Swift Server APIs: Security stream meeting 1

Initial Agenda:

- Current discussion status roundup
- Review of BlueSSLService, a layer on top of OpenSSL and macOS Secure Transport libraries: BlueSSLService Review

If you have items you want to make sure are on the agenda, please add them below:

Attendees:

- Chris Bailey
- David Sperling - scope of the group
- Gelareh Taban
- Alfredo Delli Bovi
- Bill Abt
- Alex Blewitt

Minutes:

Scope:
- not just SSL/TLS, but also crypto, keychain/certificate management, etc which crypto depends on what's available (and whether there's requirements above/beyond what we get from the macOS and/or OpenSSL/LibreSSL libs).

BlueSSLService:
This itself covers SSL/TLS - there's no crypto services etc in there.

- Provides a service that adds onto an existing socket library (currently BlueSocket, but could be easily modified to work with other socket implementations).
- Uses OpenSSL and SecureTransport
- Initial implementation used OpenSSL everywhere, and then switched to SecureTransport on macOS as the additional work was done.
- APIs are (very) different between OpenSSL and SecureTransport. The docs on SecureTransport were a challenge.
- Configuration - OpenSSL provides much more flexibility. On macOS you're pretty much limited to pkcs#12 - so that's all it supports there. Question for Luke Hiesterman as to why that might be (or if there's more support that we're not aware of). APIs exist for dealing with other formats but are not exposed.
- Cipher Suites: SecureTransport uses the hex IANA value to select cipher suites. OpenSSL lets you use text abbreviations (corresponding to the IANA values), as well as AND/OR logic. Again, documentation for Secure Transport was a big challenge.
- OpenSSL vs LibreSSL
    - Only OpenSSL provides FIPS certification.
    - Are there any other FIPS certified forks of OpenSSL? We don't think so.

- Additionally most of the forks are subsets. This is not necessarily a bad thing as they remove a lot of insecure or unused functionality which are in OpenSSL because of legacy.
- Reusing OS libs also makes it easier to get FIPS certification as they are already certified on macOS and iOS and there is a certified version of OpenSSL available for Linux.

Usage:

- 6 protocol methods need to be implemented in the socket library.
- Added a method in the SSL service to let you override the connection verification process to allow for specialized connection verification on an application basis.

The current differences are they you actually get more capabilities on Linux from OpenSSL.

- Do we need support for PEM, etc? Probably - it's much more prevalent elsewhere.

BlueSSLService doesn't work on iOS because some of the SecureTransport APIs aren't exposed.

BlueCryptor attempts to do the same thing for Crypto - using CommonCrypto on iOS and macOS, and OpenSSL on Linux.

- The thing missing on macOS / iOS is RSA, which is there (in CommonCrypto) but not exposed.
- Similar to the issue with other file formats in BlueSSLService, RSA is there in the library, but not exposed publicly. In OpenSSL, RSA is available and the API is similar enough to the CommonCrypto API that it could be easily exposed in BlueCryptor as well.

Nothing has been tried for the keychain services…

- Keychain services is great on macOS/iOS, but no equivalent in Linux
- Linux just relies on certificates etc being present on the file system
- A lot of Linux servers, particularly Java based ones, use a CMS (Certificate Management System) to manage certificates and keys. This is similar to the services provided by the Keychain API on Apple platforms.

Should we use BlueSSLService and BlueCryptor as a prototype implementation, and use that to work on the API surface?

- This seems like a reasonable approach