January 2024

Privacy Preserving Device-Bound Information

Max Bires

Seth Moore

Privacy Preserving Device-Bound Information

ABSTRACT

A common abuse scenario is one in which a malicious user repeatedly performs actions that are intended by a developer, vendor, or service provider to be limited. For example, a user may repeatedly take undue advantage of one-time free trials, coupons, or account creation flows. This disclosure describes techniques to verifiably attach data to a particular device in a privacy-preserving manner such that the data can survive a factory reset or compromise of the device. Data is bound to the device in a privacy-preserving manner such that the data cannot be used as a mechanism for identifying a particular device. Developers can bind an abuse bit to a device such that even when the device is used with a fresh user account, it can be determined that abuse has been attempted from the device previously. Counters in device-bound data enable setting thresholds to detect abuse of services or products.

KEYWORDS

- Device abuse
- Factory reset
- Remote key provisioning (RKP)
- Device-bound information
- Digital rights management (DRM)
- Attestation certificate
- Key attestation
- Authentication token
- Device identifier
- Device authentication

BACKGROUND

A common abuse scenario is one in which a malicious user repeatedly performs actions that are intended by a developer, vendor, or service provider to be limited. For example, a user may repeatedly take undue advantage of one-time free trials, coupons, or account creation flows. Repeated availing of one-time offers is made possible by changing or resetting device state, which prevents the developer, vendor, or service provider from determining whether requests originate from the same device or different devices.

Developers, service providers, and vendors (hereinafter collectively referred to as developers) benefit from the ability to identify abusive devices. Identifying such devices enables linking abuse not only to user accounts but also to specific physical devices. Currently, attackers who have their accounts banned (or limited) can use factory resets in conjunction with fresh accounts to restart abuse. Developers also benefit from the ability to count high-value operations.

Existing techniques such as device-check APIs bind relatively few bits, e.g., just two bits, to the device, and require token regeneration each time the API is used. Also, such device-check techniques bind a counter to the attested keys, not to the device, such that factory resets and/or app reinstalls enable users to bypass protections offered by the counter. Although a second layer of counters can enable determination of how many times an app has been installed, the onus is on the developer to figure out how to combine counter values. Existing device-check APIs are complex and require multiple backend API calls to fetch the per-device data.

DESCRIPTION

This disclosure describes techniques to verifiably attach data to a particular device in a privacy-preserving manner such that the data can survive a factory reset or compromise of the device. Data is bound to the device in a privacy-preserving manner such that the data cannot be

used as a mechanism for identifying a particular device. Developers can bind an abuse bit to a device such that even when the device is used with a fresh user account, it can be determined that abuse has been attempted from the device previously. Counters in device-bound data enable setting thresholds to detect abuse of services or products. The techniques build on remote key provisioning, e.g., as described in [1].

A small amount of data, e.g., as little as three bits along with a last-modified timestamp, is bound to the device. A vendor spends (expends) one bit each time they offer up one of these scarce coupons or other resources and resets that bit when sufficient time has passed. The mechanism of writing and reading privacy-preserving device-bound data, described in greater detail below, includes per-device setup, developer authentication, key-authentication certificates that uniquely identify devices, opaque authentication tokens, and enabling developers to access per-device information in a privacy-preserving manner.

Per-device setup

To give devices permanent identities and to enable looking up the identities based on privacy preserving information, the following configuration procedures are performed.

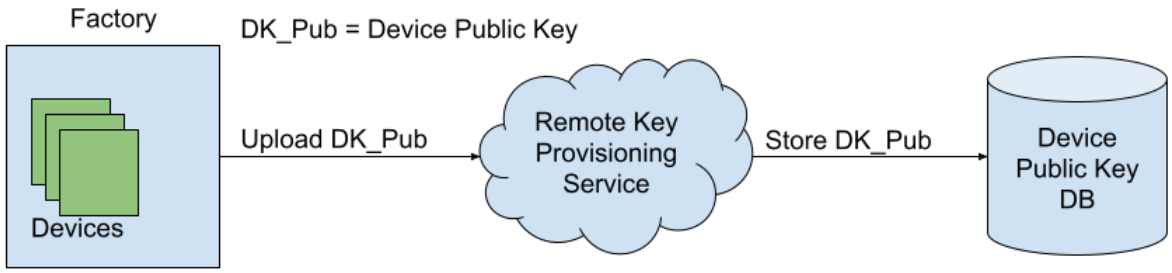

Fig. 1: Per-device setup (typically a factory operation)

As illustrated in Fig. 1, a permanent device public key (DK_Pub) is retrieved from a device and uploaded to a remote key provisioning (RKP) service during device manufacture. Original equipment manufacturers (OEMs) are authenticated with unique credentials, enabling the RKP service to trust the uploaded key and associate that key with a particular manufacturer.

The DK_Pub is used to authenticate the device and provision information on the device, including information such as digital rights management (DRM) and short-lived attestation certificates. When attestation certificates are provisioned, a database mapping the attestation certificates to the device public keys (DK_Pubs) is populated. The mapping serves as a mechanism to uniquely identify devices from public information (the certificates), and, as such, is securely guarded.

Developer identity and authentication

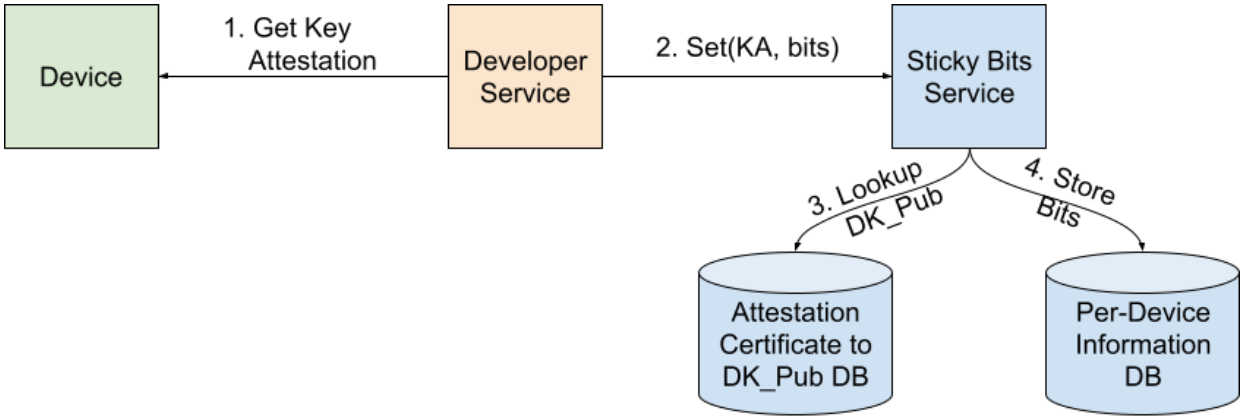

Fig. 2: Data binding

As illustrated in Fig. 2, an identity provider (not shown) authenticates a developer and provides developer identity to a sticky bits service. The developer uses the sticky bits service to access unique, per-device data. No two users of the service can see the same information. The sticky bits service uses the developer identity to partition the information.

The sticky bits service can support arbitrary partitioning of the developers, including multiple levels of identity layering. Therefore, it is possible for the service to support multiple developer ecosystems.

Device authenticators

In addition to developer identity, the sticky bits service also uses device authenticators to uniquely identify devices. Various forms of interchangeable authenticators can be supported by the sticky bits service, e.g., key attestation certificates, opaque authentication tokens, etc. These are explained in greater detail below.

Key Attestation Certificate Exchange for A Token

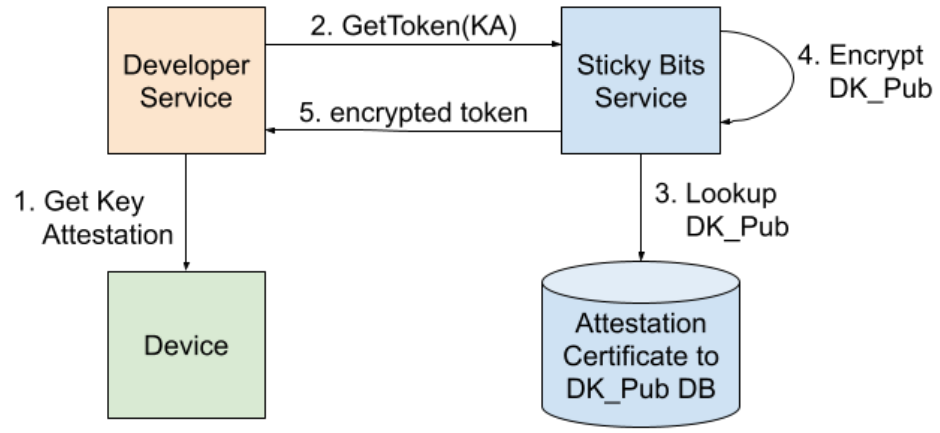

Fig. 3: Authentication token creation

Key attestation certificates are traditionally used to measure the integrity of a device. As illustrated in Fig. 3, in combination with sticky bits, key attestation certificates can additionally be used to uniquely identify devices using the following mechanism.

An on-device application generates a key attestation using standard APIs. The application communicates with the developer backend service, transmitting the key attestation certificate

chain to the developer backend service. The developer backend service authenticates with an identity provider that is also recognized by the sticky bits service. The developer backend service communicates with the sticky bits service, providing both proof of developer identity and the key attestation certificate.

The sticky bits service verifies the developer identity and, using the database mapping described in per-device setup, performs a reverse lookup of the DK_pub to which the key attestation certificate was provisioned. Once the DK_pub is identified, the device itself is uniquely identified and per-device information can be accessed in a manner described in greater detail below.

Opaque authentication tokens

The sticky bits service supports the creation of opaque authentication tokens, as illustrated in Fig. 3. The developer can use the encrypted token for future calls to the sticky bits service in lieu of a key attestation certificate. When the sticky bits service receives an authentication token, it first decrypts the token, revealing the DK_pub. At this point, the sticky bits service can proceed to access per-device information (described in greater detail below) without performing any additional database lookups. Compared to key attestation certificates, opaque authentication tokens can have lower latency, as there are fewer database lookups required on the hot path when per-device data is accessed and can also have more predictable (consistent) expiration times.

Accessing per-device information

Once a developer has a device authenticator, they can interact with the sticky bits service to access per-device information. The developer authenticates themselves to the sticky bits front

end. As mentioned above (in *developer identity*), this authentication relies on an identity provider. The developer passes the device authenticator to the backend along with an indicator of the operation (read, write, increment-counter, etc.) requested. The sticky bits service validates the authenticator and performs the appropriate operation on a per-developer, per-device basis.

Advantages

Unlike traditional counterabuse mechanisms, a trust binding is created between the manufacturer, the RKP service, and the identity provider service. Abuse can no longer scale infinitely from a limited number of compromised devices. An attacker is defeated by their malicious devices becoming permanently marked upon detected attempts to engage in malicious behaviors such as account creation, spamming, free-offer abuse, etc. There is no known mechanism for an attacker acting from the device to break the security of the described mechanism.

In contrast to current device-check APIs, the described techniques enable an authenticator token to live for a longer period of time without requiring token regeneration, reducing latency for applications. The use of sticky bits reduces the burden on developers to count the number of times an app has been installed, resulting in a simpler, less error-prone solution that binds all counters directly to the device itself, preventing even factory resets from changing values. The number of server calls is streamlined, and, in contrast to existing device-check APIs, per-device data can be fetched in as little as a single backend API call.

The described techniques can be implemented as part of a device operating system and provided as an application programming interface (API) to developers. The techniques can support multiple developer ecosystems, e.g., sources from where applications can be installed on

a device. The techniques are applicable to any device that can run developer-provided applications, such as smartphones, tablets, smartwatches, etc.

CONCLUSION

This disclosure describes techniques to verifiably attach data to a particular device in a privacy-preserving manner such that the data can survive a factory reset or compromise of the device. Data is bound to the device in a privacy-preserving manner such that the data cannot be used as a mechanism for identifying a particular device. Developers can bind an abuse bit to a device such that even when the device is used with a fresh user account, it can be determined that abuse has been attempted from the device previously. Counters in device-bound data enable setting thresholds to detect abuse of services or products.

REFERENCES

1. "Upgrading Android Attestation: Remote Provisioning" available online at https://android-developers.googleblog.com/2022/03/upgrading-android-attestation-remote.html accessed Jan. 19, 2024.
2. "DeviceCheck," available online at https://developer.apple.com/documentation/devicecheck accessed Jan. 3, 2024.
3. Zia, Hannia, Heman Khanna, Animesh Chatterji, Stavan Parikh, Ridhima Kedia, and Bogdan Brinzarea Iamandi. "Authenticator App for Consent Architecture." U.S. Patent Application 17/633,509, filed September 22, 2022. Assigned to Google LLC.
4. Gunn, Lachlan J., N. Asokan, Jan-Erik Ekberg, Hans Liljestrand, Vijayanand Nayani, and Thomas Nyman. "Hardware platform security for mobile devices." *Foundations and Trends® in Privacy and Security* 3, no. 3-4 (2022): 214-394.