

Защита информации

Павел Юдаев

МГТУ им. Баумана, Кафедра ИУ-9

Москва, 2014

oooooo
oooooooooooo
ooo

Симметричный шифр. n пользователей. Каждый с каждым.

$n(n - 1)/2$ постоянных ключей.

Добавление польз-лей затруднено.

Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхема-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана



Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхема-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана

Инфраструктура открытого ключа (PKI)



Протокол Нидхема-Шредера (Needham-Schroeder)

Протокол с доверенным центром.

Вариант для симметричной к/с.

S - доверенный сервер.

Алиса (A) инициирует коммуникацию с Бобом (B), хочет иметь защищенный сеанс связи. Боб хочет быть при этом уверен, что к нему обратилась именно Алиса.

Обозн.:

- A, B - Алиса и Боб (их ID, известны всем)
- K_{XY} - симметричный ключ, известный только X и Y.
- N_A and N_B - nonce, генерируемые A и B.
- $\{M\}_{K_{XY}}$ - сообщение M, зашифрованное ключом K_{XY}

S лично вручил A и B ключи K_{AS}, K_{BS} .



Протокол:

1) $A \rightarrow S : \{A, B, N_A\}$ - открытый текст: "Хочу связаться с B".

2) $S \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$

4) $B \rightarrow A : \{N_B\}_{K_{AB}}$

5) $A \rightarrow B : \{N_B + 1\}_{K_{AB}}$

A производит фиксированную операцию с N_B , показывая B, что она все еще в сети и помнит этот сессионный ключ.



Атака воспроизведения (replay attack):

если сессионный ключ N_{AB} стал известен Е, и Е запомнил сообщение $\{K_{AB}, A\}_{K_{BS}}$,

$$E \rightarrow B : \{K_{AB}, A\}_{K_{BS}},$$

В примет его, т.о. установив сеанс связи с Е (а не А) с ключом N_{AB} .



Защита:

00) $A \rightarrow B : A$ - открытый текст

0) $B \rightarrow A : \{A, N'_B\}_{K_{BS}}$

В отвечает поспе, зашифрованным своим ключом.

1') $A \rightarrow S : \{A, B, N_A, \{A, N'_B\}_{K_{BS}}\}$

А связ-ся с S: "Хочу связаться с В". Открытый текст и сообщ. от В.

2') $S \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A, N'_B\}_{K_{BS}}\}_{K_{AS}}$

3') $A \rightarrow B : \{K_{AB}, A, N'_B\}_{K_{BS}}$



Шаги 4),5) без изменений:

$$4) B \rightarrow A : \{N_B\}_{K_{AB}}$$

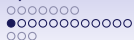
$$5) A \rightarrow B : \{N_B - 1\}_{K_{AB}}$$

Добавление попсо N'_B не дает возм-ть Е переслать повторно сообщение $\{K_{AB}, A, N'_B\}_{K_{BS}}$ с скомпрометированным ключом K_{AB} : этот N'_B уже использован и не ожидается.

Kerberos: протокол Н.-Ш. с $N'_B = \text{timestamp}$ и др. модификациями.

oooooooo●
oooooooooooo
ooo

TODO: Добавить вариант для асимметричной к/с? Из 2011 года. Лекция 11а. Не сложный.



Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхема-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана

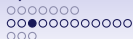
Инфраструктура открытого ключа (PKI)



Система авторизации Kerberos.

Протокол Нидхема-Шредера с двумя существенными изменениями:

- уменьшено количество сообщений между клиентом и сервером аутентификации.
- введение TGT (Ticket Granting Ticket, билет для получения билета) - концепции, позволяющей пользователям авторизовываться на несколько сервисов один раз.



Kerberos:

- сервер аутентификации (AS) и
- сервер выдачи билетов (TGS - Ticket Granting Server).

Цель:

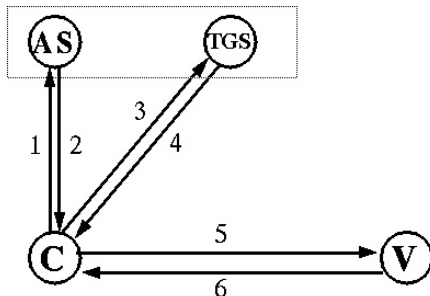
- я запустил программу, она от моего имени авторизуется для доступа к сервисам системы.

Простой подход: хранить мой пароль в памяти.

Н.-Ш. и Kerberos: не хранить мой пароль.

- защита от подслушивания и атаки воспроизведения.
- от других атак стандарт Kerberos не защищает.

Исп-е: Windows 2000 и более поздние, также все Unix и гетерогенные сети.



Чтобы связаться с другим сервером V', C повторит шаги 3)-5).
1),2) - повторить по истечении срока авторизации.



Kerberos.

Обозначения:

AS = сервер авторизации (authentication server)

TGS = сервер выдачи билетов (ticket granting server)

C = программа-клиент (client)

S = сервис системы (service provider)

A = сетевой адрес клиента (client's network address)

V = срок действия билета (validity period for a ticket)

t = текущее время (timestamp)

K_X = секретный ключ X ($K_{X,AS}$)

$K_{X,Y}$ = сессионный ключ для X,Y

$\{m\}_{K_X}$ = сообщение m , зашифр. ключом K_X

$T_{X,Y}$ = билет: X может пользоваться Y

$A_{X,Y}$ = токен авторизации X для Y



Билет $T_{X,Y} = Y, \{X, A, V, t, K_{X,Y}\}K_Y$

Создает TGS или AS, с новой меткой времени t .

Токен авторизации X для Y :

$A_{X,Y} = \{X, t, [key]\}K_{X,Y}$

Создает X , с новой t .

key - опционально.



Билет $T_{X,Y} = Y, \{X, A, V, t, K_{X,Y}\}$

Токен авторизации: $A_{X,Y} = \{X, t, [key]\}$

Описание протокола:

1) $C \rightarrow AS : \{C, TGS, t\}$ - время клиента верное? (± 5 минут)

Запрос к AS на билет для доступа к TGS. Открытый текст.

2) $AS \rightarrow C : \{T_{C,TGS}\}K_{TGS}, \{K_{C,TGS}\}K_C$

TGT для доступа к TGS. Время жизни TGT 8-10 часов. Потом надо получить новый.



$$3) C \rightarrow TGS : \{A_{C,S}\}K_{C,TGS}, \{T_{C,TGS}\}K_{TGS}$$

Запрос билета для связи с сервером S

$$4) TGS \rightarrow C : \{K_{C,S}, S, V\}K_{C,TGS}, \{T_{C,S}\}K_S$$

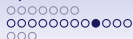
Server Ticket (имеет срок действия) и сессионный ключ

$$5) C \rightarrow S : \{A'_{C,S}\}K_{C,S}, \{T_{C,S}\}K_S$$

Предъявляет Server Ticket; шифрует метку времени \Rightarrow знает ключ.

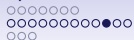
$$6) S \rightarrow C : \{t_{A'} + 1\}K_{C,S}$$

Шифрует текущее время \Rightarrow знает ключ.



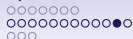
Проверка метки времени подразумевает, что часы всех машин в сети синхронизованы в пределах нескольких минут. Если отличие метки времени от текущего времени выходит за эти пределы, запрос отвергается как попытка атаки воспроизведения.

TGS также хранит базу токенов авторизации, чтобы исключить атаки воспроизведения со все еще действительной меткой времени. Повторный запрос с тем же токеном авторизации и меткой времени будет проигнорирован.



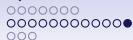
C и S могут зашифровать свою сессию ключом $K_{C,S}$ или ключом key из authenticator $A_{C,S}$. (Разные версии протокола.)

Оба этих ключа известны только им, т.е. их использование явл. свидетельством авторства сообщения.



Уязвимости:

- чтобы искл. атаку воспроизведения в течение временного окна после метки времени, каждый сервер должен хранить все действительные тикеты своих клиентов. Если их много, то проблема.
- если хосту может быть передано по сети неправильное время, можно воспроизвести старые сообщения. Т.е. требуется безопасный сетевой протокол точного времени.
- можно угадать пароль польз-ля. (Подбором: люди часто выбирают плохие пароли.)
- вредоносное ПО. Установить на клиентской стороне ПО, которое, кроме осн. ф-й, собирает пароли. В реальной жизни это едва ли не основная угроза.



Улучшение:

- авторизация клиента перед AS с использованием ЭЦП сообщения и сертификата пользователя.

На основе асимметричного протокола Нидхема-Шредера.



Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхема-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана

Инфраструктура открытого ключа (PKI)



Домены Kerberos (realms)

Нужны в большой распределенной сети

$$3) C \rightarrow TGS : \{A_{C,TGS2}\}K_{C,TGS}, \{T_{C,TGS}\}K_{TGS}$$

$$3a) TGS \rightarrow C : \{K_{C,TGS2}, S, V\}K_{C,TGS}, \{T_{C,TGS2}, \underline{r(TGS)}\}K_{c-r}$$

$$3b) C \rightarrow TGS2 : \{A'_{C,S}\}K_{C,TGS2}, \{T_{C,TGS2}, \underline{r(TGS)}\}K_{c-r}$$

$$4) TGS2 \rightarrow C : \{K_{C,S}, S, V\}K_{C,TGS2}, \{T_{C,S}\}K_S - \text{как внутри домена}$$

5),6) без изменений

$r()$ - realm, домен. K_{c-r} - cross-realm key.



Kerberos V4: каждый домен с каждым. Квадратичный рост числа ключей.

Kerberos V5: древовидная структура.

Пример

graphics.cs.mit.edu ↔ cs.mit.edu

parallel.cs.mit.edu ↔ cs.mit.edu

cs.mit.edu ↔ mit.edu

mit.edu ↔ biology.mit.edu

mit.edu ↔ chemistry.mit.edu

В билете записывается путь, пройденный по дереву.
Проверяется на входе в домен.

Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхем-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана

Инфраструктура открытого ключа (PKI)

oooooo
oooooooooooo
ooo

Было: доверенный центр создает и раздает сессионные ключи.
А без него?

Протокол Диффи-Хеллмана

p - большое простое, $g \in \mathbb{Z}_p^*$ - порождающий элемент (генератор) группы.

Цель - создать общий ключ для A, B .

$A : a \xleftarrow{R} \mathbb{Z}_p^*$ - число

$B : b \xleftarrow{R} \mathbb{Z}_p^*$

$A \rightarrow B : g^a \bmod p$, "A"

$B \rightarrow A : g^b \bmod p$, "B"

$A : (g^b)^a = g^{ab} = k_{AB} = (g^a)^b : B$ - общий ключ

ooooooo
ooooooooooooo
ooo

Задача DHP: даны \mathbb{Z}_p^* , g , g^a , g^b . Чему равен g^{ab} ?

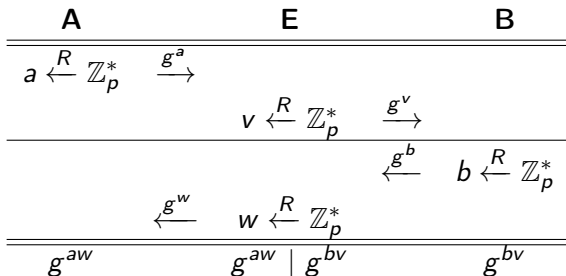
Гипотеза Диффи-Хеллмана: DHP - сложная задача. Не опровергнута за несколько десятилетий.

Для групп общего вида доказано, что DHP так же сложна, как задача поиска дискретного логарифма (найти x по g^x).

Лучший известный алг-м - субэксп-й.

(Приложение теории сложности к криптографии.)

Протокол уязвим к атаке посредника (man in the middle attack):



ooooooo
ooooooooooooo
ooo

- Если сообщения заверены ЭЦП, которые можно проверить, то MitM бессилён.
- Вариант системы: хранить g^a в “телефонном справочнике”. Те же проблемы с MitM.

oooooo
oooooooooooo
ooo

Почему g должно быть порождающим элементом?

Пусть p большое, но $|\langle g \rangle| = 10^6$. Тогда ключ g^{ab} - один из 10^6 эл-в. Легко найти перебором.

Кто выбирает p и g ? Эти пар-ры общие для всех абонентов сети, в которой работает протокол DH. Поэтому каждый из пользователей должен (благоразумно) проверить, что p - простое и g - примитивный эл-т $GF^*(p)$.

В \mathbb{Z}_p^* $\varphi(p-1)$ примитивных эл-в.

Пусть p_1, \dots, p_n - простые множители $p-1$. Чтобы проверить, явл. ли g генератором \mathbb{Z}_p^* , выч. все $g^{(p-1)/p_i}$, $i = 1..n$ и сравним с 1.

Пример

$$p = 11, p - 1 = 10 = 2 * 5.$$

$$2^2 = 4 \not\equiv 1 \pmod{11}.$$

$$2^5 = 32 \not\equiv 1 \pmod{11}$$

2 - генератор \mathbb{Z}_{11}^* .

$$3^2 = 9 \not\equiv 1 \pmod{11}.$$

$$3^5 = 243 \equiv 1 \pmod{11}.$$

3 - не генератор \mathbb{Z}_{11}^* .



Атака при наличии большого числа малых делителей.

Если В не проверяет порядок присланного эл-та, возможна атака на DH.

(В реальной реализации части протокола IPSec такой проверки не было!)

Пусть d - малый делитель $(p - 1)$. Зл-к присылает эл-т $h : | < h > | = d$. Теперь сессионный ключ $k := h^b$ польз-ля В может иметь лишь d зн-й и полностью опр-ся зн-ми h и $(b \bmod d)$. Исходящее от В сообщение можно взломать перебором, найдя таким образом k и значение $(b \bmod d)$.

Порядок любой подгруппы является делителем числа $(p - 1)$. И обратно, для любого d - делителя $(p - 1) \exists!$ подгруппа \mathbb{Z}_p^* порядка d .

Если у $(p - 1)$ много относительно малых делителей, то эту атаку можно повторять и получить набор $(b \bmod d_1), \dots, (b \bmod d_t)$. По китайской т. об остатках, можно найти $b \bmod (d_1 \cdot \dots \cdot d_t)$. Если $(d_1 \cdot \dots \cdot d_t)$ велико, зл-к получит большой объем инф-и о постоянном секретном ключе b польз-ля В.

Защита: разложить на множители $p - 1$ не можем (большое). Используем надежные простые числа $p = 2q + 1$, подгруппу квадратичных вычетов (см. далее). Для надежного простого числа каждый элемент подгруппы - ее генератор.

oooooo
oooooooooooo
ooo

Надежное простое число (safe prime)

- это достаточно большое простое ч. вида $p = 2q + 1$, где q - также простое. В этом случае у \mathbb{Z}_p^* будут только 4 подгруппы:

- $\{1\}$ - порядка 1
- $\{1, (p-1)\}$ - порядка 2
- подгруппа порядка q
- подгруппа порядка $2q$, совп. с \mathbb{Z}_p^*

Нет малых подгрупп.

ooooooo
ooooooooooooo
ooo

При этом \mathbb{Z}_p^* обладает недостатком: при ее использовании в ДН раскрывается один бит ключа.

Половина эл-в \mathbb{Z}_p^* являются квадратичными вычетами, половина - нет.

Можно быстро проверить (вычислить символ Лежандра), явл. ли число квадратом по мод. p , не находя корня из числа.

Пусть образующий эл-т g - не квадр. вычет. Тогда при четном x g^x - кв. вычет, при нечетном - нет. Это может легко проверить эл-к и найти младший бит числа x .

Поэтому надо использовать только квадратичные вычеты по мод. p . Их половина, и они образуют подгруппу порядка q . q - простое, поэтому вложенных подгрупп нет. Значит, всякий ее эл-т им. порядок q .

Использование надежного простого числа.

- Выберите пару простых p, q : $p = 2q + 1$. (Напр., перебором простого числа q .)
- Выберите случ. число a из $\{2, \dots, (p - 2)\}$.
- $g := a^2 \bmod p$. Надо: $|\langle g \rangle| = q$.
Т.о., g годится *Leftrightarrow* $g \neq 1, g \neq p - 1$ и $g^q = 1 \bmod p$.
Иначе выберите др. a и проверьте снова.

Нашли g - квадратичный вычет по модулю p , из подгруппы порядка q . q простое, $\Rightarrow g$ - генератор подгруппы.

ooooooo
ooooooooooooo
ooo

Каждый раз, когда В получает элемент h , который (предпол.) является степенью g , он должен проверить этот факт. Если p - надежное простое, проверить просто: вычислить $\left(\frac{h}{p}\right)$.

ooooooo
ooooooooooooo
ooo

Использ. подгрупп меньшего размера.

Низкая эфф-ть исп-я надежных простых чисел:

если длина p - n бит, то длина q - $n - 1$ бит.

При возведении в степень все показатели степеней приводятся по $\text{mod } q$ и будут иметь длину до $n - 1$.

Средняя операция возведения в степень - ок. $3n/2$ умножений по мод. p . При больших p это влияет на эфф-ть.

Решение: подгруппы меньшего размера.

oooooo
oooooooooooo
ooo

Выбир. q - 256 битовое простое ч. Затем ищем N из нек. наперед заданного диапазона такое, что $p = Nq + 1$ - простое. Длина простого p - неск. тысяч бит.

Выберем эл-т g порядка q .

Так же, как раньше: случ. эл-т a , $g = a^N$.

Проверяем, что $g \neq 1$, $g \neq p - 1$, $g^q = 1$. Полученный набор (p, q, g) пригоден для использ. в протоколе DH.

Задача

Почему g - квадратичный вычет?

Пользователь А
Известны (p, q, g)
Проверить параметры (p, q, g)
 $x \in_R \{1, \dots, q-1\}$

$$\xrightarrow{X := g^x}$$

Пользователь Б
Известны (p, q, g)
Проверить параметры (p, q, g)

$$1 \leq X^2 < p, X^{q-1} = 1$$

$$y \in_R \{1, \dots, q-1\}$$

$$\xleftarrow{Y := g^y}$$

$$1 \leq Y^2 < p, Y^{q-1} = 1$$

$$k \leftarrow (Y)^x$$

$$k \leftarrow (X)^y$$

oooooo
oooooooooooo
ooo

$\forall s \in \langle g \rangle$ справедливо $s^q = 1$. Поэтому $s^e = s^{e \bmod q}$. Это сокращает объем выч-й, по сравнению со случаем, когда $p = 2q + 1$. Чем больше зн-е N , тем больше разница.

Замеч.: так же как раньше, необходимо проверять, что присланный s входит в $\langle g \rangle$: $1 < s < p$, $s^q = 1$.

oooooo
oooooooooooo
ooo

Размер p

Выбирается исходя из стойкости алгоритма. Для симметричных шифров для достижения стойкости порядка 2^{128} операций требовался 128-битовый ключ. В системе DH для такой же стойкости размер p - порядка 6800 бит.

Пример

длина p - 6000 бит. Сколько умножений для выч. g^x ?

$p = 2q + 1$, длина q - 5999 бит, \Rightarrow ок. 9000.

$p = Nq + 1$, длина q - 256 бит, \Rightarrow ок. 400.

Создание общего секрета с асимм. к/с (*)

$$\begin{array}{ccc}
 & A & B \\
 pk, sk \leftarrow G() & \xrightarrow{pk, "A"} & \\
 x = D(sk, c) & \xleftarrow{c := E(pk, x), "B"} & x \xleftarrow{R} \{0, 1\}^{128}
 \end{array}$$

x - общий секрет A, B

```

○○○○○○○
○○○○○○○○○○○○○
○○○
    
```

(E, D) - семантически стойкий отн. атаки с выбранным о.т., поэтому зл-к не может отличить $E(pk, x)$ от $E(pk, y)$.

Протокол уязвим к атаке посредника (MitM):

A	E	B
$pk, sk \leftarrow G()$	\xrightarrow{pk}	
	$pk', sk' \leftarrow G()$	$\xrightarrow{pk'}$
	$x = D(sk', c)$	$\xleftarrow{E(pk', x)}$
	$y \xleftarrow{R} \{0, 1\}^{128}$	$x \xleftarrow{R} \{0, 1\}^{128}$
$y = D(sk, c)$	$\xleftarrow{E(pk, y)}$	
y	y x	x

Защита от атаки посредника для асимм. к/с (*)

A докажет, что это именно она.

A

B

$$pk, sk \leftarrow G() \xrightarrow{pk, "A", \text{Удост. личности A}}$$

$$x = D(sk, c) \xleftarrow{c := E(pk, x), "B"} x \xleftarrow{R} \{0, 1\}^{128}$$

Удостоверение личности A - напр., $\text{cert}_T(A, pk)$.

Тогда третий шаг, напр. передать $\text{HMAC}(x, "A")$. (*)

Задача


Как можно изменить протокол, чтобы B тоже удостоверил свою личность?


ooooooo
ooooooooooooo
ooo

www.facebook.com ✕

Identity verified

Permissions **Connection**

 The identity of this website has been verified by DigiCert High Assurance CA-3 but does not have public audit records.
[Certificate information](#)

 Your connection to www.facebook.com is encrypted with 128-bit encryption.


The connection uses TLS 1.2.


The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_ECDSA as the key exchange mechanism.

www.google.ru ✕

Identity verified

Permissions **Connection**

 The identity of this website has been verified by Google Internet Authority G2 but does not have public audit records.
[Certificate information](#)

 Your connection to www.google.ru is encrypted with 256-bit encryption.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using CHACHA20_POLY1305 and uses ECDHE_RSA as the key exchange mechanism.

oooooo
oooooooooooo
ooo

CHACHA20_POLY1305 ??? *

Google Online Security Blog: **Speeding up and strengthening HTTPS connections for Chrome on Android**

Posted: Thursday, April 24, 2014

Posted by Elie Bursztein, Anti-Abuse Research Lead

Earlier this year, we deployed a new TLS cipher suite in Chrome that operates three times faster than AES-GCM on devices that don't have AES hardware acceleration, including most Android phones, wearable devices such as Google Glass and older computers. This improves user experience, reducing latency and saving battery life by cutting down the amount of time spent encrypting and decrypting data.

[http://googleonlinesecurity.blogspot.ru/2014/04/
speeding-up-and-strengthening-https.html](http://googleonlinesecurity.blogspot.ru/2014/04/speeding-up-and-strengthening-https.html)

ooooooo
ooooooooooooo
ooo

Комментарий в том блоге:

Darragh McCurragh said:

While I am glad to hear that and I know Google has access to the brightest brains, probably even outdoing Microsoft as an employer in this respect: since the NSA revelation by a certain “Russian agent” called Snowden now we all fret how all of your efforts might be compromised right from the start. Unexpected payloads, undetected trapdoors, man in the middle, anyone?

May 28, 2014 at 12:28 PM

- Намек на сотрудничество Google и NSA.

ooooooo
ooooooooooooo
ooo

Google: весь поиск по HTTPS.

Google.ru details:

Google.ru is shown with a location in Mountain View, California, United States.

IP Address: 173.194.71.94

Yandex: весь поиск по HTTP. Все видят твои запросы!

Раздел 14 - Управление ключами

Протокол Kerberos

Протокол Нидхем-Шредера

Протокол Kerberos

Домены Kerberos

Протокол Диффи-Хеллмана

Инфраструктура открытого ключа (PKI)

oooooo
oooooooooooo
ooo

(TODO: Давать после рассказа о цифровой подписи.)

- система, с пом. которой можно определить, кому принадл.
тот или иной ключ.

Административная, а не математическая/сложностная.

Реально исп-ся.

А генерирует пару (о.к, з.к.) и передает свой о.к. центру
сертификации. ЦС (СА) проверяет, что А тот, за кого он себя
выдает, затем подписывает ключ pk_A своей ЭЦП (создает
сертификат) и возвращает его А.

Теперь, чтобы получать сообщения от В, А отправляет ему свой
ключ и сертификат. В знает о.к. ЦС, проверяет сертификат и
убеждается, что это действительно ключ А.

Идеал: единый ЦС, всемирная инфр-ра о.к.

Невозможно.

ooooooo
ooooooooooooo
ooo

Опр.

Сертификат польз-ля А - это

- сообщение "*pk* - публичный ключ А"
- доп. инф-я, напр., что может делать А
- все это подписано ЭЦП центра сертификации

oooooo
oooooooooooo
ooo

Пример

- 1) VPN. IT-отдел компании - ЦС, предоставляет сотрудникам компании сертификаты.
- 2) Электронные платежные поручения. Банк заверяет о.к. своих клиентов.
- 3) Ассоциация кредитных карт: клиент банка А совершает покупку по кредитке в точке, обслуживающейся в банке В. Банк А должен рассчитаться с банком В. Для этого они должны идентифицировать друг друга. ЦС заверяет ключи каждого банка.

oooooo
oooooooooooo
ooo

Многоуровневые сертификаты.

Банков слишком много, поэтому главный ЦС создает региональные отделения, заверяет их ключи, добавляя в сертификат описание: “предъявитель данного ключа - рег. отделение X. Оно может удостоверять этим ключом ключи других банков”.

После этого сертификат о.к. банка состоит из двух частей: сертификата ЦС, авторизующего ключ регионального отделения, и сертификата отделения о том, что это - ключ данного банка.

Цепочка сертификатов - больше вычислений и потенциальных мест для атаки. Вариант свертывания: получив двухуровневый сертификат, банк отправляет его ЦС. Тот возвращает одноур. сертификат, подписанный ЦС. На практике - не используется.

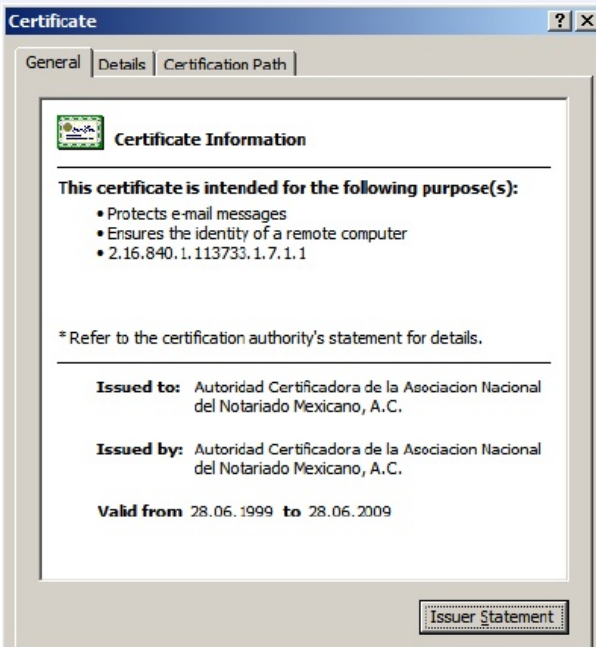
ooooooo
ooooooooooooo
ooo

Параметры сертификата:

- дата начала действия
- дата конца действия
- кем выдан
- кому выдан
- разрешенные действия
- техническая инф-я: значение о.к., тип алгоритма о.к., ...
- и т.д.

Самый распространенный формат сертификатов - X.509 v3.

oooooo
oooooooooooo
ooo



Проблемы:

1) Клиенты должны доверять корневому ЦС. Эти сертификаты подписал их предъявитель (self-signed certificate): издатель=субъект. Пример: GeoTrust Global CA.

В браузере предустановлен список корневых сертификатов, которым можно доверять. Не могут быть отозваны.

Т.е. проверка любого сертификата проходит оффлайн.

И нужно “надежно” распространить лишь сертификаты корневых ЦС. (Вместе с операционной системой.)

2) В сертификат можно упаковать разрешения (запрещения) на очень большое число разнородных действий: открытие файлов, доступ к БД и т.д. Поэтому реализация системы доступа, основанная на сертификатах, получается сложной и далекой от идеальной. (Сообщения о передаче полномочий и т.д.)

3) Отзыв сертификатов.

Отзыв сертификатов.

Клиент скомпрометировал или потерял свой секретный ключ, или сотрудник уволился.

Варианты решения:

1) Список отзыва сертификатов (CRL).

Проверка сертификата - онлайн.

- Каждый член PKI должен иметь доступ к центральной БД с этим списком. Плохо: - точка атаки - не только взлом, но и DoS.

- Распределенная БД. Репликация дорогая. Кроме того, как част репл-ся?

ooooooo
ooooooooooooo
ooo

2) Online Certificate Status Protocol.

- запрос статуса - ответ. Открытым текстом. Меньше нагрузка.

3) Быстрое устаревание.

Сертификат клиента действует от 10 мин. до 24ч. Когда он почти устарел (но еще не), клиент обращается в ЦС за новым сертификатом. Не треб-ся список отозванных сертификатов. Но для обновления сертификата клиент должен быть “постоянно” подключен к ЦС.

ooooooo
ooooooooooooo
ooo

4) Не PKI, а централизованный сервер ключей. Доверенный центр Т, с которым может общаться каждый клиент. Если А хочет послать сообщение В, он получает ключ K_{AB} и его зашифр. копию $E_{K_B}(K_{AB})$, оба подписаны Т. А посылает В подписанный зашифр. сессионный ключ $E_{K_B}(K_{AB})$ и они могут общаться.

Пример 4): система упр. ключами Kerberos.

Замеч.: для сравнительно небольших систем сложность PKI слишком велика, исп-ся серверы ключей. Для большой, распределенной, слабо контролируемой - PKI.

ooooooo
ooooooooooooo
ooo

Жизненный цикл ключа в PKI.

- 1) Создание. Польз-ль А создает пару (з.к., о.к.)
- 2) Сертификация. ЦС сертифицирует о.к. А, наделяет того полномочиями.
- 3) Распространение о.к.
- 4) Активное использование о.к. в транзакциях.
- 5) Пассивное использование. О.к. не применяется в новых транзакциях, но принимаются старые с его использованием. (Пример: эл. почта доходит не мгновенно.)
- 6) Окончание срока действия.

HET