Sample Code

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.

Download

macOS 12.0+ | Xcode 13.0+

Overview

This sample code project implements the Apple Encrypted Archive library to compress and encrypt the contents of a single file using a `SymmetricKey`. The sample saves the encrypted file to the user's temporary directory and then calls a second function that decrypts the contents of the archive and recreates the original file.

Configure the Sample Code Project

Before running the sample code project in Xcode, ensure you have a file in your temporary directory (see: `NSTemporaryDirectory()`) named `file.txt`.

Define File Paths

The sample code project defines `FilePath` structures that represent the locations of the source file, the encrypted version of the source file, and the recreated, unencrypted version of the source file.

```
// The sample defines a `FilePath` structure that represents the path of
// the source file. Before running the same, you must have a file in
// your temporary directory named `file.txt`.
```

```
let sourceFilePath = FilePath(NSTemporaryDirectory() + "file.txt")

// The sample writes the encrypted source file to the path defined by
// the `encryptedFilePath` file path structure.
let encryptedFilePath = FilePath(NSTemporaryDirectory() + "file.encrypted")

// The sample defines a `FilePath` structure that represents the path of
// the decrypted recreation of the original file.
let decryptedFilePath = FilePath(NSTemporaryDirectory() + "file.decrypted.txt")
```

Generate a Symmetric Key

The sample imports the Apple CryptoKit framework to generate the symmetric cryptographic key.

```
let key = SymmetricKey(size: SymmetricKeySize.bits256)
```

The sample uses the same key for encryption and decryption.

Create a Context for Encryption

An `ArchiveEncryptionContext` object contains the parameters, keys, and other data that the Apple Encrypted Archive library requires to open an encrypted archive for encryption and decryption streams. The sample initializes the context with a profile and compression algorithm, and its symmetric key set for encryption.

```
let context = ArchiveEncryptionContext(profile: .hkdf_sha256_aesctr_hmac__symmetric_
                                       compressionAlgorithm: .lzfse)
try context.setSymmetricKey(key)
```

Open Source and Destination File Streams

The sample creates a `readOnly` file stream to read the source file, and a `writeOnly` file stream to write the encrypted file to the file system.

```
guard let sourceFileStream = ArchiveByteStream.fileStream(
        path: sourceFilePath,
        mode: .readOnly,
        options: [ ],
        permissions: FilePermissions(rawValue: 0o644)),
      let destinationFileStream = ArchiveByteStream.fileStream(
```

```
        path: destinationFilePath,
        mode: .writeOnly,
        options: [ .create, .truncate ],
        permissions: FilePermissions(rawValue: 0o644)) else {
    throw Error.unableToCreateFileStream
}
```

Create the Encryption Stream

The encryption stream uses the encryption context and the destination file stream to write the encrypted string to the file system.

```
guard let encryptionStream = ArchiveByteStream.encryptionStream(
        writingTo: destinationFileStream,
        encryptionContext: context) else {
    throw Error.unableToCreateEncryptionStream
}
```

Encrypt the File

The `process(readingFrom:writingTo:)` function sends the output of the file-reading stream to the encryption stream. In turn, the compression stream sends its output to the file-writing stream and writes the encrypted file to the file system.

```
_ = try ArchiveByteStream.process(readingFrom: sourceFileStream,
                                  writingTo: encryptionStream)
```

Open the Source File Stream

The sample creates a source file stream to open the encrypted file.

```
guard let sourceFileStream = ArchiveByteStream.fileStream(
        path: sourceFilePath,
        mode: .readOnly,
        options: [ ],
        permissions: FilePermissions(rawValue: 0o644)) else {
    throw Error.unableToCreateFileStream
}
```

Create a Context for Decryption

The `ArchiveEncryptionContext` object for decryption derives its parameters, keys, and other data from the encrypted source file, and the sample sets the decryption context with the same symmetric key that was used for encryption.

```
guard let decryptionContext = ArchiveEncryptionContext(from: sourceFileStream) else {
    throw Error.unableToCreateDecryptionContext
}

// Set the key on the context.
try decryptionContext.setSymmetricKey(key)
```

Create the Decryption Stream

The decryption stream uses the encryption context and the source file stream to read the encrypted string from the file system.

```
guard let decryptionStream = ArchiveByteStream.decryptionStream(
    readingFrom: sourceFileStream,
    encryptionContext: decryptionContext) else {
    throw Error.unableToCreateFileStream
}
```

Open the Destination File Stream

The destination file stream writes the encrypted file to the file system. In this case, the file stream's mode is `writeOnly`. The options specify that the stream creates the file if it doesn't exist, and that if the file does exist, it should be truncated to zero bytes before the stream performs any operations.

```
guard let decryptedFileStream = ArchiveByteStream.fileStream(
    path: destinationFilePath,
    mode: .writeOnly,
    options: [ .create, .truncate ],
    permissions: FilePermissions(rawValue: 0o644)) else {
    throw Error.unableToCreateFileStream
}
```

Decrypt the Source Archive

The `process(readingFrom:writingTo:)` method writes the output of the decryption stream to the file-writing stream.

```
_ = try ArchiveByteStream.process(readingFrom: decryptionStream,
                                  writingTo: decryptedFileStream)
```

On return, `file.decrypted.txt` exists in `NSTemporaryDirectory()` and contains the decrypted contents of `file.encrypted`.

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 Directories

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

`class ArchiveEncryptionContext`

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