ВВЕДЕНИЕ В КРИПТОГРАФИЧЕСКИЕ ОПЕРАЦИИ В ОПЕРАЦИОННОЙ СИСТЕМЕ APPLE IOS

Вчерашний В.Е.

Вчерашний Владислав Евгеньевич - руководитель отдела,
отдел разработки мобильного приложения,
Компания Parler,
преподаватель,
курсы разработки приложений для Apple iOS,
г. Киев, Украина

Аннотация: *в статье анализируются различные криптографические операции на примере мобильного приложения под платформу Apple iOS.*
Ключевые слова: *Apple, криптография, безопасность, информационная безопасность, шифрование, хеширование, Swift, iOS, CryptoKit.*

В прошлом году в ходе ежегодной конференции для мобильных разработчиков **WWDC 19** (World Wide Developers Conference) компания Apple представила абсолютно новый инструмент, который значительно упрощает создание хэшей, шифрование данных и проверку / аутентификацию данных, под названием ***Apple CryptoKit***. Новый инструмент прост в использовании и не требует знания низкоуровневого API, единственного способа выполнить эту операцию до сих пор.

CryptoKit - это криптографическая библиотека для платформ Apple, написанная на Swift. Она предоставляет простые и удобные интерфейсы для безопасных и высокоуровневых операций шифрования. Больше не нужно беспокоиться об управлении указателями или другими низкоуровневыми концепциями, которых нет в Swift. Также не нужно вручную управлять памятью.

CryptoKit позволяет:

- Вычислить и сравнить хэши.
- Работать с криптографией с открытым ключом для создания и оценки цифровых подписей и обмена ключами.
- Использовать симметричную криптографию для аутентификации и шифрования сообщений.

Генерация хэшей

Хеш-функции генерируют уникальный ключ из определенных входных данных, который остается неизменным до тех пор, пока входные данные точно такие же. Например, это можно использовать для проверки того, что файл, размещенный на сервере, совпадает с файлом на вашем рабочем столе. CryptoKit упрощает создание различных типов хэшей с хорошей устойчивостью к столкновениям, что предотвращает возможность генерации одного и того же хеша двумя разными частями входных данных.

CryptoKit поддерживает алгоритмы **SHA256**, **SHA384** и **SHA512** (а также некоторые старые небезопасные алгоритмы, которые не следует использовать). По мере того, как компьютеры становились все более мощными, размеры хэшей продолжали увеличиваться, чтобы злоумышленникам было сложнее вызвать коллизии.

Следующий пример демонстрирует, как с помощью CryptoKit мы можем сгенерировать хэш з данных из строки:

```
guard let data = "Example for ScientificMagazine.ru".data(using: .utf8) else { return }
let hash = SHA256.hash(data: data)
print(hash.description)
```

Создание и проверка цифровых подписей

Цифровые подписи используются для проверки подлинности и целостности сообщения или фрагмента данных. После подписания данных закрытым ключом другие пользователи могут проверить подпись, используя ваш открытый ключ. CryptoKit имеет встроенную поддержку 4 различных типов эллиптических кривых, которые используются для создания и проверки криптографических подписей: **Curve25519**, **P521**, **P384** и **P256**. Различные типы имеют разные уровни безопасности и скорости, но Curve25519 из них лучший.

Следующий пример демонстрирует, как с помощью CryptoKit мы можем сгенерировать и проверить цифровую подпись:

```
let privateKey = Curve25519.Signing.PrivateKey()
let publicKey = privateKey.publicKey
let publicKeyData = publicKey.rawRepresentation
guard let data = "Digital signature for ScientificMagazine.ru".data(using: .utf8), let signature = try? privateKey.signature(for: data) else { return }
print(signature.rawRepresentation.)
```

Для того, чтобы проверить валидность цифровой подписи нужно использовать команду:

```
if publicKey.isValidSignature(signature, for: data) {
print("Всё хорошо")
}
```

Если кто-то попытается опубликовать фальшивую подпись, она будет отклонена, когда кто-то попытается ее проверить.

Шифрование данных

В настоящее время шифрование данных действительно очень важно. Это позволяет повысить безопасность вашего приложения и сделать его более привлекательным для конечных пользователей, которым действительно интересно, когда мы говорим о конфиденциальности. CryptoKit поддерживает алгоритмы **AES-GCM** и **ChaChaPoly**. Учтите, что в мобильной среде предпочтительнее использовать ChaChaPoly, потому что он быстрее.

Чтобы зашифровать файл, просто запечатайте его выбранным алгоритмом (к примеру, возьмём ChaChaPoly):

```
let key = SymmetricKey(size: .bits256)
guard let filePath = Bundle.main.path(forResource: "somefile", ofType: "txt"), let encryptedData = FileManager.default.contents(atPath: filePath), let sealedBox = try? ChaChaPoly.seal(encryptedData, using: key) else { return }
```

Запечатанный блок (sealedBox) содержит 3 выхода операции:

1. ***sealedBox.ciphertext*** *(зашифрованные данные, всегда того же размера, что и входные данные);*
2. ***sealedBox.tag*** *(используется для обеспечения сохранности данных)*
3. ***sealedBox.nonce*** *(который должен отличаться каждый раз при выполнении операции шифрования и генерируется случайным образом, если не предоставляется во время шифрования).*

Минусы использования CryptoKit

В настоящее время CryptoKit не поддерживает некоторые популярные алгоритмы шифрования. Хотя я и не ожидал, что Apple в ближайшее время внедрит **TwoFish** или **Serpent**, я был удивлен, когда увидел, что мы не можем получить ключи **RSA**. Хотя CryptoKit действительно хорош, поэтому я уверен, что количество поддерживаемых им наборов шифрования в будущем будет расти.

Выводы

CryptoKit - это новый и современный инструмент криптографии для платформ Apple. Он работает на очень высоком уровне, что делает его очень простым в использовании. Он поддерживает хороший набор алгоритмов криптографии, который вы ожидаете найти в любой другой библиотеке. Он поддерживает самые основные операции, такие как хеширование, шифрование и даже получение и совместное использование ключей. Это очень мощный и простой инструмент, несмотря на отсутствие других популярных алгоритмов. Я лично возлагаю большие надежды на его будущее, и это был один из моих любимых сюрпризов WWDC в 2019 году.

Список литературы

1. Документация Apple // CryptoKit. [Электронный ресурс], 2019. Режим доступа: https://developer.apple.com/documentation/cryptokit/_(дата обращения: 10.11.2020).
2. Документация Apple // Хэш-Функции [Электронный ресурс], 2019. Режим доступа: https://developer.apple.com/documentation/cryptokit/hashfunction/ (дата обращения: 10.11.2020).