Framework

Apple CryptoKit

Perform cryptographic operations securely and efficiently.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 15.0+ | macOS 10.15+ | tvOS 15.0+ | visionOS 1.0+ | watchOS 8.0+

Overview

Use Apple CryptoKit to perform common cryptographic operations:

- Compute and compare cryptographically secure digests.
- Use public-key cryptography to create and evaluate digital signatures, and to perform key exchange. In addition to working with keys stored in memory, you can also use private keys stored in and managed by the Secure Enclave.
- Generate symmetric keys, and use them in operations like message authentication and encryption.

Prefer CryptoKit over lower-level interfaces. CryptoKit frees your app from managing raw pointers, and automatically handles tasks that make your app more secure, like overwriting sensitive data during memory deallocation.

Topics

Essentials

Complying with Encryption Export Regulations

Declare the use of encryption in your app to streamline the app submission process.

{} Performing Common Cryptographic Operations

Use CryptoKit to carry out operations like hashing, key generation, and encryption.

{} Storing CryptoKit Keys in the Keychain

Convert between strongly typed cryptographic keys and native keychain types.

{} Enhancing your app's privacy and security with quantum-secure workflows

Use quantum-secure cryptography to protect your app from quantum attacks.

Cryptographically secure hashes

`protocol HashFunction`

A type that performs cryptographically secure hashing.

`struct SHA512`

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 512-bit digest.

`struct SHA384`

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 384-bit digest.

`struct SHA256`

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 256-bit digest.

Message authentication codes

`struct HMAC`

A hash-based message authentication algorithm.

`struct SymmetricKey`

A symmetric cryptographic key.

`struct SymmetricKeySize`

The sizes that a symmetric cryptographic key can take.

Ciphers

`enum AES`

A container for Advanced Encryption Standard (AES) ciphers.

`enum ChaChaPoly`

An implementation of the ChaCha20-Poly1305 cipher.

Public key cryptography

`enum` `Curve25519`

An elliptic curve that enables X25519 key agreement and Ed25519 signatures.

`enum` `P521`

An elliptic curve that enables NIST P-521 signatures and key agreement.

`enum` `P384`

An elliptic curve that enables NIST P-384 signatures and key agreement.

`enum` `P256`

An elliptic curve that enables NIST P-256 signatures and key agreement.

`struct` `SharedSecret`

A key agreement result from which you can derive a symmetric cryptographic key.

`enum` `SecureEnclave`

A representation of a device's hardware-based key manager.

`enum` `HPKE`

A container for hybrid public key encryption (HPKE) operations.

Key derivation functions

`struct` `HKDF`

A standards-based implementation of an HMAC-based Key Derivation Function (HKDF).

Key encapsulation mechanisms (KEM)

`enum` `KEM`

A key encapsulation mechanism.

`enum` `MLKEM768`

The Module-Lattice key encapsulation mechanism (KEM).

`enum` `MLKEM1024`

The Module-Lattice key encapsulation mechanism (KEM).

`enum` `XWingMLKEM768X25519`

The X-Wing (ML-KEM768 with X25519) Key Encapsulation Mechanism, defined in https://datatracker.ietf.org/doc/html/draft-connolly-cfrg-xwing-kem-06

KEM keys

`protocol KEMPrivateKey`

The private key for a key encapsulation mechanism.

`protocol KEMPublicKey`

The public key for a key encapsulation mechanism.

Errors

`enum CryptoKitError`

General cryptography errors used by CryptoKit.

`enum CryptoKitASN1Error`

Errors from decoding ASN.1 content.

Legacy algorithms

`enum Insecure`

A container for older, cryptographically insecure algorithms.

Protocols

`protocol DiffieHellmanKeyAgreement`

A Diffie-Hellman Key Agreement Key

`protocol HPKEDiffieHellmanPrivateKey`

A type that represents the private key in a Diffie-Hellman key exchange.

`protocol HPKEDiffieHellmanPrivateKeyGeneration`

A type that represents the generation of private keys in a Diffie-Hellman key exchange.

`protocol HPKEDiffieHellmanPublicKey`

A type that represents the public key in a Diffie-Hellman key exchange.

`protocol HPKEKEMPrivateKey`

A type that represents the private key in HPKE.

`protocol HPKEKEMPrivateKeyGeneration`

A type that represents the generation of private keys in HPKE

`protocol` HPKEKEMPublicKey

A type that represents the public key in HPKE

`protocol` HPKEPublicKeySerialization

A type that HPKE uses to encode the public key.

Structures

`struct` CorecryptoCurveType

`struct` SHA3_256

An implementation of Secure Hashing Algorithm 3 (SHA-3) hashing with a 256-bit digest.

`struct` SHA3_256Digest

The output of a Secure Hashing Algorithm 3 (SHA-2) hash with a 256-bit digest.

`struct` SHA3_384

An implementation of Secure Hashing Algorithm 3 (SHA-3) hashing with a 384-bit digest.

`struct` SHA3_384Digest

The output of a Secure Hashing Algorithm 3 (SHA-2) hash with a 384-bit digest.

`struct` SHA3_512

An implementation of Secure Hashing Algorithm 3 (SHA-3) hashing with a 512-bit digest.

`struct` SHA3_512Digest

The output of a Secure Hashing Algorithm 3 (SHA-2) hash with a 512-bit digest.

Type Aliases

`typealias` CryptoKitMetaError

`typealias` SHA2_256

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 256-bit digest.

`typealias` SHA2_384

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 384-bit digest.

`typealias` SHA2_512

An implementation of Secure Hashing Algorithm 2 (SHA-2) hashing with a 512-bit digest.

Enumerations

`enum` MLDSA65

The MLDSA65 Digital Signature Algorithm

`enum` MLDSA87

The MLDSA87 Digital Signature Algorithm