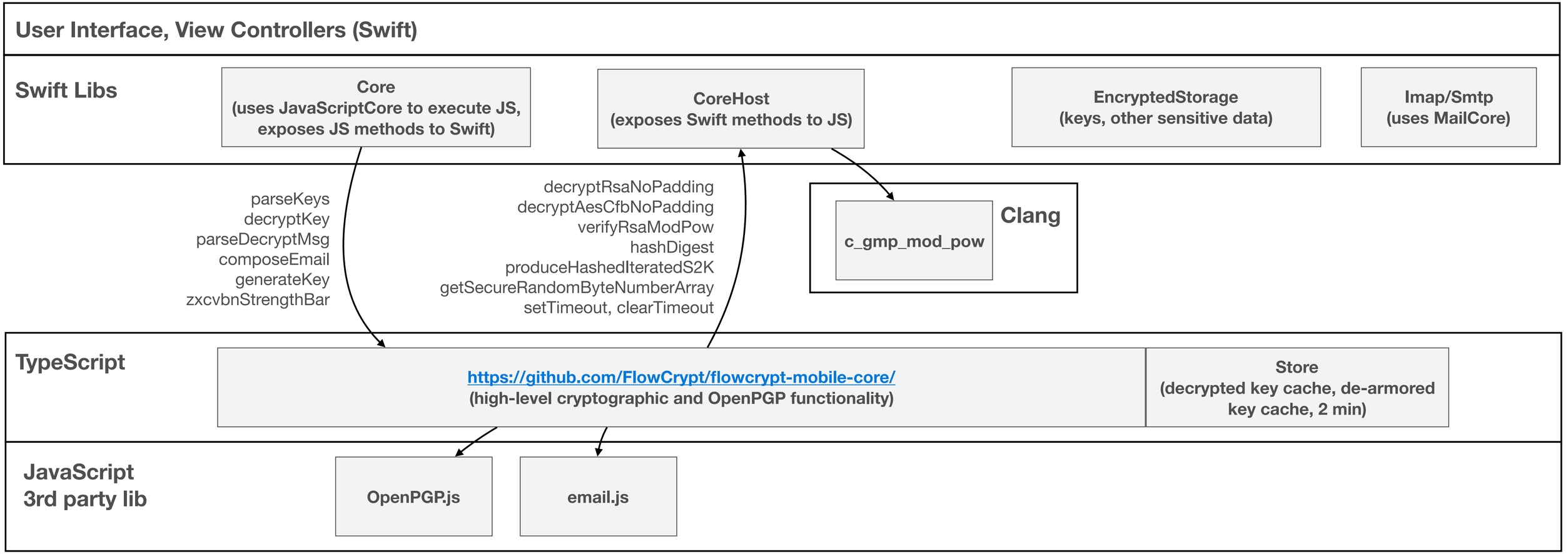
FlowCrypt iOS Architecture

Doc audience: Technical staff, security staff, penetration testers

User-facing code and business logic implemented in native Swift: https://github.com/FlowCrypt/flowcrypt-ios which will use **Core** class (with JavaScriptCore) to execute JS for OpenPGP functionality, with source at (https://github.com/FlowCrypt/flowcrypt-mobile-core) which may in turn call a Swift class **CoreHost** (exposed to JS) for low-level crypto operations, which may finally process such operation in C (in case of mod_pow) or in Swift/ObjectiveC.

Diagram below (direction of arrows indicates direction of method calls).



Decryption workflow (example, simplified):

1) user opens message
2) Message loaded using **Imap** class, keys and pass phrases loaded from **EncryptedStorage** class
3) MIME message, keys and pass phrases passed into TypeScript through **Core.parseDecryptMsg** method
4) (in TypeScript) Mime message processed using email.js lib
5) (in TypeScript) Keys compared against internal cache using "**Store**" methods. If we already have de-armored or decrypted version of the same private key in cache, the key from cache is used. Else provided private key is decrypted using provided pass phrase, and the result stored in cache.
6) (in TypeScript) Any encrypted parts are decrypted using OpenPGP.js
7) (in OpenPGP.js) to decrypt RSA, **CoreHost.decryptRsaNoPadding** is called (from JS). To decrypt AES, **CoreHost.decryptAesCfbNoPadding** is called. Also **CoreHost.getSecureRandomByteNumberArray** may be called as needed. (earlier, private key may have been decrypted using **CoreHost.produceHashedIteratedS2K**)
8) Processed/decrypted message returned back to Swift
9) Result rendered
10) After 120 seconds of inactivity, decrypted keys stored in TypeScript "Store" cache are wiped (using setTimeout and clearTimeout, which are implemented in Swift and provided to JS execution env like the rest of CoreHost methods).