Лекция 12: Цифровые подписи и PKI

Тема: Целостность данных, цифровые подписи и PKI

Технический университет Молдовы

Лектор: Максим Масютин

Введение

Добро пожаловать на лекцию 12, заключительную лекцию в нашей серии по криптографии. Сегодня мы объединим все, что изучили -- симметричное шифрование, асимметричную криптографию и хеш-функции -- для исследования цифровых подписей и инфраструктуры открытых ключей.

Задумайтесь на мгновение о физическом мире. Когда вы подписываете контракт, подпись выполняет несколько функций: она доказывает, что вы согласились с условиями, идентифицирует вас как подписавшего, и любое изменение документа будет заметно. Цифровые подписи решают те же задачи в цифровом мире, но с ещё более надёжными гарантиями.

К концу сегодняшней лекции вы поймёте, как работают цифровые подписи, изучите как классические алгоритмы, так и постквантовые альтернативы, и узнаете, как инфраструктура открытых ключей обеспечивает отношения доверия, которые делают возможной безопасную коммуникацию в интернете.

Часть 1: Основы цифровых подписей

Что такое цифровая подпись?

Цифровая подпись -- это криптографический механизм, обеспечивающий три критически важных свойства безопасности:

Аутентификация: Подпись удостоверяет личность подписавшего. Только тот, кто обладает закрытым ключом, мог создать подпись.

Целостность: Любое изменение подписанных данных, каким бы малым оно ни было, делает подпись недействительной. Изменение даже одного бита обнаруживается.

Неотрекаемость: Подписавший не может впоследствии отрицать факт подписания документа. Подпись является обязательным доказательством его

действия.

Эти свойства делают цифровые подписи юридически равноценными собственноручным подписям в большинстве юрисдикций, а во многих отношениях и превосходящими их -- цифровую подпись нельзя скопировать с одного документа на другой, и подделка всегда обнаруживается.

Процесс подписания и проверки

Цифровые подписи используют асимметричную криптографию с обратным применением ключей:

Подписание (операция с закрытым ключом):

1. Хешировать сообщение для создания дайджеста фиксированного размера
2. Применить алгоритм подписи с закрытым ключом к хешу
3. На выходе получается подпись

Проверка (операция с открытым ключом):

1. Хешировать полученное сообщение тем же алгоритмом
2. Применить алгоритм проверки с открытым ключом к подписи
3. Сравнить восстановленный хеш с вычисленным хешем
4. Совпадение указывает на действительную подпись

Хеш-функция играет ключевую роль -- она позволяет подписывать сообщения любой длины с помощью операций асимметричной криптографии фиксированного размера, а свойство однонаправленности не позволяет злоумышленникам восстанавливать поддельные сообщения из подписей.

Требования к безопасности

Безопасная схема цифровой подписи должна удовлетворять следующим требованиям:

1. **Экзистенциальная невозможность подделки**: Злоумышленник не может создать действительную подпись для какого-либо сообщения, даже после наблюдения множества подписей на других сообщениях (устойчивость к атаке с выбранным сообщением)
2. **Строгая невозможность подделки**: Даже при наличии подписи на сообщении m злоумышленник не может создать другую действительную подпись для m
3. **Атаки только по ключу**: Невозможность подделки подписей, зная только открытый ключ

4. **Атаки с известным сообщением**: Невозможность подделки подписей после наблюдения подписей на известных сообщениях
5. **Атаки с выбранным сообщением**: Невозможность подделки подписей даже после запроса подписей на сообщениях, выбранных злоумышленником

Современные схемы подписей имеют доказанную безопасность против атак с выбранным сообщением при стандартных криптографических допущениях.

Часть 2: Подписи RSA

Основы подписи RSA

Подписи RSA используют ту же математическую основу, что и шифрование RSA, но с обратным использованием ключей:

Подписание: s = m^d mod n (операция с закрытым ключом) **Проверка**: m = s^e mod n (операция с открытым ключом)

Где d -- закрытая экспонента, e -- открытая экспонента, а n -- модуль.

Как и в случае с шифрованием RSA, необработанные "учебные" подписи RSA небезопасны. Детерминированная природа допускает различные атаки, а математические свойства модульного возведения в степень могут быть использованы злоумышленником.

Подписи PKCS#1 v1.5

Более старая схема подписи PKCS#1 v1.5 добавляет дополнение (padding):

1. Хешировать сообщение
2. Применить кодирование: EM = 0x00 || 0x01 || PS || 0x00 || DigestInfo
   - PS -- строка дополнения из байтов 0xFF
   - DigestInfo содержит идентификатор алгоритма хеширования и хеш-значение
3. Преобразовать EM в целое число m
4. Вычислить подпись: s = m^d mod n

Подписи PKCS#1 v1.5 не считаются взломанными, в отличие от шифрования PKCS#1 v1.5. Однако ошибки реализации приводили к уязвимостям (подделка подписи Блейхенбахера), и схема не имеет доказательства безопасности.

RSA-PSS (вероятностная схема подписи)

RSA-PSS, стандартизированная в PKCS#1 v2.1 и RFC 8017, является рекомендуемой схемой подписи RSA. Она добавляет рандомизацию и имеет

доказательство безопасности в модели случайного оракула.

Кодирование PSS:

1. Сгенерировать случайную соль
2. Вычислить M' = (0x00 * 8) || mHash || salt
3. Вычислить H = Hash(M')
4. Сформировать DB = PS || 0x01 || salt
5. Вычислить dbMask = MGF(H, length)
6. Вычислить maskedDB = DB XOR dbMask
7. EM = maskedDB || H || 0xBC
8. Подписать: s = EM^d mod n

Соль обеспечивает рандомизацию, делая подписи вероятностными -- одно и то же сообщение, подписанное дважды, даёт разные подписи.

Преимущества RSA-PSS по сравнению с PKCS#1 v1.5:

- Доказуемая безопасность (модель случайного оракула)
- Рандомизация (отсутствие детерминированных атак)
- Более строгие границы безопасности

Параметры RSA-PSS:

- Хеш-функция: рекомендуются SHA-256 или SHA-384
- Функция генерации маски: MGF1 с той же хеш-функцией
- Длина соли: равна длине выхода хеш-функции (32 байта для SHA-256)

Размеры ключей для подписи RSA

Те же рекомендации по размеру ключей применимы, что и для шифрования RSA:

Размер ключа	Уровень безопасности	Рекомендация
RSA-1024	~80 бит	НЕ ИСПОЛЬЗОВАТЬ
RSA-2048	~112 бит	Минимально допустимый
RSA-3072	~128 бит	Рекомендуемый
RSA-4096	~140+ бит	Высокая безопасность

RSA-2048 -- текущий минимум для новых систем. RSA-3072 рекомендуется для данных, которые должны оставаться защищёнными после 2030 года.

Часть 3: Подписи на эллиптических кривых

ECDSA (алгоритм цифровой подписи на эллиптических кривых)

ECDSA -- это вариант DSA (алгоритма цифровой подписи) на эллиптических кривых. Он обеспечивает тот же уровень безопасности, что и подписи RSA, при значительно меньшем размере ключей.

Генерация ключей:

1. Выбрать параметры кривой (кривая, базовая точка G, порядок n)
2. Выбрать случайный закрытый ключ d из [1, n-1]
3. Вычислить открытый ключ Q = d * G

Подписание:

1. Хешировать сообщение: e = H(m)
2. Выбрать случайное k из [1, n-1]
3. Вычислить точку (x1, y1) = k * G
4. Вычислить r = x1 mod n (перезапустить, если r = 0)
5. Вычислить s = k^(-1) *(e* + *r*d) mod n (перезапустить, если s = 0)
6. Подпись -- это (r, s)

Проверка:

1. Проверить r, s из [1, n-1]
2. Хешировать сообщение: e = H(m)
3. Вычислить u1 = e * s^(-1) mod n
4. Вычислить u2 = r * s^(-1) mod n
5. Вычислить точку (x1, y1) = u1*G* + *u2*Q
6. Подпись верна, если r = x1 mod n

ECDSA с кривыми NIST:

- P-256: подписи 64 байта, безопасность ~128 бит
- P-384: подписи 96 байт, безопасность ~192 бита
- P-521: подписи 132 байта, безопасность ~256 бит

Критическое требование к случайным числам

ECDSA имеет критическую уязвимость: случайное число k, уникальное для каждой подписи, должно быть действительно случайным и секретным. Если k

когда-либо повторяется или k имеет какую-либо предвзятость, закрытый ключ может быть восстановлен.

Известные случаи провала:

- Подпись кода PlayStation 3: Sony использовала постоянное k, закрытый ключ был восстановлен
- Биткоин-кошельки на Android: слабый генератор случайных чисел, средства похищены
- Различные реализации: утечки через тайминг раскрывали значения k

Именно поэтому был разработан детерминированный ECDSA (RFC 6979), который вычисляет k из сообщения и закрытого ключа, устраняя требование случайности.

EdDSA (алгоритм цифровой подписи на кривых Эдвардса)

EdDSA, определённый в RFC 8032, -- это современная схема подписи, разработанная для предотвращения недостатков ECDSA:

Ключевые особенности:

- Детерминированность: не требуется случайное число для каждой подписи
- Быстродействие: одно скалярное умножение для подписания
- Безопасность против атак по побочным каналам по проектированию
- Компактные подписи: 64 байта
- Построен на кривых Эдвардса (быстрых и безопасных)

Ed25519 -- это EdDSA с использованием Curve25519 (точнее, edwards25519):

- 256-битная кривая, безопасность ~128 бит
- 32-байтовые открытые ключи, 64-байтовые подписи
- Хеш-функция: SHA-512

Ed448 -- это EdDSA с использованием Curve448:

- 448-битная кривая, безопасность ~224 бита
- 57-байтовые открытые ключи, 114-байтовые подписи
- Хеш-функция: SHAKE256

Процесс подписания EdDSA:

1. Хешировать закрытый ключ для получения секретного скаляра и префикса
2. Вычислить открытый ключ A = scalar * G
3. Вычислить r = H(prefix || message) (детерминированно!)
4. Вычислить R = r * G

5. Вычислить k = H(R || A || message)
6. Вычислить S = r + k * scalar
7. Подпись -- это (R, S) -- всего 64 байта

Ed25519 -- это рекомендуемый алгоритм подписи для новых приложений. Он используется в:

- SSH (по умолчанию для новых ключей)
- Протоколе Signal
- TLS 1.3
- WireGuard
- Многих блокчейн-платформах

Часть 4: Дополнительные схемы подписей

Подписи Шнорра

Подписи Шнорра, изобретённые Клаусом-Петером Шнорром в 1989 году, элегантны и эффективны. Они были защищены патентом до 2008 года, что привело к разработке DSA и ECDSA в качестве альтернатив. Теперь, когда патент истёк, подписи Шнорра набирают популярность.

Преимущества подписей Шнорра:

- Проще, чем ECDSA
- Доказуемая безопасность в модели случайного оракула
- Линейность: подписи могут быть агрегированы
- Пакетная проверка: проверка множества подписей быстрее, чем по отдельности

Агрегация ключей особенно ценна: несколько подписывающих сторон могут создать единую подпись, которая проверяется по агрегированному открытому ключу. Это позволяет:

- Мультиподписные схемы с экономией места
- Масштабирование блокчейнов (Taproot в Bitcoin)
- Пороговые подписи

Ed25519, по сути, является вариантом подписей Шнорра на кривой Эдвардса, унаследовавшим эти преимущества.

Подписи BLS

Подписи BLS (Boneh-Lynn-Shacham) используют билинейные спаривания на эллиптических кривых. Они обладают уникальными свойствами, невозможными в других схемах:

Агрегация подписей: Множество подписей на разных сообщениях от разных подписывающих сторон могут быть объединены в одну подпись. Для проверки нужны все сообщения и открытые ключи, но только одно вычисление спаривания на подпись.

Пороговые подписи: Схема (t, n) с порогом требует подписей t из n участников, создавая подпись стандартного вида. Другие не могут определить, что это пороговая подпись.

Применения:

- Консенсус в блокчейнах (Ethereum 2.0, Chia)
- Распределённые системы
- Агрегация сертификатов

Недостатки BLS:

- Медленнее, чем EdDSA (спаривания вычислительно затратны)
- Более сложная реализация
- Более широкие криптографические допущения

Постквантовые подписи

Как было рассмотрено в лекции 10, квантовые компьютеры угрожают всей существующей криптографии с открытым ключом. NIST (National Institute of Standards and Technology, Национальный институт стандартов и технологий) стандартизировал постквантовые алгоритмы подписей:

ML-DSA (FIPS 204) -- на основе Dilithium:

- На основе решёток
- Быстрое подписание и проверка
- Умеренный размер подписей (2,4-4,6 КБ)
- Основная замена для ECDSA/EdDSA

SLH-DSA (FIPS 205) -- на основе SPHINCS+:

- На основе хешей (без допущений о решётках)
- Очень консервативная безопасность
- Большие подписи (17-50 КБ)
- Медленный, но надёжный

Для текущего развёртывания:

- Используйте Ed25519 или ECDSA для немедленной безопасности
- Добавьте ML-DSA в гибридных схемах для квантовой устойчивости
- Используйте SLH-DSA для долгосрочных архивных подписей, где допустим большой размер

Часть 5: Основы инфраструктуры открытых ключей

Проблема доверия

Цифровые подписи доказывают, что сообщение было подписано тем, кто владеет определённым закрытым ключом. Но как узнать, какой открытый ключ кому принадлежит?

Если Алиса хочет проверить подпись Боба, ей нужен открытый ключ Боба. Но если злоумышленник передаст Алисе поддельный "открытый ключ Боба" (который на самом деле является открытым ключом злоумышленника), злоумышленник сможет выдать себя за Боба.

Это проблема доверия, которую решает инфраструктура открытых ключей.

Компоненты PKI

Инфраструктура открытых ключей состоит из:

Удостоверяющий центр (CA): Доверенная организация, которая проверяет личность и выпускает сертификаты. CA подписывает сертификаты своим закрытым ключом, подтверждая связь между личностью и открытым ключом.

Центр регистрации (RA): Проверяет личность заявителей на получение сертификатов до того, как CA выпустит сертификаты. Может быть той же организацией, что и CA.

Репозиторий сертификатов: Хранит и распространяет сертификаты, часто через LDAP, HTTP или встроенные в приложения.

Инфраструктура отзыва сертификатов: Механизмы для аннулирования сертификатов до истечения срока действия (CRL, OCSP).

Конечные субъекты: Пользователи, серверы, устройства или приложения, имеющие сертификаты.

Модель сети доверия

Прежде чем обсуждать модель CA, стоит упомянуть альтернативу: сеть доверия, используемую в PGP/GPG.

В сети доверия:

- Пользователи подписывают ключи друг друга напрямую
- Доверие транзитивно: если вы доверяете Алисе, а Алиса подписала ключ Боба, вы имеете определённую степень доверия к ключу Боба
- Нет центрального органа

Преимущества:

- Децентрализованность, отсутствие единой точки отказа
- Пользователи контролируют свои собственные решения о доверии

Недостатки:

- Сложные вычисления доверия
- Трудность начальной настройки (новые пользователи не имеют связей доверия)
- Плохая масштабируемость для идентификации в масштабах интернета

Сеть доверия работает для сообществ, но не масштабируется до глобального интернета, поэтому для TLS и подписи кода доминирует модель CA.

Часть 6: Сертификаты X.509

Структура сертификата

X.509 -- это стандартный формат сертификатов открытых ключей, определённый в RFC 5280. Сертификат содержит:

Версия: В настоящее время v3 (обозначается значением 2, так как версии индексируются с нуля)

Серийный номер: Уникальный идентификатор, назначаемый удостоверяющим центром

Алгоритм подписи: Алгоритм, использованный CA для подписания (например, sha256WithRSAEncryption)

Издатель: Уникальное имя (DN) удостоверяющего центра

Период действия: Даты "Не ранее" и "Не позднее"

Субъект: Уникальное имя владельца сертификата

Информация об открытом ключе субъекта: Алгоритм и открытый ключ

Расширения (v3): Дополнительная информация, включая:

- Key Usage: Для чего может использоваться ключ
- Extended Key Usage: Конкретные области применения (TLS-сервер, подпись кода)
- Subject Alternative Name: Дополнительные идентификаторы (DNS-имена, IP-адреса)
- Basic Constraints: Является ли данный сертификат сертификатом CA
- Certificate Policies: Идентификаторы и квалификаторы политик
- CRL Distribution Points: Где найти информацию об отзыве
- Authority Information Access: Местоположение OCSP-респондера, сертификат CA

Подпись: Подпись CA на содержимом сертификата

Уникальные имена

Уникальное имя (Distinguished Name, DN) идентифицирует субъект в формате каталога X.500:

```
CN=www.example.com
O=Example Corporation
OU=IT Department
L=San Francisco
ST=California
C=US
```

Для TLS-сертификатов Common Name (CN) традиционно содержало имя хоста, но современная практика вместо этого использует расширение Subject Alternative Name (SAN), так как CN для этой цели считается устаревшим.

Типы сертификатов

Сертификат с проверкой домена (DV): CA проверяет только контроль над доменом (например, электронное письмо на admin@domain или HTTP-запрос). Выпускается за минуты, минимальная стоимость, подходит для шифрования.

Сертификат с проверкой организации (OV): CA проверяет идентичность организации по деловым реестрам. Занимает дни, умеренная стоимость, отображает название организации.

Сертификат с расширенной проверкой (EV): Строгая проверка личности, проверка юридического существования, проверка физического присутствия. Занимает недели, наивысшая стоимость. Браузеры раньше отображали зелёную строку (в настоящее время устарело в большинстве браузеров).

С точки зрения шифрования, все три типа обеспечивают идентичную безопасность. Различие заключается в степени проверки личности.

Цепочки сертификатов

Сертификаты образуют цепочки доверия:

1. **Корневой сертификат CA**: Самоподписанный, является доверенным, поскольку установлен в хранилище доверия
2. **Промежуточный сертификат CA**: Подписан корневым или другим промежуточным центром
3. **Сертификат конечного субъекта**: Подписан промежуточным CA

Зачем нужны промежуточные центры?

- Корневые ключи хранятся в автономном режиме в защищённых помещениях
- Компрометация промежуточного центра затрагивает только выпущенные им сертификаты
- Можно иметь разные промежуточные центры для разных целей/политик

При проверке сертификата:

1. Построить цепочку от конечного субъекта до доверенного корня
2. Проверить каждую подпись в цепочке
3. Проверить сроки действия
4. Проверить статус отзыва
5. Проверить ограничения на имя и использование ключа

Часть 7: Жизненный цикл сертификата

Выпуск сертификата

Традиционный процесс выпуска сертификата:

1. **Генерация пары ключей**: Конечный субъект создаёт пару открытого и закрытого ключей
2. **Создание CSR**: Запрос на подпись сертификата содержит открытый ключ и информацию об идентификации, подписанные закрытым ключом для подтверждения владения
3. **Отправка CSR**: Передача в CA или RA
4. **Проверка идентичности**: CA проверяет идентичность в соответствии с типом сертификата

5. **Выпуск сертификата**: CA создаёт и подписывает сертификат
6. **Доставка сертификата**: Сертификат отправляется заявителю

Отзыв сертификата

Сертификаты могут потребовать отзыва до истечения срока действия:

- Компрометация закрытого ключа
- Владелец сертификата покидает организацию
- Изменение информации в сертификате
- Компрометация CA

Списки отозванных сертификатов (CRL):

- CA публикует подписанный список отозванных серийных номеров
- Клиенты загружают и кешируют CRL
- Проблемы: CRL увеличиваются в размере, кеширование задерживает отзыв

Протокол онлайн-проверки статуса сертификата (OCSP):

- Проверка статуса сертификата в реальном времени
- Клиент отправляет серийный номер сертификата OCSP-респондеру
- Респондер возвращает подписанный статус (действителен, отозван, неизвестен)
- Проблемы: конфиденциальность (CA видит каждое соединение), зависимость от доступности

Прикрепление OCSP (OCSP Stapling):

- Сервер периодически запрашивает свой собственный OCSP-ответ
- Включает ответ в рукопожатие TLS
- Клиент получает актуальный статус без обращения к CA
- Лучшее из обоих подходов: статус в реальном времени, конфиденциальность сохранена

Краткосрочные сертификаты

Современный подход: выпуск сертификатов с очень коротким сроком действия (от часов до дней) вместо использования механизма отзыва:

- Не требуется инфраструктура отзыва
- Окно компрометации ограничено
- Требуется автоматический выпуск (см. ACME)

Часть 8: TLS 1.3

Почему TLS важен

Протокол Transport Layer Security (TLS) обеспечивает безопасность большей части интернет-коммуникаций:

- Веб-сайты HTTPS
- Защищённая электронная почта (IMAPS, SMTPS)
- VPN (виртуальные частные сети)
- API-коммуникации

TLS 1.3, стандартизированный в 2018 году (RFC 8446), представляет собой крупную ревизию с улучшенной безопасностью и производительностью.

Рукопожатие TLS 1.3

Рукопожатие TLS 1.3 значительно упрощено:

Полное рукопожатие (1-RTT):

1. **ClientHello**: Клиент отправляет поддерживаемые версии, наборы шифров, ключевые доли (например, публичное значение X25519) и поддерживаемые алгоритмы подписей
2. **ServerHello**: Сервер выбирает набор шифров и отправляет свою ключевую долю

- С этого момента связь зашифрована
- **EncryptedExtensions**: Дополнительная конфигурация сервера
- **Certificate**: Цепочка сертификатов сервера
- **CertificateVerify**: Подпись сервера, подтверждающая владение сертификатом
- **Finished**: MAC сервера на рукопожатие

3. **Клиент**: Проверяет сертификат и подпись

- **Finished**: MAC клиента на рукопожатие
- Данные приложения теперь могут передаваться

Возобновление 0-RTT: Клиенты могут отправлять зашифрованные данные в первом сообщении, используя предварительно согласованный ключ из предыдущей сессии. Это имеет уязвимость к повторному воспроизведению, поэтому безопасно только для идемпотентных запросов.

Наборы шифров TLS 1.3

TLS 1.3 значительно сократил сложность наборов шифров. Определены только пять наборов:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256 (для устройств с ограниченными ресурсами)

Все требуют:

- Аутентифицированное шифрование AEAD (GCM, CCM или ChaCha20-Poly1305)
- Прямую секретность (эфемерный обмен ключами)
- Современный обмен ключами (только ECDHE или DHE)

Удалено из TLS 1.2:

- Обмен ключами RSA (без прямой секретности)
- Статический DH (без прямой секретности)
- RC4, DES, 3DES, экспортные шифры
- MD5, SHA-1 для подписей

Проверка сертификатов в TLS

TLS 1.3 определяет алгоритмы подписей для сертификатов:

- rsa_pkcs1_sha256, rsa_pkcs1_sha384, rsa_pkcs1_sha512
- ecdsa_secp256r1_sha256, ecdsa_secp384r1_sha384
- rsa_pss_rsae_sha256, rsa_pss_rsae_sha384, rsa_pss_rsae_sha512
- ed25519, ed448

PSS обязателен для новых подписей RSA; PKCS#1 v1.5 сохраняется для совместимости с существующими сертификатами.

Ed25519 и Ed448 полностью поддерживаются и рекомендуются для выпуска новых сертификатов при наличии совместимости.

Часть 9: Прозрачность сертификатов

Проблема

Удостоверяющие центры обладают абсолютной властью в рамках своей области доверия. Если CA скомпрометирован или действует злонамеренно, он может выпустить действительные сертификаты для любого домена. Такие случаи происходили:

- DigiNotar, нидерландский удостоверяющий центр, (2011): Скомпрометирован, выпущены сертификаты для google.com
- Symantec (2015-2017): Выпуск несанкционированных сертификатов, в итоге исключён из доверия
- Различные менее масштабные инциденты

Прозрачность сертификатов (Certificate Transparency, CT) обеспечивает публичное журналирование для обнаружения ошибочного выпуска.

Как работает CT

CT требует, чтобы CA предоставляли сертификаты в публичные журналы перед выпуском:

1. CA отправляет сертификат (или пре-сертификат) в журналы CT
2. Журнал возвращает подписанную временную метку сертификата (SCT)
3. CA включает SCT в сертификат или предоставляет через OCSP/расширение TLS
4. Браузеры требуют SCT от нескольких журналов
5. Мониторы отслеживают журналы на предмет несанкционированных сертификатов

Журналы CT

Журналы CT представляют собой деревья Меркла, допускающие только добавление:

- Сертификаты могут только добавляться, но не изменяться и не удаляться
- Структура дерева Меркла обеспечивает эффективные доказательства
- Любой может проверить целостность журнала
- Множество независимых журналов обеспечивают избыточность

Браузеры требуют SCT как минимум от двух журналов. Это означает, что компрометация одного журнала не может обеспечить незамеченный выпуск

сертификата.

Мониторинг

Владельцы доменов должны отслеживать журналы CT на предмет несанкционированных сертификатов:

- Сервисы, такие как crt.sh, Censys, Facebook CT monitor
- Автоматический мониторинг быстро обнаруживает ошибочный выпуск
- Обеспечивает быструю реакцию (отзыв, обращение к CA)

CT успешно обнаружила множество ошибочно выпущенных сертификатов, которые иначе остались бы незамеченными.

Часть 10: Протокол ACME

Автоматизированное управление сертификатами

Традиционно получение TLS-сертификатов требовало:

- Ручного создания CSR
- Проверки личности через электронную почту или веб-интерфейс
- Ручной установки сертификата
- Ручного продления до истечения срока действия

Этот процесс был подвержен ошибкам и приводил к истечению сроков действия сертификатов.

Обзор ACME

ACME (Automatic Certificate Management Environment), стандартизированный в RFC 8555, автоматизирует весь процесс:

1. **Создание учётной записи**: Клиент создаёт учётную запись на ACME-сервере
2. **Создание заказа**: Клиент запрашивает сертификат для определённых идентификаторов
3. **Авторизация**: Клиент подтверждает контроль над идентификаторами через проверки
4. **Прохождение проверки**: Клиент выполняет проверочные задания
5. **Выпуск сертификата**: Сервер выпускает сертификат
6. **Установка сертификата**: Клиент устанавливает сертификат

7. **Продление**: Клиент автоматически повторяет процесс до истечения срока действия

Проверочные задания ACME

ACME поддерживает несколько типов проверочных заданий:

HTTP-01: Разместить файл по адресу http://domain/.well-known/acme-challenge/token

- Работает для веб-серверов
- Требуется доступ к порту 80

DNS-01: Создать TXT-запись _acme-challenge.domain

- Работает для любого сервиса
- Требуется для сертификатов с подстановочными знаками (wildcard)
- Сложнее автоматизировать

TLS-ALPN-01: Ответить на TLS-соединение специальным сертификатом

- Работает, когда доступен только порт 443
- Используется реже

Let's Encrypt

Let's Encrypt, запущенный в 2015 году, произвёл революцию в развёртывании TLS:

- Бесплатные сертификаты DV
- Полная автоматизация через ACME
- Срок действия сертификата 90 дней (требуется автоматизация)
- Выпущено более 300 миллионов сертификатов

В сочетании с такими инструментами, как Certbot, это сделало HTTPS доступным для всех и способствовало росту его использования с ~40% до >90% веб-трафика.

Часть 11: Подпись кода

Зачем нужна подпись кода?

Подпись кода использует цифровые подписи для проверки подлинности программного обеспечения:

- Подтверждает, что программа поступила от идентифицированного издателя

- Обнаруживает подделку с момента подписания
- Позволяет принимать решения о доверии на основе репутации издателя
- Требуется на многих платформах (Windows SmartScreen, macOS Gatekeeper, магазины мобильных приложений)

Сертификаты подписи кода

Сертификаты подписи кода отличаются от TLS-сертификатов:

- Extended Key Usage включает OID подписи кода
- Часто требуется проверка OV или EV
- Временная метка (timestamp) обеспечивает действительность после истечения срока сертификата
- Требования к защите закрытого ключа (для EV часто требуются аппаратные модули безопасности)

Временные метки

Код, подписанный просроченным сертификатом, обычно недействителен. Временные метки решают эту проблему:

1. Хеш подписи отправляется службе штампов времени (TSA)
2. TSA возвращает подписанную временную метку
3. Временная метка доказывает, что подпись существовала до истечения срока действия сертификата
4. Код остаётся действительным, пока временная метка может быть проверена

Это критически важно для программного обеспечения, которое должно оставаться действительным в течение многих лет.

Особенности для разных платформ

Windows: Подписи Authenticode, репутация SmartScreen **macOS**: Сертификаты Developer ID, требуется нотаризация **Android**: Подпись APK, Play App Signing **iOS**: Сертификаты разработчика, обязательны для App Store **Java**: Подпись JAR

Каждая платформа имеет свои специфические требования и модели доверия.

Часть 12: Лучшие практики

Выбор алгоритма подписи (2025-2026)

Для новых приложений:

- Основной: Ed25519
- При необходимости соответствия NIST: ECDSA с P-256
- Большие сообщения/документы: RSA-PSS (3072+ бит)
- При необходимости постквантовой защиты: ML-DSA-65 (гибридный с Ed25519)
- Максимальный консерватизм: SLH-DSA (на основе хешей)

Следует избегать:

- RSA с подписями PKCS#1 v1.5 (предпочтительнее PSS)
- RSA < 2048 бит
- ECDSA с кривыми < 256 бит
- Любой алгоритм без корректной реализации

Управление сертификатами

1. **Автоматизируйте жизненный цикл сертификатов**: Используйте ACME и инструменты вроде Certbot
2. **Отслеживайте прозрачность сертификатов**: Следите за несанкционированными сертификатами
3. **Используйте краткосрочные сертификаты**: Снижает зависимость от механизма отзыва
4. **Внедрите прикрепление OCSP**: Лучше, чем CRL или проверка OCSP в реальном времени
5. **Планируйте на случай потери доверия к CA**: Имейте готовые альтернативные CA
6. **Ведите инвентарь всех сертификатов**: Знайте, какие сертификаты у вас есть и когда истекает их срок

Конфигурация TLS

1. **Используйте TLS 1.3 по возможности**
2. **Отключите TLS 1.0 и 1.1** (устарели)
3. **Отдавайте предпочтение наборам шифров TLS 1.3**: AES-GCM, ChaCha20-Poly1305

4. **Используйте надёжный обмен ключами**: X25519 или P-256
5. **Включите прикрепление OCSP**
6. **Настройте HSTS** (HTTP Strict Transport Security)

Защита закрытого ключа

1. **Генерируйте ключи на защищённых системах**
2. **Используйте аппаратные модули безопасности (HSM) для ценных ключей**: Ключи CA, подпись кода
3. **Шифруйте закрытые ключи при хранении**
4. **Внедрите контроль доступа**
5. **Никогда не передавайте закрытые ключи**: Генерируйте новые ключи, если нескольким субъектам нужны сертификаты
6. **Имейте процедуры на случай компрометации ключей**: Знайте, как отозвать и перевыпустить

Заключение

Сегодня мы рассмотрели последние элементы нашей криптографической мозаики: цифровые подписи обеспечивают аутентификацию, целостность и неотрекаемость, а инфраструктура открытых ключей предоставляет основу доверия, которая делает криптографию с открытым ключом практичной в масштабах интернета.

Ключевые выводы:

1. **Цифровые подписи подтверждают подлинность и целостность** с использованием асимметричной криптографии
2. **Ed25519 -- рекомендуемый алгоритм подписи** для новых приложений -- он быстрый, безопасный и лишён проблем ECDSA, связанных со случайными числами
3. **Постквантовые подписи (ML-DSA, SLH-DSA)** уже стандартизированы и должны развёртываться в гибридных схемах для чувствительных приложений
4. **PKI решает проблему доверия**, позволяя доверенным удостоверяющим центрам подтверждать связи открытых ключей
5. **Прозрачность сертификатов** обеспечивает публичный аудит, который выявил многочисленные ошибки CA
6. **ACME и Let's Encrypt** произвели революцию в развёртывании сертификатов благодаря автоматизации

7. **TLS 1.3** -- современный стандарт транспортной безопасности с упрощённым рукопожатием и надёжными наборами шифров
8. **Подпись кода** распространяет преимущества цифровых подписей на распространение программного обеспечения

Криптография, которую мы рассмотрели в этих четырёх лекциях -- симметричное шифрование, асимметричная криптография, хеширование и цифровые подписи -- составляет основу современной информационной безопасности. Освойте эти концепции, используйте проверенные реализации, следуйте лучшим практикам, и вы создадите системы, которые защитят данные от противников как нынешних, так и будущих.

Вопросы для обсуждения

1. Каковы были бы последствия компрометации крупного удостоверяющего центра, и как организации могут подготовиться к этому?
2. Как автоматизация управления сертификатами через ACME (Automatic Certificate Management Environment, автоматическое управление сертификатами) и Let's Encrypt изменила ландшафт безопасности?
3. Какие проблемы создаёт постквантовая криптография для существующей экосистемы PKI (Public Key Infrastructure, инфраструктура открытых ключей)?
4. Когда вы посещаете веб-сайт с иконкой замка, что именно этот замок гарантирует и чего не гарантирует?
5. Должны ли правительства иметь возможность принуждать удостоверяющие центры выпускать сертификаты, позволяющие вести слежку? Каковы последствия для безопасности?

Благодарю за внимание на протяжении всей серии лекций. Миру нужно больше инженеров, осознающих важность безопасности, и я надеюсь, что эти лекции заложили прочный фундамент для вашего дальнейшего обучения.

Контрольные вопросы

1. Объясните, как цифровые подписи обеспечивают аутентификацию, целостность и неотрекаемость.
2. Что такое цифровой сертификат и какую роль играет удостоверяющий центр в инфраструктуре открытых ключей?
3. Что такое прозрачность сертификатов и как она помогает обнаруживать ошибочно выпущенные сертификаты?

4. Как протокол ACME (Automatic Certificate Management Environment, автоматическое управление сертификатами) автоматизирует управление сертификатами, и почему это было важно?
5. Что такое подпись кода, и почему временные метки важны для долгоживущего программного обеспечения?
6. В чём разница между сертификатами DV (Domain Validation, проверка домена), OV (Organization Validation, проверка организации) и EV (Extended Validation, расширенная проверка)?

Ключевые термины

- **ACME**: Automatic Certificate Management Environment -- протокол для автоматизации выпуска сертификатов
- **CA**: Удостоверяющий центр (Certificate Authority) -- доверенная организация, выпускающая цифровые сертификаты
- **Прозрачность сертификатов (CT)**: Система публичного журналирования для обнаружения ошибочно выпущенных сертификатов
- **Подпись кода**: Процесс цифровой подписи программного обеспечения для проверки его подлинности и целостности
- **CRL**: Список отозванных сертификатов (Certificate Revocation List) -- список отозванных сертификатов, публикуемый CA
- **DANE**: DNS-based Authentication of Named Entities -- использование DNS-записей для проверки TLS-сертификатов
- **Цифровая подпись**: Криптографический механизм, обеспечивающий аутентификацию, целостность и неотрекаемость
- **DSA**: Алгоритм цифровой подписи (Digital Signature Algorithm) -- федеральный стандарт цифровых подписей
- **ECDSA**: Алгоритм цифровой подписи на эллиптических кривых (Elliptic Curve Digital Signature Algorithm)
- **Ed25519**: Алгоритм цифровой подписи на кривых Эдвардса, обеспечивающий быстрые детерминированные подписи
- **EdDSA**: Алгоритм цифровой подписи на кривых Эдвардса (Edwards-curve Digital Signature Algorithm) -- семейство детерминированных схем подписей на скрученных кривых Эдвардса
- **HSTS**: HTTP Strict Transport Security -- принуждение браузеров к использованию HTTPS
- **Let's Encrypt**: Бесплатный автоматизированный удостоверяющий центр, использующий протокол ACME

- **ML-DSA**: Алгоритм цифровой подписи на основе модулярных решёток (Module-Lattice-Based Digital Signature Algorithm, ранее Dilithium) -- постквантовый стандарт подписи NIST
- **OCSP**: Протокол онлайн-проверки статуса сертификата (Online Certificate Status Protocol) -- для проверки статуса отзыва сертификата в реальном времени
- **Прикрепление OCSP (OCSP Stapling)**: Доставка OCSP-ответа на стороне сервера, улучшающая производительность и конфиденциальность
- **PKI**: Инфраструктура открытых ключей (Public Key Infrastructure) -- фреймворк для управления цифровыми сертификатами и открытыми ключами
- **RSA-PSS**: Вероятностная схема подписи RSA (RSA Probabilistic Signature Scheme) -- рекомендуемое дополнение для подписи RSA
- **SCT**: Подписанная временная метка сертификата (Signed Certificate Timestamp) -- подтверждение отправки сертификата в журнал CT
- **SLH-DSA**: Алгоритм цифровой подписи на основе хешей без состояния (Stateless Hash-Based Digital Signature Algorithm, ранее SPHINCS+) -- консервативный постквантовый стандарт подписи
- **TLS**: Transport Layer Security -- протокол, обеспечивающий безопасность интернет-коммуникаций
- **X.509**: Стандартный формат сертификатов открытых ключей