Задача: Реализация защищённого протокола обмена сообщениями с использованием ECDH, AES-256-GCM, HKDF и ECDSA

Условие:

Необходимо разработать криптографический протокол для безопасного обмена сообщениями между двумя сторонами (Михаил и Александра). Протокол должен обеспечивать конфиденциальность, целостность и аутентичность сообщений. Для этого требуется:

1. **Обмен ключами:**
    - Использовать эллиптическую криптографию с алгоритмом ECDH (Elliptic Curve Diffie-Hellman) на кривой secp256k1 для установления общего секрета между сторонами RFC 7748 - Elliptic Curves for Security.
    - Каждая сторона должна сгенерировать свою ключевую пару (приватный и публичный ключи) с использованием elliptic curve cryptography NIST SP 800-56A - Recommendation for Pair-Wise Key Establishment.
2. **Вывод ключей:**
    - Из общего секрета с помощью HKDF (HMAC-based Key Derivation Function) вывести два субключа:
        - Ключ шифрования (enc_key, 32 байта).
        - Ключ подписи (sig_key, 32 байта, опционально).
    - Использовать фиксированную соль для воспроизводимости RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF).
    - Хеш-функция: SHA-256 FIPS 180-4 - Secure Hash Standard.
3. **Шифрование:**
    - Зашифровать сообщение с использованием AES-256 в режиме GCM (Galois/Counter Mode) NIST SP 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode.
    - Генерировать случайный 12-байтовый вектор инициализации (IV) для каждого сообщения (рекомендация для GCM).
    - Поддерживать дополнительные аутентифицированные данные (AAD).
    - Генерировать 16-байтовый тег аутентификации для обеспечения целостности.
4. **Подпись:**
    - Подписать зашифрованный текст с использованием ECDSA (Elliptic Curve Digital Signature Algorithm) на кривой secp256k1 FIPS 186-4 - Digital Signature Standard.

- Обеспечить возможность проверки подписи получателем OpenSSL ECDSA Documentation.

5. **Передача:**
   - Сформировать структуру данных, содержащую:
     - Зашифрованный текст.
     - IV (12 байт).
     - Тег аутентификации (16 байт).
     - Подпись (переменной длины).
     - AAD (переменной длины).
     - Публичный ключ отправителя (в формате точки на эллиптической кривой).
   - Симулировать передачу данных между сторонами.
6. **Проверка и дешифрование:**
   - Получатель должен:
     - Проверить подпись с использованием публичного ключа отправителя (ECDSA).
     - Дешифровать сообщение с проверкой тега аутентификации (AES-GCM).
     - Вывести исходный текст при успешной верификации.

Входные данные:

- Сообщение: строка текста (например, "Привет, Александра! Это секретное сообщение.")
- AAD: строка дополнительных данных (например, "Михаил -> Александра")
- Соль для HKDF: фиксированная строка (например, "protocol-salt")

Выходные данные:

- Зашифрованное сообщение с метаданными (IV, тег, подпись).
- Результат проверки и расшифрованный текст.

Требования:

- Использовать библиотеку OpenSSL для реализации криптографических операций OpenSSL Documentation.
- Обеспечить корректное управление памятью (выделение и освобождение).
- Обработать возможные ошибки с выводом диагностики через ERR_print_errors_fp.

- Код должен быть написан на языке C.

Ограничения:

- Размер ключа AES: 256 бит.
- Размер IV: 12 байт (рекомендуемый для GCM, см. NIST SP 800-38D).
- Размер тега: 16 байт.
- Кривая для ECDH и ECDSA: secp256k1 SEC 2: Recommended Elliptic Curve Domain Parameters.
- Хеш-функция для HKDF: SHA-256.

Пример выполнения:

1. Михаил генерирует ключевую пару ECDH и отправляет свой публичный ключ Александре.
2. Александра генерирует свою ключевую пару и отправляет свой публичный ключ Михаилу.
3. Обе стороны вычисляют общий секрет через ECDH.
4. Из секрета с помощью HKDF выводятся субключи (enc_key, sig_key).
5. Михаил шифрует сообщение "Привет, Александра!" с AAD "Михаил -> Александра " с использованием AES-256-GCM, подписывает его с ECDSA и "отправляет" Александре.
6. Боб проверяет подпись с публичным ключом Михаил, дешифрует сообщение с проверкой тега и выводит: "Привет, Александра!".

Дополнительно (опционально):

- Добавить поддержку двустороннего обмена, как в TLS Handshake RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3.
- Реализовать сериализацию данных в бинарный формат для сетевой передачи.
- Интегрировать с сетевыми сокетами для реального обмена Beej's Guide to Network Programming.

Источники:

1. **ECDH:** RFC 7748 - Elliptic Curves for Security, NIST SP 800-56A.
2. **HKDF:** RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function.
3. **AES-GCM:** NIST SP 800-38D - Recommendation for Block Cipher Modes of Operation: GCM.
4. **ECDSA:** FIPS 186-4 - Digital Signature Standard, OpenSSL ECDSA.
5. **Кривая secp256k1:** SEC 2: Recommended Elliptic Curve Domain Parameters.

6. **SHA-256:** FIPS 180-4 - Secure Hash Standard.
7. **OpenSSL:** OpenSSL Documentation.