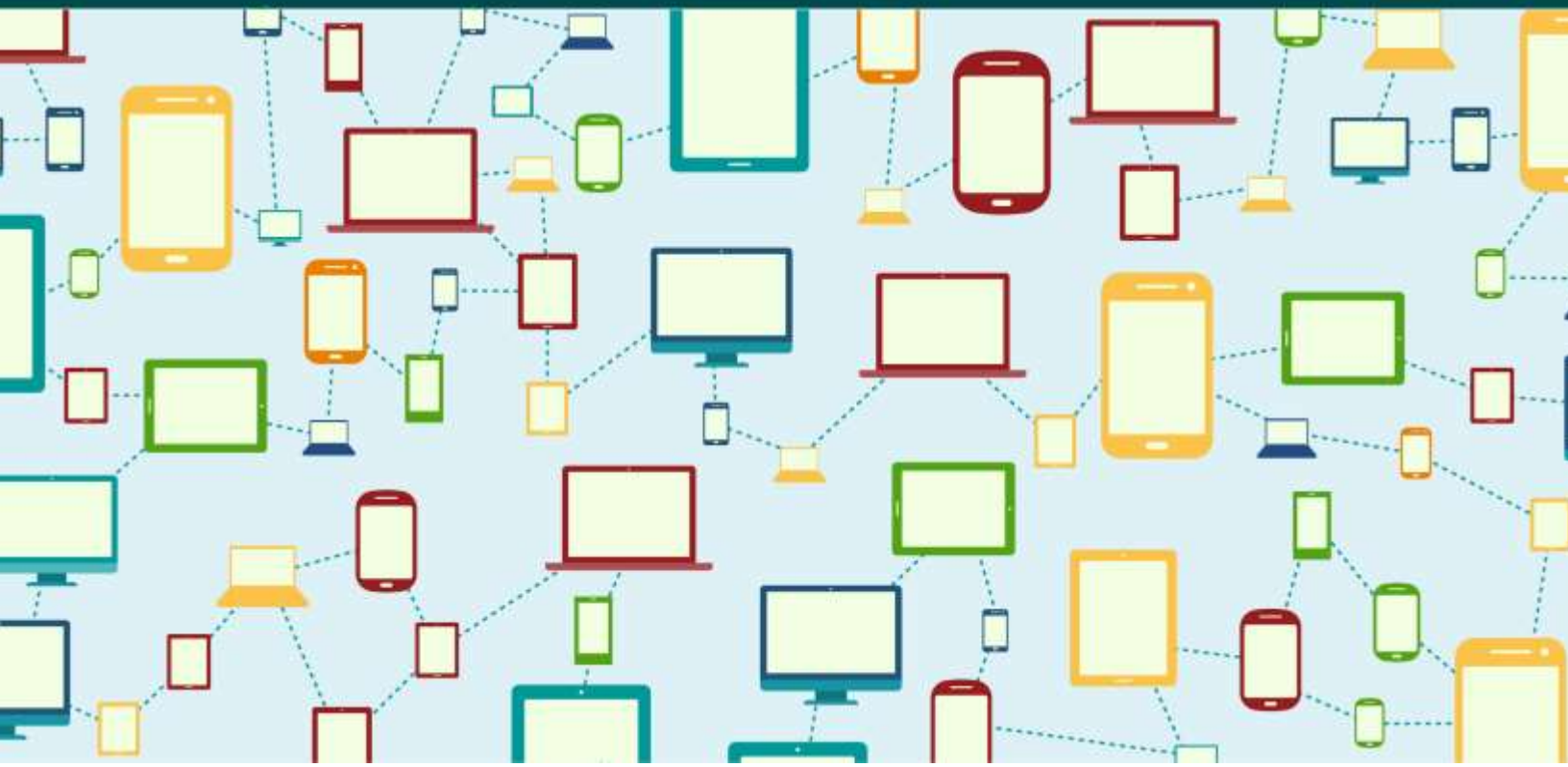
fido ALLIANCE | simpler stronger authentication



FIDO Alliance White Paper:

Hardware-backed Keystore Authenticators (HKA) on Android 8.0 or Later Mobile Devices

Enabling Any Relying Parties to Create FIDO UAF (1.1 or later) Client Apps

June 2018

Editors: Dr. Max Hata, NTT DOCOMO, Inc.
Dr. Rolf Lindemann, Nok Nok Labs

Table of Contents

Audience

This paper is aimed at Relying Parties and Android application developers who are interested in developing or deploying FIDO authentication systems that leverage the FIDO UAF protocol.

Summary

In 2017, Android 8.0 began supporting Android Keystore with hardware-backed key attestation which enables your servers to verify provenance of the Keystore implementation. Almost all Android mobile devices (8.0 or later) now support Keystore with key attestation and fingerprint sensors in hardware. This milestone allows Android mobile device app developers and Relying Parties to build and deploy FIDO UAF authenticators without being dependent on underlying UAF specific hardware. This is an epoch-making event to implement FIDO UAF authenticators since Relying Parties (RPs) and app developers can enable such Android mobile devices to become secure FIDO UAF authenticators by just adding applications. This paper introduces the details of such an implementation approach, based on the first commercial deployment [4]. It takes advantage of Android Keystore with key attestation and fingerprint sensors in hardware on standard off-the-shelf Android mobile devices. The first deployment of this approach has been proven to work successfully in the ecosystem with all the other UAF devices based on the conventional device implementation approach.

1. Introduction

The typical way to develop secure authenticators on Android mobile devices (smartphones and tablets) is to use a secure hardware-backed operating environment (referred to in this paper as a Restricted Operating Environment or ROE). This includes technology such as a TEE (Trusted Execution Environment) that performs cryptographic and other sensitive operations including digital signing and biometric data processing used to support FIDO functionality. FIDO Alliance relies on its Authenticator Certification program to define authenticator security levels based on such an ROE as Level 2. Using a ROE, a Level 2 authenticator defends against large scale attacks and OS compromise.

Android Keystore allows app developers to store cryptographic keys in a container and use them in cryptographic operations via APIs. Android also offers protection of fingerprint sensor data via a TEE, which allows for encryption and cryptographic authentication.

In 2017, Android 8.0 was released. In addition to the Keystore and fingerprint sensor, Android 8.0 or later supports key attestation in secure hardware[1]. "Key attestation affirms that a crypto key lives in secure hardware and has specific characteristics [1]". Google provides the root CA and certifies attestation keys. Thus, an application can attest the provenance of the Keystore implementation as well as if it supports secure hardware. This is done by examining the certificates provided through Android APIs.

UAF 1.1 supports Android key attestation among other attestation schemes; Android key attestation defined in UAF 1.1 complies with Android's key attestation. One of the basic requirements for a FIDO Level 2 authenticator is that it must support attestation. Android key attestation satisfies this requirement. Android Keystore implementations with hardware-backed key attestation with fingerprint sensors could satisfy all the FIDO Level 2 authenticator certification requirements and are referred to in this paper as "L2-candidate".

Adding a small application to complement some functions that are necessary to perform as a UAF 1.1 authenticator enables Android Keystore with hardware-backed key attestation and fingerprint sensor to become a secure FIDO UAF 1.1 authenticator. In this paper, such an authenticator is called a "**H**ardware-backed **K**eystore **A**uthenticator (HKA)".

HKAs are secure thanks to hardware-backed Keystore with Android key attestation and fingerprint sensor data processing executed in a ROE, which is an L2-candidate.

Until HKAs became available, only handset vendors were able to develop and offer FIDO Level 2 UAF authenticators for Android. Such authenticators require OS customization to support hardware-backed attestation keys and the Full Basic Attestation[2] defined in the UAF specifications. These implementations offer a strong security level that can achieve FIDO Level 2 or higher. This document, however, does not go into the details of such implementations as their customizations may differ from one handset vendor to another.

With HKAs, Relying Parties (RP) and application developers are now able to enable a standard off-the-shelf Android (8.0 or later) mobile device with a fingerprint sensor to become a secure FIDO UAF 1.1 authenticator by adding a small application leveraging the building blocks provided by the L2-candidate platform. No OS customization is

[1] Before Android 8, Android 7 first started supporting key attestation primarily in software. Only a small number of devices running Android 7 support hardware-backed key attestation [9] and therefore it is not relevant to this white paper that is focused on hardware-backed key attestation.

[2] Full Basic Attestation is one of the attestation flavors that FIDO UAF supports. It is based on an attestation private key shared among a class of authenticators (e.g. same model). FIDO Servers verify the signature signed by an attestation private key using a trust anchor which is the root certificate of the public attestation keys.

required, and RPs/app developers can now offer their own secure FIDO UAF authenticator instead of relying on handset vendors to support the functionality.

The vast majority of coming Android mobile devices will be the target of HKAs because Android's compliance requirements mandate supporting key attestation and strongly recommends supporting fingerprint sensors.

All the essential building-blocks to develop a secure UAF authenticator are now provided by Android: Keystore for cryptographic operations, key attestation and related certificates chaining up to the Google root CA, and fingerprint sensors with secure processing in a TEE.

This white paper introduces the details of HKAs for UAF 1.1.

The first HKAs based on UAF 1.1 [4] have successfully been deployed for various services in the same way as all the current FIDO UAF authenticators that were based on OS customization. HKAs are commercially proven to provide the same quality, usability and security as those built with OS customizations.

2. Android Keystore and Key Attestation

This section gives an overview of the essential building-blocks in Android (8.0 or later) that enable HKAs.

2.1 Keystore

The Android Keystore[5] allows app developers to store cryptographic keys in a container and use them in cryptographic operations through the KeyChain API or the Keystore API.

The Keystore keeps the key material out of the app's process space so the application cannot inadvertently reveal it to the user.

Many Android devices also provide hardware-backed security for Keystore keys in secure hardware such as TEEs. This keeps the key material out of the Android system entirely, and the key material cannot be leaked even by a Linux kernel compromise.

To mitigate unauthorized use of keys on the device, Keystore lets applications specify authorized uses of their keys when generating or importing the keys (e.g. fingerprint-based user verification). Once a key is generated or imported, its authorizations cannot be changed. Authorizations are then enforced by the Keystore whenever the key is used. Concerning the security of an HKA, this is how the trust relationship between the hardware-backed Keystore and the application is ensured.

2.2 Key Attestation

Key attestation [1][2][3] allows the server to verify that the requested key lives in secure hardware, i.e., the attestation signing key is protected by secure hardware such as TEEs and signing is performed in the secure hardware. It also allows servers to verify that each use of the key is gated by user verification, preventing unauthorized uses of the key.

The attestation statement is signed by an attestation key injected into the secure hardware at the factory. Attestation statements are produced in the form of an X.509 certificate. Google provides the root CA and certifies attestation keys to each vendor (Appendix-A shows an example of the attestation certificates). This satisfies the basic requirements of FIDO Level 2 authenticators, i.e., operation contained in a ROE and also provides support for attestation. The attestation signing keys must be shared across a large enough number (100,000 or more) of devices to prevent the keys from being used as device identifiers. The last feature fulfills one of the key principles of FIDO specifications, which is to protect users' privacy.

2.3 Fingerprint Sensor

Android 8.0 or later strongly recommends that Android mobile devices include a fingerprint sensor (7.3.10., CDD 8.1[2]). Further, if a device includes a fingerprint sensor and makes it available to third-party applications, it mandates:

1) a hardware-backed Keystore implementation, and perform the fingerprint matching in a Trusted Execution Environment (TEE) or on a chip with a secure channel to the TEE, and

2) all identifiable fingerprint data encrypted and cryptographically authenticated such that they cannot be acquired, read or altered outside of the Trusted Execution Environment (TEE).

These mandatory requirements ensure that fingerprint sensors function securely, protect users' privacy, and are consistent with the cryptographic operations in Keystore.

Additional device requirements, if fingerprint sensors are supported, include the mandate that they:

- MUST have a false acceptance rate (FAR) not higher than 0.002%.
- Are STRONGLY RECOMMENDED to have a spoof and imposter acceptance rate not higher than 7%. (New from 8.1)
- MUST rate limit attempts for at least 30 seconds after five false trials for fingerprint verification.
- MUST prevent adding a fingerprint without first establishing a chain of trust by having the user confirm existing or add a new device credential (PIN/pattern/password) that's secured by TEE.
- MUST NOT enable 3rd-party applications to distinguish between individual fingerprints.

These requirements are adequate for Android 8.0 or later mobile devices to qualify as an L2-candidate.

3. The Trend of Android Mobile Devices

Android compliance is defined by the Compatibility Definition Document (CDD) [6]. The trend of Android mobile devices relative to Keystore and biometric sensors, as defined by the CDD[3], is summarized as follows:

A) Mandatory support of the Keystore implementation with secure hardware like a TEE

B) Mandatory support of key attestation where the attestation signing key is protected by secure hardware and signing is performed in secure hardware, and

C) Strong recommendation to include a fingerprint sensor.

Thus, we can expect **almost all coming Android mobile devices (8.0 or later) will support all these features and can be the target for HKAs.**

The details of the relevant CDD requirements are shown in Appendix-B.

[3] As shown in Appendix-B, all these mandatory requirements are conditional on "If device implementations include a secure lock screen, …". Since almost all modern commercial mobile devices (smartphones and tablets) include a secure lock screen, these conditional requirements can be considered as mandatory requirements for such mobile devices.

4. Implementation of HKA using UAF 1.1

4.1 Architecture of UAF 1.1 HKAs

An example of typical architecture of UAF 1.1 HKAs is shown in Figure 1.

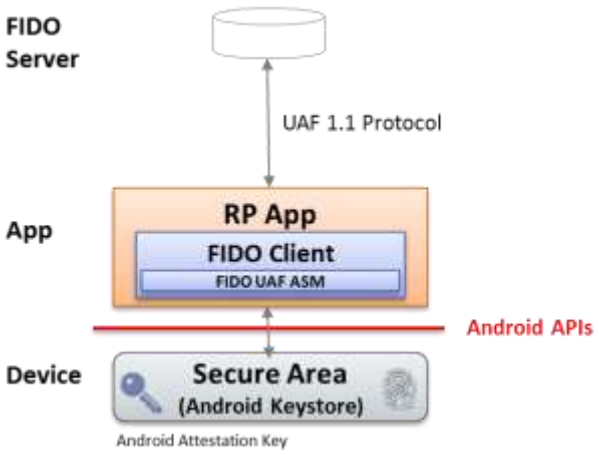


Figure 1. An example of architecture of HKAs.

In the case of HKA, the FIDO UAF ASM also implements the non-security critical aspects of the authenticator that are not offered by the Keystore.

The FIDO ASM module talks to the Keystore and fingerprint sensor through the standard Android APIs. The approach is similar to what has been used to enable iOS for UAF except that an HKA includes key attestation. Key attestation is a particularly important feature for a multi-vendor ecosystem, such as Android, to ensure the provenance and security of the Keystore that can achieve Level 2 FIDO Authenticator Certification.

“Secure Area” in Figure 1 means a Keystore implementation with Android key attestation in a secure environment such as a Trusted Execution Environment (TEE) or a Hardware Security Module (HSM). Fingerprint matching is performed in the TEE or on a chip with a secure channel to the TEE. An HSM is a class of security hardware, e.g., embedded Secure Elements (eSE), to support a StrongBox Keymaster that was introduced in Android P [7]. The HSM adds mechanisms to resist package tampering and unauthorized sideloading of apps. When checking keys stored in the StrongBox Keymaster, the system corroborates a key's integrity with the TEE.

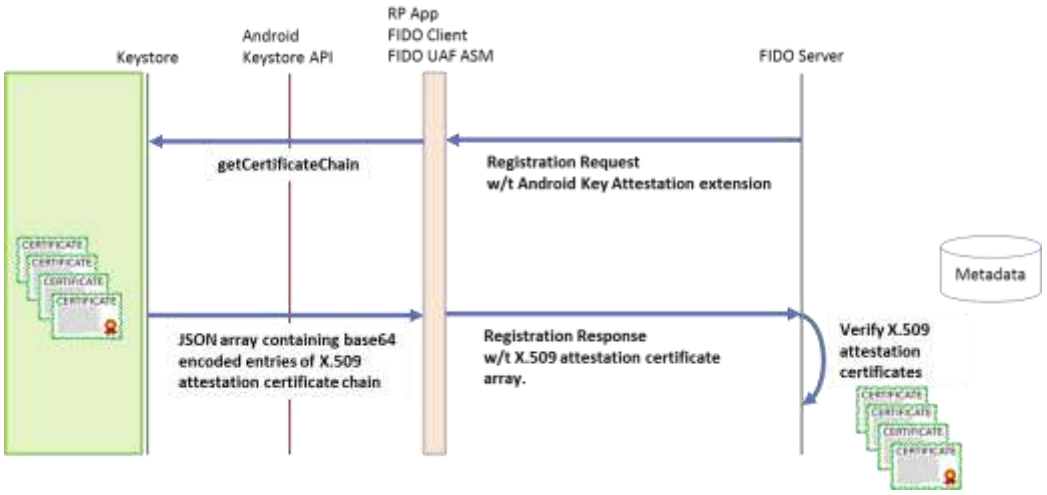


Figure 2. Registration Sequence of the Android Key Attestation Extension.

Figure 2 shows the registration sequence of the Android key attestation extension. The FIDO server sends a registration request with the Android key attestation extension. The request causes the application to trigger a call into the FIDO UAF ASM. From there, a call is made into the Android Keystore API to generate the authentication key pair and also a call into the API with getCertificateChain to get the attestation for the key pair. The Keystore returns a JSON array containing base 64 encoded entries of the X.509 attestation certificate chain. These certificates chain up to the root certificate of Google's root CA. The leaf certificate is the signed message attesting that the key was generated by the hardware-backed Keystore. This array is added as an extension to the FIDO UAF response that is sent back to the FIDO server. The server verifies the certificate chain and registers the authenticator, referencing the metadata. The FIDO server will know, based on the authenticator model, whether or not to expect the Android key attestation extension in the response.

As discussed in the Keystore section above, the trust relationship between the Keystore and the calling application is ensured by the Keystore.

4.2 Implementation Notes

1) Client Side

- You must support Android key attestation as defined in UAF 1.1.
- You need to declare isKeyRestricted=false in the Metadata Statement. By setting isKeyRestricted=false, it tells the server that the authenticator doesn't restrict the authentication private key to only sign valid FIDO signature assertions. If this field is missing, the assumed value is isKeyRestricted=true [14]. If isKeyRestricted=true, then FIDO authenticators other than keystore based ones, such as an HKA, restrict the private key for authentication only to sign FIDO signature assertions.

2) Server Side

- A UAF 1.1 server must support UAF 1.1 Errata [12] to process X.509 certificates of Android key attestation correctly. There is a minor deviation in Google's certificates from a typical X.509 certificate (See Appendix-A). The Errata describes the details on the issue.

3) **Conformance Test Tools**

- FIDO's UAF Test Tool already supports these features for certification testing. The test tools are available for download from https://fidoalliance.org/certification/conformance/.

4.3 Implementation Options

One can develop the application from scratch based on the UAF 1.1 specification or alternatively can use one of the SDKs that software vendors are offering in order to minimize complexity and time-to-market.

5. Design Considerations

When designing FIDO authentication systems based on HKAs, a few aspects may need to be considered.

5.1 User Verification Index (UVI) for Fingerprint

Android does not support features to enable the user verification index (UVI) for any applications like an HKA. The user verification index (UVI) is defined by the FIDO UAF specification as one of the optional extensions. A UVI is a value uniquely identifying a user verification data record to one specific relying party account. For fingerprint authentication, it can be used by a FIDO authenticator to let the server know specifically whether the finger used for authentication was exactly the same as the one used for registration. This concept allows relying parties to distinguish legitimate users from "friends" that might have enrolled their biometric to the same device (e.g. for playing games) – this is also known as protection against "friendly fraud".

Android only supports the so-called "Any Finger Matching". It tells the server that fingerprint verification was successful, but it does not tell which individual finger was verified. Thus, an HKA can only support Any Finger Matching. The following 2 requirements are the relevant ones in the Android CDD [2]:

> [C-1-8] MUST prevent adding a fingerprint without first establishing a chain of trust by having the user confirm existing or add a new device credential (PIN/pattern/password) that's secured by TEE; the Android Open Source Project implementation provides the mechanism in the framework to do so.
>
> [C-1-9] MUST NOT enable 3rd-party applications to distinguish between individual fingerprints.

Note that the first requirement prevents friendly fraud.

5.2 Multiple Applications and AppID/FacetIDs

As defined by the UAF specification, AppID/FacetIDs enables multiple applications (or Facets) on various platforms to access the same FIDO credentials when those Facets belong to the same Application Identifier. For example, the MyBank application may have an Android app, an iOS app, and a Web app. These are all facets of the MyBank application. In FIDO UAF, the relying party can specify a TrustedFacetsList, i.e. a list of FacetIDs that belong to one AppID. The AppID is the URL to download the TrustedFacetsList from (see FIDO AppID and Facet Specification).

A FacetID is a platform-specific identifier (URI) for an application facet. For Android applications, the FacetID is the URI android:apk-key-hash:<hash-of-apk-signing-cert>.

An AppID is sent from the FIDO server in a registration (or authentication) request to the FIDO Client and is a URL pointing to the list of FacetIDs related to the AppID. The Client downloads the FacetIDs based on the AppID and determines if the application can register to (or authenticate with) the authenticator. By listing all the FacetIDs with the AppID, the FacetIDs/AppID feature enables multiple client applications to access the credentials in the authenticator.

Some existing deployments use this feature to enable multiple applications, installed on the same Android device, for FIDO servers belonging to the same RP sharing user accounts. AppID/FacetIDs allows use of applications that are installed on the device and listed in the FacetIDs, without registering with each application once the user has registered with one of the applications. Some OEM-based device implementations expose an ASM interface to applications that allows multiple applications to access the ASM interface to support the AppID/FacetIDs feature in a straightforward way.

In the case of HKAs, Android allows only one application with the access-privilege to access the corresponding credential in the Keystore. So, the generic AppID/FacetID approach specified by FIDO UAF is restricted by the Keystore concept (Figure 3 (a)). Consequently, multiple HKA applications cannot share the same key in the Keystore. This separation is quite a reasonable security measure for Android as a platform to protect credentials in the Keystore from malicious applications. But it may pose a challenge when attempting to realize similar multi-application use cases in some deployments that use this approach.

One of the solutions is illustrated in Figure 3 (b), where a 'Proxy App' with the privilege access to the FIDO credentials -- whereas other applications talk to the Proxy App using the app-to-app communications.

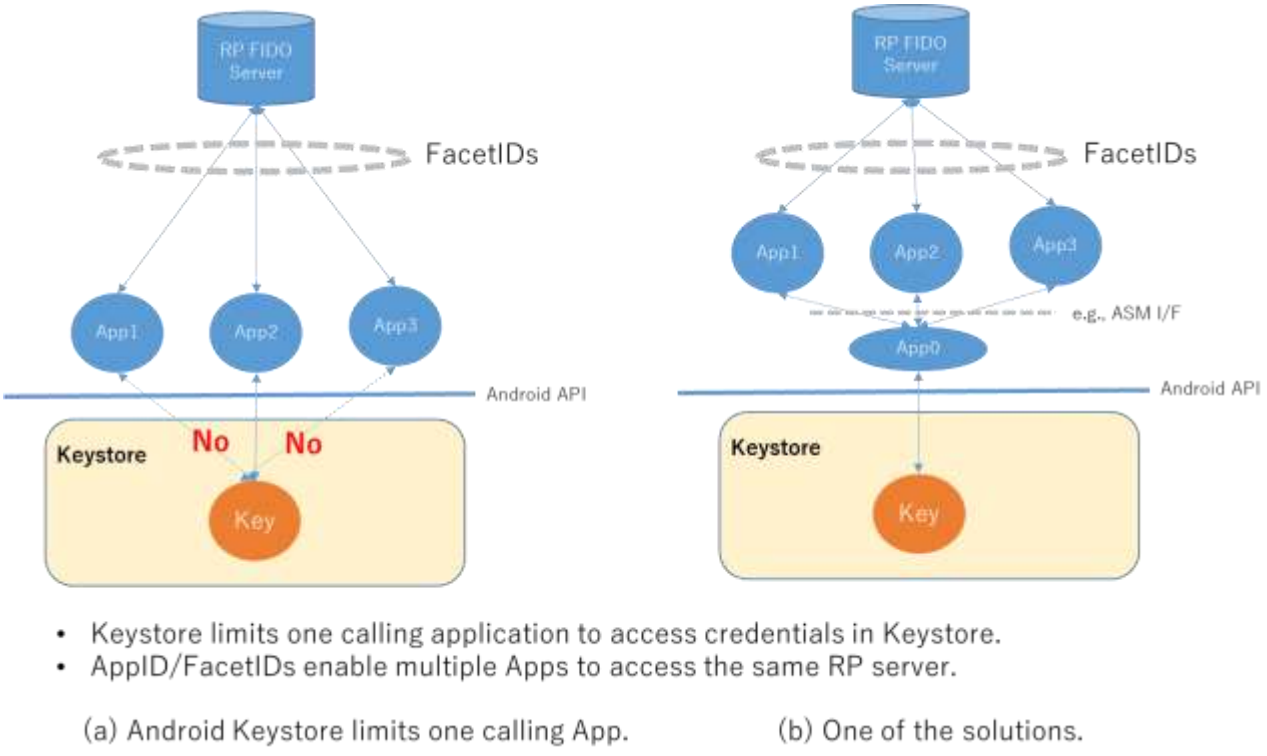


Figure 3. (a) Android Keystore limits credential access to one calling App, (b) One of the solutions using a 'Proxy App' (App0).

In cases where such a solution like Figure 3 (b) does not meet your goal, there are other architectures to avoid the one-device-with-multiple-applications situation and allow multiple servers to share a FIDO key in the Keystore.

One example is use of federation protocols such as OpenID Connect [8]. Federation can enable linking many different RP services, including the ones supported by that RP, as well as ones from third-parties, where the user needs to register and authenticate with one server from one application on each device (each application on

different devices is listed in the FacetIDs). It provides an open, flexible and extensible architecture to enable various multiple services sharing a FIDO key.

6. Advantages of HKA

This section discusses the advantages of an HKA.

Figure 4. Advantages of an HKA.

1) Relying Parties

The HKA approach enables anyone to develop FIDO UAF enabled apps for almost all new Android mobile devices, using the standard Android APIs and FIDO UAF Client SDK. Furthermore, the HKA approach reduces development costs and increases device coverage for RP services. Only minor updates are required to support UAF 1.1 (for HKA) on RP UAF 1.0 servers. (Namely, FIDO servers need to send registration or authentication requests with the Android key attestation extension and process Android key attestation certificates that are sent from the devices.)

2) FIDO Ecosystem

HKA stands to dramatically accelerate FIDO adoption by rapidly growing both the number of FIDO supporting devices and service deployments supporting FIDO authentication.

3) End Users

The HKA approach increases the range of device models supporting FIDO biometric authentication.

4) Handset Vendors

No OS customization is required to enable UAF authenticators to be candidates for FIDO Level 2 Certification. The comparison of the two typical implementation approaches is shown in Figure 5. This new approach saves time and cost. Note that OS customization is still required for other modalities, e.g., iris[4].

[4] In May 2018, Android P announced a new API to support Face and Iris authentication in addition to Fingerprint authentication [13].

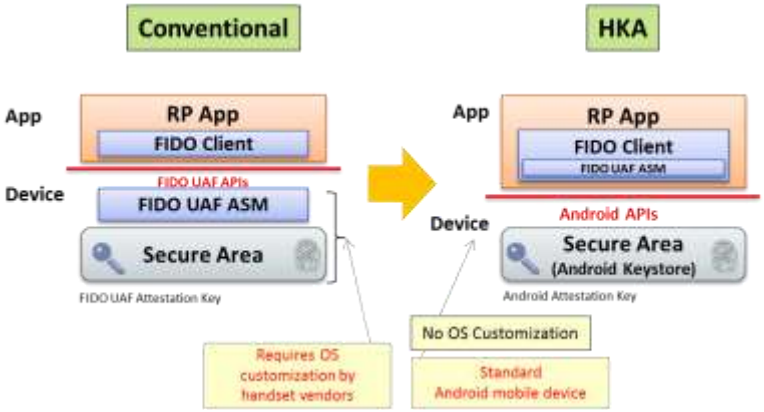


Figure 5. Comparison of conventional implementation and new HKA implementation.

7. Conclusions

This paper introduces the HKA implementation approach for UAF 1.1, based on the successful first commercial deployment [4]. Almost all the coming Android mobile devices will be the target of HKAs because Android compliance mandates supporting key attestation and strongly recommends supporting fingerprint sensors. All the essential building-blocks to develop a secure UAF authenticator are now provided in Android.

Anyone is now able to enable a standard off-the-shelf Android (8.0 or later) mobile device with fingerprint sensor to become a secure FIDO UAF 1.1 authenticator. All it takes is a small application leveraging the security features of the L2-candidate platform. No OS customization is required, and RPs/app developers are now capable of offering their own FIDO UAF authenticator instead of relying on handset vendors.

HKA dramatically accelerates FIDO adoption, the number of devices that support FIDO, and service deployments supporting FIDO authentication.

It is highly recommended to consider adopting the HKA implementation approach for the coming Android mobile devices.

8. Acknowledgements

The authors acknowledge the following people for their valuable feedback and comments:

- Laurence Lundblade, Qualcomm
- Christiaan Brand, Google
- Koichi Moriyama, NTT DOCOMO
- Giridhar Mandyam, Qualcomm
- Ki-Eun Shin, SK Planet
- Hisashi Yoshinaga, NTT DOCOMO
- John Fontana, Yubico
- Béatrice Peirani, Gemalto
- Naga Nagarajan, Nok Nok Labs

- Andrew Shikiar, FIDO Alliance

9. References

[1] Keystore key attestation, 28 September 2017, Shawn Willden, Software Engineer

https://android-developers.googleblog.com/2017/09/keystore-key-attestation.html

[2] Android CDD 8.1, December 5, 2017

https://source.android.com/compatibility/8.1/android-8.1-cdd.pdf

[3] Android Bootcamp 2016, Android Keystore Attestation, January 21, 2016,

https://source.android.com/security/reports/Android-Bootcamp-2016-Android-Keystore-Attestation.pdf

[4] FIDO Press Release: "First FIDO UAF 1.1 Implementations Ease Deployment of Advanced Biometric Authentication on Android Devices", December 7, 2017

https://fidoalliance.org/first-fido-uaf-1-1-implementations-ease-deployment-advanced-biometric-authentication-android-devices/

[5] Hardware-backed Keystore, https://source.android.com/security/keystore/

[6] Android Compatibility Definition Document (CDD), https://source.android.com/compatibility/cdd

[7] Hardware security module, https://developer.android.com/preview/features/security.html

[8] FIDO Alliance White Paper: Enterprise Adoption Best Practices (December 2017),

https://fidoalliance.org/wp-content/uploads/Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf

[9] Certificate Extension Data Schema, https://developer.android.com/training/articles/security-key-attestation.html#certificate_schema

[10] Key and ID Attestation, https://source.android.com/security/keystore/attestation

[11] Keymaster Authorization Tags, https://source.android.com/security/keystore/tags

[12] FIDO Alliance Errata UAF 1.1, https://fidoalliance.org/download/

[13] Android Developers Blog, https://android-developers.googleblog.com/2018/05/whats-new-in-android-p-beta.html

[14] FIDO Metadata Statements, https://fidoalliance.org/specs/fido-uaf-v1.1-ps-20170202/fido-metadata-statement-v1.1-ps-20170202.html#widl-MetadataStatement-isKeyRestricted

10. Appendix

10.1 Appendix-A An Example of an Attestation Certificate Chain

This section introduces an example of an X.509 certificate chain of Android key attestation.

It consists of 4 certificates and they are standard X.509 certificates with optional extensions. The certificate chain is extracted from a sample device, an off-the-shelf Pixel 2, using an application developed for this paper.

Certificate 0 is the certificate of a public key that was generated for an attested key from an application for this paper, whereas Certificate 3 is the root certificate. Certificates 1,2, and 3 are injected in the Keymaster of the device by each vendor, and Certificate 0 is generated by an Android application.

The optional attestation extension in the first certificate (i.e., Certificate 0) contains various descriptions of the attested key. They are defined by the ASN.1 schema [9][10][11].

Certificate 0

-----BEGIN CERTIFICATE-----

MIICgjCCAiigAwIBAgIBATAKBggqhkjOPQQDAjAbMRkwFwYDVQQFExBjNjA0NzU3MWQ4ZjBkMTdjMB4XDTcwMDEwMTAwMDAwMFoXDTM4MDExOTAzMTQwN1owHzEdMBsGA1UEAwwUQW5kcm9pZCBLZXlzdG9yZSBLZXkwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQhAZeExQaRmffwzDPu/UpO/eh4L7Kxa/S8EmRXYPosgOWgqgEWpMiYZS5kSKCRQ4rOTfQPiZN6sCd+ZmfZaarCo4IBVzCCAVMwDgYDVR0PAQH/BAQDAgeAMIIBPwYKKwYBBAHWeQIBEQSCAS8wggErAgECCgEBAgEDCgEBBCCZxmGfwTUq41TvlFBOq2hiCdIV+wacJ7OZB/kqu5MavQQAMIGEv4MQAwIBAL+DEQgCBgHz///8GL+DEggCBgHz///8GL+FPQgCBgFizZRu2b+FRVUEUzBRMSswKQQiY29tLmV4YW1wbGUuYW5kcm9pZC5rZXlhdHRlc3RhdGlvbgIDAWgoMSIEIBg8po4noiIv3i1GPeehbaP88dqAjfxC3hMmyPY/YZTTMHKhBTEDAgECogMCAQOjBAICAQClBTEDAgEEqgMCAQG/g3gDAgECv4U+AwIBAL+FPwIFAL+FQCowKAQgnBLP3ATHRYTXh6w9I3chMsGFJLx6so3sQhm4/FtCX3ABAQAKAQK/hUEFAgMBOOS/hUIFAgMDFEwwCgYIKoZIzj0EAwIDSAAwRQIgDhWogzzymtmnVC/R697228lhKAdG2c7xr7fSUJ7a3oQCIQCaGNyoAHmAh6wj8Hl0pUZHJkUsVXNpZErjeWXbblOdaA==

-----END CERTIFICATE-----

Certificate 1

-----BEGIN CERTIFICATE-----

MIICKzCCAbKgAwIBAgIKEXhzFQJ5hgIAEDAKBggqhkjOPQQDAjAbMRkwFwYDVQQFExA4N2Y0NTE0NDc1YmEwYTJiMB4XDTE2MDUyNjE3MTUwMloXDTI2MDUyNDE3MTUwMlowGzEZMBcGA1UEBRMQYzYwNDc1NzFkOGYwZDE3YzBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABOoJkrjZcbPu6IcDxyrvlugASVQm5MX7OGGT0T34rzlwwbR9UV2ATu6aMiEa8uuQdP3iy5qSUYeCzUuneIdo7dujgd0wgdowHQYDVR0OBBYEFHlfwP7+91r1xLPq/o7/eYXAU9ocMB8GA1UdIwQYMBaAFDBEI+Wi9gbhUKt3XxYWu5HMY8ZZMAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMCQGA1UdHgQdMBugGTAXghVpbnZhbGlkO2VtYWlsOmludmFsaWQwVAYDVR0fBE0wSzBJoEegRYZDaHR0cHM6Ly9hbmRyb2lkLmdvb2dsZWFwaXMuY29tL2F0dGVzdGF0aW9uL2NybC8xMTc4NzMxNTAyNzk4NjAyMDAxMDAKBggqhkjOPQQDAgNnADBkAjAMOvX7podpWf2gJjzut3Woz/bq1B42pC7Bu511pv1zj4jbtsdhhYCo/u/pnylG3LMCMCgdkdZQBPOEaJuBTYmxGiWrqVFe6vTsX60SJ4vqa1PruSZzEFcyukXMckPn1wcz8A==

-----END CERTIFICATE-----

Certificate 2

-----BEGIN CERTIFICATE-----

MIIDwzCCAaugAwIBAgIKA4gmZ2BliZaFdTANBgkqhkiG9w0BAQsFADAbMRkwFwYDVQQFExBmOTIwMDllODUzYjZiMDQ1MB4XDTE2MDUyNjE3MDE1MVoXDTI2MDUyNDE3MDE1MVowGzEZMBcGA1UEBRMQODdmNDUxNDQ3NWJhMGEyYjB2MBAGByqGSM49AgEGBSuBBAAiA2IABGQ7VmgdJ/rEgs9sIE3rzvApXDUMAaqMMn8+1fRJrvQpZkJfOT2EdjtdrVaxDQRZxixqT5MlVqiSk8PRTqLx3+8OPLoicqMiOeGytH2sVQurvFynVeKqSGKK1jx2/2fccqOBtjCBszAdBgNVHQ4EFgQUMEQj5aL2BuFQq3dfFha7kcxjxlkwHwYDVR0jBBgwFoAUNmHhAHyIBQlRi0RsR/8aTMnqTxIwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAYYwUAYDVR0fBEkwRzBFoEOgQYY/aHR0cHM6Ly9hbmRyb2lkLmdvb2dsZWFwaXMuY29tL2F0dGVzdGF0aW9uL2NybC9FOEZBMTk2MzE0RDJGQTE4MA0GCSqGSIb3DQEBCwUAA4ICAQBAOYqLNryTmbOlnrjnIvDoXxzaLOgCXu29l7KpbFHacVLxgYuGRiIEQqzZBqUYSt9Pgx+P2KvoHtz99sEZr2xTe0Dw6CTHTAmxWXUFdrlvEMm2GySfvJRfMNCuX1oIS/M5PfREY2YZHyLq/sn1sJr3FjbKMdUMBo5AcamcD3H8wl9O/6qfhX+57iXzoK6yMzJRG/Mlkm58/sFk0pjayUBchmUJL0FQ6IhKYgy8RKE2UDyXKOE7+ZMSMUUkAdzyn2PFv7TvQtDk0ge2mkVrNrfPSglMzBNvrSDHPBmTktXzwseVagIRT5WI91OrUOYPFgostsfH42hs5w

JtAFGPwDg/1mNa8UyH9k1bMrRq3Srez1XG0Ju7SGN/uNX5dkcwvfAmadtmM7Pp+l2VHRYRR600jAcM2+7bl8egqfM/A7vy
DLZqPIxDwkLXj2eN99nJZJVaGfB9dHyFOqBqBM6SdyV6MSIr3AHoo6u+BWIX9+q8n1qg5I6JWeEe+K58SbRDVoNQgsKP9/iP
ruXMU5rm2ywPxICVGysl1GgAP+FJ3X6oP0tXFWQlYoWdSloSVHNZQqj2ev/69sMnGsTeJw1V7I0gR+eZNEfxe+vZD4KP88Kx
uiPCe94rp+Aqs5/YwuCo6rQ+HGi5OZNBsQXYIufClSBje+OpjQb7HJgihJdzo2/IBw==

-----END CERTIFICATE-----

Certificate 3 (root)

-----BEGIN CERTIFICATE-----

MIIFYDCCA0igAwIBAgIJAOj6GWMU0voYMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNVBAUTEGY5MjAwOWU4NTNiNmIwND
UwHhcNMTYwNTI2MTYyODUyWhcNMjYwNTI0MTYyODUyWjAbMRkwFwYDVQQFExBmOTIwMDllODUzYjZiMDQ1MIICIjAN
BgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAr7bHgiuxpwHsK7Qui8xUFmOr75gvMsd/dTEDDJdSSxtf6An7xyqpRR90PL2a
bxM1dEqlXnf2tqw1Ne4Xwl5jlRfdnJLmN0pTy/4lj4/7tv0Sk3iiKkypnEUtR6WfMgH0QZfKHM1+di+y9TFRtv6y//0rb+T+W8a
9nsNL/ggjnar86461qO0rOs2cXjp3kOG1FEJ5MVmFmBGtnrKpa73XpXyTqRxB/M0n1n/W9nGqC4FSYa04T6N5RIZGBN2z2
MT5IKGbFlbC8UrW0DxW7AYImQQcHtGl/m00QLVWutHQoVJYnFPlXTcHYvASLu+RhhsbDmxMgJJ0mcDpvsC4PjvB+TxywE
lgS70vE0XmLD+OJtvsBslHZvPBKCOdT0MS+tgSOIfga+z1Z1g7+DVagf7quvmag8jfPioyKvxnK/EgsTUVi2ghzq8wm27ud/mI
M7AY2qEORR8Go3TVB4HzWQgpZrt3i5MIlCaY504LzSRiigHCzAPlHws+W0rB5N+er5/2pJKnfBSDiCiFAVtCLOZ7gLiMm0jhO
2B6tUXHI/+MRPjy02i59lINMRRev56GKtcd9qO/0kUJWdZTdA2XoS82ixPvZtXQpUpuL12ab+9EaDK8Z4RHJYYfCT3Q5vNAX
aiWQ+8PTWm2QgBR/bkwSWc+NpUFgNPN9PvQi8WEg5UmAGMCAwEAAaOBpjCBozAdBgNVHQ4EFgQUNmHhAHyIBQlRi0
RsR/8aTMnqTxIwHwYDVR0jBBgwFoAUNmHhAHyIBQlRi0RsR/8aTMnqTxIwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8B
Af8EBAMCAYYwQAYDVR0fBDkwNzA1oDOgMYYvaHR0cHM6Ly9hbmRyb2lkLmdvb2dsZWFwaXMuY29tL2F0dGVzdGF0aW9
uL2NybC8wDQYJKoZIhvcNAQELBQADggIBACDIw41L3KlXG0aMiS//cqrG+EShHUGo8HNsw30W1kJtjn6UBwRM6jnmiwfBP
b8VA91chb2vssAtX2zbTvqBJ9+LBPGCdw/E53Rbf86qhxKaiAHOjpvAy5Y3m00mqC0w/Zwvju1twb4vhLaJ5NkUJYsUS7rm
JKHHBnETLi8GFqiEsqTWpG/6ibYCv7rYDBJDcR9W62BW9jfIoBQcxUCUJouMPH25lLNcDc1ssqvC2v7iUgI9LeoM1sNovqPm
QUiG9rHli1vXxzCyaMTjwftkJLkf6724DFhuKug2jITV0QkXvaJWF4nUaHOTNA4uJU9WDvZLI1j83A+/xnAJUucIv/zGJ1AMH
2boHqF8CY16LpsYgBt6tKxxWH00XcyDCdW2KlBCeqbQPcsFmWyWugxdcekhYsAWyoSf818NUsZdBWBaR/OukXrNLfkQ79I
yZohZbvabO/X+MVT3rriAoKc8oE2Uws6DF+60PV7/WIPjNvXySdqspImSN78mflxDqwLqRBYkA3I75qppLGG9rp7UCdRjxMl
8ZDBld+7yvHVgt1cVzJx9xnyGCC23UaicMDSXYrB4I4WHXPGjxhZuCuPBLTdOLU8YRvMYdEvYebWHMpvwGCF6bAx3JBpIe
OQ1wDB5y0USicV3YgYGmi+NZfhA4URSh77Yd6uuJOJENRaNVTzk

-----END CERTIFICATE-----

Here are decoded X.509 certificates:

Certificate 0 (Leaf)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: ecdsa-with-SHA256
        Issuer:
            serialNumber              = c6047571d8f0d17c
        Validity
            Not Before: Jan  1 00:00:00 1970 GMT
            Not After : Jan 19 03:14:07 2038 GMT
        Subject:
            commonName                = Android Keystore Key
```

```
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:21:01:97:84:c5:06:91:99:f7:f0:cc:33:ee:fd:
                    4a:4e:fd:e8:78:2f:b2:b1:6b:f4:bc:12:64:57:60:
                    fa:2c:80:e5:a0:aa:01:16:a4:c8:98:65:2e:64:48:
                    a0:91:43:8a:ce:4d:f4:0f:89:93:7a:b0:27:7e:66:
                    67:d9:69:aa:c2
                ASN1 OID: prime256v1
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature
            1.3.6.1.4.1.11129.2.1.17:
```

Please see Table 1 below for the summary of the decoded extensions of the ASN.1 schema.

```
    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:20:0e:15:a8:83:3c:f2:9a:d9:a7:54:2f:d1:eb:de:
         f6:db:c9:61:28:07:46:d9:ce:f1:af:b7:d2:50:9e:da:de:84:
         02:21:00:9a:18:dc:a8:00:79:80:87:ac:23:f0:79:74:a5:46:
         47:26:45:2c:55:73:69:64:4a:e3:79:65:db:6e:53:9d:68
```

Table-1 shows a summary of the decoded values of the attestation extensions of the ASN.1 schema in the first certificate with the descriptions from the definitions [9][10][11].

Table-1 The decoded ASN.1 attestation extensions of the sample device, Pixel 2, used for this paper.

Field Name [5]	AL [6]	Value	Description	Note
attestationVersion		2	KM3	The version of the key attestation feature.
attestationSecurityLevel		1	SecurityLevel is TrustEnvironment.	TrustEnvironment means the code that creates or manages the relevant element (attestation or key) is

[5] The numbers in [ ] correspond to those that are defined in [10].

[6] AuthorizationList: “SW”: softwareEnforced AuthorizationList, “TEE”: teeEnforced AuthorizationList.

softwareEnforced: (Optional) The Keymaster authorization list that is enforced by the Android system, not by the device's TEE.

teeEnforced: (Optional) The Keymaster authorization list that is enforced by the device's TEE.

				implemented in a Trusted Execution Environment (TEE). If the value is '0', it is Software. It means the code that creates or manages the relevant element (attestation or key) is implemented in the Android system and could be altered if that system is compromised.
keymasterVersion		3	The version of the Keymaster hardware abstraction layer (HAL).	Version 3.0? Android developer page [9] may not be updated yet (?)
keymasterSecurityLevel		1	SecurityLevel is TrustEnvironment.	Hardware-backed Keystore. See above.
attestationChallenge		99 C6 61 9F C1 35 2A E3 54 EF 94 50 4E AB 68 62 09 D2 15 FB 06 9C 27 B3 99 07 F9 2A BB 93 1A BD		
purpose [1]	TEE	2	SIGN	Specifies the set of purposes for which the key may be used.
algorithm [2]	TEE	3	EC	Specifies the cryptographic algorithm with which the key is used.
keySize [3]	TEE	256	256bit	
digest [5]	TEE	4	SHA_2_256	Specifies the digest algorithms that may be used with the key to perform signing and verification operations.
ecCurve [10]	TEE	1	P_256	The EC curve that is used.
activeDateTime [400]	sw	0		Specifies the date and time at which the key becomes active. Prior to this time, any attempt to use the key fails. The value is a 64-bit integer representing milliseconds since January 1, 1970.
originationExpireDateTime [401]	sw	2147483647000		A 64-bit integer representing milliseconds since January 1, 1970.

usageExpireDateTime [402]	sw	2147483647000		A 64-bit integer representing milliseconds since January 1, 1970.
userAuthType [504]	TEE	2	Fingerprint	
creationDateTime [701]	sw	1523867479769		A 64-bit integer representing milliseconds since January 1, 1970.
Origin [702]	TEE	0	GENERATED	Specifies where the key was created, if known.
rollbackResistant [703]	TEE	NULL		
rootOfTrust [704]	TEE	verifiedBootKey: 9c 12 cf dc 04 c7 45 84 d7 87 ac 3d 23 77 21 32 c1 85 24 bc 7a b2 8d ec 42 19 b8 fc 5b 42 5f 70 deviceLocked: false (*1) verifiedBootState: 2, Unverified (*2) *1: deviceLocked: True if the device's bootloader is locked, which enables Verified Boot checking and prevents an unsigned device image from being flashed onto the device [9]. *2: “Unverified” indicates that the user can modify the device freely. Therefore, the user is responsible for verifying the device's integrity [9].		
osVersion [705]	TEE	80100	OS Version: 8.01.00	
osPatchLevel [706]	TEE	201804	OS Patch Level: 201804	
attestationApplicationId [709]	sw	package_name: com.example.android.attestation, version: 92200, signature_digests: 183CA68E27A2222FDE2D463DE7A16DA3FCF1DA808DFC42DE1326C8F63F6194D3 N.B.: The package_name is the name of the Android application that created the keys for this paper (underlined text below in ASCII) 30 51 31 2B 30 29 04 22 63 6F 6D 2E 65 78 61 6D 70 6C 65 2E 61 6E 64 72 6F 69 64 2E 6B 65 79 61		

		74 74 65 73 74 61 74 69 6F 6E 02 03 01 68 28 31 22 04 20 18 3C A6 8E 27 A2 22 2F DE 2D 46 3D E7 A1 6D A3 FC F1 DA 80 8D FC 42 DE 13 26 C8 F6 3F 61 94 D3

Certificate 1

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            11:78:73:15:02:79:86:02:00:10
    Signature Algorithm: ecdsa-with-SHA256
        Issuer:
            serialNumber               = 87f4514475ba0a2b
        Validity
            Not Before: May 26 17:15:02 2016 GMT
            Not After : May 24 17:15:02 2026 GMT
        Subject:
            serialNumber               = c6047571d8f0d17c
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:ea:09:92:b8:d9:71:b3:ee:e8:87:03:c7:2a:ef:
                    96:e8:00:49:54:26:e4:c5:fb:38:61:93:d1:3d:f8:
                    af:39:70:c1:b4:7d:51:5d:80:4e:ee:9a:32:21:1a:
                    f2:eb:90:74:fd:e2:cb:9a:92:51:87:82:cd:4b:a7:
                    78:87:68:ed:db
                ASN1 OID: prime256v1
        X509v3 extensions:
            X509v3 Subject Key Identifier:
```

```
            79:5F:C0:FE:FE:F7:5A:F5:C4:B3:EA:FE:8E:FF:79:85:C0:53:DA:1C
        X509v3 Authority Key Identifier:
            keyid:30:44:23:E5:A2:F6:06:E1:50:AB:77:5F:16:16:BB:91:CC:63:C6:59


        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Key Usage: critical
            Digital Signature ....... Note-1
        X509v3 Name Constraints:
            Permitted:
              DNS:invalid;email:invalid


        X509v3 CRL Distribution Points:

            Full Name:
              URI:https://android.googleapis.com/attestation/crl/11787315027986020010


    Signature Algorithm: ecdsa-with-SHA256
         30:64:02:30:0c:3a:f5:fb:a6:87:69:59:fd:a0:26:3c:ee:b7:
         75:a8:cf:f6:ea:d4:1e:36:a4:2e:c1:bb:9d:75:a6:fd:73:8f:
         88:db:b6:c7:61:85:80:a8:fe:ef:e9:9f:29:46:dc:b3:02:30:
         28:1d:91:d6:50:04:f3:84:68:9b:81:4d:89:b1:1a:25:ab:a9:
         51:5e:ea:f4:ec:5f:ad:12:27:8b:ea:6b:53:eb:b9:26:73:10:
         57:32:ba:45:cc:72:43:e7:d7:07:33:f0
```

Note-1: This is the issue that is discussed in the Errata UAF 1.1 [12]:

*The Server "MUST verify the syntax of the key attestation extension and it MUST perform RFC5280 compliant chain validation of the entries in the array to one attestationRootCertificate specified in the Metadata Statement - **accepting that that the keyCertSign bit in the key usage extension of the certificate issuing the leaf certificate is NOT set (which is a deviation from RFC5280).**"*

Certificate 2

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            03:88:26:67:60:65:89:96:85:75
    Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            serialNumber               = f92009e853b6b045
        Validity
            Not Before: May 26 17:01:51 2016 GMT
            Not After : May 24 17:01:51 2026 GMT
        Subject:
            serialNumber               = 87f4514475ba0a2b
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:64:3b:56:68:1d:27:fa:c4:82:cf:6c:20:4d:eb:
                    ce:f0:29:5c:35:0c:01:aa:8c:32:7f:3e:d5:f4:49:
                    ae:f4:29:66:42:5f:39:3d:84:76:3b:5d:ad:56:b1:
                    0d:04:59:c6:2c:6a:4f:93:25:56:a8:92:93:c3:d1:
                    4e:a2:f1:df:ef:0e:3c:ba:22:72:a3:22:39:e1:b2:
                    b4:7d:ac:55:0b:ab:bc:5c:a7:55:e2:aa:48:62:8a:
                    d6:3c:76:ff:67:dc:72
                ASN1 OID: secp384r1
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                30:44:23:E5:A2:F6:06:E1:50:AB:77:5F:16:16:BB:91:CC:63:C6:59
            X509v3 Authority Key Identifier:
                keyid:36:61:E1:00:7C:88:05:09:51:8B:44:6C:47:FF:1A:4C:C9:EA:4F:12
```

```
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:https://android.googleapis.com/attestation/crl/E8FA196314D2FA18
```

Certificate 3 (Root)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 16787758474318772760 (0xe8fa196314d2fa18)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            serialNumber               = f92009e853b6b045
        Validity
            Not Before: May 26 16:28:52 2016 GMT
            Not After : May 24 16:28:52 2026 GMT
        Subject:
            serialNumber               = f92009e853b6b045
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:af:b6:c7:82:2b:b1:a7:01:ec:2b:b4:2e:8b:cc:
                    54:16:63:ab:ef:98:2f:32:c7:7f:75:31:03:0c:97:
                    52:4b:1b:5f:e8:09:fb:c7:2a:a9:45:1f:74:3c:bd:
```

```
9a:6f:13:35:74:4a:a5:5e:77:f6:b6:ac:35:35:ee:
17:c2:5e:63:95:17:dd:9c:92:e6:37:4a:53:cb:fe:
25:8f:8f:fb:b6:fd:12:93:78:a2:2a:4c:a9:9c:45:
2d:47:a5:9f:32:01:f4:41:97:ca:1c:cd:7e:76:2f:
b2:f5:31:51:b6:fe:b2:ff:fd:2b:6f:e4:fe:5b:c6:
bd:9e:c3:4b:fe:08:23:9d:aa:fc:eb:8e:b5:a8:ed:
2b:3a:cd:9c:5e:3a:77:90:e1:b5:14:42:79:31:59:
85:98:11:ad:9e:b2:a9:6b:bd:d7:a5:7c:93:a9:1c:
41:fc:cd:27:d6:7f:d6:f6:71:aa:0b:81:52:61:ad:
38:4f:a3:79:44:86:46:04:dd:b3:d8:c4:f9:20:a1:
9b:16:56:c2:f1:4a:d6:d0:3c:56:ec:06:08:99:04:
1c:1e:d1:a5:fe:6d:34:40:b5:56:ba:d1:d0:a1:52:
58:9c:53:e5:5d:37:07:62:f0:12:2e:ef:91:86:1b:
1b:0e:6c:4c:80:92:74:99:c0:e9:be:c0:b8:3e:3b:
c1:f9:3c:72:c0:49:60:4b:bd:2f:13:45:e6:2c:3f:
8e:26:db:ec:06:c9:47:66:f3:c1:28:23:9d:4f:43:
12:fa:d8:12:38:87:e0:6b:ec:f5:67:58:3b:f8:35:
5a:81:fe:ea:ba:f9:9a:83:c8:df:3e:2a:32:2a:fc:
67:2b:f1:20:b1:35:15:8b:68:21:ce:af:30:9b:6e:
ee:77:f9:88:33:b0:18:da:a1:0e:45:1f:06:a3:74:
d5:07:81:f3:59:08:29:66:bb:77:8b:93:08:94:26:
98:e7:4e:0b:cd:24:62:8a:01:c2:cc:03:e5:1f:0b:
3e:5b:4a:c1:e4:df:9e:af:9f:f6:a4:92:a7:7c:14:
83:88:28:85:01:5b:42:2c:e6:7b:80:b8:8c:9b:48:
e1:3b:60:7a:b5:45:c7:23:ff:8c:44:f8:f2:d3:68:
b9:f6:52:0d:31:14:5e:bf:9e:86:2a:d7:1d:f6:a3:
bf:d2:45:09:59:d6:53:74:0d:97:a1:2f:36:8b:13:
ef:66:d5:d0:a5:4a:6e:2f:5d:9a:6f:ef:44:68:32:
bc:67:84:47:25:86:1f:09:3d:d0:e6:f3:40:5d:a8:
96:43:ef:0f:4d:69:b6:42:00:51:fd:b9:30:49:67:
```

```
                3e:36:95:05:80:d3:cd:f4:fb:d0:8b:c5:84:83:95:
                26:00:63
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                36:61:E1:00:7C:88:05:09:51:8B:44:6C:47:FF:1A:4C:C9:EA:4F:12
            X509v3 Authority Key Identifier:
                keyid:36:61:E1:00:7C:88:05:09:51:8B:44:6C:47:FF:1A:4C:C9:EA:4F:12

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:https://android.googleapis.com/attestation/crl/

    Signature Algorithm: sha256WithRSAEncryption
         20:c8:c3:8d:4b:dc:a9:57:1b:46:8c:89:2f:ff:72:aa:c6:f8:
         44:a1:1d:41:a8:f0:73:6c:c3:7d:16:d6:42:6d:8e:7e:94:07:
         04:4c:ea:39:e6:8b:07:c1:3d:bf:15:03:dd:5c:85:bd:af:b2:
         c0:2d:5f:6c:db:4e:fa:81:27:df:8b:04:f1:82:77:0f:c4:e7:
         74:5b:7f:ce:aa:87:12:9a:88:01:ce:8e:9b:c0:cb:96:37:9b:
         4d:26:a8:2d:30:fd:9c:2f:8e:ed:6d:c1:be:2f:84:b6:89:e4:
         d9:14:25:8b:14:4b:ba:e6:24:a1:c7:06:71:13:2e:2f:06:16:
         a8:84:b2:a4:d6:a4:6f:fa:89:b6:02:bf:ba:d8:0c:12:43:71:
         1f:56:eb:60:56:f6:37:c8:a0:14:1c:c5:40:94:26:8b:8c:3c:
         7d:b9:94:b3:5c:0d:cd:6c:b2:ab:c2:da:fe:e2:52:02:3d:2d:
         ea:0c:d6:c3:68:be:a3:e6:41:48:86:f6:b1:e5:8b:5b:d7:c7:
```

```
30:b2:68:c4:e3:c1:fb:64:24:b9:1f:eb:bd:b8:0c:58:6e:2a:
e8:36:8c:84:d5:d1:09:17:bd:a2:56:17:89:d4:68:73:93:34:
0e:2e:25:4f:56:0e:f6:4b:23:58:fc:dc:0f:bf:c6:70:09:52:
e7:08:bf:fc:c6:27:50:0c:1f:66:e8:1e:a1:7c:09:8d:7a:2e:
9b:18:80:1b:7a:b4:ac:71:58:7d:34:5d:cc:83:09:d5:b6:2a:
50:42:7a:a6:d0:3d:cb:05:99:6c:96:ba:0c:5d:71:e9:21:62:
c0:16:ca:84:9f:f3:5f:0d:52:c6:5d:05:60:5a:47:f3:ae:91:
7a:cd:2d:f9:10:ef:d2:32:66:88:59:6e:f6:9b:3b:f5:fe:31:
54:f7:ae:b8:80:a0:a7:3c:a0:4d:94:c2:ce:83:17:ee:b4:3d:
5e:ff:58:83:e3:36:f5:f2:49:da:ac:a4:89:92:37:bf:26:7e:
5c:43:ab:02:ea:44:16:24:03:72:3b:e6:aa:69:2c:61:bd:ae:
9e:d4:09:d4:63:c4:c9:7c:64:30:65:77:ee:f2:bc:75:60:b7:
57:15:cc:9c:7d:c6:7c:86:08:2d:b7:51:a8:9c:30:34:97:62:
b0:78:23:85:87:5c:f1:a3:c6:16:6e:0a:e3:c1:2d:37:4e:2d:
4f:18:46:f3:18:74:4b:d8:79:b5:87:32:9b:f0:18:21:7a:6c:
0c:77:24:1a:48:78:e4:35:c0:30:79:cb:45:12:89:c5:77:62:
06:06:9a:2f:8d:65:f8:40:e1:44:52:87:be:d8:77:ab:ae:24:
e2:44:35:16:8d:55:3c:e4
```

10.2 Appendix-B Android CDD Relative to Key Attestation and Fingerprint Sensors

The followings are the requirements of Android 8.1 (latest one at the time of this drafting) related to Keystore, key attestation and fingerprint sensors [2].

Note: All these mandatory requirements are conditional on “If device implementations include a secure lock screen, …”. Since almost all modern commercial mobile devices (smartphones and tablets) include a secure lock screen, these conditional requirements can be considered as mandatory requirements for such mobile devices.

- If device implementations include **a secure lock screen**, they
  - SHOULD include **a fingerprint sensor** (7.3.10., CDD 8.1[2]).
  - **MUST** (9.11., CDD 8.1[2])
    - [C-1-1] back up the **keystore implementation with secure hardware**.
    - [C-1-3] perform the lock screen authentication in the isolated execution environment and only when successful, allow the authentication-bound keys to be used.

- [C-1-4] **support key attestation where the attestation signing key is protected by secure hardware and signing is performed in secure hardware.** The attestation signing keys MUST be shared across large enough number of devices to prevent the keys from being used as device identifiers. One way of meeting this requirement is to share the same attestation key unless at least 100,000 units of a given SKU are produced. If more than 100,000 units of an SKU are produced, a different key MAY be used for each 100,000 units.

- If device implementations include a fingerprint sensor and make the sensor available to third-party Apps, they:
  - **[C-1-6] MUST have a hardware-backed keystore implementation, and perform the fingerprint matching in a Trusted Execution Environment (TEE) or on a chip with a secure channel to the TEE.**
  - **[C-1-7] MUST have all identifiable fingerprint data encrypted and cryptographically authenticated such that they cannot be acquired, read or altered outside of the Trusted Execution Environment (TEE) as documented in the implementation guidelines on the Android Open Source Project site.**
  - [C-1-3] MUST have a false acceptance rate not higher than 0.002%.
  - [SR] Are STRONGLY RECOMMENDED to have a spoof and imposter acceptance rate not higher than 7%. (New from 8.1)
  - [C-1-5] MUST rate limit attempts for at least 30 seconds after five false trials for fingerprint verification.
  - [C-1-8] MUST prevent adding a fingerprint without first establishing a chain of trust by having the user confirm existing or add a new device credential (PIN/pattern/password) that's secured by TEE; the Android Open Source Project implementation provides the mechanism in the framework to do so.
  - [C-1-9] MUST NOT enable 3rd-party applications to distinguish between individual fingerprints.