Лекция 10: Асимметричная криптография

Тема: Криптография - симметричная и асимметричная

Технический университет Молдовы

Лектор: Максим Масютин

Введение

Добро пожаловать на лекцию 10. На прошлой неделе мы рассмотрели симметричную криптографию и её мощные алгоритмы, такие как AES и ChaCha20. Однако мы оставили без ответа один ключевой вопрос: как две стороны, которые никогда не встречались, могут безопасно обменяться секретным ключом?

Это проблема распределения ключей, и на протяжении веков она казалась неразрешимой. Если для безопасной связи необходим общий секретный ключ, но для передачи ключа нужна безопасная связь, мы получаем проблему курицы и яйца. До 1970-х годов единственными решениями были личная встреча или использование доверенных курьеров.

Сегодня мы рассмотрим, как асимметричная криптография, также называемая криптографией с открытым ключом, элегантно решает эту проблему. Мы изучим основополагающие алгоритмы, которые обеспечили безопасность интернет-эпохи, и заглянем в будущее — к постквантовой криптографии, которая обеспечит безопасность в эпоху квантовых вычислений.

Часть 1: Принципы криптографии с открытым ключом

Революционная идея

В 1976 году Уитфилд Диффи и Мартин Хеллман опубликовали статью под названием "New Directions in Cryptography", которая произвела революцию в данной области. Основная идея была обманчиво простой: что, если для шифрования и расшифрования использовались бы разные ключи?

В криптографии с открытым ключом каждый участник обладает двумя математически связанными ключами:

- **Открытый ключ**, который может свободно распространяться
- **Закрытый ключ**, который должен храниться в абсолютном секрете

Эти ключи имеют особую связь: данные, зашифрованные открытым ключом, могут быть расшифрованы только соответствующим закрытым ключом. Во многих схемах верно и обратное — данные, "зашифрованные" закрытым ключом, могут быть "расшифрованы" открытым ключом (это основа цифровых подписей, которые мы рассмотрим в лекции 12).

Теперь проблема распределения ключей решена. Алиса может опубликовать свой открытый ключ на веб-сайте, в каталоге, где угодно. Боб шифрует своё сообщение, используя открытый ключ Алисы. Только Алиса, обладая своим закрытым ключом, может его расшифровать. Им никогда не нужно было встречаться или обмениваться секретами по защищённому каналу.

Математические основы: однонаправленные функции

Криптография с открытым ключом опирается на математические задачи, которые легко вычислить в одном направлении, но вычислительно неосуществимо обратить. Они называются однонаправленными функциями или функциями с секретом.

Наиболее известные примеры:

1. **Факторизация целых чисел**: Факторизация означает разложение числа на множители, то есть нахождение двух чисел, которые нужно перемножить, чтобы получить данное число. Умножение двух больших простых чисел тривиально, но обратная операция, факторизация произведения обратно в исходные простые множители, считается вычислительно неосуществимой для достаточно больших чисел. Это основа RSA.
2. **Задача дискретного логарифмирования**: Для данного генератора g, простого числа p и y = g^x mod p нахождение x считается вычислительно неосуществимым. На этом основаны алгоритмы Диффи-Хеллман и DSA.
3. **Задача дискретного логарифмирования на эллиптических кривых**: Аналогично задаче дискретного логарифмирования, но над группами эллиптических кривых. На этом основаны ECDH и ECDSA с существенно более короткими ключами.
4. **Решёточные задачи**: Такие задачи, как обучение с ошибками (LWE) и Module-LWE, считаются трудными даже для квантовых компьютеров. Они составляют основу постквантовой криптографии.

Компромиссы

Асимметричная криптография сопряжена со значительными компромиссами по сравнению с симметричной криптографией:

Производительность: Операции с открытым ключом на порядки медленнее симметричных операций. Шифрование RSA может быть в 1000 раз медленнее шифрования AES для тех же данных.

Размер ключа: Открытые ключи значительно больше симметричных ключей. Открытый ключ RSA-2048 занимает 256 байт, по сравнению с 32 байтами для ключа AES-256.

Размер сообщения: Асимметричное шифрование имеет ограничения на максимальный размер сообщения. RSA-2048 может непосредственно зашифровать лишь около 190 байт.

По этим причинам асимметричная криптография практически никогда не используется для шифрования больших объёмов данных. Вместо этого она применяется для обмена симметричными ключами, которые затем шифруют фактические данные. Это называется гибридным шифрованием, и именно так работают TLS и практически все безопасные протоколы.

Часть 2: Алгоритм RSA

История и обзор

RSA был изобретён в 1977 году Роном Ривестом, Ади Шамиром и Леонардом Адлеманом в MIT — отсюда и название RSA. Это была первая практическая система шифрования с открытым ключом, и она остаётся наиболее широко развёрнутым асимметричным алгоритмом, хотя криптография на эллиптических кривых (ECC) постепенно догоняет.

Безопасность RSA основана на сложности факторизации больших составных чисел. Хотя мы можем легко умножить два 1024-битных простых числа и получить 2048-битное произведение, ни один известный алгоритм не может эффективно факторизовать это произведение обратно в исходные простые числа.

Генерация ключей

Генерация ключей RSA выполняется следующим образом:

1. Выбираются два больших случайных простых числа, p и q, каждое примерно вдвое короче желаемой длины ключа
2. Вычисляется n = p * q (это модуль)

3. Вычисляется функция Эйлера: phi(n) = (p-1)(q-1)
4. Выбирается открытая экспонента e такая, что 1 < e < phi(n) и НОД(e, phi(n)) = 1 (обычно e = 65537)
5. Вычисляется закрытая экспонента d такая, что d * e = 1 mod phi(n)

Открытый ключ — это (n, e). Закрытый ключ — это (n, d). Простые числа p и q должны быть надёжно удалены после генерации ключей.

Шифрование и расшифрование

Базовое шифрование RSA математически просто:

- Шифрование: c = m^e mod n
- Расшифрование: m = c^d mod n

Однако базовое RSA (так называемое "учебное RSA") никогда не используется на практике, поскольку имеет серьёзные уязвимости:

- Детерминированность: один и тот же открытый текст всегда даёт один и тот же шифротекст
- Ковкость: злоумышленники могут математически манипулировать шифротекстами
- Подверженность различным атакам при неправильном использовании

Размеры ключей RSA

Рекомендуемые в настоящее время размеры ключей RSA:

- **RSA-2048**: Минимум для новых приложений, обеспечивает примерно 112 бит безопасности
- **RSA-3072**: Обеспечивает примерно 128 бит безопасности, эквивалент AES-128
- **RSA-4096**: Обеспечивает примерно 140+ бит безопасности, рекомендуется для приложений с высокой ценностью

RSA-1024 более не считается безопасным и не должен использоваться. Факторизация 768-битного модуля RSA была продемонстрирована в 2009 году, а 1024-битный находится в пределах досягаемости государственных акторов.

OAEP — оптимальное дополнение для асимметричного шифрования

RSA-OAEP (Optimal Asymmetric Encryption Padding) — это стандартный способ использования RSA для шифрования. Стандартизированный в PKCS#1 v2 и RFC 8017, OAEP обеспечивает:

- Вероятностное шифрование (одно и то же сообщение шифруется по-разному каждый раз)
- Семантическую безопасность (шифротекст не раскрывает информацию об открытом тексте)
- Защиту от атак с подобранным шифротекстом

OAEP работает следующим образом:

1. Дополнение сообщения случайными данными
2. Применение структуры, подобной двухраундовой сети Фейстеля, с использованием хеш-функций
3. Результат затем шифруется с помощью RSA

Всегда используйте RSA-OAEP для шифрования. Никогда не используйте дополнение шифрования PKCS#1 v1.5, которое уязвимо к атаке Блейхенбахера.

PSS — вероятностная схема подписи

RSA-PSS (Probabilistic Signature Scheme) — это рекомендуемое дополнение для подписей RSA. Подобно OAEP, оно добавляет случайность для того, чтобы сделать подписи вероятностными, и обеспечивает доказательство безопасности в модели случайного оракула.

PSS использует хеш-функцию и функцию генерации маски для:

1. Хеширования сообщения
2. Добавления случайной соли
3. Кодирования результата для подписания RSA

RSA-PSS рекомендуется вместо более старого дополнения подписи PKCS#1 v1.5, хотя последнее не является взломанным и по-прежнему широко используется для обеспечения совместимости.

Криптосистема Эль-Гамаля

Криптосистема Эль-Гамаля, опубликованная в 1985 году Тахером Эль-Гамалем, египетско-американским криптографом, представляет собой схему асимметричного шифрования и цифровой подписи, основанную на обмене ключами Диффи-Хеллмана. В отличие от RSA, безопасность которого основана на сложности факторизации целых чисел, безопасность криптосистемы Эль-Гамаля основана на сложности задачи дискретного логарифмирования.

Шифрование Эль-Гамаля работает следующим образом:

1. **Генерация ключей**: Выбирается большое простое число p, генератор g мультипликативной группы по модулю p и случайный закрытый ключ x. Открытый ключ — это (p, g, y), где y = g^x mod p.

2. **Шифрование**: Для шифрования сообщения m выбирается случайное k, вычисляется c1 = g^k mod p и c2 = m * y^k mod p. Шифротекст — это (c1, c2).
3. **Расшифрование**: Вычисляется m = c2 * (c1^x)^(-1) mod p.

Отличительная особенность криптосистемы Эль-Гамаля состоит в том, что шифрование является **вероятностным**: один и тот же открытый текст, зашифрованный дважды, даёт разные шифротексты благодаря случайному значению k. Это обеспечивает семантическую безопасность, которую базовое RSA (без OAEP) не предоставляет.

Криптосистема Эль-Гамаля имеет ряд важных применений:

- **Алгоритм цифровой подписи (DSA)**, стандартизированный NIST, является вариантом схемы подписи Эль-Гамаля
- Шифрование Эль-Гамаля используется в **PGP/GPG** для шифрования электронной почты
- Схема Эль-Гамаля может быть адаптирована для работы на эллиптических кривых, что даёт ECDSA

Основной недостаток состоит в том, что шифротексты Эль-Гамаля вдвое больше открытого текста (расширение шифротекста), тогда как в RSA размер шифротекста равен размеру ключа.

Часть 3: Криптография на эллиптических кривых

Почему эллиптические кривые?

Криптография на эллиптических кривых (ECC) обеспечивает тот же уровень безопасности, что и RSA, при значительно меньших размерах ключей. Ключ ECC длиной 256 бит обеспечивает примерно такую же безопасность, как RSA-3072. Это делает ECC особенно привлекательной для:

- Мобильных и встроенных устройств с ограниченными ресурсами
- Протоколов, где важна пропускная способность
- Приложений, требующих множества операций с ключами

Основы эллиптических кривых

Эллиптическая кривая над простым полем определяется уравнением: y^2 = x^3 + ax + b (mod p)

Точки на этой кривой, вместе с "точкой на бесконечности", образуют математическую группу относительно операции, называемой сложением точек. Ключевое наблюдение состоит в том, что умножение точки на большое целое число (многократное сложение) выполняется легко, но нахождение множителя по исходной и результирующей точке (задача дискретного логарифмирования на эллиптических кривых) считается вычислительно неосуществимым.

Для криптографических целей мы работаем с кривыми над конечными полями: либо простыми полями GF(p), либо двоичными полями GF(2^m). Кривые над простыми полями более распространены на практике.

Кривые NIST

NIST (National Institute of Standards and Technology, Национальный институт стандартов и технологий) стандартизировал несколько эллиптических кривых в FIPS 186-4:

- **P-256 (secp256r1)**: 256-битное простое поле, ~128-битная безопасность
- **P-384 (secp384r1)**: 384-битное простое поле, ~192-битная безопасность
- **P-521 (secp521r1)**: 521-битное простое поле, ~256-битная безопасность

Эти кривые широко развёрнуты в TLS, подписи кода и государственных приложениях. Однако они подвергались определённой критике:

1. Константы кривых были сгенерированы в процессе, который не является полностью прозрачным
2. Их сложно реализовать с постоянным временем выполнения (устойчивость к атакам по побочным каналам)
3. Спецификация допускает некоторую гибкость реализации, что приводило к уязвимостям

Curve25519 и X25519

В 2006 году Дэниел Бернштейн опубликовал Curve25519 — тщательно спроектированную эллиптическую кривую, которая решает многие проблемы кривых NIST:

- Спроектирована для эффективной реализации с постоянным временем выполнения
- Безопасна при скручивании (ошибки реализации не приводят к утечке ключей)
- Простая спецификация без произвольных констант
- Уровень безопасности 128 бит

X25519 — это функция Диффи-Хеллмана, использующая Curve25519. Она принимает 32-байтный закрытый ключ и 32-байтный открытый ключ (или

базовую точку) и возвращает 32-байтный общий секрет.

X25519 теперь широко развёрнут в TLS 1.3, протоколе Signal, WireGuard и многих других современных протоколах. Это рекомендуемый выбор для новых приложений ECDH.

Ed25519

Ed25519 — это алгоритм цифровой подписи на основе кривых Эдвардса, использующий Curve25519 (технически, родственную кривую edwards25519). Разработанный Бернштейном и коллегами, он обеспечивает:

- Быструю генерацию и проверку подписей
- Компактные подписи (64 байта)
- Детерминированные подписи (не требуется случайное число)
- Устойчивость к атакам по побочным каналам

Ed25519 стал алгоритмом подписи по умолчанию для многих современных систем, включая SSH, Signal и многочисленные блокчейн-платформы.

Часть 4: Обмен ключами Диффи-Хеллмана

Оригинальный протокол

Прежде чем мы сможем шифровать данные, необходимо согласовать общий секретный ключ. Обмен ключами Диффи-Хеллмана, опубликованный в 1976 году, был первым практическим методом, позволяющим двум сторонам установить общий секрет по незащищённому каналу.

Протокол работает следующим образом:

1. Алиса и Боб согласовывают публичные параметры: большое простое число p и генератор g
2. Алиса выбирает случайное закрытое значение a, вычисляет A = g^a mod p и отправляет A Бобу
3. Боб выбирает случайное закрытое значение b, вычисляет B = g^b mod p и отправляет B Алисе
4. Алиса вычисляет общий секрет: s = B^a mod p = g^(ab) mod p
5. Боб вычисляет тот же общий секрет: s = A^b mod p = g^(ab) mod p

Перехватчик видит A и B, но не может вычислить g^(ab) без решения задачи дискретного логарифмирования.

ECDH — эллиптический обмен ключами Диффи-Хеллмана

ECDH применяет тот же принцип к эллиптическим кривым:

1. Алиса и Боб согласовывают кривую и базовую точку G
2. Алиса выбирает случайное a, вычисляет A = a*G, отправляет A Бобу
3. Боб выбирает случайное b, вычисляет B = b*G, отправляет B Алисе
4. Алиса вычисляет общий секрет: s = aB = ab*G
5. Боб вычисляет то же самое: s = bA = ab*G

X25519 — это функция ECDH, использующая Curve25519. Она обеспечивает 128-битную безопасность с 32-байтными ключами и значительно быстрее классического обмена Диффи-Хеллмана при эквивалентной безопасности.

Вопросы безопасности

Обмен ключами Диффи-Хеллмана уязвим к атакам типа "человек посередине", если стороны не могут аутентифицировать друг друга. Злоумышленник может выполнить отдельные обмены Диффи-Хеллмана с обеими сторонами — Алисой и Бобом, — перехватывая и перешифровывая все сообщения.

Именно поэтому обмен Диффи-Хеллмана почти всегда сочетается с аутентификацией — через предварительно распределённые ключи, цифровые подписи или сертификаты. TLS, например, использует сертификат сервера для аутентификации параметров обмена Диффи-Хеллмана.

Совершенная прямая секретность

Совершенная прямая секретность (PFS), также называемая просто прямой секретностью, — это свойство протоколов обмена ключами, которое гарантирует, что сессионные ключи не могут быть скомпрометированы, даже если долгосрочный закрытый ключ сервера впоследствии будет раскрыт.

Без прямой секретности, если злоумышленник записывает зашифрованный трафик и впоследствии получает закрытый ключ сервера (в результате взлома, судебного решения или криптоанализа), он может расшифровать все ранее записанные сессии. При наличии прямой секретности каждая сессия использует уникальную эфемерную пару ключей, которая уничтожается после использования. Даже имея долгосрочный закрытый ключ, прошлые сессии остаются защищёнными.

Прямая секретность достигается путём использования **эфемерного обмена Диффи-Хеллмана (DHE или ECDHE)** для обмена ключами:

1. Для каждой сессии обе стороны генерируют свежие временные пары ключей Диффи-Хеллмана
2. Выполняется обмен Диффи-Хеллмана для получения сессионного ключа
3. Эфемерные закрытые ключи уничтожаются после получения сессионного ключа
4. Долгосрочный ключ сервера используется только для аутентификации (подписи параметров обмена Диффи-Хеллмана), а не для обмена ключами

TLS 1.3 требует прямую секретность, предписывая эфемерный обмен ключами (ECDHE с X25519 или P-256). Более ранние версии TLS поддерживали наборы шифров без прямой секретности (например, обмен ключами RSA), при которых сессионный ключ шифровался непосредственно открытым ключом RSA сервера. В таких случаях компрометация закрытого ключа сервера позволяет расшифровать весь ранее записанный трафик.

Различие между **совершенной секретностью** (теоретико-информационная концепция Шеннона, применимая к одноразовым блокнотам) и **совершенной прямой секретностью** (свойство протокола обмена ключами) является важным: совершенная секретность означает, что никакая информация не утекает независимо от вычислительной мощности; совершенная прямая секретность означает, что прошлые сессии защищены, если долгосрочные ключи впоследствии скомпрометированы.

Часть 5: Постквантовая криптография

Квантовая угроза

Всё, что мы обсуждали до сих пор — RSA, Диффи-Хеллман и ECC — основано на математических задачах, которые квантовые компьютеры могут эффективно решить с помощью алгоритма Шора:

- Факторизация больших целых чисел (взламывает RSA)
- Дискретное логарифмирование в конечных полях (взламывает Диффи-Хеллман)
- Дискретное логарифмирование на эллиптических кривых (взламывает ECDH, ECDSA)

Достаточно мощный квантовый компьютер мог бы взломать всю ныне развёрнутую криптографию с открытым ключом. Хотя таких компьютеров сегодня не существует, они могут появиться в течение 10-20 лет. Учитывая, что некоторые зашифрованные данные должны оставаться секретными десятилетиями, а противники могут уже сейчас сохранять зашифрованный трафик для будущей

расшифровки ("перехвати сейчас, расшифруй потом"), переход к постквантовой криптографии является неотложным.

Стандартизация постквантовой криптографии NIST

В 2016 году NIST начал конкурс по стандартизации постквантовых криптографических алгоритмов. После многолетнего анализа NIST объявил первые стандарты в 2024 году:

FIPS 203 (август 2024): ML-KEM (механизм инкапсуляции ключей на основе модульных решёток) **FIPS 204 (август 2024)**: ML-DSA (алгоритм цифровой подписи на основе модульных решёток) **FIPS 205 (август 2024)**: SLH-DSA (алгоритм цифровой подписи на основе хеш-функций без сохранения состояния)

Эти стандарты представляют фундаментальный сдвиг в криптографических основах — от теоретико-числовых задач к решёточным задачам и хеш-функциям.

ML-KEM (Kyber)

ML-KEM, основанный на алгоритме CRYSTALS-Kyber, является новым стандартом для инкапсуляции ключей. Он заменяет RSA и ECDH для целей обмена ключами.

ML-KEM основан на задаче обучения с ошибками на модульных решётках (Module-LWE), которая считается трудной даже для квантовых компьютеров. Основная идея заключается в том, что решение систем линейных уравнений с малыми ошибками вычислительно сложно.

Стандартизованы три набора параметров:

- **ML-KEM-512**: 128-битная безопасность, открытый ключ 800 байт, шифротекст 768 байт
- **ML-KEM-768**: 192-битная безопасность, открытый ключ 1184 байта, шифротекст 1088 байт
- **ML-KEM-1024**: 256-битная безопасность, открытый ключ 1568 байт, шифротекст 1568 байт

Обратите внимание на значительно большие размеры ключей и шифротекстов по сравнению с X25519 (по 32 байта). Это компромисс ради квантовой устойчивости.

ML-KEM — это механизм инкапсуляции ключей (KEM), а не схема шифрования. Он работает следующим образом:

1. Генерация ключей: создаёт открытый ключ и закрытый ключ
2. Инкапсуляция: используя открытый ключ, генерирует случайный общий секрет и шифротекст

3. Декапсуляция: используя закрытый ключ и шифротекст, восстанавливает общий секрет

Затем общий секрет используется с симметричным шифром, таким как AES-256-GCM, для фактического шифрования данных.

ML-DSA (Dilithium)

ML-DSA, основанный на CRYSTALS-Dilithium, является основным постквантовым алгоритмом цифровой подписи. Он заменяет RSA-PSS и ECDSA для большинства приложений подписи.

Подобно ML-KEM, он основан на Module-LWE и связанных решёточных задачах. Стандартизованы три набора параметров:

- **ML-DSA-44**: 128-битная безопасность, открытый ключ 1312 байт, подпись 2420 байт
- **ML-DSA-65**: 192-битная безопасность, открытый ключ 1952 байта, подпись 3309 байт
- **ML-DSA-87**: 256-битная безопасность, открытый ключ 2592 байта, подпись 4627 байт

Подписи значительно больше, чем у ECDSA (64-72 байта) или EdDSA (64 байта), но ML-DSA быстр и подходит для большинства приложений.

SLH-DSA (SPHINCS+)

SLH-DSA, основанный на SPHINCS+, является альтернативным алгоритмом подписи, полностью основанным на хеш-функциях. В отличие от ML-DSA, он не опирается на решёточные задачи, а только на безопасность хеш-функций.

SLH-DSA имеет ряд преимуществ:

- Безопасность основана только на свойствах хеш-функций (хорошо изученных)
- Консервативный выбор, если криптография на основе решёток столкнётся с неожиданными атаками
- Отсутствие структурированных математических предположений

И значительные недостатки:

- Очень большие подписи (от 17 КБ до 50 КБ в зависимости от параметров)
- Медленнее, чем ML-DSA

SLH-DSA предназначен для приложений, где долгосрочная безопасность имеет первостепенное значение и большие размеры подписей приемлемы, например, подпись кода для прошивок, которые должны оставаться безопасными более 20 лет.

Хронология миграции

NIST рекомендует переход на постквантовые алгоритмы в следующие сроки:

- **Сейчас (2024-2025)**: Начало планирования и инвентаризации криптографических систем
- **2025-2027**: Развёртывание гибридных схем, сочетающих классические и постквантовые алгоритмы
- **2030-2035**: Вывод из эксплуатации классических алгоритмов для критичных приложений
- **2035+**: Полный переход на постквантовую криптографию

Гибридный подход является ключевым в период перехода. Гибридная схема может использовать X25519 + ML-KEM-768, обеспечивая безопасность, пока хотя бы один из алгоритмов остаётся надёжным.

Часть 6: Криптографическая гибкость и гибридные схемы

Что такое криптографическая гибкость?

Криптографическая гибкость — это способность системы переключать криптографические алгоритмы без серьёзной перестройки. Учитывая, что:

1. Алгоритмы периодически взламываются или ослабляются (MD5, SHA-1, RC4, DES)
2. Возникают новые требования (переход на постквантовые алгоритмы)
3. Компромиссы производительности меняются с появлением нового оборудования

Хорошо спроектированная система должна уметь менять алгоритмы через конфигурацию, а не через изменение кода. Для этого необходимы:

- Идентификаторы алгоритмов в протоколах и форматах данных
- Уровни абстракции в коде
- Управление ключами и сертификатами с поддержкой нескольких алгоритмов
- Процедуры тестирования и валидации для новых алгоритмов

Проектирование для криптографической гибкости

Практическая криптографическая гибкость включает:

Проектирование протоколов: Включите номера версий и идентификаторы алгоритмов. TLS является хорошим примером — он согласовывает наборы шифров и может внедрять новые алгоритмы через определения новых наборов.

Проектирование форматов данных: Зашифрованные данные должны включать метаданные, идентифицирующие использованный алгоритм. В противном случае вы не сможете расшифровать их при смене алгоритмов.

Архитектура кода: Используйте абстрактные интерфейсы для криптографических операций. Ваше приложение не должно напрямую вызывать "AES_encrypt", а должно вызывать "encrypt(algorithm, key, data)", где алгоритм может быть настроен.

Управление ключами: Поддержка нескольких версий ключей и алгоритмов одновременно в период перехода.

Гибридные постквантовые схемы

В период постквантового перехода гибридные схемы сочетают классические и постквантовые алгоритмы. Например:

Гибридный KEM: Объединение X25519 и ML-KEM-768

1. Выполнить обмен ключами X25519, получить общий секрет ss1
2. Выполнить инкапсуляцию ML-KEM-768, получить общий секрет ss2
3. Итоговый общий секрет = HKDF(ss1 || ss2)

Это обеспечивает безопасность, если хотя бы один из алгоритмов надёжен. Если ML-KEM имеет неожиданную слабость, X25519 по-прежнему защищает. Если квантовые компьютеры взломают X25519, ML-KEM по-прежнему защищает.

Гибридные подписи: Объединение EdDSA и ML-DSA

- Подписать сообщение обоими алгоритмами
- Проверка требует, чтобы обе подписи были корректны
- ИЛИ: Использовать составную подпись (единый блок, содержащий обе подписи)

Основные браузеры и протоколы внедряют гибридные схемы:

- Chrome и Firefox поддерживают X25519+ML-KEM-768 для TLS
- Протокол Signal внедрил X25519+ML-KEM-768 в 2023 году
- OpenSSH поддерживает постквантовый обмен ключами

Часть 7: Практические применения

Обмен ключами в TLS 1.3

TLS 1.3 использует эфемерный ECDH (обычно X25519) для обмена ключами:

1. Клиент отправляет поддерживаемые группы и ключевые доли в ClientHello
2. Сервер выбирает группу и отвечает своей ключевой долей в ServerHello
3. Обе стороны выводят общий секрет и используют его (через HKDF) для получения сессионных ключей
4. Сертификат сервера аутентифицирует обмен

Постквантовый TLS добавляет ML-KEM к этому процессу, обычно в гибридном режиме с X25519.

Безопасный обмен сообщениями

Современные системы безопасного обмена сообщениями (Signal, WhatsApp, iMessage) используют:

1. X25519 для начального обмена ключами
2. Ed25519 для ключей идентификации
3. Алгоритм двойной трещотки (Double Ratchet) для прямой секретности (непрерывная ротация Диффи-Хеллмана)
4. AES-256-GCM или ChaCha20-Poly1305 для шифрования сообщений

Signal начал добавлять ML-KEM для обеспечения постквантовой безопасности новых сообщений.

Подпись кода

Подпись кода использует асимметричную криптографию для проверки подлинности программного обеспечения:

1. Разработчик подписывает хеш кода закрытым ключом (RSA-PSS, ECDSA или EdDSA)
2. Подпись распространяется вместе с кодом
3. Пользователи проверяют подпись с помощью открытого ключа разработчика
4. Открытый ключ аутентифицируется через цепочку сертификатов

Постквантовые подписи (ML-DSA, SLH-DSA) в конечном счёте заменят текущие схемы подписи для подписи кода.

Часть 8: Лучшие практики

Выбор алгоритмов в 2025-2026 годах

Для новых систем сегодня:

Обмен ключами:

- Основной: X25519 + ML-KEM-768 (гибридный)
- Если постквантовая защита пока не требуется: только X25519
- Обратная совместимость: ECDH с P-256

Подписи:

- Основной: Ed25519 (с ML-DSA-65 в гибридном режиме для высокой безопасности)
- Если требуются кривые NIST: ECDSA с P-256
- Обратная совместимость: RSA-PSS с 2048+ битами

Избегайте:

- RSA-1024 (слишком слаб)
- Обычный обмен Диффи-Хеллмана без эллиптических кривых (если это не требуется)
- Любой алгоритм без надлежащего дополнения (учебное RSA)
- Дополнение шифрования PKCS#1 v1.5

Рекомендации по размерам ключей

Алгоритм	Минимум	Рекомендуемый	Высокая безопасность
RSA	2048	3072	4096
ECC (NIST)	P-256	P-256	P-384
X25519/Ed25519	-	256 бит	256 бит
ML-KEM	ML-KEM-512	ML-KEM-768	ML-KEM-1024
ML-DSA	ML-DSA-44	ML-DSA-65	ML-DSA-87

Безопасность реализации

1. **Используйте проверенные библиотеки**: Никогда не реализуйте RSA или ECC самостоятельно. Используйте OpenSSL, libsodium, BoringSSL или криптографические API платформы.

2. **Операции с постоянным временем выполнения**: Атаки по побочным каналам могут извлечь закрытые ключи путём измерения времени выполнения. Используйте библиотеки, реализующие операции с постоянным временем (X25519 и Ed25519 спроектированы для этого).
3. **Генерация случайных чисел**: Генерация ключей требует высококачественных случайных чисел. Используйте предоставляемый ОС криптографический генератор случайных чисел (CryptGenRandom, /dev/urandom, getrandom).
4. **Валидация входных данных**: Проверяйте, что полученные открытые ключи являются корректными точками на кривой. Атаки с некорректной кривой могут привести к утечке закрытых ключей.
5. **Защита закрытых ключей**: Храните закрытые ключи в зашифрованном виде, используйте аппаратные модули безопасности (HSM) для ключей высокой ценности, реализуйте надлежащий контроль доступа.

Постквантовый переход

Начните подготовку уже сейчас:

1. **Инвентаризация**: Определите все системы, использующие асимметричную криптографию
2. **Оценка**: Определите требования к конфиденциальности и сроку хранения данных
3. **Планирование**: Разработайте хронологию миграции
4. **Тестирование**: Оцените влияние постквантовых алгоритмов на производительность
5. **Развёртывание**: Внедрите гибридные схемы для систем с высокой степенью критичности
6. **Мониторинг**: Следите за криптографическими разработками и корректируйте планы

Заключение

Сегодня мы рассмотрели принципы и практику асимметричной криптографии — от революционной статьи Диффи и Хеллмана 1976 года до постквантовых стандартов NIST 2024 года.

Ключевые выводы:

1. **Асимметричная криптография решает проблему распределения ключей**, но слишком медленна для массового шифрования — мы используем её для обмена ключами и подписей

2. **RSA по-прежнему широко развёрнут**, но ECC (X25519, Ed25519) превосходит его для новых приложений благодаря меньшим ключам и более быстрым операциям
3. **Квантовые компьютеры взломают текущую криптографию с открытым ключом** — переход на ML-KEM и ML-DSA необходим
4. **Гибридные схемы обеспечивают безопасность в период перехода** — сочетание классических и постквантовых алгоритмов
5. **Криптографическая гибкость необходима** — проектируйте системы с возможностью смены алгоритмов без серьёзной перестройки
6. **NIST FIPS 203, 204, 205 (2024)** определяют новые постквантовые стандарты: ML-KEM для инкапсуляции ключей, ML-DSA для подписей и SLH-DSA как консервативную альтернативу

На следующей неделе мы рассмотрим хеширование и целостность данных — алгоритмы, которые гарантируют, что данные не были изменены, и которые составляют основу многих криптографических конструкций.

Вопросы для обсуждения

1. Насколько срочно организациям следует начать миграцию на постквантовые криптографические алгоритмы?
2. Каковы практические трудности развёртывания гибридных классических/постквантовых криптографических схем?
3. Как переход от RSA к криптографии на эллиптических кривых повлиял на реальную безопасность и производительность?
4. Почему асимметричное шифрование слишком медленное для шифрования больших файлов и как гибридная схема шифрования решает эту проблему?
5. Если правительство создаст квантовый компьютер, способный взломать RSA, должно ли оно раскрыть это общественности? Каковы аргументы за и против?

Благодарю за внимание. До встречи на следующей неделе.

Контрольные вопросы

1. Объясните концепцию однонаправленных функций и их роль в асимметричной криптографии.
2. Опишите процесс генерации ключей RSA и объясните, почему большие простые числа необходимы.

3. Объясните протокол обмена ключами Диффи-Хеллмана и почему он позволяет осуществлять безопасную связь по незащищённым каналам.
4. В чём состоит угроза квантовых вычислений для асимметричной криптографии и какие алгоритмы затронуты?
5. Опишите ML-KEM (Kyber) и ML-DSA (Dilithium). На каких математических задачах они основаны?
6. Что такое криптографическая гибкость и почему она необходима для постквантового перехода?
7. Что такое криптосистема Эль-Гамаля и на какой математической задаче основана её безопасность?
8. Объясните разницу между совершенной секретностью и совершенной прямой секретностью. Почему TLS 1.3 требует прямую секретность?

Ключевые термины

- **Асимметричная криптография**: Криптография, использующая математически связанные пары ключей (открытый и закрытый)
- **Криптографическая гибкость**: Способность переключать криптографические алгоритмы без серьёзной перестройки системы
- **Curve25519**: Эллиптическая кривая, спроектированная для согласования ключей, обеспечивающая скорость и безопасность
- **Диффи-Хеллман**: Протокол обмена ключами, позволяющий двум сторонам установить общий секрет по незащищённому каналу
- **ECC (криптография на эллиптических кривых)**: Криптография с открытым ключом, основанная на алгебраической структуре эллиптических кривых над конечными полями
- **ECDH**: Эллиптический обмен ключами Диффи-Хеллмана, протокол согласования ключей с использованием эллиптических кривых
- **Ed25519**: Алгоритм цифровой подписи на кривых Эдвардса, обеспечивающий быстрые и безопасные подписи
- **Эль-Гамаль**: Асимметричная криптосистема, основанная на задаче дискретного логарифмирования, обеспечивающая вероятностное шифрование и составляющая основу DSA
- **Гибридная схема**: Сочетание классических и постквантовых алгоритмов в период перехода
- **Инкапсуляция ключа**: Механизм безопасной передачи симметричного ключа с использованием асимметричной криптографии

- **ML-DSA**: Алгоритм цифровой подписи на основе модульных решёток (ранее Dilithium), постквантовый стандарт NIST для подписей
- **ML-KEM**: Механизм инкапсуляции ключей на основе модульных решёток (ранее Kyber), постквантовый стандарт NIST
- **OAEP**: Оптимальное дополнение для асимметричного шифрования, безопасное дополнение для шифрования RSA
- **Однонаправленная функция**: Функция, которую легко вычислить, но вычислительно неосуществимо обратить
- **Совершенная прямая секретность (PFS)**: Свойство, гарантирующее, что сессионные ключи не могут быть скомпрометированы, даже если долгосрочный закрытый ключ сервера впоследствии раскрыт
- **Постквантовая криптография**: Криптографические алгоритмы, устойчивые к атакам квантовых компьютеров
- **PSS**: Вероятностная схема подписи, безопасная схема дополнения для подписей RSA
- **Инфраструктура открытых ключей (PKI)**: Фреймворк для управления открытыми ключами и цифровыми сертификатами
- **RSA**: Широко используемый асимметричный алгоритм, основанный на сложности факторизации больших чисел
- **Алгоритм Шора**: Квантовый алгоритм, способный эффективно факторизовать большие целые числа, угрожающий безопасности RSA и ECC
- **SLH-DSA**: Алгоритм цифровой подписи на основе хеш-функций без сохранения состояния (ранее SPHINCS+), консервативный постквантовый стандарт