Sistema de armazenamento de chaves do Android

O sistema de armazenamento de chaves do Android permite que você armazene chaves criptográficas em um contêiner para dificultar sua extração do dispositivo. Quando as chaves estão no armazenamento de chaves, elas podem ser usadas para operações criptográficas e o material das chaves permanece não exportável. Além disso, esse recurso oferece instalações para restringir como e quando as chaves podem ser usadas, por exemplo, solicitando a autenticação do usuário para usar as chaves ou restringindo o uso das chaves apenas em certos modos criptográficos. Consulte a seção [Recursos de segurança](https://developer.android.com/training/articles/keystore#SecurityFeatures) para saber mais.

O sistema de armazenamento de chaves é usado pela [KeyChain](https://developer.android.com/reference/android/security/KeyChain.html) API e pelo recurso de provedor de armazenamento de chaves do Android introduzido no Android 4.3 (nível de API 18). Este documento explora quando e como usar o provedor de armazenamento de chaves do Android.

Recursos de segurança

O sistema de armazenamento de chaves do Android protege o material de chaves contra o uso não autorizado. Em primeiro lugar, o armazenamento de chaves do Android mitiga o uso não autorizado do material de chaves fora do dispositivo Android ao evitar sua extração dos processos do aplicativo e do dispositivo Android como um todo. Em segundo lugar, o armazenamento de chaves do Android mitiga o uso não autorizado do material de chaves no dispositivo Android fazendo com que os aplicativos especifiquem usos autorizados de suas chaves e aplicando essas restrições fora dos processos dos aplicativos.

Prevenção de extração

O material de chaves do armazenamento do Android é protegido contra a extração por duas medidas de segurança:

* O material de chaves nunca entra no processo de aplicativos. Quando um aplicativo realiza operações criptográficas usando uma chave do armazenamento de chaves do Android, o texto simples, de cifra e as mensagens a serem assinadas ou verificadas são alimentadas, em segundo plano, a um processo de sistema que realiza as operações criptográficas. Se o processo do aplicativo for comprometido, o invasor poderá usar as chaves do aplicativo, mas não conseguirá extrair o material das chaves (por exemplo, para usá-lo fora do dispositivo Android).
* O material de chaves pode ser vinculado ao hardware protegido (por exemplo, o Trusted Execution Environment (TEE) e o Secure Element (SE)) do dispositivo Android. Quando esse recurso é ativado para uma chave, seu material nunca é exposto fora do hardware protegido. Se o sistema operacional Android for comprometido ou se um invasor conseguir ler o armazenamento interno do dispositivo, ele poderá usar as chaves do armazenamento de chaves do Android de qualquer aplicativo no dispositivo, mas não extraí-las do dispositivo. Esse recurso é ativado apenas se o hardware protegido do dispositivo oferecer suporte à combinação específica de algoritmo de chave, modos de bloqueio, esquemas de preenchimento e resumos com a qual a chave tem autorização para ser usada. Para verificar se o recurso está ativado para uma chave, obtenha um [KeyInfo](https://developer.android.com/reference/android/security/keystore/KeyInfo.html) para a chave e verifique o valor de retorno de [KeyInfo.isInsideSecurityHardware()](https://developer.android.com/reference/android/security/keystore/KeyInfo.html" \l "isInsideSecureHardware()).

Autorizações de uso de chaves

Para mitigar o uso não autorizado de chaves em um dispositivo Android, o armazenamento de chaves do Android permite que os aplicativos especifiquem usos autorizados de suas chaves ao gerá-las ou importá-las. Quando uma chave é gerada ou importada, suas autorizações não podem ser alteradas. Dessa forma, as autorizações são aplicadas pelo armazenamento de chaves do Android sempre que a chave for usada. Esse é um recurso de segurança avançado que geralmente é útil apenas se você precisar que um comprometimento do processo do seu aplicativo após uma geração/importação de chave (mas não antes ou durante) não cause usos não autorizados da chave.

As autorizações de uso de chaves com suporte se encaixam nas seguintes categorias:

* *criptografia*: algoritmos, operações ou propósitos autorizados de chaves (criptografar, descriptografar, verificar), esquemas de preenchimento, modos de bloqueio e resumos com os quais a chave pode ser usada;
* *intervalo de validade temporal*: intervalo de tempo durante o qual a chave tem o uso autorizado;
* *autenticação do usuário*: a chave só pode ser usada se o usuário tiver sido autenticado recentemente. Consulte [Exigir a autenticação do usuário para o uso de chaves](https://developer.android.com/training/articles/keystore#UserAuthentication).

Como medida de segurança adicional, para chaves cujo material se encontra em um hardware protegido (consulte [KeyInfo.isInsideSecurityHardware()](https://developer.android.com/reference/android/security/keystore/KeyInfo.html" \l "isInsideSecureHardware())), algumas autorizações de uso de chaves podem ser aplicadas pelo hardware protegido, dependendo do dispositivo Android. As autorizações criptográfica e de autenticação do usuário provavelmente serão aplicadas pelo hardware protegido. É improvável que o hardware protegido aplique autorizações de intervalo de validade temporal, pois ele normalmente não tem um relógio de tempo real protegido e independente.

É possível consultar se a autorização de autenticação do usuário de uma chave será aplicada pelo hardware protegido usando [KeyInfo.isUserAuthenticationRequirementEnforcedBySecureHardware()](https://developer.android.com/reference/android/security/keystore/KeyInfo.html#isUserAuthenticationRequirementEnforcedBySecureHardware()).

Escolher entre uma cadeia de chaves ou o provedor do armazenamento de chaves do Android

Use a [KeyChain](https://developer.android.com/reference/android/security/KeyChain.html) API quando quiser credenciais de sistema. Quando um aplicativo solicita o uso de qualquer credencial com a [KeyChain](https://developer.android.com/reference/android/security/KeyChain.html) API, os usuários podem escolher, por meio de uma IU fornecida pelo sistema, quais das credenciais instaladas podem ser acessadas por um aplicativo. Isso permite que vários aplicativos usem o mesmo conjunto de credenciais com o consentimento do usuário.

Use o provedor de armazenamento de chaves do Android para permitir que um aplicativo individual armazene suas próprias credenciais que apenas o próprio aplicativo pode acessar. Isso oferece uma maneira para que os aplicativos gerenciem credenciais que apenas eles podem usar, com os mesmos benefícios de segurança que a [KeyChain](https://developer.android.com/reference/android/security/KeyChain.html) API fornece para credenciais de sistema. Esse método não exige interação do usuário para selecionar as credenciais.

Usar o provedor do armazenamento de chaves do Android

Para usar esse recurso, use as classes padrão [KeyStore](https://developer.android.com/reference/java/security/KeyStore.html) e [KeyPairGenerator](https://developer.android.com/reference/java/security/KeyPairGenerator.html) ou [KeyGenerator](https://developer.android.com/reference/javax/crypto/KeyGenerator.html) juntamente com o provedor de AndroidKeyStore introduzido no Android 4.3 (nível de API 18).

O AndroidKeyStore é registrado como um tipo [KeyStore](https://developer.android.com/reference/java/security/KeyStore.html) para uso com o método [KeyStore.getInstance(type)](https://developer.android.com/reference/java/security/KeyStore.html" \l "getInstance(java.lang.String)) e como um provedor para uso com os métodos [KeyPairGenerator.getInstance(algorithm, provider)](https://developer.android.com/reference/java/security/KeyPairGenerator.html" \l "getInstance(java.lang.String,%20java.lang.String)) e [KeyGenerator.getInstance(algorithm, provider)](https://developer.android.com/reference/javax/crypto/KeyGenerator.html" \l "getInstance(java.lang.String,%20java.lang.String)).

Gerar uma nova chave privada

Para gerar uma nova [PrivateKey](https://developer.android.com/reference/java/security/PrivateKey.html), você também deve especificar os atributos X.509 iniciais que o certificado autoassinado terá. Você pode usar [KeyStore.setKeyEntry](https://developer.android.com/reference/java/security/KeyStore.html" \l "setKeyEntry(java.lang.String,%20java.security.Key,%20char[],%20java.security.cert.Certificate[])) para substituir o certificado posteriormente por um certificado assinado por uma Autoridade de Certificação (CA).

Para gerar a chave, use um [KeyPairGenerator](https://developer.android.com/reference/java/security/KeyPairGenerator.html) com [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();

Gerar uma nova chave secreta

Para gerar a chave, use um [KeyGenerator](https://developer.android.com/reference/javax/crypto/KeyGenerator.html) com [KeyGenParameterSpec](https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.html).

Trabalhar com entradas do armazenamento de chaves

O uso do provedor de AndroidKeyStore é realizado por pelo de todas as APIs padrão do [KeyStore](https://developer.android.com/reference/java/security/KeyStore.html).

Listar entradas

Liste entradas no armazenamento de chaves chamando o método [aliases()](https://developer.android.com/reference/java/security/KeyStore.html" \l "aliases()):

/\*  
 \* 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();

Assinar e verificar dados

Assine dados recuperando [KeyStore.Entry](https://developer.android.com/reference/java/security/KeyStore.Entry.html) do armazenamento de chaves e usando as APIs de [Signature](https://developer.android.com/reference/java/security/Signature.html), como [sign()](https://developer.android.com/reference/java/security/Signature.html" \l "sign()):

/\*  
 \* 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();

Da mesma forma, verifique dados com o método [verify(byte[])](https://developer.android.com/reference/java/security/Signature.html#verify(byte[])):

/\*  
 \* 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);

Exigir a autenticação do usuário para o uso de chaves

Ao gerar ou importar uma chave para o AndroidKeyStore, você pode especificar que a chave só pode ser usada se o usuário tiver sido autenticado. O usuário é autenticado usando um subconjunto de suas credenciais de bloqueio de tela (padrão/PIN/senha, impressão digital).

Esse é um recurso de segurança avançado que geralmente é útil apenas se você precisar que um comprometimento do processo do seu aplicativo após uma geração/importação de chave (mas não antes ou durante) não possa ignorar a exigência de autenticação do usuário para usar a chave.

Quando uma chave está autorizada a ser usada apenas se o usuário tiver sido autenticado, ela é configurada para operar em um de dois modos:

* A autenticação do usuário autoriza o uso das chaves por um período de tempo. Todas as chaves nesse modo podem ser usadas assim que o usuário desbloqueia a tela ou confirma suas credenciais de bloqueio de tela usando o fluxo [KeyguardManager.createConfirmDeviceCredentialIntent](https://developer.android.com/reference/android/app/KeyguardManager.html" \l "createConfirmDeviceCredentialIntent(java.lang.CharSequence,%20java.lang.CharSequence)). A duração da autorização é especificada para cada chave usando setUserAuthenticationValidityDurationSecondsdurante a geração ou importação da chave. Essas chaves só podem ser geradas ou importadas se a tela de bloqueio estiver ativada (consulte [KeyguardManager.isDeviceSecure()](https://developer.android.com/reference/android/app/KeyguardManager.html" \l "isDeviceSecure())). Essas chaves são permanentemente invalidadas se a tela de bloqueio for desativada (configurada como None, Swipe ou outro modo que não autentique o usuário) ou redefinida à força (por exemplo, por um administrador de dispositivo).
* A autenticação do usuário autoriza uma operação criptográfica específica associada a uma chave. Nesse modo, cada operação que envolve a chave em questão deve ser autorizada individualmente pelo usuário. No momento, a única forma de autorização desse tipo é a autenticação por impressão digital: [FingerprintManager.authenticate](https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html" \l "authenticate(android.hardware.fingerprint.FingerprintManager.CryptoObject,%20android.os.CancellationSignal,%20int,%20android.hardware.fingerprint.FingerprintManager.AuthenticationCallback,%20android.os.Handler)). Essas chaves só podem ser geradas ou importadas se pelo menos uma impressão digital tiver sido cadastrada (consulte [FingerprintManager.hasEnrolledFingerprints](https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html" \l "hasEnrolledFingerprints())). Essas chaves são invalidadas permanentemente se uma nova impressão for cadastrada ou se todas as impressões digitais forem descadastradas.