Apple Archive / ArchiveEncryptionContext

Class

ArchiveEncryptionContext

An object that encapsulates all parameters, keys, and data necessary to open an encrypted archive for both encryption and decryption streams.

iOS 15.0+ | iPadOS 15.0+ | Mac Catalyst | macOS 12.0+ | tvOS 15.0+ | visionOS | watchOS 8.0+

```
class ArchiveEncryptionContext
```

Topics

Creating an archive encryption context

`init?(from: ArchiveByteStream)`

Returns a new encryption context from the specified encrypted stream.

`init(profile: ArchiveEncryptionContext.Profile, compressionAlgorithm: ArchiveCompression, compressionBlockSize: Int)`

Returns a new encryption context from the specified profile, compression algorithm, and block size.

Setting and retrieving keys

`var mainKey: SymmetricKey?`

The main key used to append data to an existing archive.

`var symmetricKey: SymmetricKey?`

The symmetric encryption key used to encrypt or decrypt an archive.

`func generateSymmetricKey() throws -> SymmetricKey`

Generates a symmetric encryption key.

`func setSymmetricKey(SymmetricKey) throws`

Sets the symmetric encryption key that the context requires for symmetric encryption mode.

`func setRecipientPrivateKey(P256.KeyAgreement.PrivateKey) throws`

Sets the recipient private key that the context requires to decrypt an archive to a specific recipient in ECDHE encryption profiles.

`func setSigningPrivateKey(P256.Signing.PrivateKey) throws`

Sets the signing private key that corresponds to the signing public key that you used to create the archive.

`func setRecipientPublicKey(P256.KeyAgreement.PublicKey) throws`

Sets the recipient public key that the context requires to encrypt an archive to a specific recipient in ECDHE encryption profiles.

`func setSigningPublicKey(P256.Signing.PublicKey) throws`

Sets the signing public key that the context requires to unlock a signed archive.

Signing an encryption context

`static func sign(encryptedStream: ArchiveByteStream, encryptionContext: ArchiveEncryptionContext) throws`

Signs an encrypted archive using the credentials stored in the specified encryption context.

`var signatureMode: ArchiveEncryptionContext.SignatureMode`

The signature mode, such as an ECDSA Nist P-256 signature.

`struct SignatureMode`

Constants that describe the signature modes of an encryption context.

Getting and setting encryption context properties

`func decryptAttributes() -> Bool`

Validates decryption keys, collects archive attributes, and updates the context with decrypted archive attributes.

`var archiveIdentifier: Data?`

An optional set of data that represents the archive identifier.

`var authData: Data?`

An optional, unencrypted set of data that’s stored in the archive prologue.

`var checksumMode: ArchiveEncryptionContext.ChecksumMode`

The checksum mode, such as the 256-bit SHA-256 checksum.

`struct ChecksumMode`

Constants that describe the checksum modes of an encryption context.

`var containerSize: Int`

The size of the compressed and encrypted archive.

`var encryptionMode: ArchiveEncryptionContext.EncryptionMode`

The encryption mode, such as symmetric key encryption.

`struct EncryptionMode`

Constants that describe the checksum modes of an encryption context.

`var compressionAlgorithm: ArchiveCompression`

The compression algorithm, such as LZFSE.

`struct ArchiveCompression`

Constants that describe compression algorithms.

`var compressionBlockSize: Int`

The compression block size that defines the size of the blocks, in bytes, that the context splits data into.

`var paddingSize: Int`

An integer value that, if not zero, specifies that the size of the final archive is a multiple of the padding size.

`var profile: ArchiveEncryptionContext.Profile`

The profile of the archve.

`struct Profile`

Constants that describe the profile of an encryption context.

`var rawSize: Int`

The size of the archive raw data.

`var signatureEncryptionKey: SymmetricKey?`

The signature encryption key that the context requires to sign an encrypted archive.

Setting a password

`var password: String?`

The password used to encrypt or decrypt an archive.

`func generatePassword() throws -> String`

Generates a new password.

`func setPassword(String) throws`

Sets the password from the supplied string.

See Also

Apple Encrypted Archive essentials

{} Encrypting and Decrypting a String

Encrypt the contents of a string and save the result to the file system, then decrypt and recreate the string from the archive file using Apple Encrypted Archive.

{} Encrypting and Decrypting a Single File

Encrypt a single file and save the result to the file system, then decrypt and recreate the original file from the archive file using Apple Encrypted Archive.

{} Encrypting and Decrypting Directories

Compress and encrypt the contents of an entire directory or decompress and decrypt an archived directory using Apple Encrypted Archive.