Android 密钥库系统

利用 Android 密钥库系统，您可以在容器中存储加密密钥，从而提高从设备中提取密钥的难度。在密钥进入密钥库后，可以将它们用于加密操作，而密钥材料仍不可导出。此外，它提供了密钥使用的时间和方式限制措施，例如要求进行用户身份验证才能使用密钥，或者限制为只能在某些加密模式中使用。如需了解详细信息，请参阅安全功能 (#SecurityFeatures)部分。

密钥库系统由 `KeyChain` (https://developer.android.com/reference/android/security/KeyChain.html) API 以及在 Android 4.3（API 级别 18）中引入的 Android 密钥库提供程序功能使用。本文说明了何时以及如何使用 Android 密钥库提供程序。

安全功能

Android 密钥库系统可以保护密钥材料免遭未经授权的使用。首先，Android 密钥库可以防止从应用进程和 Android 设备中整体提取密钥材料，从而避免了在 Android 设备之外以未经授权的方式使用密钥材料。其次，Android 密钥库可以让应用指定密钥的授权使用方式，并在应用进程之外强制实施这些限制，从而避免了在 Android 设备上以未经授权的方式使用密钥材料。

提取防范

Android 密钥库密钥使用两项安全措施来避免密钥材料被提取：

- 密钥材料永不进入应用进程。通过 Android 密钥库密钥执行加密操作时，应用会将待签署或验证的明文、密文和消息馈送到执行加密操作的系统进程。如果应用进程受攻击，攻击者也许能使用应用密钥，但无法提取密钥材料（例如，在 Android 设备以外使用）。
- 您可以将密钥材料绑定至 Android 设备的安全硬件，例如可信执行环境 (TEE) 和安全元素 (SE)。为密钥启用此功能时，其密钥材料永远不会暴露于安全硬件之外。如果 Android 操作系统受到攻击或者攻击者可以读取设备内部存储空间，攻击者也许能在 Android 设备上使用应用的 Android 密钥库，但无法从设备上提取这些数据。只有设备的安全硬件支持密钥算法、区块模式、填充方案和密钥有权使用的摘要的特定组合时，才可启用此功能。要检查是否为密钥启用了此功能，请获取密钥的 `KeyInfo`

(https://developer.android.com/reference/android/security/keystore/KeyInfo.html) 并检查 `KeyInfo.isInsideSecurityHardware()` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html#isInsideSecureHardware()) 的返回值。

密钥使用授权

为了避免在 Android 设备上以未经授权的方式使用密钥材料，在生成或导入密钥时 Android 密钥库会让应用指定密钥的授权使用方式。一旦生成或导入密钥，其授权将无法更改。然后，每次使用密钥时，都会由 Android 密钥库强制执行授权。这是一项高级安全功能，通常仅用于有以下要求的情形：在生成/导入密钥后（而不是之前或当中），应用进程受到攻击不会导致密钥以未经授权的方式使用。

支持的密钥使用授权可归为以下几个类别：

- *加密*：授权密钥算法、运算或目的（加密、解密、签署、验证）、填充方案、区块模式以及可与密钥搭配使用的摘要；
- *时间有效性间隔*：密钥获得使用授权的时间间隔；
- *用户身份验证*：密钥只能在用户最近进行身份验证时使用。请参阅要求进行用户身份验证才能使用密钥 (#UserAuthentication)。

作为一项额外的安全措施，对于密钥材料位于安全硬件内部的密钥（请参阅 `KeyInfo.isInsideSecurityHardware()` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html#isInsideSecureHardware()) ），某些密钥使用授权可能由安全硬件实施，具体取决于 Android 设备。加密和用户身份验证授权可能由安全硬件实施。由于安全硬件一般不具备独立的安全实时时钟，时间有效性间隔授权不可能由其实施。

您可以使用 `KeyInfo.isUserAuthenticationRequirementEnforcedBySecureHardware()` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html#isUserAuthenticationRequirementEnforcedBySecureHardware()) 查询密钥的用户身份验证授权是否由安全硬件实施。

选择密钥链或 Android 密钥库提供程序

在需要系统级凭据时请使用 `KeyChain` (https://developer.android.com/reference/android/security/KeyChain.html) API。在应用通过 `KeyChain` (https://developer.android.com/reference/android/security/KeyChain.html) API 请求使用

任何凭据时，用户需要通过系统提供的 UI 选择应用可以访问已安装的哪些凭据。因此，在用户同意的情况下多个应用可以使用同一套凭据。

使用 Android 密钥库提供程序让各个应用存储自己的凭据，并且只允许应用自身访问。这样，应用可以管理仅能由自己使用的凭据，同时又可以提供等同于 `KeyChain` (https://developer.android.com/reference/android/security/KeyChain.html) API 为系统级凭据提供的安全优势。这一方法不需要用户选择凭据。

使用 Android 密钥库提供程序

要使用此功能，请使用标准的 `KeyStore` (https://developer.android.com/reference/java/security/KeyStore.html) 和 `KeyPairGenerator` (https://developer.android.com/reference/java/security/KeyPairGenerator.html) 或 `KeyGenerator` (https://developer.android.com/reference/javax/crypto/KeyGenerator.html) 类，以及在 Android 4.3（API 级别 18）中引入的 `AndroidKeyStore` 提供程序。

`AndroidKeyStore` 注册为 `KeyStore` (https://developer.android.com/reference/java/security/KeyStore.html) 类型以用于 `KeyStore.getInstance(type)` (https://developer.android.com/reference/java/security/KeyStore.html#getInstance(java.lang.String)) 方法，而在用于 `KeyPairGenerator.getInstance(algorithm, provider)` (https://developer.android.com/reference/java/security/KeyPairGenerator.html#getInstance(java.lang.String, java.lang.String)) 和 `KeyGenerator.getInstance(algorithm, provider)` (https://developer.android.com/reference/javax/crypto/KeyGenerator.html#getInstance(java.lang.String, java.lang.String)) 方法时则注册为提供程序。

生成新私钥

生成新的 `PrivateKey` (https://developer.android.com/reference/java/security/PrivateKey.html) 要求您同时指定自签署证书具备的初始 X.509 属性。之后，您可以使用 `KeyStore.setKeyEntry` (https://developer.android.com/reference/java/security/KeyStore.html#setKeyEntry(java.lang.String, java.security.Key, char[], java.security.cert.Certificate[])) 将证书替换为由证书颁发机构 (CA) 签署的证书。

要生成密钥，请使用 `KeyPairGenerator` (https://developer.android.com/reference/java/security/KeyPairGenerator.html) 和 `KeyPairGeneratorSpec` (https://developer.android.com/reference/android/security/KeyPairGeneratorSpec.html)：

```
/*
 * Generate a new EC key pair entry in the Android Keystore by
 * using the KeyPairGenerator API. The private key can only be
 * used for signing or verification and only with SHA-256 or
 * SHA-512 as the message digest.
 */
KeyPairGenerator kpg = KeyPairGenerator.getInstance(
        KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
kpg.initialize(new KeyGenParameterSpec.Builder(
        alias,
        KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY)
        .setDigests(KeyProperties.DIGEST_SHA256,
            KeyProperties.DIGEST_SHA512)
        .build());

KeyPair kp = kpg.generateKeyPair();
```

生成新密钥

要生成密钥，请使用 **KeyGenerator** (https://developer.android.com/reference/javax/crypto/KeyGenerator.html) 和 **KeyGenParameterSpec** (https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.html)。

使用密钥库条目

AndroidKeyStore 提供程序的使用通过所有的标准 **KeyStore** (https://developer.android.com/reference/java/security/KeyStore.html) API 加以实现。

列出条目

通过调用 **aliases()** (https://developer.android.com/reference/java/security/KeyStore.html#aliases()) 方法列出密钥库中的条目：

```
/*
 * Load the Android KeyStore instance using the the
 * "AndroidKeyStore" provider to list out what entries are
 * currently stored.
 */
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
ks.load(null);
Enumeration<String> aliases = ks.aliases();
```

签署和验证数据

通过从密钥库提取 **KeyStore.Entry** (https://developer.android.com/reference/java/security/KeyStore.Entry.html) 并使用 **Signature** (https://developer.android.com/reference/java/security/Signature.html) API（例如 **sign()** (https://developer.android.com/reference/java/security/Signature.html#sign())）签署数据：

```
/*
 * Use a PrivateKey in the KeyStore to create a signature over
 * some data.
 */
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
ks.load(null);
KeyStore.Entry entry = ks.getEntry(alias, null);
if (!(entry instanceof PrivateKeyEntry)) {
    Log.w(TAG, "Not an instance of a PrivateKeyEntry");
    return null;
}
Signature s = Signature.getInstance("SHA256withECDSA");
s.initSign(((PrivateKeyEntry) entry).getPrivateKey());
s.update(data);
byte[] signature = s.sign();
```

类似地，请使用 **verify(byte[])** (https://developer.android.com/reference/java/security/Signature.html#verify(byte[])) 方法验证数据：

```
/*
 * Verify a signature previously made by a PrivateKey in our
 * KeyStore. This uses the X.509 certificate attached to our
 * private key in the KeyStore to validate a previously
 * generated signature.
 */
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
ks.load(null);
KeyStore.Entry entry = ks.getEntry(alias, null);
if (!(entry instanceof PrivateKeyEntry)) {
    Log.w(TAG, "Not an instance of a PrivateKeyEntry");
    return false;
}
Signature s = Signature.getInstance("SHA256withECDSA");
s.initVerify(((PrivateKeyEntry) entry).getCertificate());
s.update(data);
boolean valid = s.verify(signature);
```

要求进行用户身份验证才能使用密钥

生成密钥或将密钥导入到 `AndroidKeyStore` 时，您可以指定密钥仅授权给经过身份验证的用户使用。用户使用安全锁定屏幕凭据（模式/PIN/密码、指纹）的子集进行身份验证。

这是一项高级安全功能，通常仅用于有以下要求的情形：在生成/导入密钥后（而不是之前或当中），应用进程受到攻击不会导致密钥被未经身份验证的用户使用。

如果密钥仅授权给经过身份验证的用户使用，可以将其配置为以下列两种模式之一运行：

- 经过身份验证的用户可以在一段时间内使用密钥。在用户解锁安全锁定屏幕或使用 `KeyguardManager.createConfirmDeviceCredentialIntent` (https://developer.android.com/reference/android/app/KeyguardManager.html#createConfirmDeviceCredentialIntent(java.lang.CharSequence, java.lang.CharSequence)) 流程确认其安全锁定屏幕凭据后，即可使用此模式中的所有密钥。每个密钥的授权持续时间各不相同，并由 `setUserAuthenticationValidityDurationSeconds` 在密钥生成或导入时指定。此类密钥只能在启用安全锁定屏幕时生成或导入（请参阅 `KeyguardManager.isDeviceSecure()` (https://developer.android.com/reference/android/app/KeyguardManager.html#isDeviceSecure()) ）。在安全锁定屏幕停用（重新配置为“无”、“滑动”或不验证用户身份的其他模式）或被强制重置（例如由设备管理员执行）时，这些密钥将永久失效。
- 用户身份验证会授权与某一密钥关联的特定加密操作。在此模式中，涉及此类密钥的每个操作都需要用户单独授权。目前，此类授权的唯一方式是指纹身份验证：`FingerprintManager.authenticate` (https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html#authenticate(android.hardware.fingerprint.FingerprintManager.CryptoObject, android.os.CancellationSignal, int, android.hardware.fingerprint.FingerprintManager.AuthenticationCallback, android.os.Handler)) 。此类密钥只能在至少注册一个指纹时生成或导入（请参阅 `FingerprintManager.hasEnrolledFingerprints` (https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html#hasEnrolledFingerprints()) ）。一旦注册新指纹或取消注册所有指纹，这些密钥将永久失效。

支持的算法

- `Cipher` (#SupportedCiphers)
- `KeyGenerator` (#SupportedKeyGenerators)
- `KeyFactory` (#SupportedKeyFactories)
- `KeyPairGenerator` (#SupportedKeyPairGenerators)
- `Mac` (#SupportedMacs)

- `Signature` (#SupportedSignatures)
- `SecretKeyFactory` (#SupportedSecretKeyFactories)

密码

算法	提供支持的 API 级别	备注
AES/CBC/NoPadding	23+	
AES/CBC/PKCS7Padding	23+	
AES/CTR/NoPadding	23+	
AES/ECB/NoPadding	23+	
AES/ECB/PKCS7Padding	23+	
AES/GCM/NoPadding	23+	仅支持 12 字节长的 IV。
RSA/ECB/NoPadding	18+	
RSA/ECB/PKCS1Padding	18+	
RSA/ECB/OAEPWithSHA-1AndMGF1Padding	23+	
RSA/ECB/OAEPWithSHA-224AndMGF1Padding	23+	
RSA/ECB/OAEPWithSHA-256AndMGF1Padding	23+	
RSA/ECB/OAEPWithSHA-384AndMGF1Padding	23+	
RSA/ECB/OAEPWithSHA-512AndMGF1Padding	23+	
RSA/ECB/OAEPPadding	23+	

KeyGenerator

算法	提供支持的 API 级别	备注
AES	23+	支持的大小：128、192、256
HmacSHA1	23+	• 支持的大小：8 - 1024（含），必须是 8 的倍数 • 默认大小：160

算法	提供支持的 API 级别	备注
HmacSHA224	23+	• 支持的大小：8 - 1024（含），必须是 8 的倍数 • 默认大小：224
HmacSHA256	23+	• 支持的大小：8 - 1024（含），必须是 8 的倍数 • 默认大小：256
HmacSHA384	23+	• 支持的大小：8 - 1024（含），必须是 8 的倍数 • 默认大小：384
HmacSHA512	23+	• 支持的大小：8 - 1024（含），必须是 8 的倍数 • 默认大小：512

KeyFactory

算法	提供支持的 API 级别	备注
EC	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)（仅私钥）、`ECPublicKeySpec` (https://developer.android.com/reference/java/security/spec/ECPublicKeySpec.html)（仅公钥）、`X509EncodedKeySpec` (https://developer.android.com/reference/java/security/spec/X509EncodedKeySpec.html)（仅公钥）
RSA	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)（仅私钥）、`RSAPublicKeySpec` (https://developer.android.com/reference/java/security/spec/RSAPublicKeySpec.html)（仅公钥）、`X509EncodedKeySpec` (https://developer.android.com/reference/java/security/spec/X509EncodedKeySpec.html)（仅公钥）

密钥库

密钥库支持的密钥类型与 `KeyPairGenerator` (#SupportedKeyPairGenerators) 和 `KeyGenerator` (#SupportedKeyGenerators) 支持的相同。

KeyPairGenerator

算法	提供支持的 API 级别	备注
DSA	*19–22*	
EC	23+	• 支持的大小：224、256、384、521 • 支持的命名曲线：P-224 (secp224r1)、P-256（又称为 secp256r1 和 prime256v1）、P-384（ 521（又称为 secp521r1） 在 API 级别 23 前，EC 密钥可使用经 "RSA" 算法初始化的 `KeyPairGeneratorSpec` (https://developer.android.com/reference/android/security/KeyPairGeneratorSpec.html) 的 KeyP 型需使用 `setKeyType(String)` (https://developer.android.com/reference/android/security/KeyPairGeneratorSpec.Builder.html# 设为“EC”。EC 曲线名称无法使用此方法指定 - NIST P 曲线将根据请求的密钥大小自动选择。
RSA	18+	• 支持的大小：512、768、1024、2048、3072、4096 • 支持的公共指数：3、65537 • 默认公共指数：65537

Mac

算法	提供支持的 API 级别	备注
HmacSHA1	23+	
HmacSHA224	23+	
HmacSHA256	23+	
HmacSHA384	23+	

算法	提供支持的 API 级别	备注
HmacSHA512	23+	

Signature

算法	提供支持的 API 级别	备注
MD5withRSA	18+	
NONEwithECDSA	23+	
NONEwithRSA	18+	
SHA1withDSA	*19–22*	
SHA1withECDSA	19+	
SHA1withRSA	18+	
SHA1withRSA/PSS	23+	
SHA224withDSA	*20–22*	
SHA224withECDSA	20+	
SHA224withRSA	20+	
SHA224withRSA/PSS	23+	
SHA256withDSA	*19–22*	
SHA256withECDSA	19+	
SHA256withRSA	18+	
SHA256withRSA/PSS	23+	
SHA384withDSA	*19–22*	
SHA384withECDSA	19+	
SHA384withRSA	18+	
SHA384withRSA/PSS	23+	
SHA512withDSA	*19–22*	
SHA512withECDSA	19+	

算法	提供支持的 API 级别	备注
SHA512withRSA	18+	
SHA512withRSA/PSS	23+	

SecretKeyFactory

算法	提供支持的 API 级别	备注
AES	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)
HmacSHA1	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)
HmacSHA224	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)
HmacSHA256	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)
HmacSHA384	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)
HmacSHA512	23+	支持的密钥规范：`KeyInfo` (https://developer.android.com/reference/android/security/keystore/KeyInfo.html)

上次更新日期：四月 25, 2018

Twitter
在 Twitter 上关注 @AndroidDev

Google+
在 Google+ 上关注 Android Developers

YouTube
在 YouTube 上访问“Android Developers”频道