Qualcomm

Qualcomm Technologies, Inc.

File based encryption

[Qualcomm® Snapdragon™ 855 Mobile Platform and beyond]

Barani Muthukumaran

June 11, 2019

Disclaimer

Qualcomm Technologies, Inc.

Qualcomm Technologies, Inc.

5775 Morehouse Drive

San Diego, CA 92121

U.S.A.

Contents

Introduction

Android 7.0 (“Nougat”) introduced File Based Encryption (FBE) with Direct boot support ensuring that a device can boot to home screen and critical services (emergency calls, alarms, etc.) are available even if the user does not authenticate. This is achieved by having two classes of storage per user where one is bound to user credentials (Credential Encrypted – CE) and the other is not (Device Encrypted – DE). With user specific keys, FBE allows Android to maintain a separate personal profile, and a work profile with its own independent lock screen.

More recently in Android 8.1 (“Oreo”), Escrow tokens were introduced, allowing device administrators to unlock a device or its work profile. These tokens can be used as backup in case a user forgets her credentials (PIN, password, pattern, etc.). It can also be used as the sole credential to unlock a device, or its work profile. As a result, user credentials cannot be directly used to protect FBE keys. To support tokens, Android creates a ‘synthetic password’ for each user and protects it using both the user credential and the token (if available).

Synthetic password protection

Android provides two methods for protecting the synthetic password. The most common method is using Gatekeeper (GK) and Keymaster (KM), and the other using Weaver. Let’s go through some key management details of the former. When the user enrolls a credential, GK uses it as context, derives a key from the device unique key, and computes a HMAC on the password handle. GK never stores the actual credential but instead derives a HMAC key to validate the credentials. In addition, GK computes an auth token with a shared secret key. This is then used as a secure signal to KM to unlock auth bound keys. A KM key bound to the user is created and used to encrypt the synthetic password.

When tokens are enrolled, the synthetic password is encrypted with the token. Because tokens have high entropy, back-off protections against brute-force attacks are not necessary.

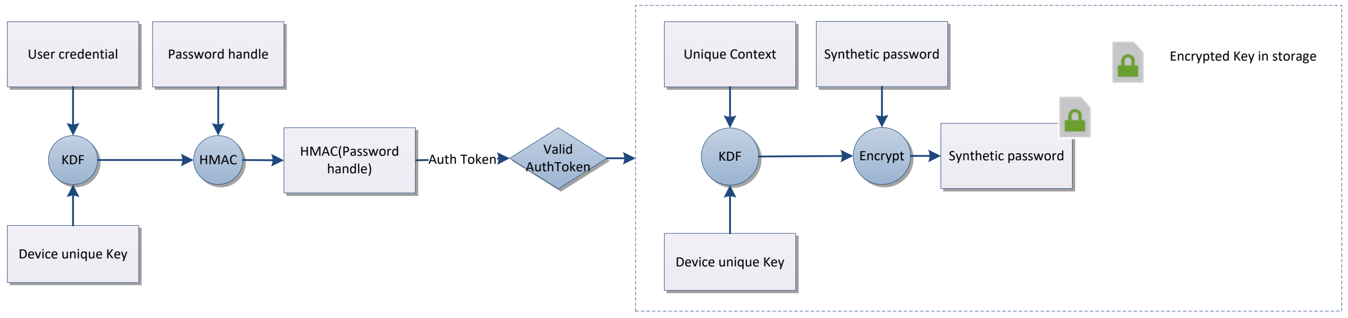

Figure 1: Synthetic Password protection - AOSP

FBE Credential Encrypted key protection

FBE Credential Encrypted (CE) class keys are generated in an Android system process called “vold.” They are protected by a secret derived from the synthetic password. When the user unlocks the device with the enrolled credential, the FBE class key is decrypted, cached in vold, and set in the Linux kernel keyring. When the File System / Storage driver reads or writes to a file in the credential encrypted class, it retrieves the appropriate key from the kernel keyring and sets the key in the Inline crypto engine to decrypt / encrypt the data.

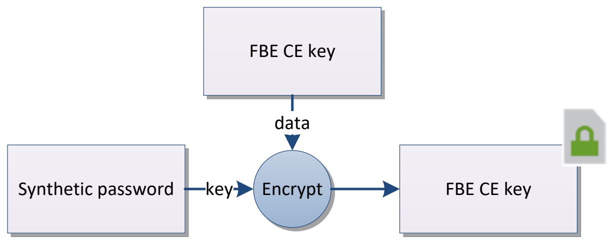

Figure 2: FBE CE key protection

Keyguard bound keys

Once the user provides the credential, CE keys are unlocked and can be used to access data protected in that class of storage. However, even after the user locks the screen this class of storage remains accessible to pictures, messages, etc. In order to protect this data, Android 9.0 (“Pie”) introduces Keyguard bound keys which are only accessible (for data decryption) when the device is unlocked. Applications that need to work in the background (such as chat apps that sync real-time messages) can use these keys to encrypt data, but they will not be able to decrypt it until the device is unlocked.

Metadata encryption

File-Based Encryption protects file names and data but does not protect metadata (such as file size or time modified). Starting with Android 9.0 (“Pie”), metadata encryption is supported by generating a Keymaster key used to encrypt everything other than file names and data.

Now that we’ve covered some of the enterprise-grade security features provided by Android, let’s examine the enhancements provided by Qualcomm Technologies, Inc. (QTI) to further bolster the security of FBE.

Cryptographic binding

In the context of data encryption, cryptographic binding is the binding of user credentials and hardware keys to keys used for protecting data.

Why is this required? Isn't this already done?

It isn't. Currently once Gatekeeper (GK) validates the user credential it provides a signal to Keymaster (KM) to unlock a key bound to the user. However, in the event of a zero-day bug that compromises the secure environment, an attacker could provide a fake signal to KM (without the user credential), thereby causing the synthetic password to be decrypted. While QTI works with OEMs to provide fixes for this issue, it could potentially be a window for attackers to compromise user data.

Cryptographic binding therefore provides a second layer of protection. This means that even on a compromised device, brute forcing of the user credential is necessary. Owing to its cryptographic computational complexity, this may consume on the order of ~100ms per attempt.

User root key

Gatekeeper generates a user specific key (the user root key). The wrapping key that's used to protect it is derived from the hardware key in combination with the user credential as context (over multiple iterations). This guarantees a minimal amount of effort needed to generate the wrapping key.

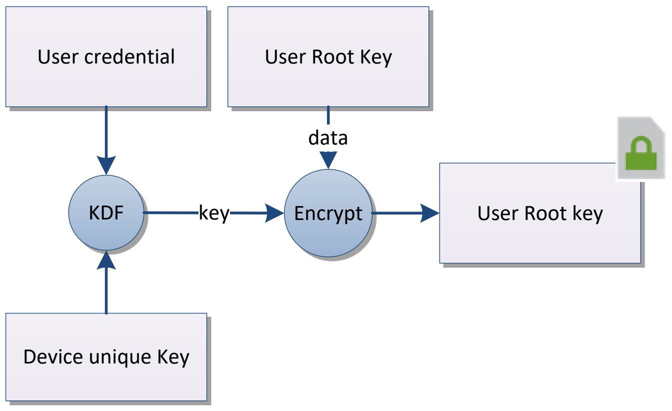

Figure 3: User Root Key Protection

Synthetic password protection

The wrapping key used to encrypt the synthetic password is derived from a Device unique key, which uses the User Root Key as its context.

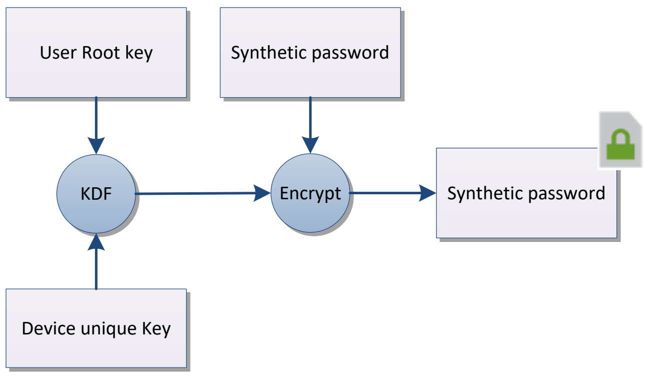

Figure 4: Synthetic Password protection

This enhancement ensures that a fake signal cannot be used to unlock the KM key used to protect the synthetic password, even in the event of a compromised secure environment. If user credentials are not present, the User Root Key cannot be retrieved, and the KM key cannot be decrypted.

Gatekeeper in Qualcomm Secure Processing Unit

While Gatekeeper is typically implemented in the Qualcomm® Trusted Execution Environment, on certain premium tier Snapdragon chipsets it is implemented in the Qualcomm Secure Processing Unit. This is engineered to provide high assurance against tampering and side-channel attacks. Qualcomm Secure Processing Unit has a dedicated CPU separate from the application processor, a true random number generator used to generate keys for Gatekeeper, a secure timer, and secure storage with replay protection to protect against brute force attacks. Since user credential validation, as well as generation and protection of the user root key, are done in the Qualcomm Secure Processing Unit, it provides an extremely high level of user data protection.

Gatekeeper throttling

The user's credential is critical to the overall security of data encryption, because it's used in conjunction with the Device Unique Key in the Qualcomm Secure Processing Unit to encrypt the User Root Key. Therefore, to help prevent an attacker from brute forcing the credential,

- Gatekeeper uses the secure timer to implement an exponential back-off mechanism, which uses secure storage (with anti-replay protection) to store the failure count. This ensures an attacker cannot reboot the device to circumvent throttling.

- Each verification consumers approximately 100ms, which is achieved by using a large iteration count utilizing the hardware key in the Qualcomm Secure Processing Unit.

Wrapped key support for FBE

These aforementioned enhancements improve security for data-at-rest, when the device is powered but the user has not yet entered her credentials. However, most often the user has already entered her credentials and the device is in use or locked. As explained, once the user provides her credential, the FBE key is cached in vold and in the Linux kernel. This exposes a sizeable attack surface which, if compromised, results in an attacker retrieving the FBE key used to encrypt sensitive user data. With these keys an attacker can decrypt sensitive information even at a future point in time, after the bug has been fixed.

The goals of wrapped keys are,

1. To ensure the FBE keys are never present in the clear in the high-level OS.
2. To have a short lifespan — until the device is rebooted or if the work profile is shutdown. This property is essential — without it, an attacker can decrypt user data without the user's credentials across boots or enterprise data even though the work profile is shutdown.

FBE keys generated in Keymaster

FBE Credential Encrypted (CE) class keys are generated in Keymaster instead of vold and the resulting keyblob is encrypted with the secret derived from the user specific synthetic password. Unless the user provides the credential, the secret used to decrypt the FBE keyblob cannot be retrieved.

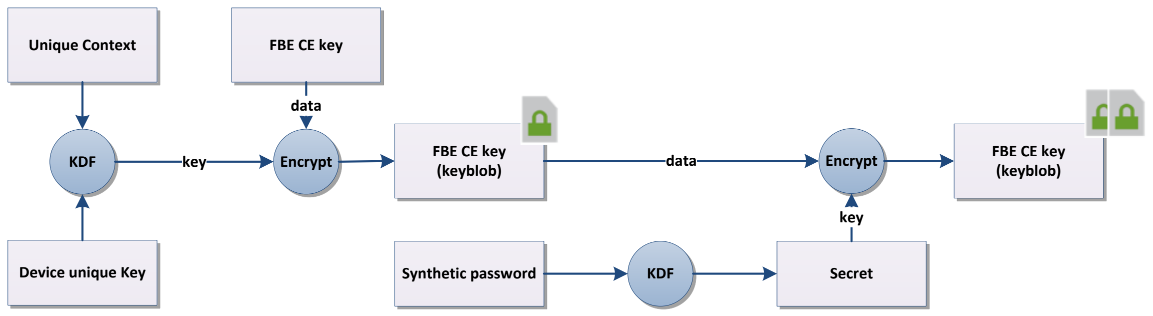

Figure 5: FBE CE key protection

Because FBE keys are generated and protected in Keymaster, plaintext keys are never present on Android. The second encryption operation, with a secret derived from synthetic password, guarantees that keys are cryptographically protected with user credentials.

Wrapping of FBE keys

Keymaster generates a per-boot / per-class / per-user ephemeral key (EK) to wrap the FBE class keys. This key is cached in Android (vold & kernel). Even if an attacker retrieves this wrapped key from the high-level OS, its validity is short-lived, and it cannot be used after a device reboot, or a work profile shutdown.

Unlocking FBE CE key

When the user unlocks the device after a reboot using the credentials, the synthetic password is retrieved. Keymaster decrypts the FBE keyblob to retrieve the FBE class key and uses the appropriate EK to wrap the FBE class key. This wrapped key is then cached in vold and the Linux kernel keyring. When the Linux kernel requires this key to read or write a file, it calls into the secure environment which unwraps the wrapped key, derives a 64-byte AES256-XTS key, and programs it in into the Inline Crypto Engine (ICE).

Secure key eviction

Android allows secondary users (Android for work profiles) to be locked, so that the user's credential is required to unlock the profile. In addition to evicting the key from the kernel keyring and vold, the EK (cached in the secure environment) is also evicted. This ensures the invalidation of any wrapped key still present in Android. If present, the actual key is also evicted from the ICE. As a result, if an attacker retrieves one such wrapped key from Android, the secure environment can no longer unwrap these keys, providing stronger protection upon locking. Finally, the next time a user unlocks the profile, a new EK will be used to wrap the CE key.