp для соединения с net-dist.mit.edu и перейдите в каталог, указанный в сессии telnet. Эту программу также можно получить ftp.ox.ac.uk, ftp.dsi.unimi.it, ftp.funet.fi, ftp.demon.co.uk, CompuServe, AOL, и т.п. Для коммерческого использования в США PGP можно приобрести - полностью, вместе с лицензиями - примерно за 100 долларов в компании ViaCrypt, 9033 N 24th Ave., Phoenix, AZ, 85021; (602) 944-0773; viacrypt@acm.org. Существуют различные средства, помогающие интегрировать PGP в MS-DOS, Microsoft Windows, Macintosh и UNIX.

О PGP написано несколько книг [601,1394,1495]. Исходный код был даже опубликован в печатном виде в [1653] при попытке обойти Госдепартамент США, который продолжает считать, что исходный код можно экспортировать только в бумажном, а не в электронном виде. Если вы доверяете IDEA, PGP позволит вам приблизиться к военному уровню шифрования.

24.13 Интеллектуальные карточки

Интеллектуальная карточка представляет собой пластиковую карточку, по размеру и форме как кредитная карточка, с встроенной компьютерной микросхемой. Идея стара - первые патенты были выданы лет 20 тому назад - но из-за практических ограничений возможность реализовать такие карточки появилась только приме рно пять лет назад. С тех пор они стали популярны, главным образом в Европе. Во многих странах интеллектуальные карточки используются для оплаты за телефоны. Существуют интеллектуальные кредитные карточки, интеллектуальные дебитные карточки, интеллектуальные карточки для чего угодно. Американские компании по выпуску кредитных карточек работают над технологией, и через несколько лет даже захудалые американцы будут носить интеллектуальные карточки в своих бумажниках.

Интеллектуальная карточка содержит маленький компьютер (обычно 8-битовый микропроцессор), ОЗУ (четверть килобайта), ПЗУ (примерно 6-8 килобайт), и несколько килобайт либо EPROM (стираемое программируемое ПЗУ) или EEPROM (электронно стираемое программируемое ПЗУ). Объем памяти в интеллектуаль-

ных карточках следующего поколения наверняка возрастет, но определенные физические ограничения затру днят такие расширения. Карточка использует свою операционную систему, программы и данные . (Чего в ней нет, так это источника питания, электроэнергия подается, когда карточку вставляют в считыватель .) Карточка безопасна. В нашем меняющемся мире, когда нельзя доверять чужому компьютеру, телефону, да чему угодно, вы можете быть уверены в своей карточке, которая хранится в вашем бумажнике .

В интеллектуальных карточках могут использоваться различные криптографические протоколы и алгоритмы. Они могут быть электронным кошельком, давая возможность тратить и получать электронные н аличные. Карточки могут использоваться в протоколах проверки подлинности с нулевым знанием, они могут обладать собственными ключами шифрования. Возможно, они позволяют подписывать документы или снимать блокировку с компьютерных программ.

Некоторые интеллектуальные карточки считаются устойчивыми к взлому, таким образом себя часто защ ищают организации, эмитировавшие карточки. Банк вовсе не хочет, чтобы вы могли влезть в их интеллектуал ьную карточку и начислить себе побольше денег.

Интеллектуальные карточки - это очень интересная тема, на которую написано множество литературы . Хорошей обзорной статьей по криптографии в интеллектуальных карточках может служить [672]. Ежегодно проводятся конференции: CARTES в октябре в Париже и CardTech в апреле в Вашингтоне, округ Колумбия. Труды двух других конференций по интеллектуальным карточкам можно найти в [342, 382]. В области интеллектуальных карточек существуют сотни патентов, частью принадлежащие европейским компаниям. Интересные вопросы будущего использования интеллектуальных карточек - проверка целостности, аудиторский контроль, защита от копирования, электронные наличные, оплата почтовых расходов - описаны в [1628].

24.14 Стандарты криптографии с открытыми ключами

Стандарты криптографии с открытыми ключами (Public-Key Cryptography Standards, PKCS) - это попытка компании RSA Data Security, Inc обеспечить промышленный стандарт для криптографии с открытыми ключами. По традиции такими делами занимался ANSI, но, учитывая текущую ситуацию в криптографической политике, RSADSI решила, что лучше они все сделают сами. Работая со множеством компаний, RSADSI разработала набор стандартов. Некоторые из них совместимы с другими стандартами, а некот орые - нет.

Эти стандарты не являются стандартами в общепринятом смысле этого слова, никто не собирался и не гол осовал за PKCS. По своим собственным словам RSADSI "будет единственной организацией, правомочной пр инимать решения о каждом стандарте, и будет пересматривать эти стандарты по мере необходимости " [803].

Даже это уже совсем. Если вы неуверены, какие структуры данных и синтаксис использовать при програ ммировании криптографии с открытыми ключами, эти стандарты не хуже каких-либо других. К тому же, так как это не настоящие стандарты, вы можете подстроить их под свои нужды .

Далее приведено краткое описание каждого PKCS (PKCS #2 и PKCS #4 были включены в PKCS #l).

PKCS #l [1345] описывает способ шифрования и дешифрирования RSA, главным образом для создания ци фровых подписей и цифровых конвертов, описанных в PKCS #7. Для цифровых подписей сообщение хэшируется, а затем хэш-значение шифруется закрытым ключом подписывающего. Совместное представление сообщения и хэш-значения подробно описано в in PKCS #7. Для цифровых конвертов (шифрованные сообщения) сообщение сначала шифруется симметричным алгоритмом, а затем ключ сообщений шифруется открытым ключом получ ателя. Совместное представление шифрованного сообщения и шифрованного ключа должно соответствовать PKCS #7. Эти два метода совместимы со стандартами PEM. Для структуры сертификатов (или их подобия) о ткрытых и закрытых ключей RSA и трех алгоритмов подписи - MD2 и RSA, MD4 и RSA, MD5 и RSA - PKCS #l также описывает синтаксис, идентичный синтаксису X.509 и PEM.

PKCS #3 [1346] описывает способ реализации обмена ключами по схеме Diffie-Hellman.

PKCS #5 [1347) описывает способ шифрования сообщений секретным ключом, полученным из пароля. Стандарт использует MD2 или MD5 для получения ключа из пароля и шифрует сообщения с помощью DES в режиме CBC. Этот метод предназначен главным образом для шифрования закрытых ключей при их передаче от одной компьютерной системы другой, но может быть использован и для шифрования сообщений .

PKCS #6 [1348] описывает стандартный синтаксис сертификатов открытых ключей. Синтаксис является надмножеством сертификата X.509, при необходимости можно извлечь и сертификат X.509. Дополнительные атрибуты не ограничивают процесс сертификации только открытым ключом. Они содержат и другую информ ацию, например, адрес электронной почты.

PKCS # 7 [1349] представляет собой общий синтаксис для подписываемых или шифруемых данных, напр имер, цифровых подписей или цифровых конвертов. Синтаксис является рекурсивным, поэтому можно организ овать вложенность конвертов или поставить чью-то подпись под ранее зашифрованными данными. Синтаксис также разрешает вместе с содержанием сообщения проверку подлинности других атрибутов, например, меток

времени. PKCS #7 с PEM, поэтому подписанные и зашифрованные сообщения могут быть преобразованы в с о-общения PEM, и наоборот, без дополнительных криптографических операций. Для управления ключами с по-мощью сертификатов PKCS #7 может поддерживать множество архитектур - одной из них является PEM.

PKCS #8 [1350] описывает синтаксис информации о закрытых ключах, включая закрытый ключ и набор а т-рибутов, и синтаксис шифрованных закрытых ключей. Для шифрования информации о закрытых ключах мож-но использовать PKCS #5.

PKCS #9 [1351] определяет избранные типы атрибутов для расширенных сертификатов PKCS #6, сообщений с цифровой подписью PKCS #7 и информации о закрытых ключах PKCS #8.

PKCS #10 [1352,] описывает стандартный синтаксис запросов сертификации. Сертификация включает инди-видуальное имя, открытый ключ и (необязательно) набор атрибутов, которые подписаны лицом, приславшим запрос. Запросы сертификации присылаются в сертифицирующий орган, который преобразует запрос либо в сертификат открытого ключа X.509, либо в сертификат PKCS #6.

PKCS #11 [1353], Стандарт API криптографической метки (Cryptographic Token API Standard), определяет интерфейс программирования, называемый "Cryptoki", для портативных криптографических устройств всех типов. Cryptoki представляет собой обобщенную логическую модель, позволяющую приложениям выполнять криптографические операции на портативных устройствах, не зная деталей используемой технологии. Этот стандарт также определяет профили приложения: наборы алгоритмов, которые может поддерживать устройство.

PKCS #12 [1354] описывает синтаксис хранения в программном обеспечении открытых ключей пользоват е-лей, защищенных закрытых ключей, сертификатов и другой связанной криптографической информации. Целью этого является стандартизация единого фала ключей, используемого многими приложениями.

Эти стандарты всесторонни, но не всеобъемлющи. Многие вопросы остались за пределами этих стандартов: проблема присвоения имен, некриптографические вопросы, касающиеся сертификации, длины ключей и усл о-вия для различных параметров. PKCS призваны обеспечить формат передачи данных, основанной на крипт о-графии с открытыми ключами, и инфраструктуру, поддерживающую такую передачу.

24.15 Универсальная система электронных платежей

Универсальная система электронных платежей (Universal Electronic Payment System, UEPS) представляет собой банковское приложение, использующее интеллектуальные карточки, первоначально разработанное для сельской Южной Африки, но позднее принятое основными банковскими группами этой страны. К началу 1995 года в ЮАР было выпущено около 2 миллионов карточек. Эта система также принята в Намибии, и разверт ы-вается по крайней мере одним российским банком.

Система позволяет использовать безопасные дебитные карточки, подходящие для регионов, в которых пл о-хая телефонная сеть делает невозможной диалоговую проверку. Карточки есть и покупателей, и у продавцов, покупатели могут использовать свои карточки для перевода денег продавцам. Продавец может воспользоваться своей карточкой, чтобы позвонить в банк и поместить деньги на свой банковский счет, покупатель может во с-пользоваться своей карточкой, чтобы позвонить в банк и перевести деньги на свою карточку. Нет необходимо-сти заботиться об анонимности, нужно обеспечить только защиту от мошенничества.

Вот как выглядит протокол связи между покупателем Алисой и продавцом Бобом (В действительности, Али-са и Боб просто вставляют свои карточки в машину и ожидают выполнения транзакции.) Когда Алиса впервые получает свою карточку, она получает и пару ключей, K_1 и K_2, банк вычисляет их, используя ее имя и некот о-рую секретную функцию. Только в карточки продавцов встроены секретные средства, необходимые для вычи с-ления ключей пользователей.

(1) Алиса посылает Бобу свое имя, A, его имя, B, и случайное число R_A, шифруя их с помощью DES: сначала ключом K_2, затем K_1. Она также посылает свое имя открытым текстом.

$$A, E_{K_1}(E_{K_2}(A,B,R_A))$$

(2) Боб вычисляет K_1 и K_2 по имени Алисы. Он расшифровывает сообщение, убеждается, что A и B правиль-ны, затем шифрует незашифрованную вторую половину сообщения Алисы ключом K_2.

$$E_{K_2}(A,B,R_A)$$

Боб не посылает это сообщение Алисе, 56 битов шифротекста становятся ключом K_3. Боб посылает Алисе свое имя, ее имя и случайное число, R_B, шифруя их с помощью DES: сначала ключом K_3, затем K_1.

$$E_{K_1}(E_{K_3}(B,A,R_B))$$

(3) Алиса аналогичным образом вычисляет K_3 и расшифровывает сообщение Боба, убеждаясь, что A и B пра-

вильны, затем шифрует незашифрованную вторую половину сообщения Боба ключом K_3.

$$E_{K_3}(B,A,R_B)$$

Алиса не посылает это сообщение Бобу, 56 битов шифротекста становятся ключом K_4. Затем Алиса посылает Бобу свое имя, его имя проверочное значение C. Это проверочное значение содержит имена отправителя и получателя, дату, контрольную сумму, количество и два MAC. Все это шифруется DES: сначала ключом K_4, затем K_1. Один из MAC может быть проверен банком Алисы, а второй может быть проверен только расчетно-кассовым центром. Алиса уменьшает свой счет на соответствующее значение.

$$E_{K_1}(E_{K_4}(A,B,C))$$

(4) Боб аналогичным образом вычисляет K_4. При условии, что все имена совпадают, и правильно выполнена проверка, он принимает платеж.

Великолепным нововведением в этом протоколе является то, что каждое сообщение зависит от предыдущего. Каждое сообщение выступает удостоверением *всех* предыдущих сообщений. Это означает, что повторить старое сообщение никому не удастся, получатель просто никогда не расшифрует его. Мне нравится эта идея, и я уверен, что она получит широкое применение, как только станет широко известна.

Другой разумной вещью в этом протоколе - навязывание правильной реализации. Если разработчик приложения неправильно реализует протокол, он просто не будет работать.

Обе карточки сохраняют записи каждой транзакции. Когда карточки рано или поздно установят диалоговое соединение с банком (продавец - положить деньги на счет, а покупатель - снять со счета), банк извлечет эти записи для последующего контроля.

Аппаратура изготавливается устойчивой к взлому, чтобы помешать любому из участников испортить данные. Алиса не сможет изменить значение своей карточки. Подробная запись обеспечивает данные для обнаружения и запрещения мошеннических транзакций. В карточках используются универсальные секреты - ключи MAC в карточках покупателей, функции для преобразования имен пользователей в K_1 и K_2 - но считается, что решение обратной задачи для этих секретов достаточно трудно.

Эта схема, конечно же, несовершенна, но она безопаснее бумажных чеков и обычных дебитных карточек. Источником угрозы мошенничества являются не военные враги, а покупатели и продавцы. UEPS предоставляет защиту от таких злоупотреблений.

Обмен сообщения является прекрасным примером устойчивого протокола: В каждом сообщении присутствую имена обеих сторон, включая информацию, уникальную для сообщения, каждое сообщение явным образом зависит от всех предыдущих.

24.16 CLIPPER

Микросхема Clipper (известная также как MYK-78T) - это разработанная в NSA, устойчивая к взлому микросхема, предназначенная для шифрования переговоров голосом. Это одна из двух схем, реализующих правительственный Стандарт условного шифрования (Escrowed Encryption Standard, EES) [1153]. VLSI Technologies, Inc. изготовила микросхему, а Mykotronx, Inc. запрограммировала ее. Сначала все микросхемы Clipper будут входить в Безопасное телефонное устройство Model 3600 AT&T (см. раздел 24.18). Микросхема реализует алгоритм шифрования Skipjack (см. раздел 13.12,), разработанный NSA секретный алгоритм с шифрованием секретным ключом, только в режиме OFB.

Самым противоречивым моментом микросхемы Clipper, и EES в целом, является протокол условного вручения ключей (см. раздел 4.14). У каждой микросхемы есть специальный, ненужный для сообщений, ключ. Этот ключ используется для шифрования копии ключа сообщений каждого пользователя. В ходе процесса синхронизации передающая микросхема Clipper генерирует и посылает принимающей Поле доступа для выполнения закона (Law Enforcement Access Field, LEAF). LEAF содержит копию текущего сеансового ключа, зашифрованного специальным ключом (называемым **ключом модуля**). Это позволяет правительственным прослушивателям получить сеансовый ключ и раскрыть открытый текст разговора.

По словам директора NIST [812]:

Предусматривается, что система "с условно врученным ключом" обеспечит использование микросхемы Clipper для защиты законопослушных американцев. В каждом устройстве, содержащем микросхему будет два уникальных "ключа", два числа, которые понадобятся уполномоченным правительственным органам для дешифрирования сообщений, зашифрованных устройством. При изготовлении устройства оба ключа будут помещены порознь в двух базах данных " условно врученных ключей", контролируемых Генеральным прокурором. Доступ к этим ключам будет разрешен только правительственным чиновникам с законным разрешением подключить подслушивающее устройство.

Правительство также собирается поощрять широкое распространение таких телефонных аппаратов, но никто не знает, что может произойти с базами данных условно врученных ключей.

Помимо политических аспектов, стоит поговорить и о внутренней структуре LEAF [812, 1154, 1594, 459, 107, 462]. LEAF - это строка, включающая достаточно информации, чтобы при обеспечении правопорядка можно было раскрыть сеансовый ключ K_s при условии, что два **условно получивших ключи учреждения** будут действовать сообща. LEAF содержит 32-битовый идентификатор модуля U, уникальный для каждой микросхемы Clipper. Оно также содержит текущий 80-битовый сеансовый ключ, зашифрованный уникальным ключом модуля микросхемы K_U, и 16-битовую контрольную сумму C, называемую идентификатором условного вручения. Контрольная сумма представляет собой функцию сеансового ключа, IV и возможно другой информации. Эти три поля шифруются фиксированным общим ключом K_F, общим для всех взаимодействующих микросхем Clipper. Общий ключ, используемые режимы шифрования, детали контрольной суммы и точная структура LEAF засекречены. Возможно это поле похоже на что-то подобное:

$$E_{K_F}(E_{K_U}(K_s, U, C))$$

K_U вводится в микросхемы Clipper при изготовлении. Этот ключ затем разделяется (см. раздел 3.5) и хранится в двух базах данных условно врученных ключей, охраняемых двумя различными учреждениями.

Чтобы Ева могла извлечь K_s из LEAF, она должна сначала расшифровать LEAF ключом K_F и получить U. Затем она должна получить постановление суда для каждого из учреждений условного вручения, каждое из которых возвращает половину K_U для данного U. Ева выполняет XOR обеих половин и получает K_U, затем она использует K_U для получения K_s, и K_s - для подслушивания разговора.

Контрольная сумма должна помешать нарушению этой схемы, принимающая микросхема Clipper не может выполнить дешифрирование, если контрольная сумма неправильна. Однако существует лишь 2^{16} возможных значений контрольной суммы, и фальшивое LEAF с правильной контрольной суммой, но неправильным ключом, может быть найдено примерно за 42 минуты [187]. Но это не очень поможет подслушать разговор, ведущийся с помощью Clipper. Так как протокол обмена ключами не является частью микросхемы Clipper, 42-минутное вскрытие грубой силой должно быть выполнено после обмена ключами, оно не может быть выполнено до телефонного звонка. Такое вскрытие может работать при передаче факсов или при использовании карточки Fortezza (см. раздел 24.17).

Предположительно микросхема Clipper должна противостоять инженерному вскрытию, выполненному "изощренным, хорошо" [1154], но по слухам в Sandia National Laboratories успешно провели исследование одной из микросхем. Даже если эти слухи ложны, я подозреваю, что самым крупным мировым производителям такое инженерное вскрытие вполне по силам, и его срок является только вопросом ресурсов и морали.

С этой темой связано множество вопросов о тайне личности. Многочисленные группы защиты гражданских свобод ведут активную компанию против любого механизма условного вручения ключей, который даст правительству право подслушивать граждан. Вся подлость в том, что, ходя эта схема никогда не проходила через Конгресс, NIST опубликовал EES в качестве FIPS [1153], обойдя болезненный законодательный процесс. Сейчас все выглядит, как если бы EES тихо и медленно умирал, но стандарты способны продолжать свою ползучую деятельность.

В 22-й перечислены различные организации, участвующие в этой программе. Как насчет идеи, чтобы оба учреждения условного вручения относились только к исполнительной ветви власти? Что вы скажете об учреждениях условного вручения, которые по сути ничего не знают о заявках на подслушивание и могут только слепо одобрять их? И что насчет идее о принятии правительством секретного алгоритма в качестве коммерческого стандарта?

Табл. 24-2.
Организации, участвующие в EES.

Министерство юстиции - Спонсор системы, владелец общего ключа
NIST - Руководство программой, хранитель условно врученной части ключа
FBI - Пользователь-дешифровщик, владелец общего ключа
Министерство финансов - Хранитель условно врученной части ключа
NSA - Разработчик программы

В любом случае, использование Clipper породит немало проблем при обращении в суд. Не забывайте, Clipper работает только в режиме OFB. Что бы вам иное не говорили, этот режим не обеспечивает целостности или проверке подлинности. Предположим, что Алиса предстала перед судом, и частью доказательств является телефонный разговор, зашифрованный микросхемой Clipper. Алиса утверждает, что она никогда не звонила, и голос - не ее. Алгоритм сжатия речи настолько плох, что опознать голос Алисы трудно, но обвинение утверждает, что, так как расшифровать разговор можно только с помощью условно врученного ключа Алисы, этот звонок был

сделан с ее телефона.

Алиса заявляет, что разговор был подделан в соответствии с [984, 1339]: даны шифротекст и открытый текст, объединив их с помощью XOR, можно получить ключевой поток. Затем этот ключевой поток можно объединить с помощью XOR с абсолютно другим открытым текстом, получая фальшивый шифротекст, который затем может быть преобразован в фальшивый открытый текст, который подается на дешифратор микросхемы. Правдив он или нет, этот довод может легко посеять сомнение в жюри присяжных, которые не сочтут телефонный разговор доказательством.

Другой способ вскрытия, называемый Втискиванием (Squeeze), позволяет Алисе выдать себя за Боба. Вот как это происходит [575]: Алиса звонит Бобу, используя Clipper. Она сохраняет копию его LEAF вместе с сеансовым ключом. Затем она звонит Кэрол (про которую известно, что ее подслушивают). При установке ключа Алиса делает сеансовый ключ идентичным тому, который она использовала для разговора с Бобом. Для этого потребуется взломать телефон, но это нетрудно. Затем вместо того, чтобы послать свое LEAF, она посылает LEAF Боба. Это правильное LEAF, поэтому телефон Кэрол ничего не заметит. Теперь она может говорить Кэрол все, что захочет - когда полиция расшифрует LEAF, она обнаружит, что оно принадлежит Бобу. Даже если Алисе не удастся выдать себя за Боба, ему придется доказывать свою невиновность в суде, что вполне может оправдать применение подобной схемы.

Органы охраны правопорядка Соединенных Штатов не должны тратить свое время, занимаясь сбором информации в уголовных расследованиях, которую нельзя использовать в суде. Даже если условное вручение ключей и являлось бы неплохой идеей, Clipper - это не лучший способ реализации этой идеи.

24.17 CAPSTONE

Capstone (известный также как MYK-80) - это другая разработанная NSA СБИС, реализующая Стандарт условного шифрования правительства США [1153]. Capstone реализует следующие функции [1155, 462]:

— Алгоритм Skipjack в любом из четырех основных режимов: ECB, CBC, CFB и OFB.

— Алгоритм обмена ключами (Key Exchange Algorithm, KEA) на базе открытых ключей, скорее всего Diffie-Hellman.

— Алгоритм цифровой подписи (Digital Signature Algorithm, DSA). *

— Алгоритм безопасного хэширования (Secure Hash Algorithm, SHA). j

— Алгоритм возведения в степень для общего назначения.

— Генератор случайных чисел с использованием истинно шумового источника.

Capstone обеспечивает криптографические возможности, необходимые для безопасной электронной торговли и других компьютерных приложений. Первым применением Capstone является карточка PCMCIA, названная Fortezza. (Сначала она называлась Tessera, пока на это не пожаловалась компания Tessera, Inc..)

NSA изучило возможность удлинения контрольной суммы LEAF в Capstone в версиях для карточек для того, чтобы помешать ранее рассмотренному вскрытию LEAF. Вместо этого была добавлена возможность выполнять перезапуск карточки после 10 неправильных LEAF. Меня это не впечатлило - время поиска правильного LEAF только на 10 процентов, до 46 минут.

24.18 Безопасный телефон AT&T MODEL 3600 TELEPHONE SECURITY DEVICE (TSD)

Безопасный телефон AT&T (Telephone Security Device, TSD) - это телефон с микросхемой Clipper. На самом деле существует четыре модели TSD. Одна содержит микросхему Clipper, другая - экспортируемый фирменный алгоритм шифрования AT&T третья - фирменный алгоритм для использования внутри страны плюс экспортируемый алгоритм, а четвертая включает Clipper, внутренний и экспортируемый алгоритмы.

Для каждого телефонного звонка TSD используют отличный сеансовый ключ. Пара TSD генерирует сеансовый ключ с помощью схемы обмена ключами Diffie-Hellman, независящей от микросхемы Clipper. Так как Diffie-Hellman не включает проверки подлинности, TSD использует два метода для предотвращения вскрытия "человек в середине".

Первым является экран. TSD хэширует сеансовый ключ и выводит хэш-значение на маленьком экране в в иде четырех шестнадцатиричных цифр. Собеседники проверяют, что на их экраны выведены одинаковые цифры. Качество голоса достаточно хорошо, чтобы они могли узнать друг друга по голосу.

Все же Ева может вскрыть эту схему. Пусть ей удалось вклиниться в линию между Бобом и Алисой. Она использует TSD на линии с Алисой и модифицированный TSD на линии с Бобом. Посередине она сопрягает два

телефонных звонка. Алиса пытается сделать разговор безопасным. Она обычным образом генерирует ключ, но общается с Евой, выдающей себя за Боба. Ева раскрывает ключ и с помощью модифицированного TSD делает так, чтобы ключ, который она сгенерировала для Боба, имел такое же хэш-значение. Это вскрытие на вид не очень реально, но для его предотвращения в TSD используется блокировка.

TSD генерирует случайные числа, используя источник шума и хаотичный усилитель с цифровой обратной связью. Он генерирует битовый поток, который пропускается через постотбеливающий фильтр на базе цифрового процессора.

Несмотря на все это в справочном руководстве TSD нет ни слова о безопасности. На самом деле там написано [70]:

> AT&T не гарантирует, что TSD защитит от вскрытия зашифрованной передачи правительственным учреждением, его агентами или третьей стороной. Более того, AT&T не гарантирует, что TSD защитит от вскрытия передаваемой информации с помощью методов, обходящих шифрование.

Глава 25
Политика

25.1 Агентство национальной безопасности (NSA)

NSA - это Агентство национальной безопасности (National Security Agency, когда-то расшифровывалось шутниками как "No Such Agency" (никакое агентство) или "Never Say Anything" (никогда ничего не скажу), но теперь они более открыты), официальный орган правительства США по вопросам безопасности. Агентство было создано в 1952 году президентом Гарри Труменом в подчинении Министерства безопасности, и многие годы в секрете хранилось сам факт его существования. NSA воспринималось как электронная разведка, в его задачи входило подслушивать и расшифровывать все иностранные линии связи в интересах Соединенных Штатов.

Следующие абзацы взяты из оригинального положения о NSA, подписанного в 1952 году президентом Труменом и рассекреченного спустя много лет [1535]:

> В задачи COMINT Агентства национальной безопасности (NSA) должны входить эффективные организация и управление разведывательной деятельности Соединенных Штатов в области телекоммуникаций, проводимой против иностранных правительств, чтобы обеспечить целостную действенную политику и соответствующие меры. Используемый в этой директиве термин "электронная разведка" ("communications intelligence") или "COMINT" обозначает все действия и методы, используемые для перехвата телекоммуникаций, исключая зарубежные прессу и радиовещание, и получения информации, предназначенной для приема другим получателем, но исключает цензуру, а также производство и распространение полученной разведывательной информации.
>
> Специальная природа действий COMINT требует, чтобы они во всех отношениях проводились отдельно от другой или общей разведывательной деятельности. Приказы, директивы, указания или рекомендации любого органа исполнительной власти, касающиеся сбора, получения, безопасности, обработки, распространения или использования разведывательной информации неприменимы в отношении действий COMINT, если это не оговорено особо, и документы не будут изданы компетентным представителем агентства, входящим в правительство. Другие директивы Национального совета безопасности директору ЦРУ и связанные директивы, изданные директором ЦРУ, не должны применяться к действиям COMINT, если это не будет специальная директива Национального совета безопасности, касающаяся COMINT.

NSA ведет исследования в области криптологии, занимаясь как разработкой безопасных алгоритмов для защиты коммуникаций Соединенных Штатов, так и криптоаналитические методы для прослушивания коммуникаций за пределами США research. Известно, что NSA является крупнейшим в мире работодателем для математиков. Оно также является крупнейшим в мире покупателем компьютерной аппаратуры. Возможно криптографический опыт NSA на много лет оторвался от состояния дел в открытой науке (в части алгоритмов, но вряд ли в части протоколов). Несомненно Агентство может взломать многие из используемых сегодня систем. Но, из соображений национальной безопасности, почти вся информация о NSA - даже ее бюджет - засекречена. (По слухам бюджет Агентства составляет около 13 миллиардов долларов в год - включая военное финансирование проектов NSA и оплату персонала - и, по слухам, в нем работает 16 тысяч человек.)

NSA использует свою власть, чтобы ограничить открытую доступность криптографии и помешать национальным врагам использовать слишком сильные методы шифрования, чтобы Агентство могло их взломать. Джеймс Массей (James Massey) анализирует эту борьбу между научными и военными исследованиями в криптографии [1007]:

> Если считать, что криптология является прерогативой правительства, то, конечно, большая часть криптологических исследований должна вестись за закрытыми дверями. Без всякого сомнения количество людей, занятых сегодня криптологическими исследованиями, намного больше, чем количество людей, работающих в открытой криптологии. Открытые криптологические исследования широко ведутся только последние 10 лет. Между этими двумя исследовательскими сообществами были и будут конфликты. Открытые исследования представляют собой обычный поиск знания, для которого жизненно важен открытый обмен идеями с помощью конференций, презентаций и публикаций в научных журналах. Но может ли правительственная организация, ответственная за вскрытие шифров других государств, приветствовать публикацию шифра, который нельзя взломать? Может ли исследователь с чистой совестью публиковать подобный шифр, которые может свести на нет все усилия взломщиков кода, находящихся на службе его правительства? Можно настаивать, что публикация доказано безопасного шифра заставит все правительства вести себя подобно "джентльменам" Стимсона, но необходимо помнить, что открытые исследования в криптографии полны политических и этических мотивов гораздо более серьезных, чем во многих других областях науки. Удивляться надо не тому, что правительственные организации на почве криптологии конфликтуют с независимыми исследователями, а тому, что эти конфликты (по крайней мере те, о которых нам известно) так незначительны и так сглажены.

Джеймс Бэмфорд (James Bamford) написал увлекательную книгу про NSA: *The Puzzle Palace* [79], (*Дворец головоломок*), недавно доработанную вместе с Вэйной Медсен (Wayne Madsen) [80].

Коммерческая программа сертификации компьютерной безопасности

Коммерческая программа сертификации компьютерной безопасности (Commercial COMSEC Endorsement Program (CCEP)), кодовое имя Overtake, - это предложение, сделанное NSA в 1984 году и призванное облегчить разработку компьютеров и средств связи с встроенными криптографическими возможностями [85, 1165]. Обычно всю разработку таких изделий оплачивали военные, и это обходилось им недешево. NSA посчитало, что если компании могут продавать аппаратуру и армии, и корпорациям, даже иностранным, это позволит уменьшить расходы к всеобщей выгоде. Агентству больше не требовалось бы проверять совместимость обору-

дования с Федеральным стандартом 102.7, и затем CCEP предоставила бы доступ к одобренному правительством криптографическому оборудованию [419].

NSA разработало ряд криптографических модулей различного назначения. В этих модулях для различных приложений используются различные алгоритмы, и производители получают возможность извлечь один модуль и вставить другой в зависимости от желаний клиента. Существуют модули для военного использования (Тип I), модули для "несекретного, но важного" правительственного использования (Тип II), модули для корпоративного использования (Тип III) и модули для экспортирования (Тип IV). Различные модули, их применение и названия сведены в 24-й.

Табл. 25-1.
Модули CCEP

Применение	Тип I	Тип II
Речь/низкоскоростная передача данных	Winster	Edgeshot
Компьютер	Tepache	Bulletproof
Высокоскоростная передача данных	Foresee	Brushstroke
Следующее поколение	Countersign I	Countersign II

Эта программа все еще действует, но она не вызвала энтузиазма ни у кого кроме правительства. Все модули были защищены от вскрытия, все алгоритмы были засекречены, а пользователи должны были получать ключи от NSA. Корпорации никогда реально не верили в идею использования секретных алгоритмов, навязанных правительством. Казалось бы, NSA получило заметный урок, чтобы больше не докучать применением Clipper, Skipjack и микросхем шифрования с условным вручением ключей.

25.2 Национальный центр компьютерной безопасности (NCSC)

Национальный центр компьютерной безопасности (National Computer Security Center, NCSC), отделение NSA, отвечает за доверенную правительственную компьютерную программу. В настоящее время центр проводит оценку продуктов компьютерной безопасности (программных и аппаратных), финансирует исследования и публикует их резаультаты, разрабатывает технические руководства и обеспечивает общую поддержку и обучение.

NCSC издает скандально известную "Оранжевую книгу" [465]. Ее настоящее название - *Department of Defense Trusted Computer System Evaluation Criteria* (Критерии оценки департамента оборонных доверенных компьютерных систем), но это так трудно выговаривать, и к тому же у книги оранжевая обложка. Оранжевая книга пытается определить требования к безопасности, дает производителям компьютеров объективный способ измерить безопасность их систем и указывает им, что необходимо встраивать в безопасные продукты. Книга посвящена компьютерной безопасности, о криптографии в ней по сути говорится не очень много.

Оранжевая книга определяет четыре широких категории защиты безопасности. В ней также определяются классы защиты внутри некоторых из этих категорий. Они сведены в 23-й.

Табл. 25-2.
Классификация Оранжевой книги

D: Minimal Security (Минимальная безопасность)
C: Discretionary Protection (Защита по усмотрению)
    C1: Discretionary Security Protection (Защита безопасности по усмотрению)
    C2: Controlled Access Protection (Защита управляемого доступа)
B: Обязательная защита
    B1: Labeled Security Protection
    B2: Structured Protection (Структурная защита)
    B3: Security Domains (Области безопасности)
A: Verified Protection (Достоверная защита)
    A1: Verified Design (Достоверная разработка)

Иногда производители любят говорить "мы обеспечиваем безопасность C2". В виду они имеют классификацию Оранжевой книги. За более подробной информацией обращайтесь к [1365]. Модель компьютерной безопасности, используемая в этих критериях, называется моделью Bell-LaPadula [100, 101, 102, 103].

NCSC издал целую серию книг по компьютерной безопасности, иногда называемую Радугой книг (все обложки имеют различные цвета). Например, *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria* [1146] (Интерпретация критериев оценки доверенных компьютерных систем в отношении

доверенных сетей), иногда называемая Красной книгой, толкует положения Оранжевой книги по отношению к сетям и сетевому оборудованию. *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* [1147] (Интерпретация критериев оценки доверенных компьютерных систем в отношении систем управления базами данных) - я даже не пытаюсь описать цвет обложки - делает то же самое для баз данных. Сегодня существует свыше 30 таких книг, цвет обложек некоторых из них отвратителен.

За полным комплектом Радуги книг обращайтесь по адресу Director, National Security Agency, INFOSEC Awareness, Attention: C81, 9800 Savage Road, Fort George G. Meade, MD 2,0755-6000; (301) 766-8729. Не говорите им, что вас послал я.

25.3 Национальный институт стандартов и техники

NIST - это Национальный институт стандартов и техники (National Institute of Standards and Technology), подразделение Министерства торговли США. Ранее он назывался Национальным бюро стандартов (NBS, National Bureau of Standards) и изменил имя в 1988 году. С помощью своей Лаборатории компьютерных систем (Computer Systems Laboratory, CSL), NIST продвигал открытые стандарты взаимодействия, которые, как он надеялся, ускорят развитие основанных на компьютерах отраслях промышленности. К настоящему времени NIST выпустил стандарты и руководства, которые, как он считает, будут приняты всеми компьютерными системами Соединенных Штатов. Официальные стандарты опубликованы как издания FIPS (Федеральные стандарты обработки информации.

Если вам нужны копии любого из FIPS (или других изданий NIST), свяжитесь с Национальной службой технической информации Министерства торговли США - National Technical Information Service (NTIS), U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161; (703) 487-4650; или посетите gopher://csrc.ncsl.nist.go*

Когда в 1987 году Конгресс принял Акт о компьютерной безопасности (Computer Security Act), NIST был уполномочен определять стандарты, обеспечивающие безопасность важной, но не секретной информации в правительственных компьютерных. (Секретная информация и данные Предупреждающей поправки находятся в сфере юрисдикции NSA.) Акт разрешает NIST в ходе оценки предлагаемых технических стандартов сотрудничать с другими правительственными организациями и частными предприятиями.

NIST издает стандарты криптографических функций. Организации правительства США обязаны использовать их для важной, но несекретной информации. Часто эти стандарты принимаются и частным сектором. NIST выпустил DES, DSS, SHS и EES.

Все эти алгоритмы разработаны с некоторой помощью NSA, начиная от анализа DES до проектирования DSS, SHS и алгоритма Skipjack в EES. Некоторые критикуют NIST за то, что NSA в большой степени может контролировать эти стандарты, хотя интересы NSA могут не совпадать с интересами NIST. Неясно, как действительно NSA может повлиять на проектирование и разработку алгоритмов. Но при ограничениях на персонал, бюджет и ресурсы NIST привлечение NSA кажется разумным. NSA обладает большими возможностями, включая лучшую в мире компьютерные средства.

Официальный "Меморандум о взаимопонимании" ("Memorandum of Understanding", MOU) между двумя организациями гласит:

> МЕМОРАНДУМ О ВЗАИМОПОНИМАНИИ МЕЖДУ ДИРЕКТОРОМ НАЦИОНАЛЬНОГО ИНСТИТУТА СТАНДАРТОВ И ТЕХНИКИ И ДИРЕКТОРОМ АГЕНТСТВА НАЦИОНАЛЬНОЙ БЕЗОПАСНОСТИ ОТНОСИТЕЛЬНО ПРИМЕНЕНИЯ ПУБЛИЧНОГО ЗАКОНА 100-235
>
> Сознавая, что:
>
> A. В соответствии с разделом 2 Акта о компьютерной безопасности от 1987 года (Публичный закон 100-235), (Акт), на Национальный институт стандартов и техники (NIST) как часть Федерального правительства возлагается ответственность за:
>
> 1. Разработку технических, административных, физических стандартов, стандартов управления и руководств для рентабельных безопасности и защищенности важной информации Федеральных компьютерных систем, определенных в Акте; и,
>
> 2. Разработку руководств по технической безопасности соответствующих компьютерных систем Агентства национальной безопасности (NSA).
>
> B. В соответствии с разделом 2 Акта NIST обязан работать в тесном взаимодействии с другими организациями, включая NSA, обеспечивая:
>
> 1. Максимальное использование всех существующих и планируемых программ, материалов, исследований и отчетов, касающихся безопасности и защищенности компьютерных систем, чтобы избежать недужного и дорогого дублирования работ; и,
>
> 2. Эти стандарты, разработанные NIST в соответствии с Актом, в максимально возможной степени должны быть согласованы и совместимы со стандартами и процедурами, разработанными для защиты секретной информации в Федеральных компьютерных системах.
>
> C. В соответствии с Актом в обязанности Министра торговли, которые он перепоручает NIST, входит назначение членов Консультативного комитета по безопасности и защищенности компьютерных систем (Computer System Security and Privacy Advisory Board), по крайней мере члена, представляющего NSA.
>
> Следовательно, для обеспечения целей данного MOU Директор NIST и Директор NSA настоящим признают следующее:
>
> I. NIST будет:
>
> 1. Назначать в Консультативный комитет по безопасности и защищенности компьютерных систем по крайней мере одного представителя, замещающего Директора NSA.

2. Опираться на разработанные NSA руководства по технической безопасности компьютерных систем до той степени, в которой NIST определяет, что эти руководства отвечают требованиям, предъявляемым к защите важной информации в Федеральных компьютерных системах.

3. Признавать сертифицированный NSA рейтинг доверенных систем в соответствии с Программой критериев оценки безопасности доверенных компьютеров без дополнительной экспертизы.

4. Разрабатывать стандарты безопасности телекоммуникаций для защиты важных несекретных компьютерных данных, максимально опираясь на результаты экспертизы и разработки Агентства национальной безопасности, чтобы выполнять эти обязанности своевременно и эффективно.

5. По возможности избегать дублирования, разграничив совместные работы с NSA для получения помощи NSA.

6. Запрашивать помощи NSA по всем вопросам, связанным с криптографическими алгоритмами и криптографическими методами, включая исследования, оценку разработки, одобрение, но не ограничиваясь этими действиями.

II. NSA будет:

1. Обеспечивать NIST техническими руководствами по доверенным технологиям, безопасности телекоммуникаций и идентификации личности, которые могут быть использованы в рентабельных системах защиты важных компьютерных данных.

2. Проводить или инициировать исследовательские и проектные программы по доверенным технологиям, безопасности телекоммуникаций, криптографическим методам и методам идентификации личности.

3. По просьбам NIST оказывать помощь в отношении всех вопросов, связанных с криптографическими алгоритмами и криптографическими методами, включая исследования, оценку разработки, одобрение, но не ограничиваясь этими действиями.

4. Устанавливать стандарты и одобрять изделия для применения в безопасных системах, охватываемых 10 USC раздел 2315 (Поправка Уорнера).

5. По требованию федеральных организаций, их подрядчиков и других финансируемых правительством субъектов проводить оценку возможности вражеской разведывательной деятельности в отношении федеральных информационных систем, а также обеспечивать техническое содействие и рекомендовать изделия, одобренные для применения в безопасных системах, чтобы противостоять такой угрозе.

III. NIST и NSA будут:

1. Координировать свои планы по обеспечению безопасности и защищенности компьютерных систем, за которые NIST и NSA несут ответственность в соответствии с разделом 6(b) Акта.

2. Обмениваться техническими стандартами и руководствами, если это необходимо для достижения целей Акта.

3. Совместно работать над достижением целей этого меморандума с максимальной эффективностью, избегая ненужного дублирования усилий.

4. Поддерживать непрерывный диалог, гарантирующий, что каждая из организаций будет находиться на одинаковом уровне современных технологий и вопросов, влияющих на безопасность автоматизированных информационных компьютерных систем.

5. Организовывать техническую рабочую группу для обзора и анализа областей совместных интересов, касающихся защиты систем, обрабатывающих важную или другую несекретную информацию. Эта Группа будет состоять из шести федеральных служащих, по трое от NIST и NSA, и при необходимости может быть увеличена за счет представителей других организаций. Темы работы группы могут определяться либо заместителем директора NSA по информационной безопасности, либо заместителем директора NIST, либо могут инициироваться самой группой с последующим одобрением заместителем директора NSA по информационной безопасности или заместителем директора NIST. В течение нескольких дней после постановки перед Группой вопроса либо заместителем директора NSA по информационной безопасности, либо заместителем директора NIST Группа должна представить отчет о выполнении работ по этому вопросу и, при необходимости, план дальнейшего анализа.

6. На ежегодной основе обмениваться планами работы по всем исследовательским и конструкторским проектам, связанным с защитой систем, обрабатывающих важную или другую несекретную информацию, включая доверенные технологии, защиту целостности и доступности данных, безопасности телекоммуникаций и методов идентификации личности. Обмен информацией по проектам должен происходить ежеквартально, и обзор состояния проектов должен любой из сторон предоставляться по запросу другой стороны.

7. Проверять обзоры технической рабочей группы до опубликования всех вопросов, касающихся техники обеспечения безопасности систем, разрабатываемых для использования при защите важной информации в федеральных компьютерных системах, чтобы гарантировать совместимость раскрытия этих тем с национальной безопасностью Соединенных Штатов. Если NIST и NSA не смогут решить подобный вопрос в течение 60 дней, любая из организаций может поднять этот вопрос перед Министром обороны и Министром торговли. Признается, что данный вопрос с помощью NSC может быть передан для решения Президенту. Никакие действия не должны предприниматься до окончательного решения в опроса.

8. Определять дополнительные рабочие соглашения, заключенные между NSA и NIST, как приложения к этому MOU.

IV. Любая из сторон может прекратить действие этого MOU письменным уведомлением, направленным за шесть месяцев до прекращения действия. Этот MOU считается действительным при наличии обеих подписей.

/подписано/

РЭЙМОНД. ДЖ. КАММЕР
Исполнительный Директор, Национальный институт стандартов и техники, 24 марта 1989 года

У. О. СТЬЮДМЕН
Вице-адмирал, ВМС США, Директор, Агентство национальной безопасности, 23 марта 1989 года

25.4 RSA Data Security, Inc.

RSA Data Security, Inc. (RSADSI) была основана в 1982 году для разработки, лицензирования и коммерческого использования патента RSA. У компании есть ряд коммерческих продуктов, включая отдельный пакет безопасности электронной почты, и различные криптографические библиотеки (доступные в виде исходных текстов или объектного кода). RSADSI также предлагает на рынке симметричные алгоритмы RC2 и RC4 (см. раздел 11.8). RSA Laboratories, исследовательская лаборатория, связанная с RSADSI, выполняет фундаментальные криптографические исследования и оказывает консультационные услуги.

При заинтересованности в лицензиях или продуктах нужно обращаться к директору по продажам (Director of Sales, RSA Data Security, Inc., 100 Marine Parkway, Redwood City, CA 94065; (415) 595-8782; факс: (415) 595-1873).

25.5 PUBLIC KEY PARTNERS

Пять патентов, перечисленных в 22-й, принадлежат Public Key Partners (PKP) из Саннивэйла (Sunnyvale), Калифорния, партнерству RSADSI и Care-Kahn, Inc. - родительской компании Cylink. (RSADSI получает 65 процентов прибыли, а Care-Kahn 35 процентов.) PKP утверждает, что эти патенты и 4218582 особенно применимы ко *всем способам использования* криптографии с открытыми ключами.

Табл. 25-3.
Патенты Public Key Partners

№ патента	Дата	Изобретатели	Название патента
4200770	29.3.80	Hellman, Diffie, Merkle	Обмен ключами Diffie-Hellman
4218582	19.8.80	Hellman, Merkle	Рюкзаки Merkle-Hellman
4405829	20.9.83	Rivest, Shamir, Adleman	RSA
4424414	3.3.84	Hellman, Pohlig	Pohlig-Hellman
4995082	19.2.91	Schnorr	Подписи Schnorr

В [574], PKP писала:

> Эти патенты [4200770, 4218582, 4405829 и 4424414] охватывают все известные методы использования искусства открытых ключей, включая варианты, обобщенно известные как ElGamal.
>
> Благодаря широкому распространению цифровых подписей RSA в международном сообществе Public Key Partners решительно одобряет их включение в стандарт цифровой подписи. Мы заверяем все заинтересованные стороны, что Public Key Partners подчинится всем решениям ANSI и IEEE, касающимся доступности лицензирования этого искусства. Особенно для поддержки любых принимаемых стандартов, использующих цифровую подпись RSA. Public Key Partners настоящим заверяет, что лицензии на использование подписей RSA будут предоставляться в разумные сроки, на разумных условиях и без какой-либо дискриминации.

Правда ли это, зависит от того, с кем вы говорите. Лицензии PKP, как правило, секретны, поэтому способа проверить, отличается ли данная лицензия от других, не существует. Хотя компания утверждает, что никому не отказала в выдаче лицензии, по крайней мере две компании говорят о том, что им лицензия выдана не была. PKP тщательно охраняет свои патенты, угрожая всем, кто использует без лицензирования криптографию с открытыми ключами. Частично это реакция на патентное законодательство США. Если владельцу патента не удается наказать нарушителя патента, он может потерять свой патент. Было много разговоров о законности этих патентов, но дальше разговоров дело не пошло. Все законные претензии к патентам PKP были урегулированы до суда.

Я не собираюсь в этой книге давать юридические советы. Может быть патент RSA не устоит перед судом. Может быть эти патенты не применимы ко всей криптографии с открытыми ключами. (Честно говоря, я не понимаю, как они охватывают ElGamal или криптосистемы с эллиптическими кривыми.) Может кому-то удастся выиграть процесс против PKP или RSADSI. Но не забывайте, что корпорации с огромными юридическими отделами, например, IBM, Microsoft, Lotus, Apple, Novell, Digital, National Semiconductor, AT&T и Sun, лицензировали RSA для использования в своих продуктах, а не обращались в суд. Boeing, Shell Oil, DuPont, Raytheon и Citicorp - все лицензировали RSA для своего внутреннего использования.

В одном случае PKP возбудило процесс против TRW Corporation по поводу использования без лицензирования алгоритма ElGamal. TRW утверждала, что ей не нужна лицензия. PKP и TRW достигли соглашения в июне 1992. Подробности урегулирования конфликта неизвестны, но среди них - согласие TRW получить лицензию на патенты. Это не предвещает ничего хорошего. TRW могла позволить себе хороших юристов. Я могу только предположить, что, если бы TRW была уверена, что сможет выиграть процесс, не потратив невероятного количества денег, она бы не отказалась от борьбы.

Тем не менее в PKP существуют свои внутренние проблемы. В июне 1994 года Care-Kahn подала в суд на RSADSI, заявив, среди всего остального, что патент RSA неправилен и неприменим [401]. Оба партнера попытались разорвать свое партнерство. Законны патенты или нет? Нужно ли будет пользователям получать лицензию от Care-Kahn, чтобы пользоваться алгоритмом RSA? Кому будет принадлежать патент Schnorr? Возможно это дело будет урегулировано к моменту выхода этой книги.

Патенты действительны лишь в течение Patents 17 лет и не могут быть возобновлены. 29 марта 1997 года обмен ключами Diffie-Hellman (и алгоритм ElGamal) станут общедоступными. 20 сентября 2000 года станет общедоступным и RSA. Пометьте на своих календарях.

25.6 Международная ассоциация криптологических исследований

Международная ассоциация криптологических исследований (International Association for Cryptologic Research, IACR) - это всемирная криптографическая исследовательская организация. Ее целью является развитие теории и практики криптологии и связанных областей. Ее членом может стать любой. Ассоциация выступает

спонсором двух ежегодных конференций, Crypto (проводится в августе в Санта-Барбаре) и Eurocrypt (проводится в в Европе), и ежеквартально издает *The Journal of Cryptology* и *IACR Newsletter*.

Адрес штаб-квартиры IACR меняется вместе со сменой президента. Текущий адрес: IACR Business Office, Aarhus Science Park, Custav Wieds Vej 10, DK-8000 Aarhus C, Denmark.

25.7 Оценка примитивов целостности RACE (RIPE)

Программа исследования и развития передовых средств связи в Европе (Research and Development in Advanced Communication Technologies in Europe, RACE) была инициирована Европейским сообществом для поддержки предварительной проработки телекоммуникационных стандартов и технологий, поддерживающих Интегрированные высокоскоростные средства связи (Integrated Broadband Communication, IBC). В качестве части этой работы RACE учредило консорциум для Оценки примитивов целостности RACE (RACE Integrity Primitives Evaluation, RIPE), чтобы собрать в одно целое пакет технологий, соответствующих возможным требованиям к безопасности IBC.

Консорциум RIPE образовали шесть ведущих европейских криптографических исследовательских групп: Центр по математике и компьютерным наукам (Center for Mathematics and Computer Science), Амстердам; Siemens AG; Philips Crypto BV; Royal PTT Nederland NV, PTT Research; Katholieke Univesiteit Leuven и Aarhus Universitet. После объявлений о приеме алгоритмов в 1989 и 1991 годах [1564], подачи 32 заявок, присланных со всего мира, и собственно оценивающего проекта длительностью 350 человеко-месяцев, консорциум опубликовал *RIPE Integrity Primitives* [1305, 1332]. Отчет содержит введение, несколько основных концепций целостности и их примитивы: MDC-4 (см. раздел 14.11), RIPE-MD (см. раздел 14.8), RIPE-MAG (см. раздел 14.14), IBC-HASH, SKID (см. раздел 3.2), RSA, COMSET (см. раздел 16.1) и генерацию ключей RSA.

25.8 Условный доступ для Европы (CAFE)

Условный доступ для Европы (Conditional Access for Europe, CAFE) - это проект в рамках программы ESPRIT Европейского сообщества [204, 205]. Работа началась в декабре 1992 года и по плану должна закончиться к концу 1995 года. Образованный консорциум состоит из груп социальных исследований и исследований рынка (Cardware, Institut fur Sozialforschung), изготовителей программного обеспечения и аппаратуры (DigiCash, Cemplus, Ingenico, Siemens), а также криптографов (CWI Amsterdam, PTT Research Netherlands, SPET, Sintef Delab Trondheim, Universities of Arhus, Hildesheim and Leuven).

Целью проекта является разработка системы условного доступа, особенно для цифровых платежных систем. Платежные системы должны обеспечивать надежность для каждого пользователя и требовать как можно меньше веры в себя - надежность не должна зависеть от устойчивости устройств к взлому.

Основным устройством CAFE служит электронный бумажник: маленький компьютер, очень похожий на карманный калькулятор. У него есть батарейка, клавиатура, экран и инфракрасный канал для связи с другими бумажниками. У каждого пользователя свой собственный бумажник, который обеспечивает его права и гарантирует его безопасность.

У устройства с клавиатурой и экраном есть определенное преимущество перед интеллектуальной картой - оно может работать независимо от терминала. Пользователь может непосредственно ввести свой пароль и сумму платежа. Отличие от кредитной карты пользователю не нужно отдавать свой бумажник кому-то, чтобы выполнить транзакцию. Дополнительными возможностями являются:

— Автономные транзакции. Система предназначена для замены обращения небольших сумм наличных, диалоговая система была бы слишком громоздка.

— Устойчивость к потерям. Если пользователь потеряет свой бумажник, или бумажник сломается, или его украдут, пользователь не потеряет свои деньги.

— Поддержка различных валют.

— Открытая архитектура и открытая система. Пользователь должен иметь возможность заплатить за произвольные услуги, например, покупки в магазине, телефон, общественный транспорт, предоставляемые различными поставщиками. Система должна обеспечивать взаимодействие любого количества эмитентов электронных денег, а также взаимодействие бумажников различных типов и производителей.

— Низкая стоимость.

К моменту написания этой книги существует только программная версия системы, и консорциум плотно работает над аппаратным прототипом.

25.9 ISO/IEC 9979

В середине 80-х ISO стандартизировать DES, который уже использовался в качестве FIPS и стандарта ANSI. После некоторой политической возни ISO решило не стандартизировать криптографические алгоритмы, а регистрировать их. Зарегистрировать можно только алгоритмы шифрования, регистрировать хэш-функции и схемы подписи нельзя. Зарегистрировать алгоритм может любая национальная организация.

В настоящее время поданы заявки на регистрацию трех алгоритмов (см. 21-й). Подача заявки включает информацию об использовании, параметрах, реализациях, режимах и тестовых векторах. Подробное описание необязательно, можно подавать на регистрацию и секретные алгоритмы.

Факт регистрации алгоритма ничего не говорит о его качестве. Регистрация не является и одобрением алгоритма ISO/IEC, она просто показывает, что одна из национальных организаций хочет зарегистрировать алгоритм, независимо от критериев, используемых данной организацией.

Меня не впечатлила эта идея. Регистрация мешает процессу стандартизации. Вместо того, чтобы принять несколько алгоритмов, ISO регистрирует любой алгоритм. При таком контроле можно зарегистрировать все, что угодно, и далее с полным правом сопровождать свой алгоритм звучной добавкой "Зарегистрирован ISO/IEC 9979 ". В любом случае реестр ведет National Computer Centre Ltd., Oxford Road, Manchester, MI 7ED, United Kingdom.

Табл. 25-4.
Зарегистрированные алгоритмы ISO/IEC 9979

Регистрационный номер	Название
0001	B-CRYPT
0002	IDEA
0003	LUC

25.10 Профессиональные и промышленные группы, а также группы защитников гражданских свобод

Информационный центр по электронной тайне личности (EPIC)

Информационный центр по электронной тайне личности (Electronic Privacy Information Center, EPIC) был учрежден в 1994 году для привлечения общественного внимания к возникающим вопросам тайн личности, связанным с Национальной информационной инфраструктурой, таких как микросхемы Clipper, предложения по цифровой телефонии, национальные системы идентификационных номеров, тайны историй болезни и продажа сведений о потребителях. EPIC ведет судебные процессы, спонсирует конференции, публикует отчеты, издает *EPIC Alert* и проводит кампании по вопросам тайны личности. Желающие присоединиться могут обратиться по адресу Anyone interested in joining should contact Electronic Privacy Information Center, 666 Pennsylvania Avenue SE, Suite 301, Washington, D.C. 20003 (202,) 544-9240; факс: (202) 547-5482; Internet: info@epic.org.

Фонд электронного фронтира (EFF)

Фонд электронного фронтира (Electronic Frontier Foundation, EFF) посвятил себя защите гражданских прав в киберпространстве. Рассматривая криптографическую политику США, EFF считает, что информация и доступ к криптографии являются фундаментальными правами, и поэтому с них должны быть сняты правительственные ограничения. Фонд организовал рабочую группу по цифровой безопасности и тайне личности (Digital Privacy and Security Working Croup), которая является коалицией 50 организаций. Группа противодействует закону о цифровой телефонии и инициативе Clipper. EFF также содействует ведению процессов против контроля за экспортом криптографии [143]. Желающие присоединиться к EFF могут связаться с Electronic Frontier Foundation, 1001 C Street NW, Suite 950E, Washington, D.C. 20001; (202) 347 5400, факс: (202) 393-5509; Internet: eff@eff.org.

Ассоциация по вычислительной технике (ACM)

Ассоциация по вычислительной технике (Association for Computing Machinery, ACM) - это международная компьютерная промышленная организация. В 1994 году Комитет общественной политики ACM США представил прекрасный отчет о криптографической политике США [935]. Его стоит прочитать каждому, кто интересуется политикой в криптографии. Его можно получить с помощью анонимного ftp с info.acm.org в /reports/acm. crypt_study/acm_crypto_study.ps.

Институт инженеров по электричеству и радиоэлектронике (IEEE)

Институт инженеров по электричеству и радиоэлектронике (Institute of Electrical and Electronics Engineers , IEEE) - это другая профессиональная организация . Отделение в США изучает вопросы, связанные с тайной личности, включая криптографическую политику, идентификационные номера, и защита тайн в Internet, и разрабатывает соответствующие рекомендации .

Ассоциация производителей программного обеспечения (SPA)

Ассоциация производителей программного обеспечения (Software Publishers Association , SPA) - это торговая ассоциация, в которую входят свыше 1000 компаний, разрабатывающих программное обеспечение для перс о-нальных компаний. Они выступают за ослабление экспортного контроля в криптографии и поддерживают пер е-чень коммерчески доступных зарубежных продуктов .

25.11 Sci.crypt

Sci.crypt - это телеконференция Usenet по криптологии. Ее читают примерно 100000 человек по всему миру . Большинство сообщений - обычная чепуха, перебранка ли и то, и другое одновременно. Некоторые сообщения касаются политики, а большинство остальных - просьбы предоставить сведения или общие . Иногда в этой телеконференции случайно попадаются различные самородки и некоторая полезная информация . Если читать sci.crypt регулярно, можно узнать, как использовать нечто, называемое файлом-убийцей .

Другой телеконференцией Usenet является sci.crypt.research, более умеренная телеконференция, посвященная обсуждению криптологических исследований . В ней меньше сообщений, и они гораздо интереснее .

25.12 Шифропанки

Шифропанки (Cypherpunks) - это неформальная группа людей, заинтересованных в обучении и изучении криптографии. Они также экспериментируют с криптографией, пытаясь ввести ее в обиход . По их мнению все криптографические исследования не принесли обществу ничего хорошего, так как оно не воспользовалось до с-тижениями криптографии.

В "Манифесте шифропанков" Эрик Хьюз (Eric Hughes) пишет [744]:

> Мы, Шифропанки, стремимся создать анонимные системы . Мы защищаем наши тайны с помощью криптографии, с п о-мощью систем анонимной отправки почты, с помощью цифровых подписей и электронных денег .
>
> Шифропанки пишут код. Мы знаем, что кто-то должен написать программное обеспечение, защищающее тайны личн о-сти, и так как пока это не сделано, мы не сможем обеспечить сохранение своих тайн, мы собираемся написать такие програ м-мы. Мы публикуем наш код, чтобы наши друзья Шифропанки могли попрактиковаться и поиграть с ним . Наш код свободно может использовать кто угодно и где угодно . Нас не очень волнует, нравятся ли вам программы, которые мы пишем . Мы зна-ем, что программное обеспечение невозможно разрушить, и что невозможно прекратить работу рассеянных систем .

Те, кто хочет присоединиться к списку рассылки шифропанков в Internet, должны отправлять почту в адрес majordomo@toad.com. Список рассылки хранится на ftp.csua.berkeley.edu в /pub/cypherpunks.

25.13 Патенты

Вопрос о программных патентах невозможно втиснуть в рамки этой книги . Хороши они или нет, они суще-ствуют. В Соединенных Штатах можно патентовать алгоритмы , в том числе и криптографические . IBM владеет патентами DES [514]. IDEA запатентован. Запатентованы почти все алгоритмы с открытыми ключами . NIST даже запатентовал DSA. Действие ряда криптографических патентов было блокировано вмешательством NSA, в соответствии с Актом о секретности изобретений (Invention Secrecy Act) от 1940 года и Актом о национальной безопасности (National Security Act) от 1947 года. Это означает, что вместо патента изобретатель получает се к-ретное постановление, и ему запрещается обсуждать его изобретение с кем-нибудь еще .

У NSA есть особые возможности при патентовании . Агентство может обратиться за патентом и затем бл о-кировать его выдачу . Снова появляется секретное постановление, но теперь NSA одновременно и изобретатель, и издатель постановления . Когда спустя некоторое время секретное постановление отменяется , регистрационная контора выдает патент, действующий стандартные 17 лет years. Это более явно защищает изобретение, чем хранение его в секрете . Если кому-нибудь удастся изобрести то же самое , NSA уже подало заявку на патент . Если никому другому не удастся изобрести то же самое, изобретение остается се кретным.

Несмотря на то, что процесс патентования должен не только защищать изобретения, но и раскрывать их, благодаря этой уловке NSA может держать патент дольше 17 лет. Отсчет 17-летнего срока начинается с моме н-та выдачи патента, а не подачи заявки . Пока неясно, как все может измениться в связи с ратификацией договора о GATT Соединенными Штатами .

25.14 Экспортное законодательство США

Согласно правительству США криптография относится к военному снаряжению. Это означает, что криптография подчиняется тем же законам, что и ракета TOW или танк M1 Абрамс. Если вы продаете криптографический продукт без соответствующей экспортной лицензии, то вы - международный контрабандист оружием. Если вы не хотите испортить ваше резюме строкой о пребывании в федеральной тюрьме, обратите внимание на законодательство.

С началом в 1949 году холодной войны все страны НАТО (кроме Исландии), а затем Австралия, Япония и Испания, образовали КОКОМ - Координационный комитет для многостороннего контроля за экспортом (CoCom, Coordinating Committee for Multilateral Export Controls). Это неофициальная организация, призванная координировать национальные ограничения, касающиеся экспорта важных военных технологий в Советский Союз, другие страны Варшавского Договора и Китайскую Народную Республику. Примерами контролируемых технологий являются компьютеры, станки для металлопроката и криптография. Целью этой организации являлось замедление передачи технологий в указанные страны, и сдерживание, таким образом, их военного потенциала.

С концом холодной войны страны КОКОМ осознали, что выполняемый ими контроль большей частью устарел. В настоящее время, по видимому, идет процесс формирования "Нового форума", другой международной организации, которая собирается остановить поток военных технологий в страны, которые не нравятся членам организации.

В любом случае экспортная политика США в отношении стратегических товаров регулируется Правительственным актом об экспорте (Export Administration Act), Актом о контроле над экспортом вооружения (Arms Export Control Act), Актом об атомной энергии (Atomic Energy Act) и Актом о нераспространении ядерных вооружений (Nuclear Non-Proliferation Act). Контроль, установленный этим законодательством, реализуется с помощью многих подзаконных актов, ни один из них не координирует другой. Свыше дюжины организаций, включая военные службы, осуществляют контроль, часто их деятельность перекрывается и конфликтует.

Подконтрольные технологии фигурируют в нескольких списках. Криптография, по традиции относящаяся к вооружению, появляется в Перечне вооружений США (U.S. Munitions List, USML), Международном перечне вооружений (International Munitions List, IML), Перечне контроля за торговлей (Commerce Control List, CCL) и Международном промышленном перечне (International Industrial List, IIL). Госдепартамент отвечает за USML, он публикуется как часть Регулирования международного трафика оружия (International Traffic in Arms Regulations, ITAR) [466, 467].

Экспорт криптографии в США контролируется двумя правительственными организациями. Одной является Комитет по управлению экспортом (Bureau of Export Administration, BXA) в Министерстве торговли, уполномоченный Правилами регулирования экспорта (Export Administration Regulations, EAR). Другая - это Управление по регулированию продажи средств обороны (Office of Defense Trade Controls, DTC) в Государственном департаменте, уполномоченное ITAR. По опыту требования BXA из Министерства торговли менее строги, но сначала весь криптографический экспорт просматривается DTC из Госдепартамента (которое получает советы по технике и национальной безопасности от NSA и, кажется, всегда следует этим советам), которое может отказать передать право решения BXA.

ITAR регулирует этот процесс. (До 1990 года Управление DTC называлось Управлением по контролю над вооружением, возможно, эти усилия в области "паблик рилейшнз" направлены на то, чтобы мы забыли, что мы имеем дело с бомбами и пушками.) Исторически DTC сопротивлялось выдаче экспортных лицензий на средства шифрования сильнее определенного уровня - хотя о том, каков этот уровень, никогда не сообщалось.

Следующие разделы взяты из ITAR [466, 467]:

> § 120.10 Технические данные.
> Технические данные - это, в настоящем подпункте:
> (1) Информация, отличная от программного обеспечения, определенного в 120.10(d), которая нужна для проектирования, разработки, производства, обработки, изготовления, сборки, работы, ремонта, поддержки или модификации средств обороны. Это, например, информация в форме светокопий, чертежей, фотографий, планов, инструкций и документации;
> (2) Секретная информация, касающаяся средств обороны и оборонной деятельности;
> (3) Информация, охватываемая постановлением о засекречивании изобретения;
> (4) Программное обеспечение, определенное в разделе 121.8(f) и непосредственно связанное со средствами обороны
> (5) Это определение не включает информацию, касающуюся общенаучных, математических или инженерных принципов, обычно изучаемых в общедоступных школах, колледжах и университетах, как определено в § 120.11. Оно также не включает базовую рыночную информацию о функции, назначении или общесистемном описании средств обороны.
>
> § 120.11 Открытый доступ.
> Открытый доступ обозначает информацию, которая опубликовывается и может быть общедоступной:
> (1) С помощью продажи в киосках и книжных магазинах;
> (2) С помощью подписки, которая доступна без ограничений для любого, кто хочет получить или приобрести опубликованную информацию;
> (3) С помощью почтовых привилегий второго класса, выданных правительством США;
> (4) В библиотеках, открытых для публики, или в которых публика может получить документы;
> (5) С помощью патентов, доступных в любой патентной конторе;

(6) С помощью неограниченного распространения на конференции, встрече, семинаре, презентации или выставке, доступных обычной публике в Соединенных Штатах;

(7) С помощью сообщений для печати (т.е., неограниченное распространение) в любой форме (например, необязательно опубликованной), одобренных компетентными органами США (см. также § 125.4(b)(13)).

(8) С помощью фундаментальных исследований в науке и технике в аккредитованных высших учебных заведениях США, где полученная информация обычно публикуется и широко распространяется в научном сообществе. Фундаментальными называются базовые и прикладные исследования в науке и технике, когда полученная информация обычно публикуется и широко распространяется в научном сообществе в отличие от исследований, результаты которых не разглашаются из-за прав собственности или определенного контроля доступа и распространения правительством США. Университетские исследования не считаются фундаментальными, если:

(i) Университет или его исследователи соглашаются с другими ограничениями на публикацию научно-технической информации, полученной в результате работы над проектом, или

(ii) Исследования финансируются правительством США, а доступ к результатам исследований и их распространение находится ограничены с целью защиты информации.

§ 120.17 Экспорт.

Под экспортом понимается:

(1) Передача или вывоз средств обороны за пределы Соединенных Штатов любым способом, кроме путешествия за пределы Соединенных Штатов лица, чьи личные знания включают технические данные; или

(2) Передача иностранному лицу прав регистрации, управления или собственности на любой самолет, судно или спутник, присутствующий в Перечне вооружений США, в Соединенных Штатах или за их пределами; или

(3) Раскрытие (в том числе устное или визуальное) или передача в Соединенных Штатах любых средств обороны посольству, учреждению или подразделению иностранного правительства (например, дипломатическим миссиям); или

(4) Раскрытие (в том числе устное или визуальное) или передача технических данных иностранному лицу в Соединенных Штатах или за их пределами; или

(5) Выполнение оборонной деятельности от имени или для выгоды иностранного лица в Соединенных Штатах или за их пределами.

(6) Запускаемый аппарат или полезная нагрузка не должны, при запуске такого аппарата, рассматриваться как экспорт. Однако для определенных целей (см § 126.1 этого подпункта), положения этого подпункта применимы к продажам и другим способам передачи средств обороны или продуктов оборонительной деятельности.

Часть 121- Перечень вооружений США

§ 121.1 Общие положения. Перечень вооружений США

Category XIII—Дополнительное военное снаряжение

(1) Криптографические (включая управление ключами) системы, аппаратура, конструкции, модули, интегральные схемы, компоненты или программное обеспечение с возможностью поддержки секретности или конфиденциальности информации или информационных систем, кроме следующего криптографического оборудования и программного обеспечения:

(i)Специально спроектированное для выполнения защищенным от копирования программным обеспечением только функций дешифрирования при условии, что управление дешифрированием недоступно пользователю.

(ii) Специально спроектированное, разработанное или модифицированное для использования в машинах для банковских операций или денежных транзакций, которое можно использовать только для таких транзакций. Машины для банковских операций или денежных транзакций включают автоматические кассовые аппараты, самообслуживаемые печатающие устройства, торговые терминалы или оборудование для шифрования межбанковских транзакций.

(iii) Использующее только аналоговые методы для криптографической обработки, которая обеспечивает безопасность информации в следующих приложениях. . . .

(iv) Персональные интеллектуальные карточки, использование которых возможно только в оборудовании или системах, не попадающих под регулирование USML.

(v) С ограничением доступа, такие как автоматические кассовые аппараты, самообслуживаемые печатающие устройства или торговые терминалы, которые обеспечивают защиту паролей или персональных идентификационных номеров (PIN) или аналогичных данных, чтобы предотвратить несанкционированный доступ к средствам, но не могут шифровать файлы или тексты, не посредственно не связанные с защитой паролей или PIN.

(vi) Осуществляющее только проверку подлинности данных с помощью вычисления кода проверки подлинности сообщения (MAC) или аналогичной функции для проверки, что в текст не было внесено изменений, или для проверки подлинности пользователей, но которое нельзя использовать для шифрования данных, текста или другой информации помимо необходимой для проверки подлинности.

(vii) Использующее только фиксированные методы сжатия и кодирования данных.

(viii) Используемое только для радиовещания, платного телевидения или аналогичных телевизионных систем с ограниченной аудиторией, без цифрового шифрования, и в которых цифровое дешифрирование ограничено только видео- и аудиофункциями или управлением.

(ix) Программное обеспечение, спроектированное или модифицированное для защиты от злоумышленных компьютерных повреждений, (например, вирусов).

(2) Криптографические (включая управление ключами) системы, аппаратура, конструкции, модули, интегральные схемы, компоненты или программное обеспечение с возможностью генерации распространяемых кодов для большого количества систем или устройств:

(3) Криптографические системы, аппаратура, конструкции, модули, интегральные схемы, компоненты или программное обеспечение.

§ 125.2 Экспорт несекретных технических данных.

(a) Общие положения. Для экспорта несекретных технических данных необходима лицензия (DSP-5), если эти данные не исключены из лицензирующих требований данного подпункта. В случае планового визита детали предполагаемых дискуссий должны быть переданы в Управление по регулированию продажи средств обороны для экспертизы технических данных. Должно быть предоставлено семь копий технических данных или тем ди скуссий.

(b) Патенты. При экспорте технических данных требуется лицензия, выданная Управлением по регулированию продажи средств обороны, если данные превышают необходимые для заполнения внутренней патентной заявки или для заполнения иностранной патентной заявки, если внутренняя заявка не была заполнена. Заявки на патентование за рубежом, выполнение в таких патентах улучшений, модификаций или дополнений должны регулироваться Управлением по патентам и торговым знакам США в соответствии с 37 CFR, часть 5. Экспорт технических данных, необходимых для патентования в других странах, является субъектом норм, издаваемых Управлением по патентам и торговым знакам США, в соответствии с 35 U.S.C. 184.

(c) Раскрытия. Для устного, визуального или документального раскрытия технических данных гражданами США иностранным лицам требуется лицензия, если в данном подпункте не оговорено иное. Лицензия требуется независимо от формы передачи технических данных (например, лично, по телефону, в переписке, электронными средствами, и т.д.). Лицензия тре-

буется для таких раскрытий, делаемых гражданами США при посещении иностранных дипломатических миссий и консульств.

И так далее. В этом документе намного больше информации. Если вы собираетесь экспортировать криптографию, я советую вам добыть его копию и воспользоваться услугами юриста, который во всем этом разбирается.

В действительности экспорт криптографических продуктов контролируется NSA. Если вам нужно получить свидетельство о признании вашего продукта предметом общего потребления (Commodity Jurisdiction, CJ), вы должны представить ваш продукт на одобрение в NSA и подать в Государственный департамент заявку на получение CJ. После одобрения в Госдепартаменте дело попадает под юрисдикцию Министерства торговли, которое никогда особенно не интересовалось криптографией. Однако Государственный департамент никогда не выдаст CJ без одобрения NSA.

В 1977 году Джозеф А. Мейер (Joseph A. Meyer), служащий NSA, написал письмо - несанкционированное, в соответствии с официальной историей инцидента - в IEEE, предупреждающее, что планируемое представление оригинальной работы RSA нарушит ITAR. Из *The Puzzle Palace*:

> Вот его точка зрения. ITAR охватывает всю "несекретную информацию, которая может быть использована, или адаптирована для использования, при проектировании, производстве, изготовлении, ремонте, капитальном ремонте, переработке, конструировании, разработке, действии, поддержке или восстановлении" перечисленных материалов, также как и "любую технологию, которая развивает определенное умение или создает новое в области, которая имеет важное военное применение в Соединенных Штатах." И экспорт действительно включал передачу информации как в письменном виде, так и с помощью устных или визуальных средств, включая краткие обсуждения и симпозиумы, на которых были представлены иностранцы.
>
> Но, буквально следуя туманному, часто слишком пространному законодательству, кажется, требуется, чтобы каждый, кто собирается написать или заявить что-то на тему, касающуюся Перечня вооружений, сначала получил бы одобрение Государственного департамента - эта унылая перспектива явно противоречит Первой поправке и требует подтверждения Верховным судом.

В конце концов NSA признало действия Мейера несанкционированными, и работа по RSA была опубликована, как планировалось. Против изобретателей не было предпринято никаких действий, хотя может быть доказано, что их работа увеличила возможности зарубежной криптографии гораздо больше, чем что-нибудь, опубликованное до того.

Экспорт криптографии обсуждается в следующем заявлении NSA [363]:

> Криптографические технологии считаются жизненно важными для интересов национальной безопасности, включая экономические интересы, военные интересы и интересы внешней политики.
>
> Мы не согласны с заявлениями, сделанными 7 мая 1992 года на слушаниях Судебного комитета, и последними газетными статьями, которые заявляют, что экспортные законы США мешают американским фирмам изготавливать и использовать современное шифровальное оборудование. Нам неизвестно ни об одном случае, когда из-за экспортных ограничений США американской фирме помешали изготавливать и использовать аппаратуру шифрования внутри страны, или американской фирме либо ее дочерней компании помешали использовать аппаратуру шифрования за пределами США. В действительности, NSA всегда поддерживало использование шифрования в американском бизнесе для защиты важной информации как дома, так и за границей.
>
> Для экспорта в другие страны NSA, являющееся частью Министерства обороны, (вместе с Государственным департаментом и Министерством торговли просматривает экспортные лицензии в поисках технологий информационной безопасности, попадающих под действие Экспортного правительственного законодательства или Регулирования международного трафика оружия. Аналогичная система контроля экспорта действует во всех странах КОКОМ и во многих других странах, так как эти технологии повсеместно считаются важными. Не существует общего запрета на экспорт подобных технологий, каждый случай рассматривается отдельно. При этом может потребоваться получить лицензии на такие системы, при получении которых анализируется влияние экспорта этой системы на интересы национальной безопасности - включая интересы экономической, военной и политической безопасности. Экспортные лицензии выдаются или не выдаются в зависимости от типа задействованного оборудования, предполагаемого использования и предполагаемого пользователя.
>
> Наш анализ показывает, что США лидирует в мировом производстве и экспорте технологий информационной безопасности. NSA одобряет для экспорта свыше 90% криптологических продуктов, направленных в NSA Государственным департаментом для лицензирования. Экспортные лицензии на продукты информационной безопасности, попадающие под юрисдикцию Министерства торговли, выдаются без участия NSA или Министерства обороны. Среди них - продукты, использующие такие методы, как DSS и RSA, обеспечивающие проверку подлинности и контроль доступа к компьютерам и сетям. На самом деле, в прошлом NSA играло главную роль в успешном отстаивании ослабления экспортного контроля над RSA и близкими технологиями для проверки подлинности. Эти методы особенно важны при решении проблемы хакеров и несанкционированного использования ресурсов.

Итак, заявлено, что NSA ограничивает экспорт только продуктов шифрования, но не проверки подлинности. Если вы собираетесь экспортировать продукт только для проверки подлинности, получение разрешение ограничится демонстрацией того, что ваш продукт нельзя без значительных переделок использовать для шифрования. Более того, бюрократическая процедура для продуктов проверки подлинности намного проще, чем для продуктов шифрования. Для системы проверки подлинности получать одобрение Госдепартамента (CJ), система шифрования требует повторного одобрения для каждой версии продукта или даже при каждой продаже.

Без CJ вам придется запрашивать разрешение на экспорт всякий раз, когда вы захотите экспортировать продукт. Государственный департамент не разрешает экспортировать продукты с сильным шифрованием, даже использующие DES. Отдельные исключения были сделаны для дочерних фирм американских компаний для возможности закрытой связи с, для некоторых банковских приложений и экспорт для военных пользователей США. Ассоциация производителей программного обеспечения (SPA) вела переговоры с правительством об ослаблении ограничений на экспортные. Соглашение, заключенное SPA и Госдепартаментом в 1992 году, облег-

чило правила выдачи экспортных лицензий для двух алгоритмов, RC2 и RC4, при условии, что длина используемого ключа не превысит 40 битов. Подробности можно найти в разделе 7.1.

В 1993 году в Палате представителей Мария Кантвелл (Maria Cantwell) (D-WA) по просьбе компаний-разработчиков программного обеспечения внесла законопроект, ослабляющий экспортный контроль за программами. Сенатор Пэтти Мюррей (Patty Murray) (D-WA) внесла соответствующий билль в сенате. Законопроект Кантвелл был добавлен к общему закону о контроле над экспортом, проходящему через Конгресс, но был удален Комитетом по разведке под сильным давлением NSA. Когда NSA что-нибудь делает, оно прикладывает все усилия - комитет единодушно проголосовал за удаление формулировки. За последнее время я не припомню другого случая, чтобы группа законодателей что-то сделала единодушно.

В 1995 году Дан Бернштейн (Dan Bernstein) при поддержке EFF подал в суд на правительство США, пытаясь помешать правительству ограничивать публикации криптографических документов и программного обеспечения [143]. В иске утверждалось, что законы об экспортном контроле неконституционны и вносят "непозволительные априорные ограничения высказываний в нарушение Первой поправки". Конкретно в иске утверждалось, что современный процесс контроля над экспортом:

— Позволяет бюрократам ограничивать публикации без решения суда.

— Обеспечивает слишком мало процедурных возможностей защиты прав в соответствии с Первой поправкой.

— Требует от издателей регистрироваться в правительстве, создавая эффект "лицензированной прессы".

— Отказывает в общих публикациях, требуя идентифицировать каждого получателя.

— Достаточно запутан, чтобы простые люди не могли знать, какое поведение правильно, а какое - нет.

— Слишком пространен, так как запрещает поведение, которое явно защищается (например, разговор с иностранцами внутри Соединенных Штатов).

— Применяется слишком широко, запрещая экспорт программного обеспечения не содержащего криптографии, исходя из соображений, что криптография может быть добавлена позже.

— Явно нарушает Первую поправку, запрещая частные беседы по криптографии, так как правительство желает вместо этого навязывать публике свои криптографические взгляды.

— Многими способами превышает полномочия, предоставленные как Конгрессом в экспортном законодательстве, так и Конституцией.

Можно предвидеть, что решение этого дела займет несколько лет, но предвидеть, чем оно закончится, невозможно.

Тем не менее, Консультативный комитет по безопасности и защищенности (Computer Security and Privacy Advisory Board), официальный консультант NIST, в марте 1992 года проголосовал за то, чтобы пересмотреть в национальной политике криптографические вопросы, включая экспортную политику. Было заявлено, что экспортная политика определяется только организациями, отвечающими за национальную безопасность, без учета точки зрения организаций, связанных с развитием торговли. Эти связанные с национальной безопасностью организации делают все возможно, чтобы ничего не изменилось, но необходимость перемен уже назрела.

25.15 Экспорт и импорт криптографии за рубежом

В других странах существует свое экспортное и импортное право [311]. Приведенный обзор неполон и возможно устарел. Страны могут издать законы и не обращать на них внимания, или не иметь законов, но каким-то образом ограничивать экспорт, импорт и использование.

— Австралия требует наличия сертификата у импортируемого криптографического продукта только по требованию страны-экспортера.

— В Канаде нет контроля импорта, а контроль экспорта аналогичен американскому. Экспорт продуктов из Канада может быть ограничен, если они включены в Перечень контроля экспорта, соответствующий Акту разрешений экспорта и импорта. В отношении криптографических технологий Канада следует ограничениям КОКОМ. Шифровальные устройства описаны под категорией пять, части два канадских правил экспорта. These provisions аналогичны категории пять в Правительственных правилах экспорта в США.

— Китай использует схему лицензирования импортируемых продуктов, экспортеры должны заполнить заявку в Министерстве зарубежной торговли. На основе китайского Перечня запрещенного и ограниченного экспорта и импорта, принятого в 1987 году, Китай ограничивает импорт и экспорт устройств кодирования речи.

— Во Франции нет специального законодательства относительно импорта криптографии, но существуют за-

коны, касающиеся продажи и использования криптографии в стране. Продукты должны быть сертифицированы: либо они должны соответствовать опубликованным спецификациям, либо фирменная спецификация компании должна быть предоставлена правительству. Правительство может также затребовать два устройства для собственного использования. У компаний должна быть лицензия на продажу криптографии во Франции, в лицензии указывается рыночное назначение. У пользователей должна быть лицензия на покупку и использование криптографии, в лицензию включено положение о том, что пользователи должны быть готовы передать свои ключи правительству в течение четырех месяцев после использования. Это ограничение иногда допускает исключения: банков, больших компаний, и т.д. Для криптографии, экспортируемой из США, лицензионные требования отсутствуют.

— Германия следует положениям КОКОМ, требуя лицензировать экспорт криптографии. Проводится специальный контроль общедоступного криптографического программного обеспечения.

— В Израиле есть ограничения импорта, но, по видимому, никто не знает какие.

— Бельгия, Италия, Япония, Нидерланды и Великобритания следуют положениям КОКОМ, требуя лицензировать экспорт криптографии.

— В Бразилии, Индии, Мексике, России, Саудовской Аравии, Испании, Южной Африке, Швеции и Швейцарии контроль экспорта или импорта криптографии отсутствует.

25.16 Правовые вопросы

Являются ли цифровые подписи настоящими подписями? Будут ли они признаны судом? Некоторые предварительные правовые исследования привели к мнению, что цифровые подписи будут соответствовать требованиям законных обязующих подписей для большей части применений, включая коммерческое использование, определенное в Едином своде законов о торговле (Uniform Commercial Code, UCC). Решение Управления по общей бухгалтерии (GAD, General Accounting Office), вынесенное по просьбе NIST, утверждает, что цифровые подписи соответствуют правовым стандартам для рукописных подписей [362].

Акт о цифровых подписях штата Юта вступил в действие 1 мая 1995 года, обеспечивая законную основу использования цифровых подписей в системе судопроизводства. Калифорния рассматривает соответствующий законопроект, а в Орегоне и Вашингтоне разрабатывают свои законы. Техас и Флорида дышат им в затылок. К моменту издания книги большинство штатов пройдет этот путь.

Американская юридическая ассоциация (Отдел EDI и информационных технологий секции науки и техники) разработала образец акта, который может быть использован штатами в процессе законотворчества. Акт пытается вписать цифровые подписи в существующую для подписей правовую инфраструктуру: Единый свод законов о торговле, Законы Федеральной резервной системы Соединенных Штатов, общее право о контрактах и подписях, Конвенция ООН по контрактам для международной продажи товаров и Конвенция ООН по международным законам о комитетах по биржам и долговым обязательствам. В акт включены положения об ответственности и обязанностях сертифицирующих органов, вопросы ответственности, а также ограничения и политика.

В Соединенных Штатах законы о подписях, контрактах и торговых операциях находятся в юрисдикции штатов, поэтому этот акт-образец разработан для штатов. Окончательной целью является федеральный акт, но если все начинается на уровне штатов, у NSA меньше возможностей все испоганить.

Даже при этом, пока правильность цифровых подписей не будет оспорена в суде, их правовой статус останется неопределенным. Для того, чтобы цифровые подписи обладали теми же идентификационными возможностями, что и рукописные подписи, они сначала должны быть использованы для подписания законного, затем оспорены в суде одной из сторон. Тогда суд рассмотрит безопасность схемы подписи и вынесет решение. Спустя некоторое время, когда повторится подобный случай, решения о том, какие методы цифровой подписи и какие размеры ключей понадобятся, чтобы цифровая подпись была признана законной, будет вынесено на основе предыдущих решений. Возможно для этого потребуются годы.

До тех пор, если два человека хотят использовать цифровые подписи для контракта(для заявок на покупку, для приказов по работе, и т.д.), рекомендуется, чтобы они подписали на бумаге контракт, с котором они согласятся в будущем признавать любые документы, подписанные их цифровыми подписями [1099]. В этом документе должны определяться алгоритм, размер ключа и все остальные параметры. В нем должен, к тому же, быть определен способ разрешения споров.

Послесловие Мэтта Блейза

Одним из самых опасных моментов криптологии (и, следовательно, данной книги), является то, что вам почти удается измерить ее. Знание длины ключей, способов разложения на множители и криптоаналитических методов позволяет оценить (в отсутствии настоящей теории проектирования шифров) " коэффициент работы", необходимый для вскрытия конкретного шифра. Слишком велик соблазн неправильно использовать эти оценки в качестве общей меры безопасности систем. В реальном мире у взломщика есть куда больше возможностей, чем использование одного криптоанализа. Часто успех достигается с помощью вскрытий протоколов, троянских коней, вирусов, электромагнитного контроля, физической компрометации, шантажа и запугивания владельцев ключа, ошибок операционной системы и прикладных программ, аппаратных ошибок, ошибок пользователей, физического подслушивания, прикладной социологии, анализ содержимого свалок, и это далеко не все.

Высококачественные шифры и протоколы являются важными средствами, но сами по себе они не заменяют реалистичных, критических размышлений о том, что действительно нужно защитить, и как могут быть взломаны различные уровни обороны (взломщики, в конце концов, редко ограничиваются чистыми, хорошо определенными моделями научного мира). Росс Андерсон (Ross Anderson) приводит примеры криптографически сильных систем (в банковской индустрии), которые не устояли перед угрозами реального мира [43, 44]. Даже когда у взломщика есть доступ только к шифротексту, через кажущиеся незначительными бреши в других частях системы может просочиться достаточно информации, чтобы сделать хорошую криптосистему бесполезной . Союзники во второй мировой войне взломали трафик немецкой Энигмы, главным образом тщательно используя ошибки операторов [1587].

NSA в ответ на вопрос, может ли правительство вскрывать DES, язвительно заметило, что реальные системы настолько небезопасны, что об этом даже не стоит беспокоиться. К сожалению, не существует простых рецептов, как сделать систему безопасной, заменить тщательное проектирование и критический анализ невозможно. Хорошие криптосистемы делают жизнь взломщика намного труднее, чем жизнь законного пользователя, но это не так в отношении почти всех остальных аспектов безопасности компьютеров и систем связи . Рассмотрим следующие (наверняка не все) "Десять главных угроз безопасности реальных систем ", каждую из которых легче осуществить, чем предотвратить.

1. Печальное состояние программного обеспечения. Всем известно, что никто не знает, как писать программное обеспечение. Современные системы сложны, включают сотни тысяч строк кода, любая из них может повредить безопасности. Из программных модулей, связанных с безопасностью извлекать ошибки еще труднее.

2. Неэффективная защита против вскрытий с отказом от услуг. В некоторых криптографических протоколах допускается анонимность. Использование анонимных протоколов может быть особенно опасным, если они увеличивают возможность неопознанного вандала нарушить предоставление услуги Поэтому анонимные системы должны быть особенно устойчивы к вскрытиям с отказом от услуг . В устойчивых сетях поддерживать анонимность может быть легче - ведь вряд ли кого-то сильно волнует наличие миллионов анонимных входных точек в большинстве устойчивых сетей, таких как телефонная сеть или почтовая система, где отдельному пользователю относительно трудно (или дорого) вызвать крупномасштабные аварии.

3. Нет места для хранения секретов. Криптосистемы защищают большие секреты малыми (ключами) . К сожалению, современные компьютеры не особенно хороши для защиты даже маленьких секретов . Многопользовательские сетевые рабочие станции могут быть взломаны, а их память - скомпромет ирована. Отдельно стоящие, однопользовательские машины могут быть украдены или скомпрометированы вирусами, которые организуют асинхронную утечку секретов . Удаленные серверы, где может и не быть пользователя, вводящего парольную фразу (но см. угрозу №5), представляют собой особенно трудную проблему.

4. Плохая генерация случайных чисел. Для ключей и сеансовых переменных нужны хорошие источники непредсказуемых битов. Энтропия работающего компьютера велика, но редкое приложение в состоянии правильно использовать ее. Было предложено множество методов получать истинно случайные числа программным образом (используются непредсказуемость времени выполнения операций ввода вывода, расхождения тактовой частоты и таймера, и даже турбулентность воздуха внутри корпуса твердого диска), но все они очень чувствительны к незначительным изменениям сред, в которых они используются.

5. Слабые парольные фразы. Большинство криптографического программного обеспечения решает проблемы хранения и генерации ключей на основе создаваемых пользователем парольных фраз , которые считаются достаточно непредсказуемыми для генерации хорошего ключевого материала, и которые также легко запоминаются и поэтому не требуют безопасного хранения . В то время, как словарные вскрытия являются хорошо известной проблемой для коротких паролей , о способах вскрытия ключей,

созданных на основе выбранных пользователями парольных фраз, известно мало. Шеннон показал, что энтропия английского текста чуть больше 1 бита на символ, что, по видимому, позволяет использовать против парольных фраз грубую силу. Однако пока не вполне понятно, для этого как упорядочивать парольные фразы. Пока мы не разберемся как следует, как вскрывать парольные фразы, мы не поймем, насколько они слабы или сильны.

6. Неправильное доверие. Почти все доступное криптографическое программное обеспечение предполагает, что пользователь находится в непосредственном контакте с системой ли пользуется надежным способом доступа. Например, интерфейсы к программам, подобным PGP, предполагают, что их парольные фразы поступают от пользователя по надежному пути, например, с локальной консоли. Но это не всегда так, рассмотрим проблему чтения вами шифрованной почты при подключении по сети. То, что проектировщик системы считает надежным, может не соответствовать потребностям или ожиданиям реальных пользователей, особенно когда программным обеспечением можно управлять удаленно по небезопасным каналам.

7. Плохо понимаемое взаимодействие протоколов и услуг. С ростом и усложнением систем часто происходят странные вещи, и бывает трудно что-нибудь понять что-нибудь, даже когда произойдет какая-нибудь авария. Червь Internet распространялся с помощью туманного и с виду вполне невинного средства программы передачи почты. Сколько еще возможностей и в каком количестве программ обладают неожиданными следствиями, которые только ждут своего о ткрытия?

8. Нереалистичная оценка угрозы и риска. Эксперты по безопасности стремятся сконцентрировать свои усилия на угрозах, которые известно как моделировать и предотвращать. К сожалению, взломщики выполняют вскрытия на базе собственных знаний, и две эти области редко совпадают. Слишком много "безопасных" систем было спроектировано без учета реально возможных действий взломщика.

9. Интерфейсы, которые делают безопасность дорогой и неудобной. Если нужно использовать средства обеспечения безопасности, то они должны быть удобными и достаточно прозрачными, чтобы люди действительно пользовались ими. Нетрудно спроектировать механизмы шифрования, которые работают только за счет производительности или простоты использования, и еще легче создать механизм, который провоцирует ошибки. Безопасность должно быть труднее выключить, чем включить; к несчастью, лишь немногие системы действительно так р аботают.

10. Слишком всеобъемлющие требования к безопасности. Эта проблема хорошо известна почти всем, чье счастье связано с продажей продуктов и услуг безопасности. Пока существует широко распространенное требование всеобъемлющей безопасности, средства и инфраструктура, обеспечивающие его реализацию, будут дороги и недоступны для многих приложений. Частично это проблема понимания и раскрытия угроз и опасностей в реальных приложениях, а частично проблема проектирования систем, в которых безопасность не закладывается изначально, а добавляется позже.

Более полный список и обсуждение подобных угроз может легко заполнить книгу такого же размера, при этом проблема будет лишь едва затронута. Что делает их особенно трудными и опасными, так это то, что не существует никакого магического способа избавиться от них, кроме хорошего анализа и хорошей инженерной работы. Честолюбивый криптограф должен ощущать границы искусства.

Мэтт Блейз
Нью-Йорк