

Лекция 15: Основы сетевой безопасности

Юрий Литвинов
yurii.litvinov@gmail.com

06.06.2022

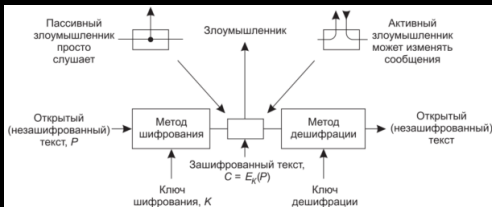
Сетевая безопасность

- ▶ 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

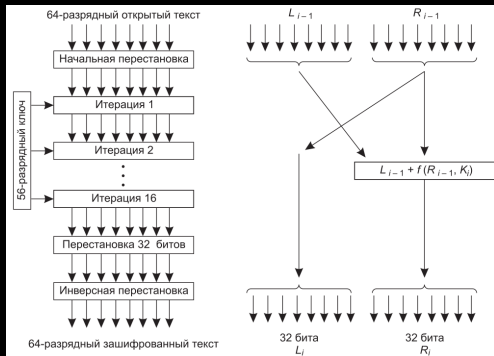


© Э. Таненбаум

- ▶ 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)



Режимы шифрования, ЕСВ

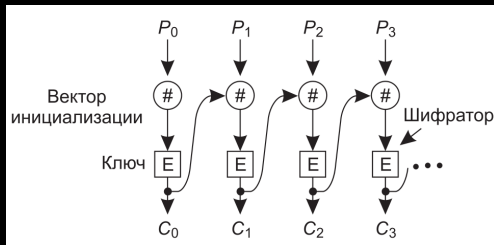
- ▶ Electronic Code Book — один ключ применяется ко всем блокам
 - ▶ Быстро, надёжно, но не криптостойко

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 →

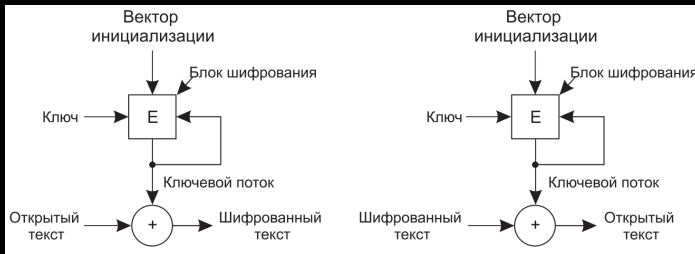
Режимы шифрования, CBC

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



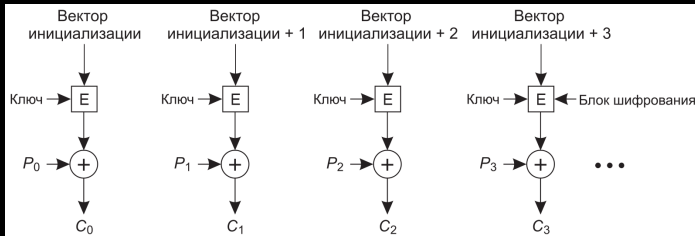
Режимы шифрования, SCM

- ▶ 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 $((P_0 \oplus K_0) \oplus (Q_0 \oplus K_0))$

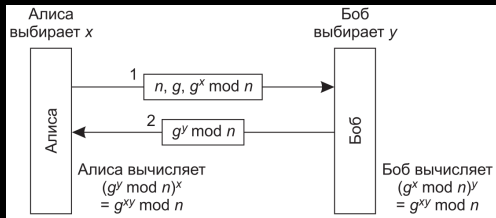


Режимы шифрования, Counter Mode

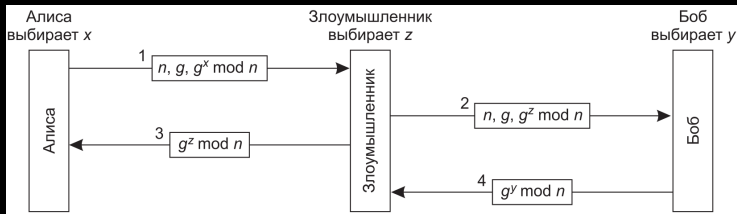
- ▶ Counter Mode — шифруем $IV + i$ для каждого i -го блока
 - ▶ И xor-им его с шифруемым текстом
 - ▶ Для произвольного доступа к зашифрованным блокам



Алгоритм Диффи-Хеллмана



Атака “Man In The Middle”



Шифрование с открытым ключом

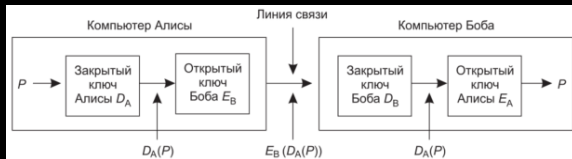
Или почему нельзя отдать ключи от Telegram

- ▶ 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

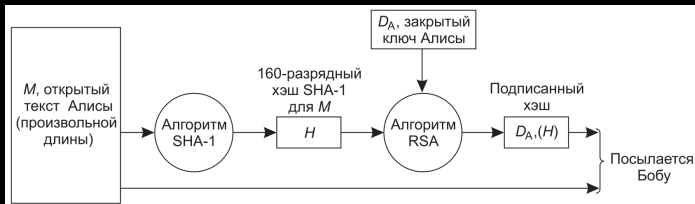


© Э. Таненбаум

- ▶ 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 $MD(P') = MD(P)$



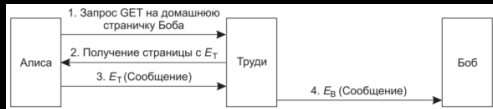
Атака дней рождения

Уважаемый господин декан,

Это [письмо | обращение] отражает мое [искреннее | откровенное] [мнение | суждение] о проф. Томе Уилсоне, являющемся [кандидатом | претендентом] на профессорскую должность в [настоящее время | этом году]. Я [знакома | работала] с проф. Уилсоном в течение [почти | около] шести лет. Он является [слабым | недостаточно талантливым] [исследователем | ученым], почти не известным в той области науки, которой он занимается. В его работах практически не заметно понимания [ключевых | главных] [проблем | вопросов] современности.

[Более | Кроме] того, он также не является сколько-нибудь [уважаемым | ценным] [преподавателем | педагогом]. Его студенты дают его [занятиям | лекциям] [самые низкие | негативные] оценки. Он самый непопулярный [преподаватель | учитель] нашей кафедры, [славящийся | печально известный] своей [привычкой | склонностью] [высмеивать | ставить в неудобное положение] студентов, осмелившихся задавать вопросы на его [лекциях | занятиях].

Сертификаты



© Э. Таненбаум

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

Сертификаты (2)

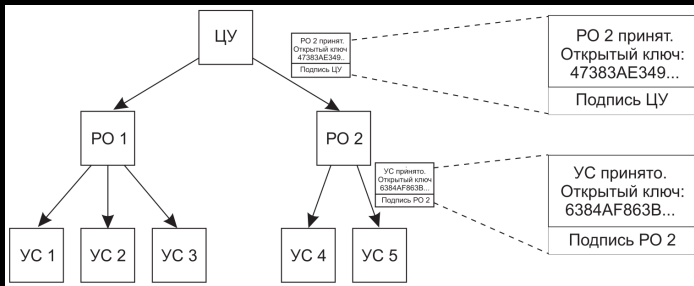
Настоящим удостоверяю, что открытый ключ
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
принадлежит
Роберту Джону Смигу
Университетская улица 12345
Беркли, CA 94702
1958 род. 5 июля 1958Kг.
Электронный адрес: bob@superdupernet.com

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 (например, по паспорту)
 - ▶ Есть бесплатно и без бюрократии, но ими далеко не всё можно подписать
- ▶ Сертификаты всегда выдаются на фиксированное время
- ▶ Сертификат можно отозвать
- ▶ Куча несовместимых форматов: .pem, .p12, .pfx, .der, .cer, .crt

Certificate Authority

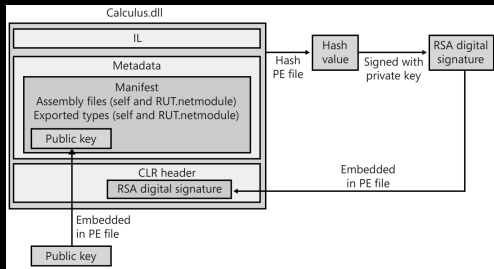


© Э. Таненбаум

- <https://letsencrypt.org/> — автоматически и бесплатно даёт сертификаты, но они подтверждают только владение доменом, а не личность хозяина

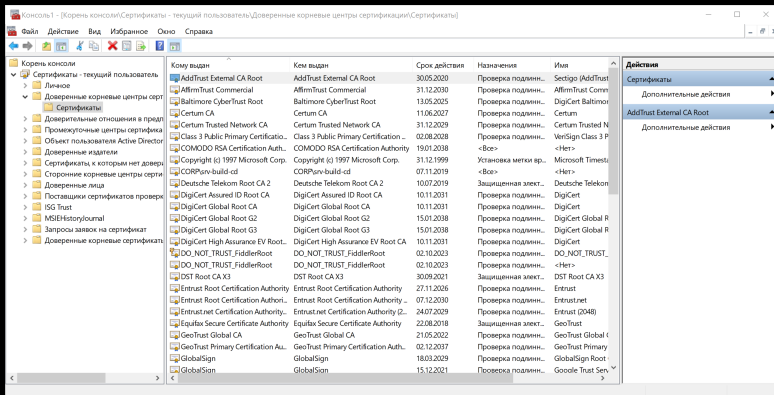
Применения сертификатов

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



Менеджер сертификатов, Windows

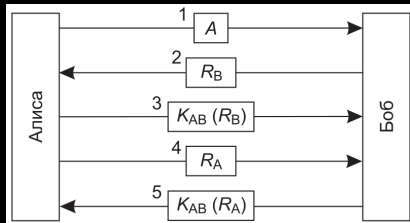
Snap-In в MMC



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`

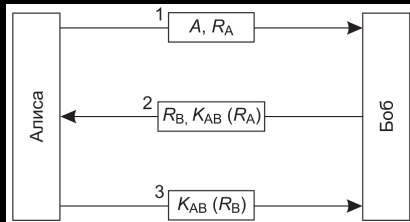
Аутентификация Challenge-Response с общим ключом



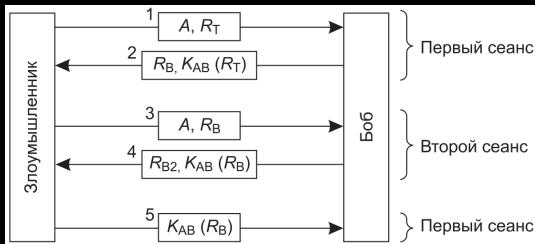
© Э. Таненбаум

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

“Упрощённый” протокол



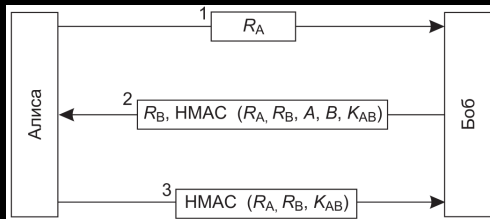
Зеркальная атака



© Э. Таненбаум

Разработать корректный протокол аутентификации сложнее, чем это может показаться

Правильный протокол



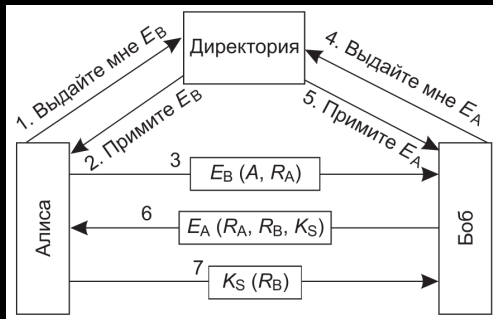
© Э. Таненбаум

- 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

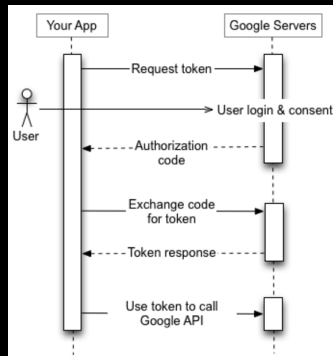


Детали

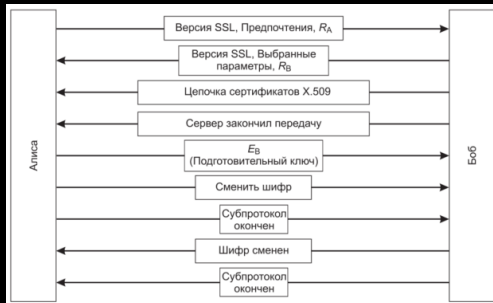
- ▶ Access Token — выдаётся авторизационным сервером и посылается с каждым запросом, ограниченное время жизни
- ▶ Refresh Token — выдаётся авторизационным сервером, используется для получения нового Access Token
- ▶ Scope — к какой части ресурса даёт доступ Access Token

Пример: Google OAuth 2.0

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



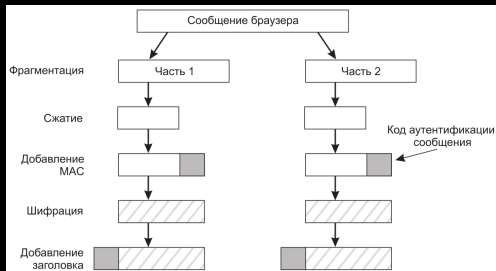
HTTPS



© Э. Таненбаум

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

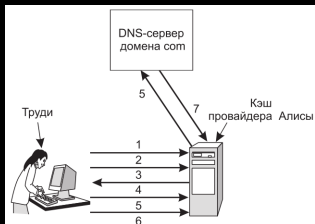
SSL, транспортный субпротокол



© Э. Таненбаум

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

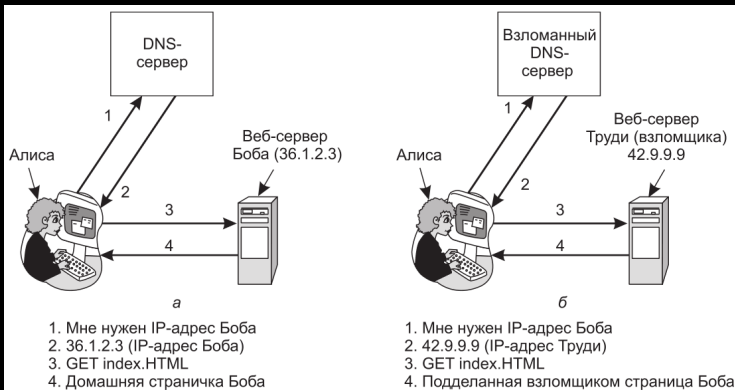
DNS Spoofing



© Э. Таненбаум

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



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