Application Key Management

Workspace ONE for Android

The Workspace ONE SDK for Android provides various tools for managing an application's cryptographic keys. Tools include secure random number generators, OpenSSL-based cryptographic functions, and passcode-based encrypted storage for a master data key.

This document is a collection of content from the earlier developer guide that hasn't yet been moved to the new integration guide set.

Table of Contents

Application Key Management Guidelines

Keep to the following guidelines for key management in your application.

The Master Data Key is a key that wraps and unwraps all an application's keys. Applications must designate one key as the master data key (MDK) and use it during wrapping and unwrapping operations of other application keys. You can use the Workspace ONE SDK's secure preferences for storage and protection of MDK. Access it through SDKContext:

```
SDKContextManager.getSDKContext().appSecurePreferences
```

Usually, keys that are stored and retrieved go through several phases.

- Store keys
  1. Get key to be stored
  2. AES key wrap with masterDataKey
  3. Encode to Base64
  4. Store key in storage
- Retrieve keys
  1. Get key from storage
  2. Decode from base 64
  3. AES key unwrap with masterDataKey
  4. Return key with lifecycle tracking

Key Generation

One option to generate cryptographically strong keys is using SecureRandom, and applications can use the default SecureRandom implementation that platforms provide.

Apps can also use the implementation provided by the Workspace ONE SDK for Android.

```
SecureRandom.getInstance(AWSecurityProvider.AW_OPENSSL_SECURE_RANDOM)
```

Key Wrapping and Unwrapping

The designated MDK must wrap and unwrap all application keys before the keys can be stored on a nonvolatile medium. It is an industry standard and a common security practice to do wrapping and unwrapping operations instead of conventional encryption operations for anything that can be considered a key material. Cryptographic APIs are provided by the Workspace ONE SDK for Android to perform these operations.

```
private val cryptUtil = OpenSSLCryptUtil.getInstance()!!
val unWrappedKey = cryptUtil.aesUnwrapKey(masterDataKey, wrappedKey)
val wrappedKey = cryptUtil.aesWrapKey(masterDataKey, key!!)
```

The `appUnwrappedKey` and `appWrappedKey` refer to the application's keys in unwrapped and wrapped form.

Key Lifecycle Tracking

All keys in an application are vulnerable to memory dump attacks, which compromise the security and integrity of the application. The Workspace ONE SDK for Android provides tools to register keys to be tracked and destroyed whenever the SDK session locks out.

```
KeyGuard.secure(keyToBeProtected, KeyGuard.KeyLifespan.CONTEXT)
```

Note: Any keys that are deep copied or duplicated without a reference to the original key are not protected. It is the application's responsibility to secure the deep copied or duplicated keys in such cases.

Key Storage

Store keys only after the wrapping operation. You can store them in a non-secure space, like an SQLite database. Consider using SQLite with Room from AndroidX Jetpack for convenience and because it is standard for Android applications.

Note: It is the application's responsibility to clear all the application's keys when Workspace ONE SDK for Android performs a wipe. Failure to clear the application's keys leaves keys in an irretrievable state.

Sample Code

The Workspace ONE SDK for Android comes with a sample implementation that does all the key management operations. Find the implementation in the Workspace ONE SDK for Android release bundle, in the included SampleApp, in the package:
`com.sample.framework.dataKey`

You can copy it and use it as is, or you can use it as a reference to build a custom workflow based on your application's requirements.

Document Information

Revision History

30jul2020	First publication, for 20.7 SDK for Android.
18Feb2025	Brand Revision

Legal