**Technical Problem Share [16/05/2020]**

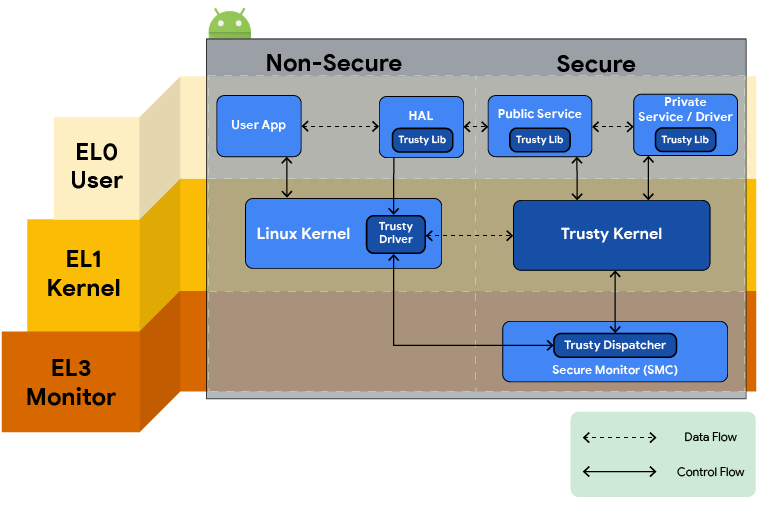
Edit by Liu Yuancheng

1. **Problem**

How smart phones/devices store the encrypted key in their local storage.

1. **Problem Back Ground**

When we are developing the Android App or IOS App and adding some key exchange feature, how do we protect the saved local key from data extraction? What kind of security data protection API should we use for different operating system?



1. **Problem Analysis**
   1. Android App key storage feature

The Android ‘Keystore’ API set is designed to generate and protect the local encryption keys, for the key storage function it stores the cryptographic keys in a container by using KeyChain API and provides different cipher algorithm API with different API level. The security Key storage contents two parts (software part and hardware part):

 Software part (Make sure ‘Key material’ never enters the application process):

1. Android Keystore mitigates unauthorized use of key material outside of the Android device by preventing extraction of the key material from application processes and from the Android device as a whole.
2. Android KeyStore mitigates unauthorized use of key material on the Android device by making apps specify authorized uses of their keys and then enforcing these restrictions outside of the apps' processes.

Hardware part (Isolate the key storage hardware from the normal system access):

1. Key material is bound to the secure hardware (e.g., Trusted Execution Environment (TEE), Secure Element (SE)) of the Android device. The module contains the following:

* Its own CPU.
* Secure storage.​
* A true random-number generator.​
* Additional mechanisms to resist package tampering and unauthorized side loading of apps.

1. TEE/SE is widely available on current Android handset.  The Trusty is compatible with ARM and Intel processors and most of current Android phone/pad has TEE/SE feature. Based on the android trusty introduction doc, if a device can run android with version 6.0 or upper, it must have TEE. If a device has fingerprint sensor, it must have TEE as the fingerprint data security need to use TEE.

When we call different cipher algorithm API function, the way they use the local key are also different:

* API Level < 18: Android Keystore not present. Request a password to the user, derive an encryption key from the password, the drawback is that you need to prompt for the password when application starts. The encryption key it is not stored in the device. It is calculated each time when the application is started using the password.
* API Level >=18 <23: Android Keystore available without AES support. Generate a random AES key using the default cryptographic provider (not using AndroidKeystore). Generate a RSA key pair into Android Keystore, and encrypt the AES key using RSA public key. Store encrypted AES key into Android SharedPreferences. When application starts, decrypt the AES key using RSA private key.
* API Level >=23: Android Keystore available with AES support. Generate a random AES key using into Android Keystore. You can use it directly.
  1. Apple IOS key storage

Apple IOS has similar feature. But the difference compared with Android is IOS don't use different API level. In iOS, each type of customer data uses its own unique key. These keys are stored in the iOS keychain and encrypted by a master key. And the feature is not named as "Trusted Execution Environment", in IOS it is called "Secure Enclave" and it is available on all devices with Apple A7 or later A-series processors.

**4. Reference**

Android development document:

<https://source.android.com/compatibility/android-cdd.pdf>

Apple IOS development document: <https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave>