身份验证

Android 采用通过用户身份验证把关的加密密钥机制，该机制需要以下组件：

- **加密密钥存储和服务提供程序。** 存储加密密钥并基于这些密钥提供标准加密例程。Android 支持由硬件支持的 Keystore (/security/keystore/index.html) 和 Keymaster 这两种加密服务，其中包括由硬件支持的密钥存储加密服务，该服务可能包括可信执行环境 (TEE) 或安全元件 (SE)，例如 Strongbox。
- **用户身份验证程序。** 证明相应用户存在并/或已成功通过身份验证。Android 支持 Gatekeeper (/security/authentication/gatekeeper.html)（用于 PIN 码/解锁图案/密码身份验证）和 Fingerprint (/security/authentication/fingerprint-hal.html)（用于指纹识别身份验证）。搭载 Android 9 及更高版本的设备可以使用 `BiometricPrompt` (https://developer.android.com/reference/android/hardware/biometrics/BiometricPrompt) 作为指纹和其他生物识别技术的单一集成点。这些组件通过已经过身份验证的渠道与 Keystore 服务沟通身份验证状态。（Keystore 服务还支持框架级 Android Keystore 系统 (https://developer.android.com/training/articles/keystore.html)。）

Gatekeeper、Fingerprint 和 Biometric 组件能够与 Keystore 及其他组件协同运作，以支持使用由硬件支持的身份验证令牌 (#authtoken_format) (AuthToken)。

注册

在设备恢复出厂设置后首次启动时，所有身份验证程序均会做好接受用户通过凭据注册的准备。用户必须先通过 Gatekeeper 注册一个 PIN 码/解锁图案/密码。该首次注册会随机生成一个 64 位的用户安全标识符 (SID)，该用户 SID 将用作用户的标识符以及用户加密材料的绑定令牌。该用户 SID 会以加密形式绑定到用户的密码，成功通过 Gatekeeper 的身份验证后，会相应生成 AuthToken，其中包含用于该密码的用户 SID。

用户如果想要更改凭据，则必须提供现有凭据。如果现有凭据成功通过验证，则与现有凭据关联的用户 SID 将转移到新凭据，这样用户便可以在更改凭据后保留访问密钥。如果用户未提供现有凭据，系统会使用一个完全随机的用户 SID 为其注册一个新凭据。用户可以访问设备，但基于旧用户 SID 创建的密钥将永久丢失。这种情况称为“不可信注册”。

一般情况下，Android 框架不允许进行不可信注册，因此大多数用户根本看不到该功能。不过，如果设备管理员或攻击者强制重置密码，则可能会发生这种情况。

身份验证

用户设置凭据并收到用户 SID 后，便可以开始进行身份验证，身份验证从用户提供 PIN 码、解锁图案、密码或指纹开始。所有 TEE 组件都共用一个密钥来验证对方的消息。

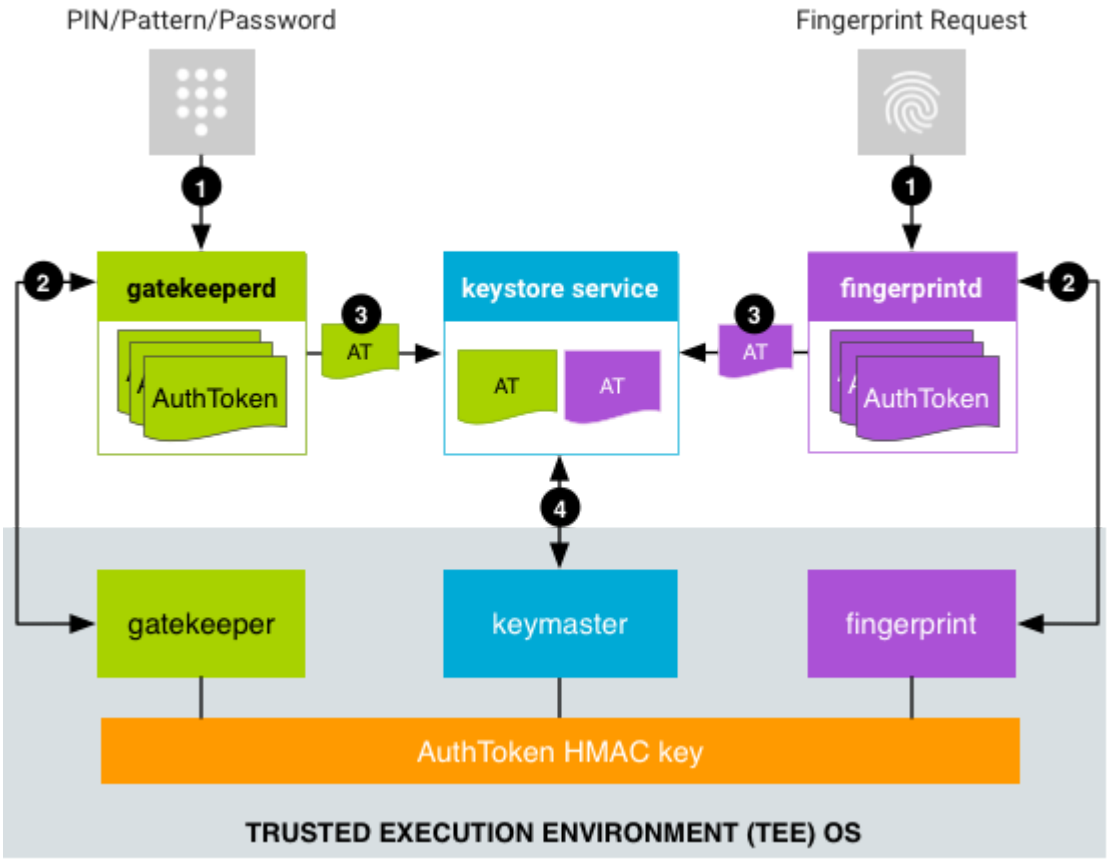

图 1. 身份验证流程

1. 用户提供身份验证方法，然后关联的服务向关联的守护进程发出请求。
    - 对于 PIN 码、解锁图案或密码，`LockSettingsService` 会向 `gatekeeperd` 发出请求。
    - 基于生物识别技术的身份验证流程取决于 Android 版本。在搭载 Android 8.x 及更低版本的设备上，`FingerprintService` 会向 `fingerprintd` 发出请求。在搭载 Android 9 及更高版本的设备上，`BiometricPrompt` 会使用合适的 ***`Biometric`***`Manager` 类（如 `FingerprintManager` 或 `FaceManager`）向相应的生物识别守护进程发出请求（例如，若是指纹识别身份验证，则向 `fingerprintd` 发出请求；若是人脸识别身份验证，则向 `faced` 发出请求）。无论什么版本，生物识别身份验证都会在请求发出后异步进行。
2. 守护进程将数据发至其副本，后者生成 AuthToken：
    - 对于 PIN 码/解锁图案/密码身份验证，`gatekeeperd` 将 PIN 码、解锁图案或密码哈希发送到 TEE 中的 Gatekeeper。如果 TEE 中的身份验证成功，TEE 中的 Gatekeeper 会将包含相应用户 SID（已使用 AuthToken HMAC 密钥签名）的 AuthToken 发送到它在 Android 操作系统中的副本。

- 对于指纹识别身份验证，**fingerprintd** 会监听指纹事件并将数据发送到 TEE 中的 Fingerprint。如果 TEE 中的身份验证成功，TEE 中的 Fingerprint 会将 AuthToken（已使用 AuthToken HMAC 密钥签名）发送到它在 Android 操作系统中的副本。
- 对于其他生物识别身份验证，相应的生物识别守护进程会监听生物识别事件，并将其发送到相应的生物识别 TEE 组件。

3. 守护进程收到经过签名的 AuthToken，并通过 Keystore 服务 Binder 接口的扩展程序将 AuthToken 传递给 Keystore 服务。（**gatekeeperd** 还会在设备被重新锁定以及设备密码发生变化时通知 Keystore 服务。）
4. Keystore 服务将 AuthToken 传递给 Keymaster，并使用与 Gatekeeper 和支持的生物识别 TEE 组件共用的密钥来验证这些 AuthToken。Keymaster 会将令牌中的时间戳视为最后一次身份验证的时间，并根据该时间戳做出密钥发布决定（以允许应用使用相应密钥）。

注意：设备重新启动后，AuthToken 即作废。

AuthToken 格式

为了确保在各种语言和组件之间实现令牌的共用和兼容，**hw_auth_token.h** (https://android.googlesource.com/platform/hardware/libhardware/+/master/include/hardware/hw_auth_token.h) 中规定了 AuthToken 的格式。该格式是一个简单序列化协议，具有以下固定大小的字段。

字段	类型	必需	说明
AuthToken 版本	1 个字节	是	下方所有字段的组代码。
质询	64 位未签名整数	否	用于防范重播攻击的随机整数，通常是所请求的加密操作的 ID。目前由交易指纹授权使用。如果质询存在，AuthToken 将仅对包含该相同质询的加密操作有效。
用户 SID	64 位未签名整数	是	不重复的用户标识符，以加密形式绑定到与设备身份验证关联的所有密钥。如需了解详情，请参阅 Gatekeeper (/security/authentication/gatekeeper.html)。
身份验证程序 ID (ASID)	64 位未签名整数，按网络字节序保存	否	绑定到特定身份验证程序政策时使用的标识符。所有身份验证程序都有自己的 ASID 值，它们可以根据自己的要求更改该值。

身份验证程序类型	32 位未签名整数，按网络字节序保存	是	• 0x00 代表 Gatekeeper。 • 0x01 代表 Fingerprint。
时间戳	64 位未签名整数，按网络字节序保存	是	自最近一次系统启动以来已经过的时间（以毫秒为单位）。
AuthToken HMAC (SHA-256)	256 位 Blob	是	除 HMAC 字段以外所有字段的已加密 SHA-256 MAC。

设备启动流程

每次设备启动时，都必须生成 AuthToken HMAC 密钥并由所有 TEE 组件（Gatekeeper、Keymaster 以及支持的生物识别 Trustlet）共用该密钥。因此，为了加强对重播攻击的防范力度，每次设备重新启动时都必须随机生成 HMAC 密钥。

关于与所有组件共用此 HMAC 密钥的协议是一项依赖于平台的实现功能。**在任何情况下都不能**将该密钥设为在 TEE 之外可用。如果 TEE 操作系统缺少内部进程间通信 (IPC) 机制，需要通过不可信操作系统传输数据，那么传输操作必须通过安全的密钥交换协议进行。

与 Android 并排运行的 Trusty (/security/trusty/index.html) 操作系统就是一种 TEE，不过也可以使用其他 TEE。Trusty 使用内部 IPC 机制在 Keymaster 和 Gatekeeper 或相应的生物识别 Trustlet 之间直接进行通信。HMAC 密钥只保存在 Keymaster 中，Fingerprint 和 Gatekeeper 会在每次使用时向 Keymaster 请求该密钥，而不会保留或缓存该密钥的值。

由于一些 TEE 缺少 IPC 基础架构，因此 TEE 中的小程序之间不会进行通信。这还使得 Keystore 服务因知晓系统中的身份验证表而能够快速拒绝注定会失败的请求，从而避免向 TEE 发送可能会占用大量资源的 IPC。