

# Практика 15: Архитектурные аспекты сетевой безопасности

Юрий Литвинов  
yurii.litvinov@gmail.com

16.05.2022

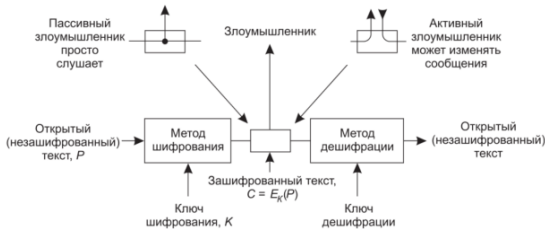
# Сетевая безопасность

- ▶ Почти все сервисы требуют авторизации и обеспечения безопасности
- ▶ Аутентификация — установление личности (точнее, идентичности) участника взаимодействия
  - ▶ Обычно взаимна
- ▶ Авторизация — установление прав на выполнение операции
- ▶ Шифрование — обеспечение конфиденциальности передаваемой информации
- ▶ Также важны:
  - ▶ Целостность — злоумышленник ничего не поменял
  - ▶ Актуальность — злоумышленник не проиграл старое сообщение

# Некоторые соображения

- ▶ Основные уязвимости в современных системах не технические по характеру
- ▶ Большинство попыток взлома — изнутри организации
- ▶ Сетевая безопасность — игра против живого, умного и часто хорошо оснащённого противника
  - ▶ Задача средств безопасности — не сделать взлом невозможным, а сделать его нерентабельным
- ▶ За протоколами безопасности стоит большая наука
  - ▶ Придумать свой хитрый шифр или протокол аутентификации в общем случае очень плохая идея
- ▶ tradeoff между безопасностью и удобством использования

# Шифрование

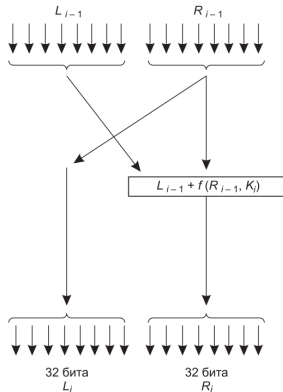
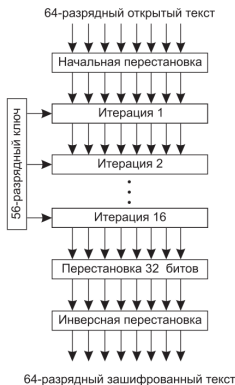


© Э. Таненбаум

- ▶ Алгоритм шифрования считается известным, секретен только ключ
- ▶ Усложнение алгоритма шифрования не всегда повышает криптостойкость

# Шифрование с симметричным ключом

- ▶ Data Encryption Standard (DES, Triple DES)
- ▶ Advanced Encryption Standard (AES, он же Rijndael)



© Э. Таненбаум

# Режимы шифрования, ECB

- ▶ Electronic Code Book — один ключ применяется ко всем блокам
  - ▶ Быстро, надёжно, но не криптостойко

Имя																Должность								Премия															
А   д   а   м   с   ,   Л																е   с   л   и								К   л   е   р   к								\$           1   0							
Б   л   э   к   ,   Р   о																б   и   н								Б   о   с   с								\$   5   0   0   ,   0   0   0							
К   о   л   л   и   н   з   ,																К   и   м								М   е   н   е   д   ж   е   р								\$   1   0   0   ,   0   0   0							
Д   э   в   и   с   ,   Б																о   б   б   и								У   б   о   р   щ   и   к								\$             5							

Байты

← 16 →

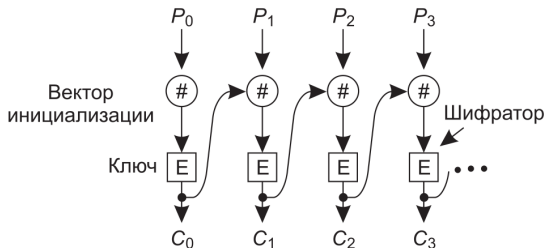
← 8 →

← 8 →

© Э. Таненбаум

# Режимы шифрования, CBC

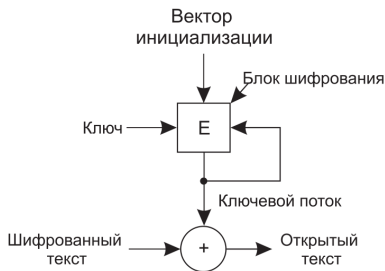
- ▶ Cipher Block Chaining — хог-им следующий блок с зашифрованным предыдущим перед шифровкой
  - ▶ Более криптостоек, не устойчив к ошибкам передачи
  - ▶ Initialization Vector (IV)



© Э. Таненбаум

# Режимы шифрования, SCM

- ▶ Stream Cipher Mode — шифруем IV ключом снова и снова, генерируя ключ бесконечной длины
  - ▶ И xor-им его с шифруемым текстом
  - ▶ Устойчив к ошибкам передачи, довольно быстр
  - ▶ Уязвим к Keystream Reuse Attack  $((P_0 \oplus K_0) \oplus (Q_0 \oplus K_0))$

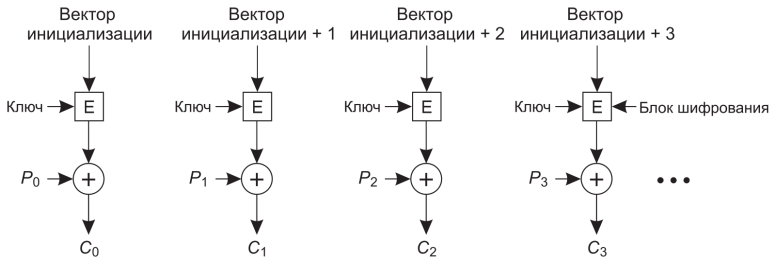


© Э. Таненбаум



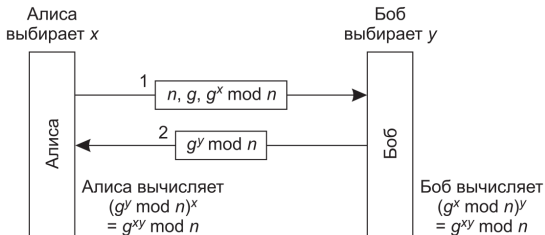
# Режимы шифрования, Counter Mode

- ▶ Counter Mode — шифруем  $IV + i$  для каждого  $i$ -го блока
  - ▶ И xor-им его с шифруемым текстом
  - ▶ Для произвольного доступа к зашифрованным блокам



© Э. Таненбаум

# Алгоритм Диффи-Хеллмана



© Э. Таненбаум

# Атака “Man In The Middle”



© Э. Таненбаум

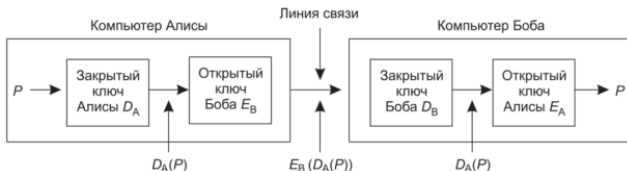
# Шифрование с открытым ключом

- ▶ Алгоритм делится на две части,  $D$  и  $E$ , так, что  $D(E(P)) = P$
- ▶  $D$  очень сложно получить по  $E$ 
  - ▶ Например, найти простые сомножители огромного числа или дискретный логарифм по заданному модулю
- ▶  $E$  не ломается атакой “произвольного открытого текста”
- ▶  $D$  (ключ от  $D$ ) держится в секрете,  $E$  выкладывается в открытый доступ
- ▶ Если Боб хочет послать Алисе сообщение, он берёт её открытый ключ  $E_A$ , шифрует им сообщение  $P$  и отправляет Алисе
- ▶ Алиса дешифрует сообщение, вычисляя  $D_A(E_A(P))$
- ▶ У каждого пользователя своя пара ключей
- ▶ Алгоритмы: RSA, ElGamal, эллиптические шифры

# Цифровые подписи, задачи

- ▶ Получатель может установить личность отправителя
- ▶ Отправитель не может отрицать, что он подписал сообщение
- ▶ Получатель не может сам подделать сообщение и сделать вид, что его послал отправитель

# Цифровые подписи, реализация

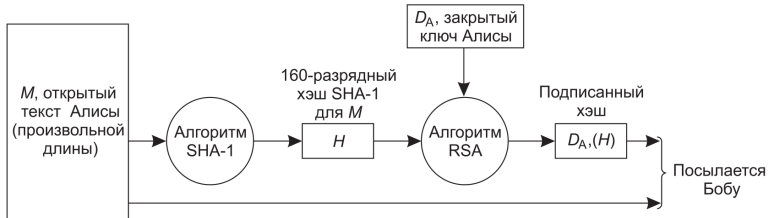


© Э. Таненбаум

- ▶ Надо, чтобы  $D(E(P)) = P$  (это так для большинства криптосхем)
- ▶ Шифровать всё сообщение слишком медленно
- ▶ Message Digest-ы — хорошие хеши сообщений
  - ▶ MD5, SHA-1
- ▶ Подписывается только хеш, это почти так же криптостойко, но в сотни раз быстрее

# SHA-1

- ▶ Считается блоками по 512 бит, возвращает 160-битный дайджест
- ▶ Изменение в одном бите входа даёт совершенно другой выход
- ▶ Если известен  $P$ , очень сложно найти такой  $P'$ , что  $MD(P') = MD(P)$



© Э. Таненбаум

# Сертификаты



© Э. Таненбаум

- ▶ Сертификат — сообщение, подтверждающее идентичность ключа, подписанное Certificate Authority (стандарт X.509)
- ▶ Цепочка сертификатов — CA верхнего уровня подписывает сертификаты CA уровнем ниже, чтобы они могли подписывать сертификаты пользователей
- ▶ Корневые сертификаты — сертификаты, которым принято доверять
- ▶ Самоподписанные сертификаты — не доверенные, используются для отладки



## Сертификаты (2)

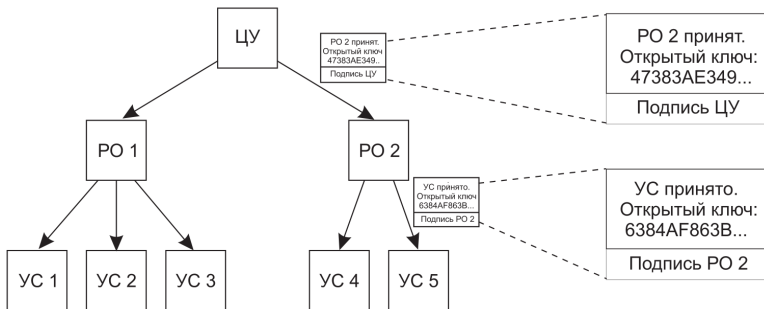
Настоящим удостоверяю, что открытый ключ  
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A  
принадлежит  
Роберту Джону Смиту  
Университетская улица 12345  
Беркли, СА 94702  
1958 род. 5 июля 1958Кг.  
Электронный адрес: bob@superdupernet.com

Хеш SHA-1 данного сертификата подписан закрытым ключом Управления сертификации

© Э. Таненбаум

- ▶ Подписанный у СА сертификат стоит денег (от \$7 до более \$200 в год, в зависимости от типа)
  - ▶ И требует идентификации личности (по паспорту или чему-то такому)
- ▶ Сертификаты всегда выдаются на фиксированное время
- ▶ Сертификат можно отозвать
- ▶ Куча несовместимых форматов: .pem, .p12, .pfx, .der, .cer, .crt

# Certificate Authority

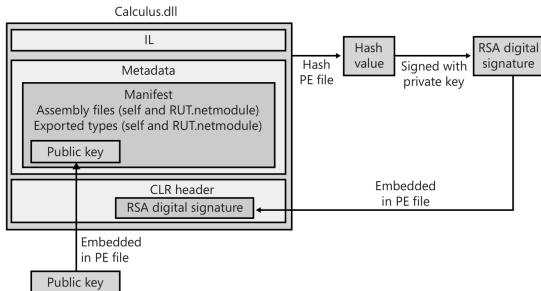


© Э. Таненбаум

- <https://letsencrypt.org/> — автоматически и бесплатно даёт сертификаты, но им почти никто не доверяет

# Применения сертификатов

- ▶ Протокол HTTPS, проверка идентичности сервера
- ▶ Подписывание кода (Windows SmartScreen, Apple Code Signing)
- ▶ Подписывание сборок, сильные имена сборок в .NET



© J. Richter

# Менеджер сертификатов, Windows

## Snap-In в MMC

Консоль1 - [Корень консоли]\Сертификаты - текущий пользователь\Доверенные корневые центры сертификации\Сертификаты]

Файл Действие Вид Избранное Окно Справка

Корень консоли

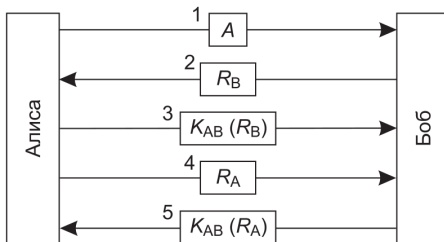
- Сертификаты - текущий пользователь
  - Личное
  - Доверенные корневые центры серт
    - Сертификаты**
    - Доверительные отношения в пред...
    - Промежуточные центры сертифика...
    - Объект пользователя Active Director...
    - Доверенные издатели
    - Сертификаты, к которым нет довери...
    - Сторонние корневые центры серти...
    - Доверенные лица
    - Поставщики сертификатов проверк...
    - ISG Trust
    - MSIEHistoryJournal
    - Запросы заявок на сертификат
    - Доверенные корневые сертификаты

Кому выдан	Кем выдан	Срок действия	Назначения	Имя	Действия
AddTrust External CA Root	AddTrust External CA Root	30.05.2020	Проверка подлинн...	Sectigo (AddTrust	Сертификаты
AffirmTrust Commercial	AffirmTrust Commercial	31.12.2030	Проверка подлинн...	AffirmTrust Comm	Дополнительные действия
Baltimore CyberTrust Root	Baltimore CyberTrust Root	13.05.2025	Проверка подлинн...	DigiCert Baltimor	AddTrust External CA Root
Certum CA	Certum CA	11.06.2027	Проверка подлинн...	Certum	Дополнительные действия
Certum Trusted Network CA	Certum Trusted Network CA	31.12.2029	Проверка подлинн...	Certum Trusted N	
Class 3 Public Primary Certification...	Class 3 Public Primary Certification...	02.08.2028	Проверка подлинн...	VeriSign Class 3 P	
COMODO RSA Certification Auth...	COMODO RSA Certification Authority	19.01.2038	<Все>	<Her>	
Copyright (c) 1997 Microsoft Corp.	Copyright (c) 1997 Microsoft Corp.	31.12.1999	Установка метки вр...	Microsoft Timeste	
CORP/srv-build-cd	CORP/srv-build-cd	07.11.2019	<Все>	<Her>	
Deutsche Telekom Root CA 2	Deutsche Telekom Root CA 2	10.07.2019	Защищенная элект...	Deutsche Telekom	
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	10.11.2031	Проверка подлинн...	DigiCert	
DigiCert Global Root CA	DigiCert Global Root CA	10.11.2031	Проверка подлинн...	DigiCert	
DigiCert Global Root G2	DigiCert Global Root G2	15.01.2038	Проверка подлинн...	DigiCert Global R	
DigiCert Global Root G3	DigiCert Global Root G3	15.01.2038	Проверка подлинн...	DigiCert Global R	
DigiCert High Assurance EV Root...	DigiCert High Assurance EV Root CA	10.11.2031	Проверка подлинн...	DigiCert	
DO_NOT_TRUST_FiddlerRoot	DO_NOT_TRUST_FiddlerRoot	02.10.2023	Проверка подлинн...	DO_NOT_TRUST_	
DO_NOT_TRUST_FiddlerRoot	DO_NOT_TRUST_FiddlerRoot	02.10.2023	<Her>	<Her>	
DST Root CA X3	DST Root CA X3	30.09.2021	Защищенная элект...	DST Root CA X3	
Entrust Root Certification Authority	Entrust Root Certification Authority	27.11.2026	Проверка подлинн...	Entrust	
Entrust Root Certification Authori...	Entrust Root Certification Authority ..	07.12.2030	Проверка подлинн...	Entrust.net	
Entrust.net Certification Authority...	Entrust.net Certification Authority (2...	24.07.2029	Проверка подлинн...	Entrust (2048)	
Equifax Secure Certificate Authority	Equifax Secure Certificate Authority	22.08.2018	Защищенная элект...	GeoTrust	
GeoTrust Global CA	GeoTrust Global CA	21.05.2022	Проверка подлинн...	GeoTrust Global C	
GeoTrust Primary Certification Au...	GeoTrust Primary Certification Auth...	02.12.2037	Проверка подлинн...	GeoTrust Primary	
GlobalSign	GlobalSign	18.03.2029	Проверка подлинн...	GlobalSign Root	
GlobalSign	GlobalSign	15.12.2021	Проверка подлинн...	Google Trust Ser	

# OpenSSL

- ▶ OpenSSL — библиотека и набор инструментов для криптографии и работы с протоколами SSL/TLS
- ▶ Стандарт де-факто для работы с открытыми ключами, сертификатами и т.д.
- ▶ Как сгенерить самоподписанный сертификат:  
`openssl req -x509 -nodes -days 365`  
`-newkey rsa:2048 -keyout privatekey.key`  
`-out certificate.crt`

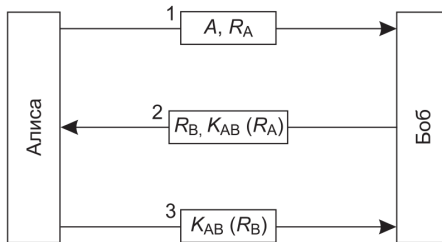
# Аутентификация Challenge-Response с общим ключом



© Э. Таненбаум

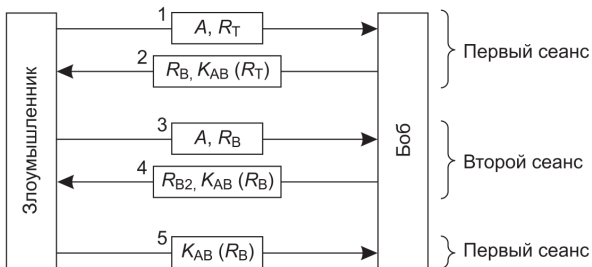
- ▶  $R_B$  — **nonce** (number used once), для предотвращения атаки повтором
- ▶  $K_{AB}$  — общий ключ

# “Упрощённый” протокол



© Э. Таненбаум

# Зеркальная атака

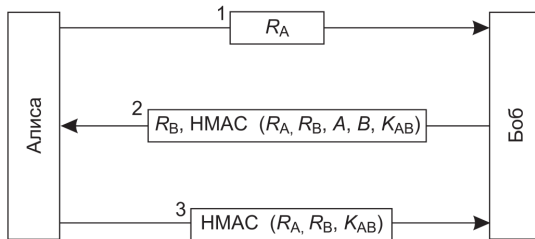


© Э. Таненбаум

**Разработать корректный протокол аутентификации сложнее, чем это может показаться**



# Правильный протокол



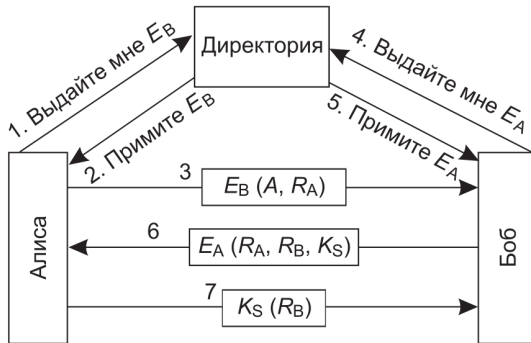
© Э. Таненбаум

## ► HMAC — Hashed Message Authentication Code

# Как на самом деле

- ▶ Basic Authentication — логин и пароль передаются нешифрованными в заголовке HTTP-запроса
- ▶ HTTPS обеспечивает безопасность
- ▶ Сервер возвращает Access Token
- ▶ Access Token предъявляется при каждом следующем запросе
  - ▶ Имеет ограниченное время жизни, но его можно продлить
- ▶ Пароли не хранятся на сервере, хранятся их хеши
  - ▶ Salt — случайное число, дописываемое к паролю на стороне сервера, хранится вместе с хешем пароля
  - ▶ Если базу паролей украдут, узнать исходные пароли очень сложно

# Аутентификация с открытым ключом



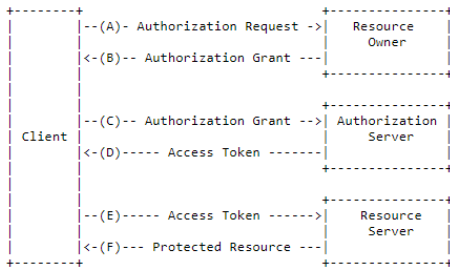
© Э. Таненбаум

- ▶  $E_A, E_B$  — открытые ключи Алисы и Боба
- ▶  $R_A, R_B$  — nonce

# OAuth 2

- ▶ Позволяет разрешить пользование ресурсом, не раскрывая хозяину ресурса логин и пароль пользователя
  - ▶>Login по аккаунту в Google или аккаунту в VK
- ▶ Роли:
  - ▶ Client — приложение, пытающееся получить доступ
  - ▶ Resource Server — сервер, хранящий защищённую информацию. К нему пытается получить доступ клиент
  - ▶ Resource Owner — пользователь, владеющий защищённой информацией
  - ▶ Authorization Server — сервер, выдающий клиенту токен на доступ к ресурсному серверу

# Протокол



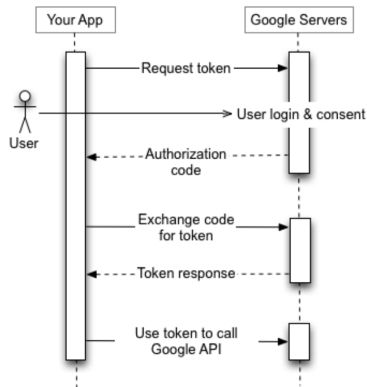
© RFC 6749

# Детали

- ▶ Access Token — выдаётся авторизационным сервером и посылается с каждым запросом, ограниченное время жизни
- ▶ Refresh Token — выдаётся авторизационным сервером, используется для получения нового Access Token
- ▶ Scope — к какой части ресурса даёт доступ Access Token

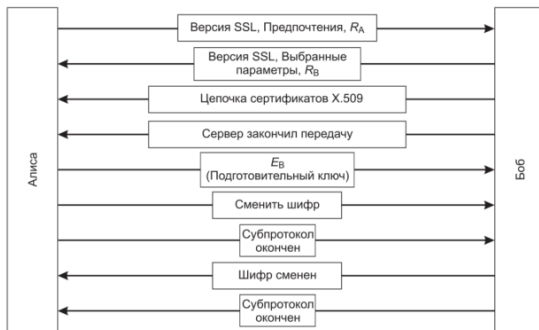
# Пример: Google OAuth 2.0

- ▶ Google Developer Console, Client ID и Client Secret
- ▶ Scope
- ▶ Consent Screen



© <https://developers.google.com>

# HTTPS

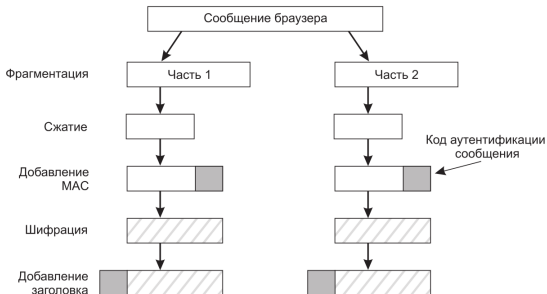


© Э. Таненбаум

- ▶ SSL (Secure Sockets Layer)
- ▶ HTTPS — HTTP через SSL
- ▶ Порт 443
- ▶ Аутентифицируется только сервер



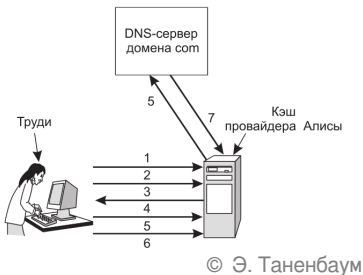
# SSL, транспортный субпротокол



© Э. Таненбаум

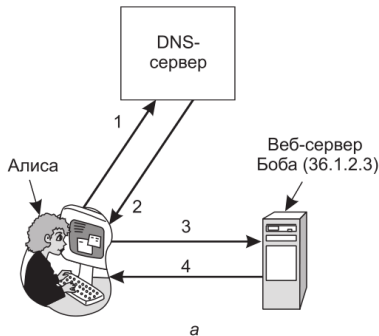
- ▶ Triple DES + SHA-1
- ▶ Или RC4 со 128-битным ключом + MD5
- ▶ TLS — Transport Layer Security (продвинутый SSL)

# DNS Spoofing



1. Запрос foobar.trudy-the-intruder.com (чтобы trudy-the-intruder.com попал в кеш провайдера)
2. Запрос www.trudy-the-intruder.com (чтобы получить следующий порядковый номер провайдера)
3. Запрос об адресе www.trudy-the-intruder.com к нашему DNS
4. Запрос к bob.com
5. Запрос о bob.com к DNS зоны com
6. Подделанный ответ о bob.com
7. Настоящий ответ, отвергнутый, потому что уже поздно

# Результат



1. Мне нужен IP-адрес Боба
2. 36.1.2.3 (IP-адрес Боба)
3. GET index.HTML
4. Домашняя страничка Боба



1. Мне нужен IP-адрес Боба
2. 42.9.9.9 (IP-адрес Трудя)
3. GET index.HTML
4. Подделанная взломщиком страница Боба

© Э. Таненбаум

# Как это всё отлаживать

И ломать

- ▶ Fiddler — кроссплатформенный отладочный прокси
  - ▶ Перехват HTTP-трафика
  - ▶ Man-In-The-Middle-атака с самоподписанными сертификатами
    - ▶ Расшифровка HTTPS-трафика на лету
  - ▶ Возможность модифицировать HTTP-пакеты, повторять пакеты и т.д.
- ▶ Wireshark — когда Fiddler-а мало
  - ▶ Перехват пакетов на низком уровне
  - ▶ Умеет даже ставить себя как драйвер USB и читать USB-пакеты