



# Android Security: Taming the Complex Ecosystem

**Stanford CS155 guest lecture**

2019-05-23

René Mayrhofer, Director of Android Platform Security

Personal Twitter: @rene\_mobile



# Outline

1. The Ecosystem and State of the Union (*The Marketing Part*)
2. Android Platform Security Model and Implementation (*The Systematic Part*)
3. Taming Complexity (*The “What I learned” Part*)
4. Where do we go from here (*The Future Part*)?

# State of the (Android) Union



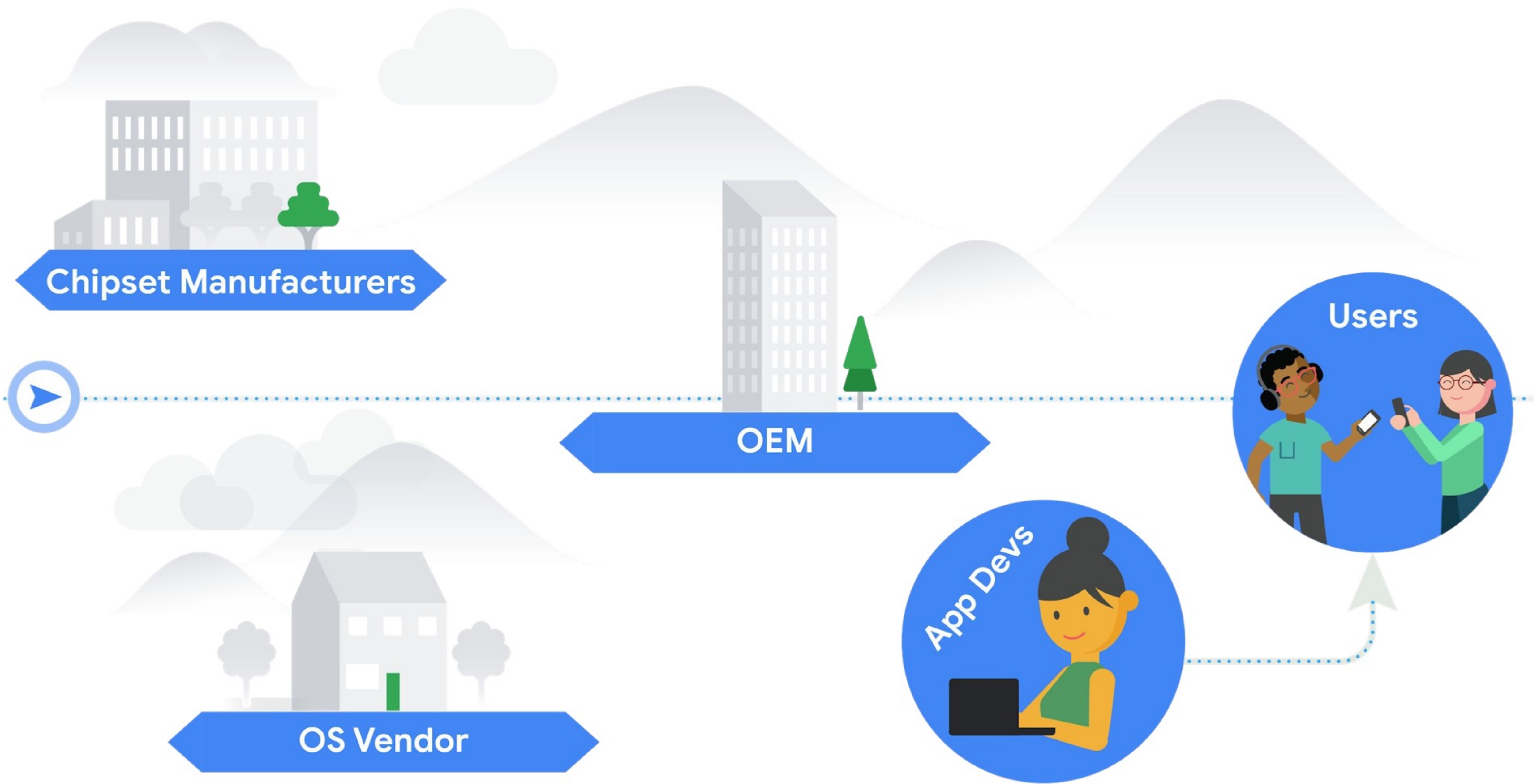
# The Android ecosystem in numbers

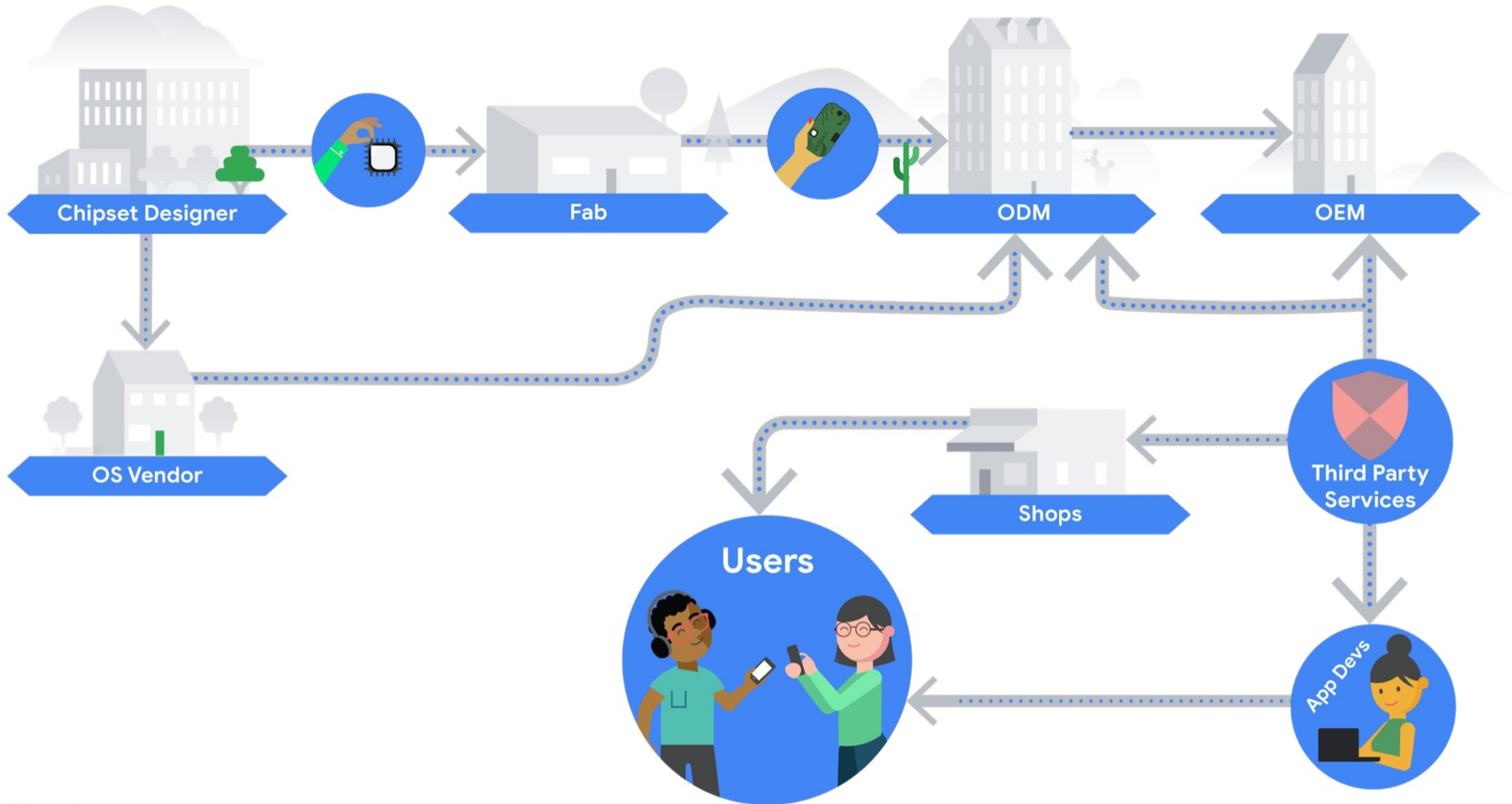
**> 1.300  
brands**

**> 24.000  
devices**

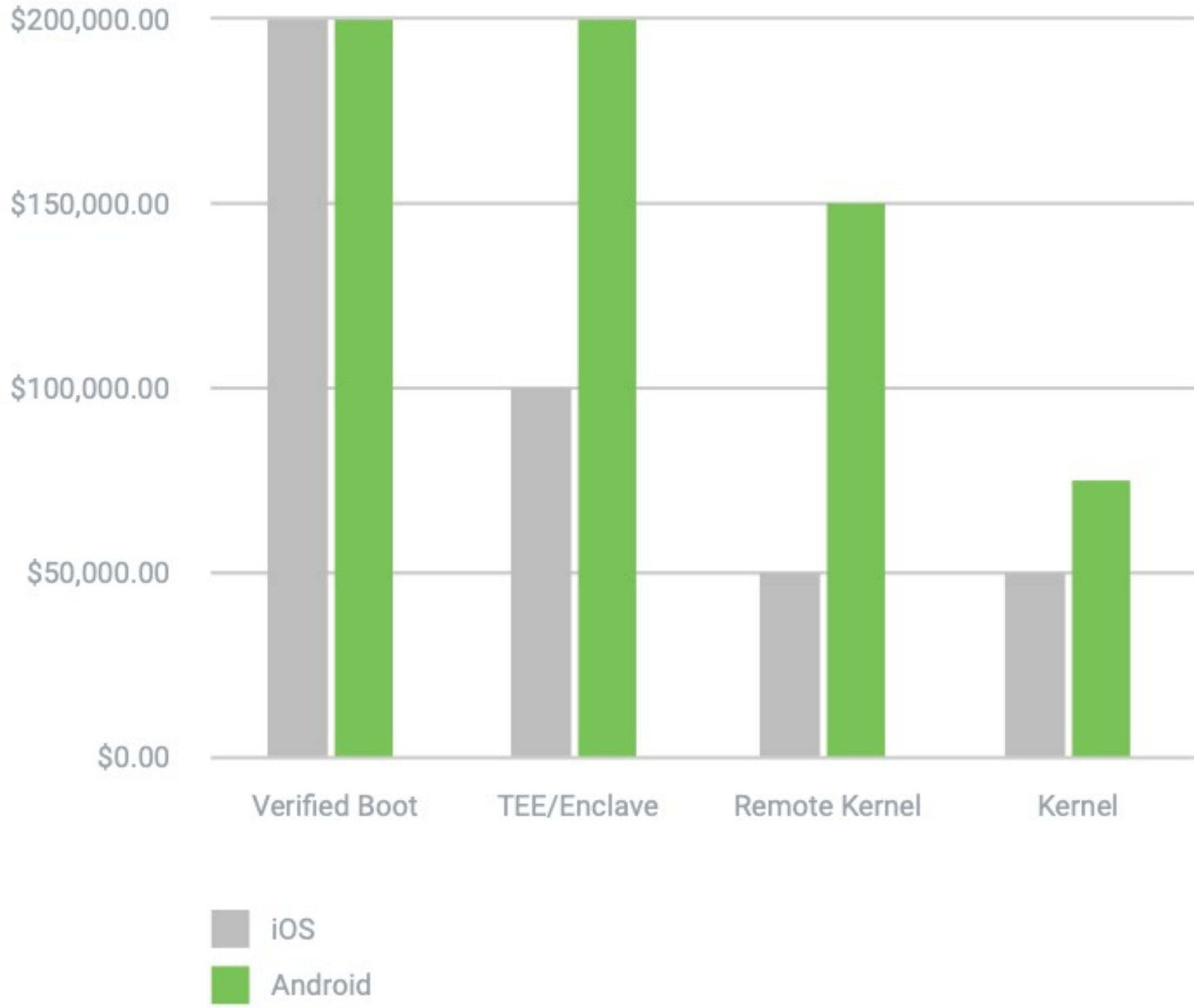
**> 1 M  
apps**

**> 2 B  
users**





# Measuring exploitation difficulty: 0-day pricing

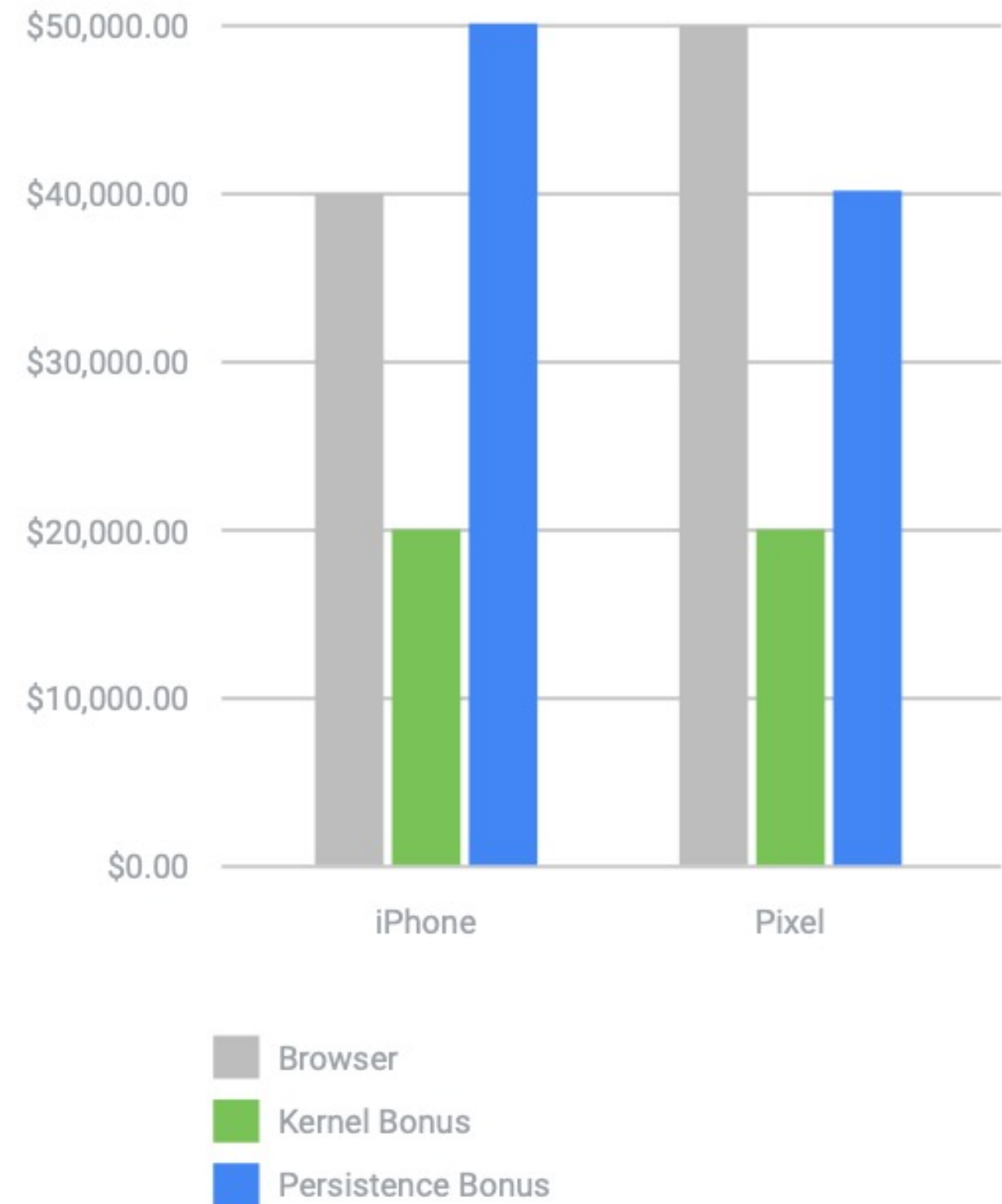


# Measuring exploitation difficulty: 0-day pricing

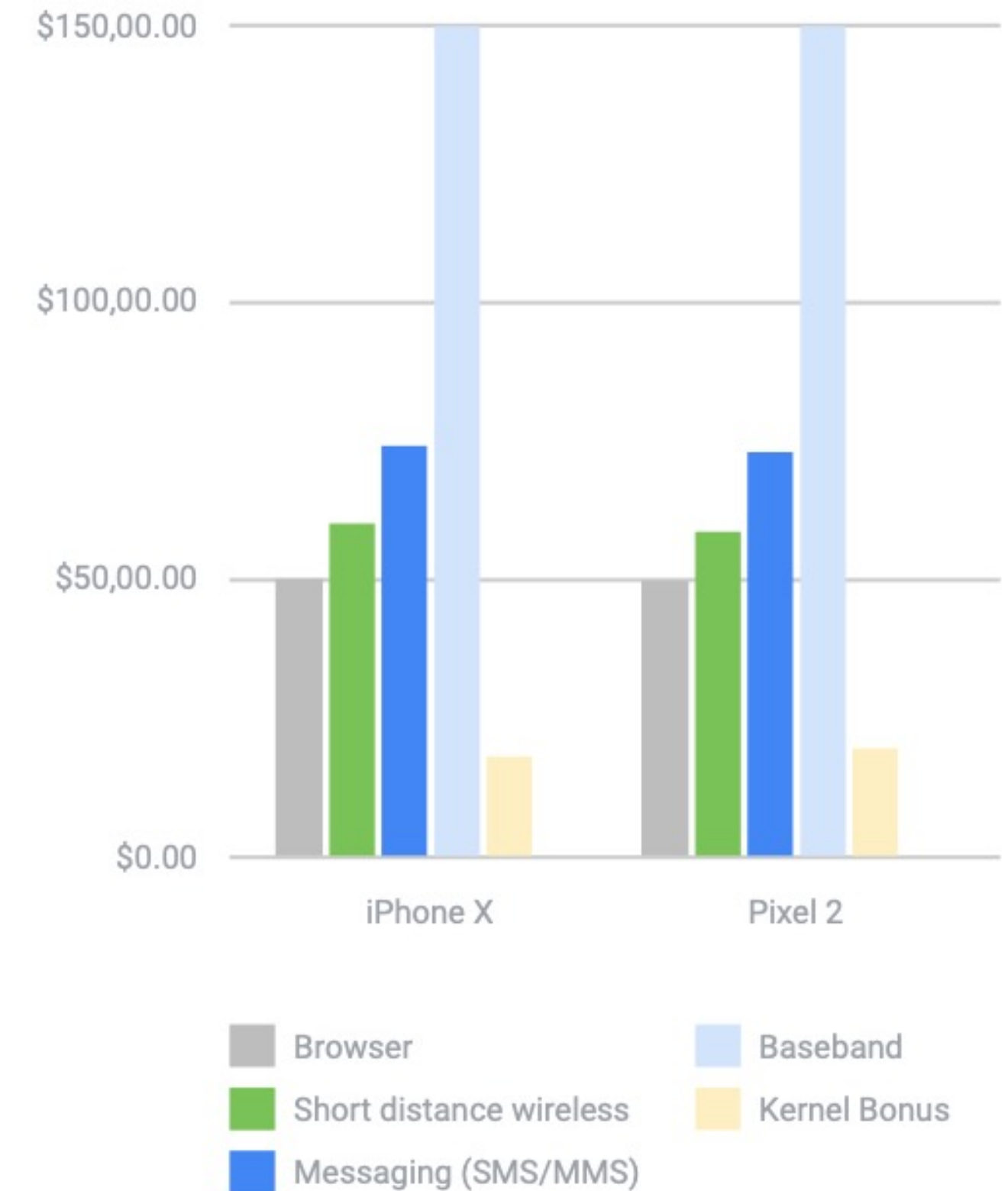
Mobile Pwn2Own 2016



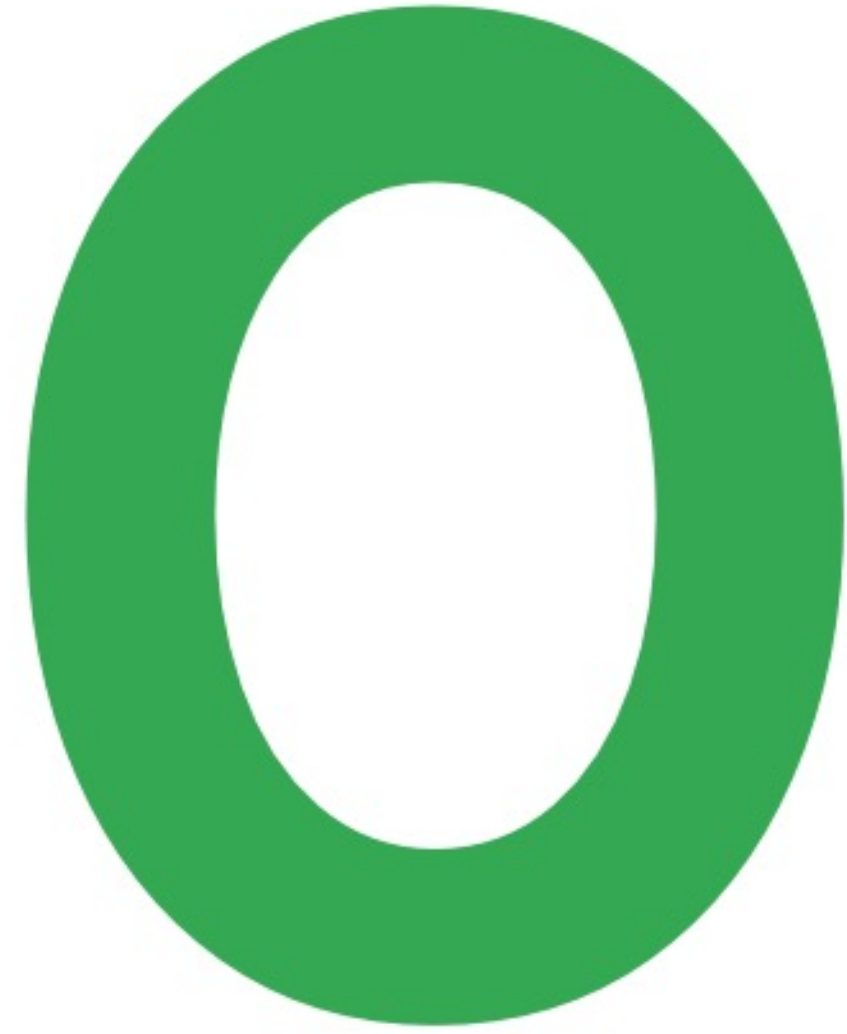
Mobile Pwn2Own 2017



Mobile Pwn2Own 2018

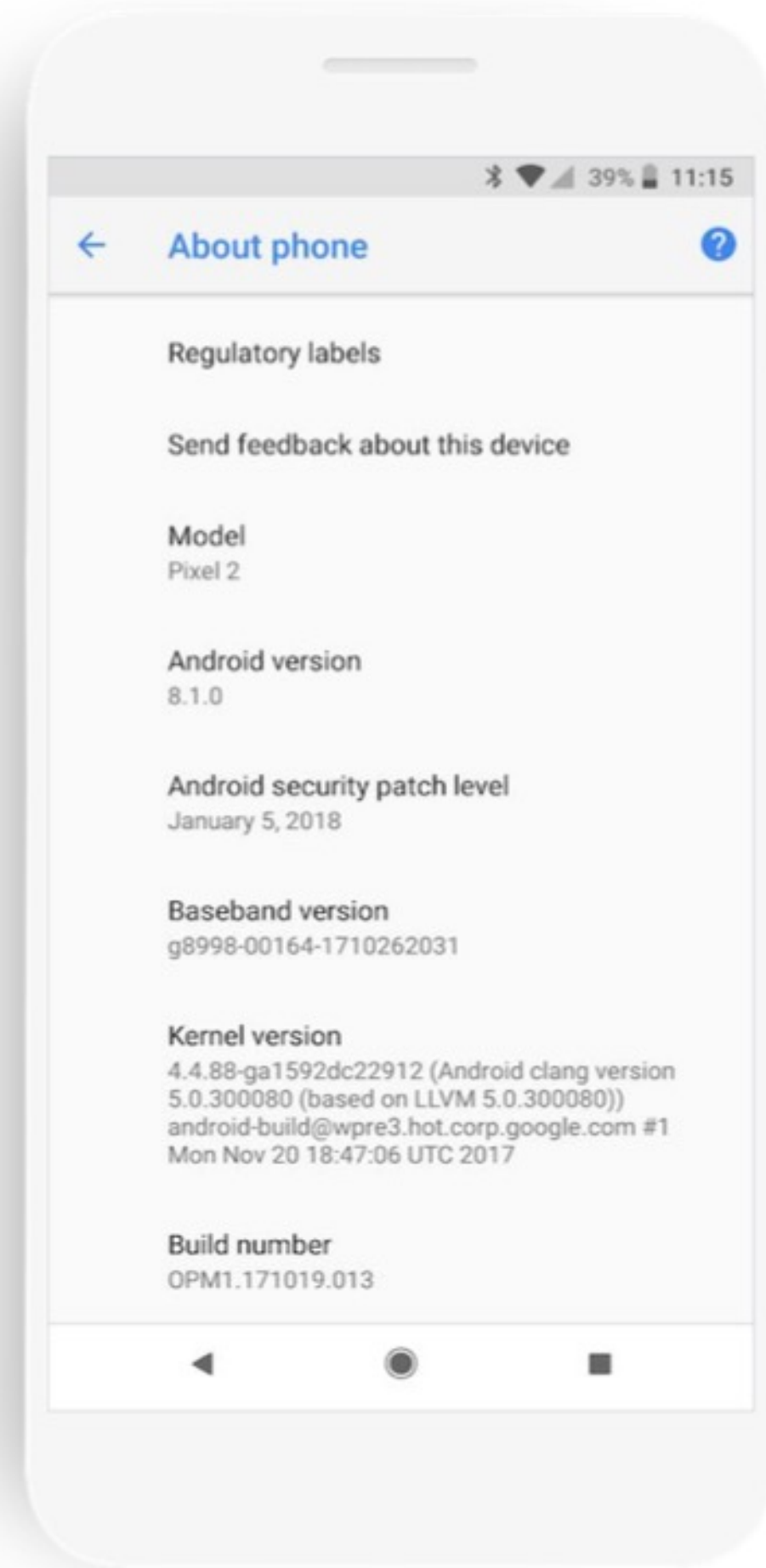






**critical security vulnerabilities** affecting the  
Android platform in 2018 publicly disclosed without  
a security update or mitigation available

# Android patching has improved



**1B**

Devices patched  
in 2018.

**29%**

more devices  
patched QoQ  
in Q4.

**84%**

more devices  
patched in Q4  
than same time  
last year.

# Malware is a universal risk

## Security researchers discover iOS version of Exodus Android spyware

Exodus iOS spyware used against Italian and Turkmenistan users.



By Catalin Cimpanu for Zero Day | April 8, 2019 -- 21:25 GMT (14:25 PDT) | Topic: Security

“This year, we celebrated the 30th anniversary of the World Wide Web. Fast forward thirty years and the threat landscape is exponentially more complex, and the available attack surface is growing faster than it has at any other point in the history of technology,” commented Ondrej Vlcek, President of Consumer at Avast.



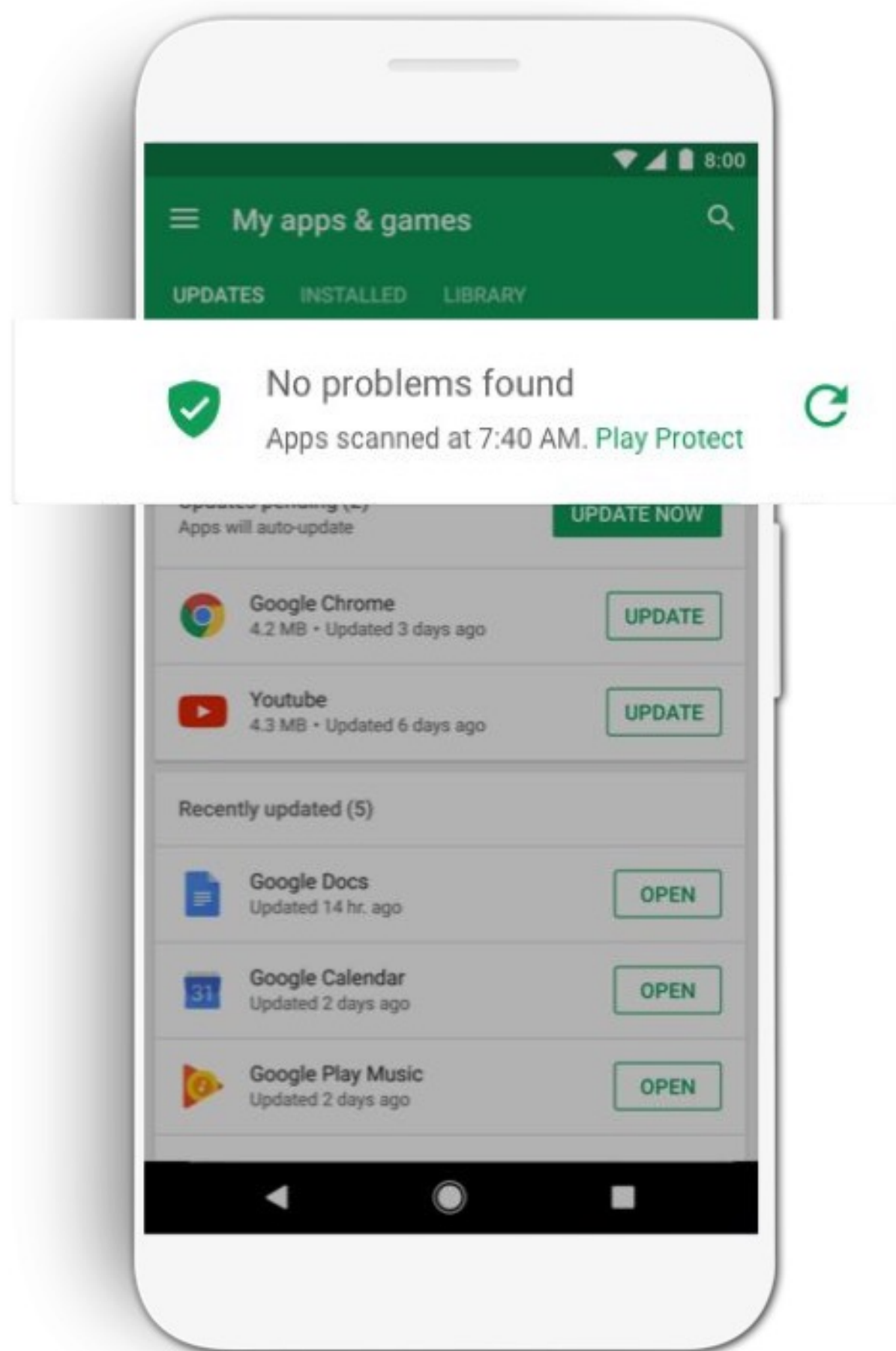
## New research reveals a dozen iPhone apps linked to Golduck malware

08/01/2019 Promon's Security Team

Android Security, App Security, Application Security, Application Shielding, Malware, Mobile App Security

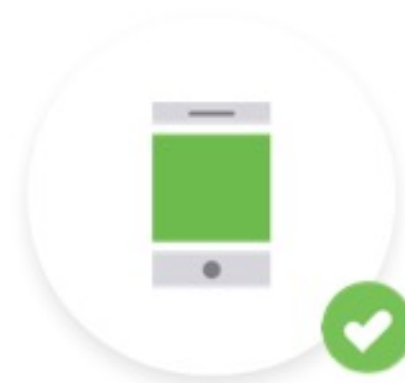
New research from [Wandera](#) find over a dozen iPhone apps linked to Golduck malware. The findings underline that fake apps is by no means an Android-only problem.

# World's most widely used Anti-Malware solution



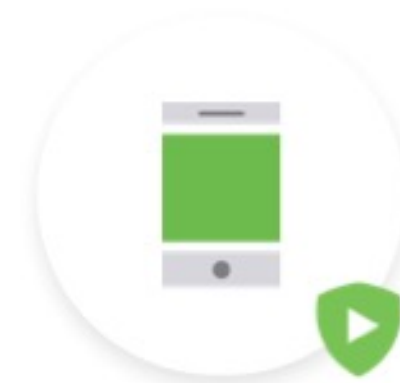
Security protection for everyone (Play and off-Play).

Always updating to provide the latest protections from **Google AI**.



**50B**

Apps verified  
per day



**2+B**

Devices  
protected



**500K**





Apps analyzed  
per day

Scans apps daily - from both within Google Play and outside of it.

Remediates by removing potentially harmful apps (PHA).

# Play App Security Improvement Program

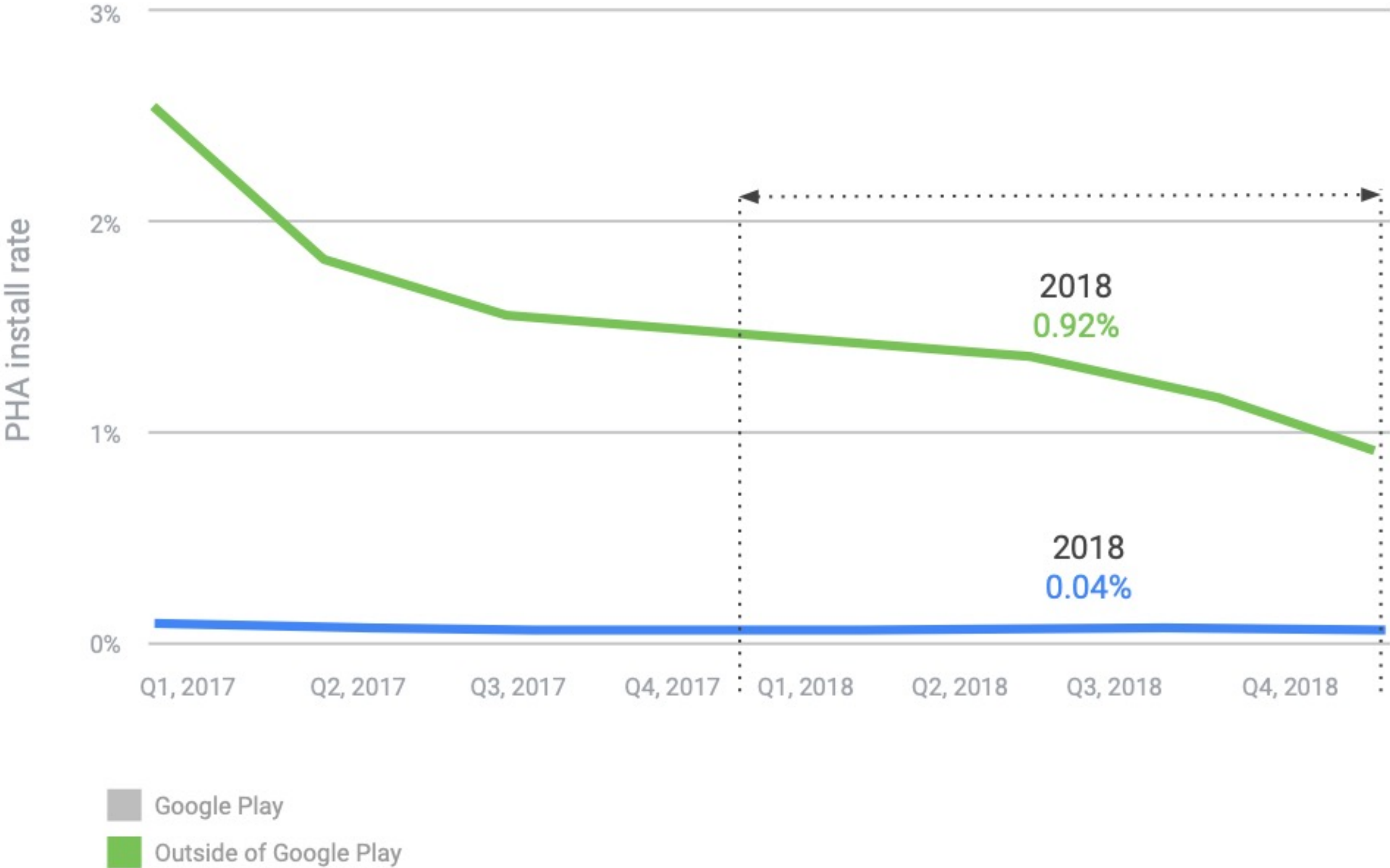
Identifies potential security enhancements when apps are uploaded to Play **300,000** developers have fixed **1,000,000+** Play apps.

|  |          |             |                  |   |
|--|----------|-------------|------------------|---|
|  <b>Test App : Vungle Alert 1.0.0</b> | USD 0.99 | Jun 8, 2015 | <b>Published</b> |  |
|  <b>Test App-Apache Cordova 2.0.0</b> | Free     |             |                  |   |
|  <b>W24m Android Auto 1.0.0</b>       | Free     |             |                  |   |

**Security alert**  
Your app is statically linking against a version of Vungle ad library that has multiple security vulnerabilities. Please see the alerts page for more information.

# Android PHA install rates over time

In 2018, downloading a PHA from Google Play was **0.04%**, and outside of Google Play was **0.92%**.



# The Android Platform Security Model

# Security Goals

## 1. Protecting user data

- a. Usual: device encryption, user authentication, memory/process isolation
- b. Upcoming: personalized ML on device

## 2. Protecting device integrity

- a. Usual: malicious modification of devices
- b. Interesting question: against whom?

## 3. Protecting developer data

- a. Content
- b. IP



# Threat Model

- Adversaries can get physical access to Android devices
  - Powered off
  - Screen locked
  - Screen unlocked by different user
  - Physical proximity
- Network communication and sensor data are untrusted
  - Passive eavesdropping
  - Active MITM
- Untrusted code is executed on the device
- Untrusted content is processed by the device
- **New: Insiders** can get access to signing keys

# Principles



**Actors control  
access to the  
data they create**

**Safe by  
design/default**

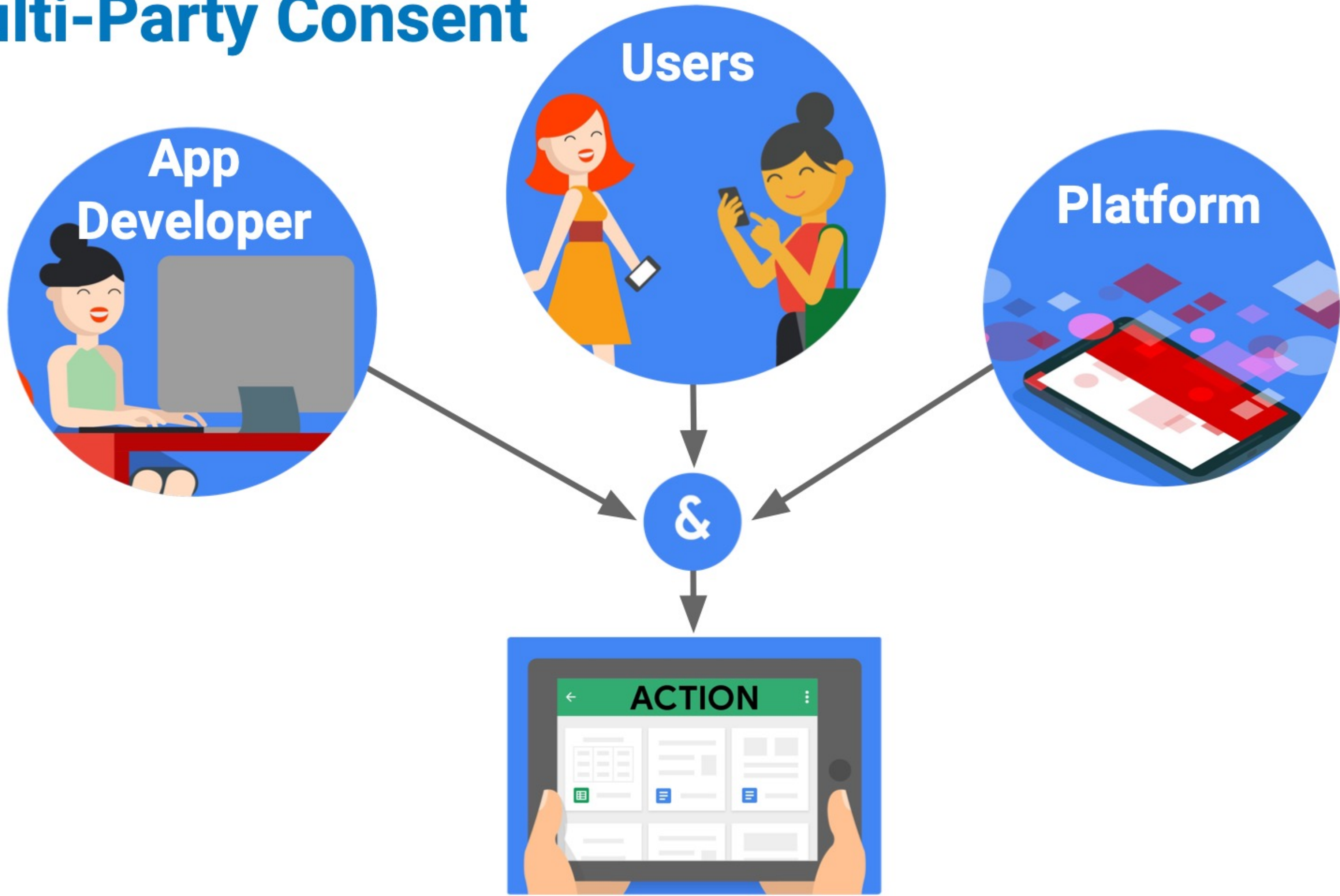
**Consent is  
Informed and  
meaningful**

**Defense in depth**



# The Android Platform Security Model

## (1) Multi-Party Consent



# The Android Platform Security Model

**(2) Open ecosystem access**

**(3) Security is a compatibility requirement**

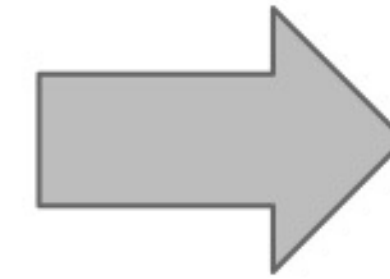
**(4) Factory reset restores the device to a safe state**

**(5) Applications are security principals**

# Implementing the Security Model

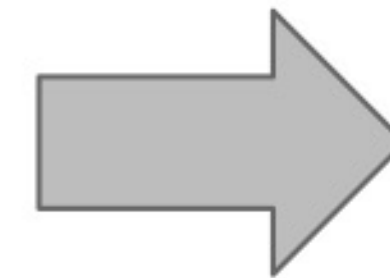
# Strategies

- **Contain:** isolating and de-privileging components, particularly ones that handle untrusted content.
  - **Access control:** adding permission checks, increasing the granularity of permission checks, or switching to safer defaults (for example, default deny).
  - **Attack surface reduction:** reducing the number of entry/exit points (i.e. principle of least privilege).
  - **Architectural decomposition:** breaking privileged processes into less privileged components and applying attack surface reduction.



**reduce  
reachability  
of code**

- **Mitigate:** Assume vulnerabilities exist and actively defend against classes of vulnerabilities or common exploitation techniques.



**reduce  
impact of  
reachable  
bugs**

# It all starts with secure hardware



**TEE (Trusted execution environment)** used for key generation, key import, signing and verification services are executed in hardware.

• **Secure Lock Screen, PIN verification & Data encryption** (PIN+HW key) used to derive encryption keys.

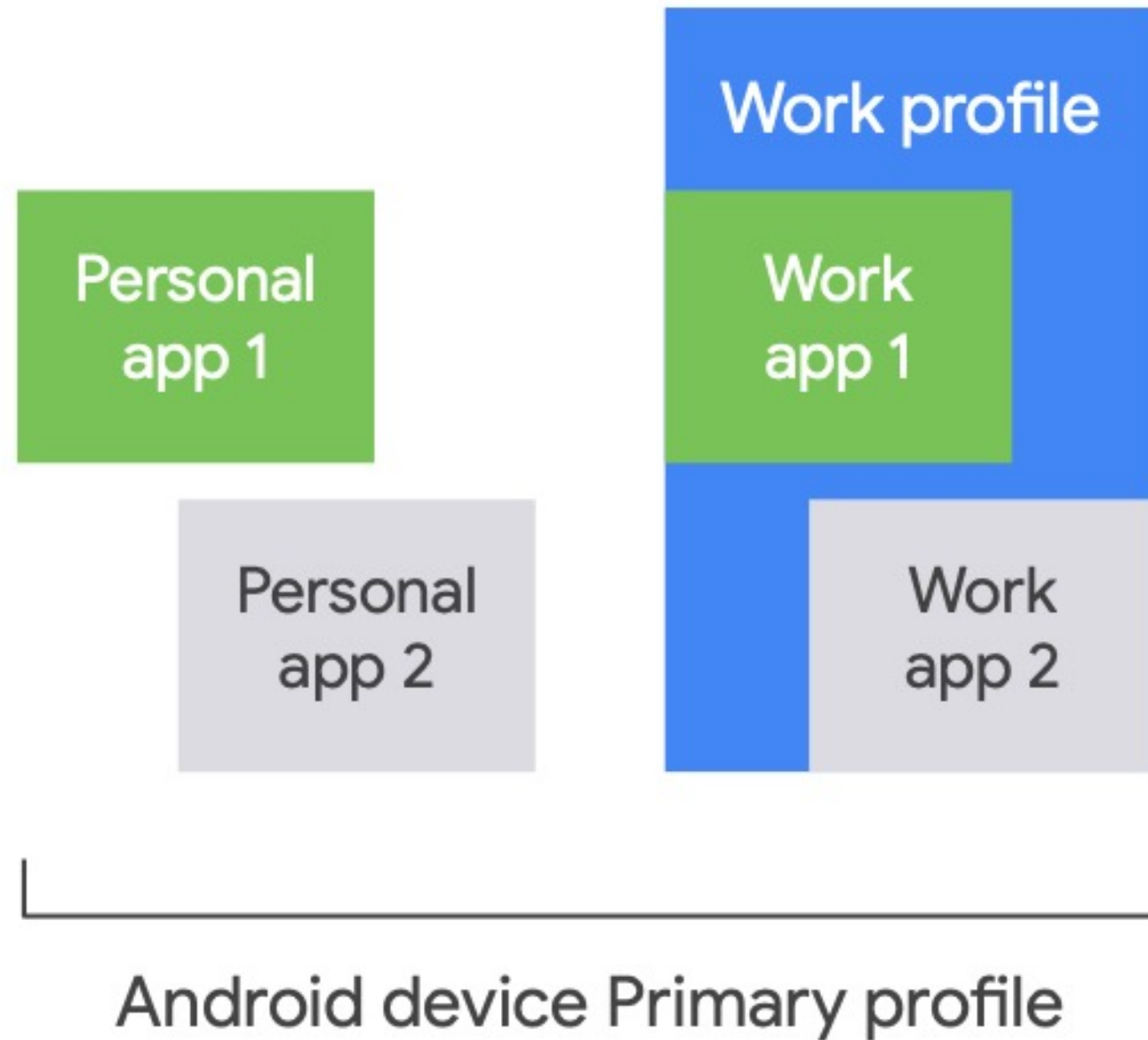
**Version binding** ensures keys created with a newer OS cannot be used by older OS versions.

**Rollback prevention** (8.0+) prevents downgrading OS to an older less secure version or patch level.

**Verified Boot** provides cryptographic verification of OS to ensure devices have not been tampered with.

**Tamper-resistant hardware** (Android Pie) offers support to execute cryptographic functions in dedicated hardware.

# SELinux, process isolation and sandboxing



**Android is built on SELinux** where if an exploit is found, the attack vector is limited to the domain the exploit is able to execute in.

**Application sandboxing** ensures that application and system data is inaccessible from other apps. Each app runs in its own user ID (UID) - limiting exposure of apps to get data from one another.

**Work profile** apps are prevented from communicating with personal apps by default. Work profile apps run in a separate user space with separate encryption keys from personal apps, further limiting exposure, EMMs cannot manage the personal device when the device is managed only via the Work Profile.



# The tiered authentication model

## Primary Auth

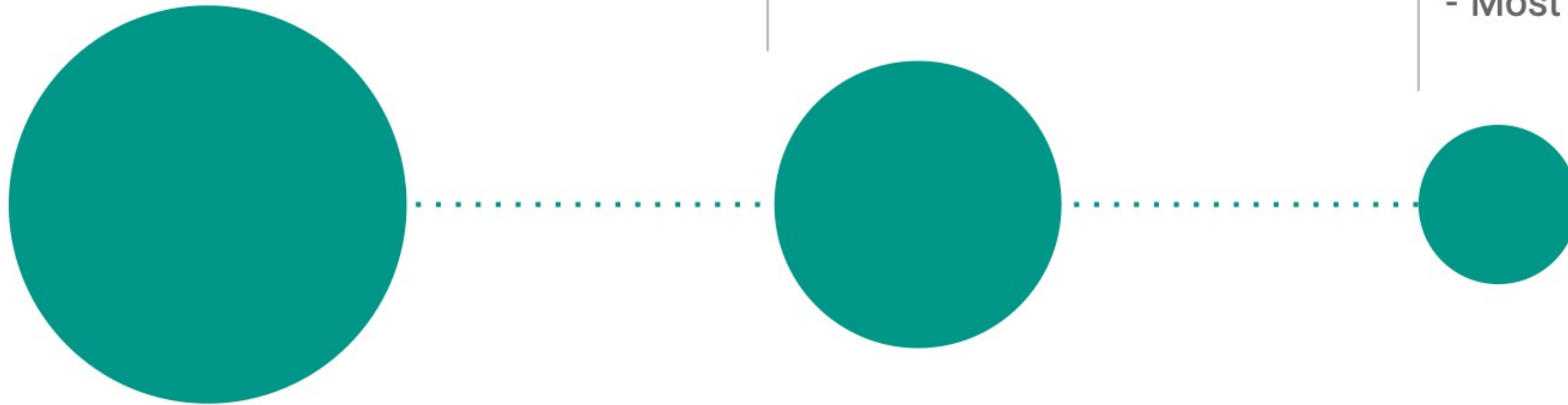
- Knowledge-factor based
- Most secure

## Secondary Auth

- Needs primary auth
- Less secure
- Somewhat constrained

## Tertiary auth

- Needs primary auth
- Least secure
- Most constrained



# Taming Complexity



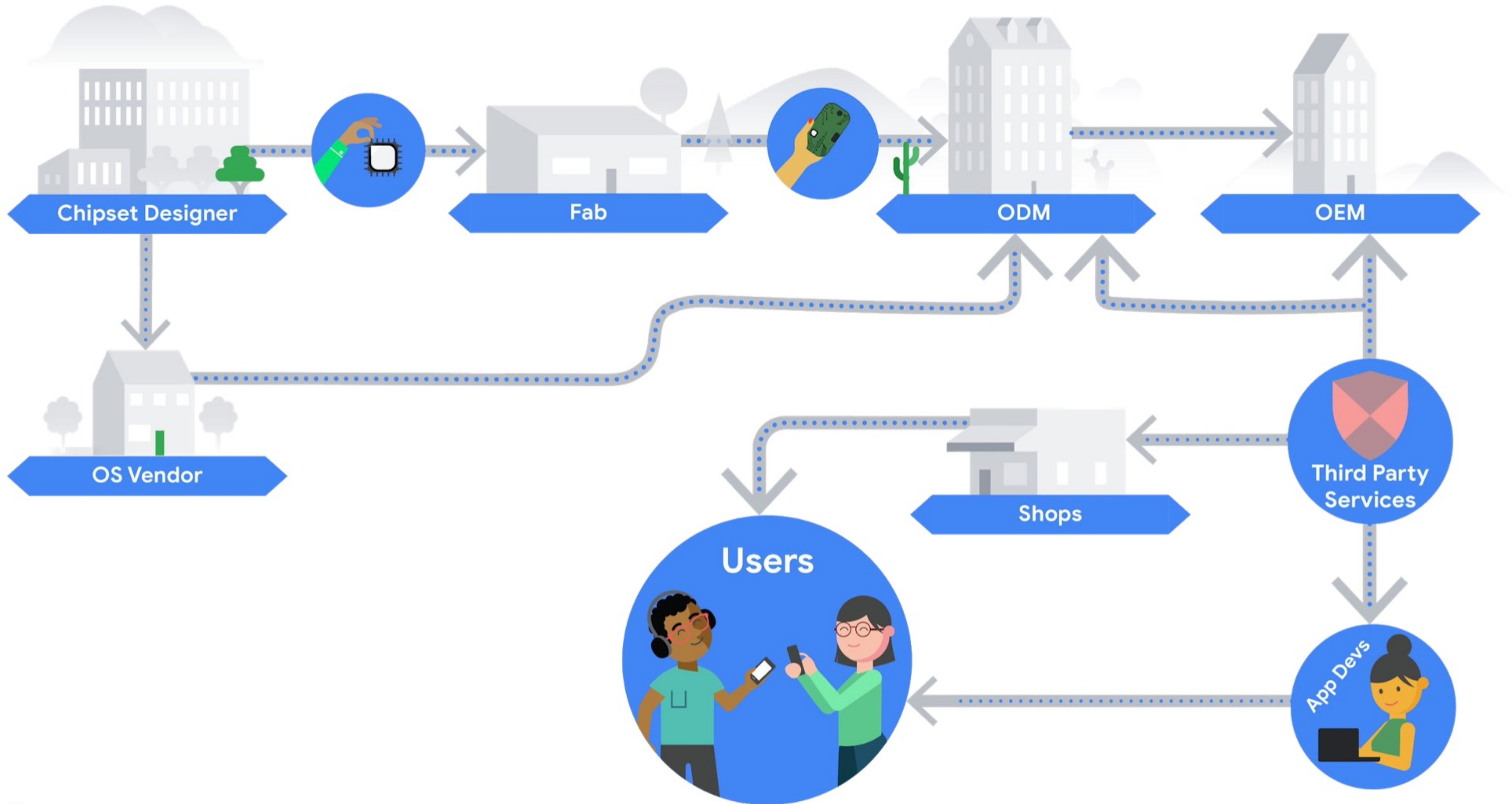
# Many variants and stakeholders: Enabling an active ecosystem

**> 1.300  
brands**

**> 24.000  
devices**

**> 1 M  
apps**

**Can be written in any language**



# Taming complexity in variants

## Compatibility Definition Document (Standards)

- Defines requirements a device needs to fulfill to be considered "Android"
- Updated for every Android release
  - Many changes scoped to apps targeting this version
- Needs to strike a balance between strong standard base and openness for innovation
  - Some requirements scoped to hardware capabilities (e.g. form factors)
- Updating security requirements is one important means of driving ecosystem to improvement

## Compatibility/Vendor/Security/... Test Suite (Enforcement)

- Tests need to be run by device manufacturer
- Guaranteed conformance to (testable parts of) CDD

*In Android Q, ca. **800 tests for SELinux policy***

- Usability of Android trademark and Google apps bound to passing tests
- Complexity in test execution:
  - Automation of test cases
  - Visibility on "user" firmware builds

# Changing the ecosystem is hard - Various strategies

1. Introducing new requirements initially as optional, becoming mandatory only in future releases → time for development, testing, adaptation

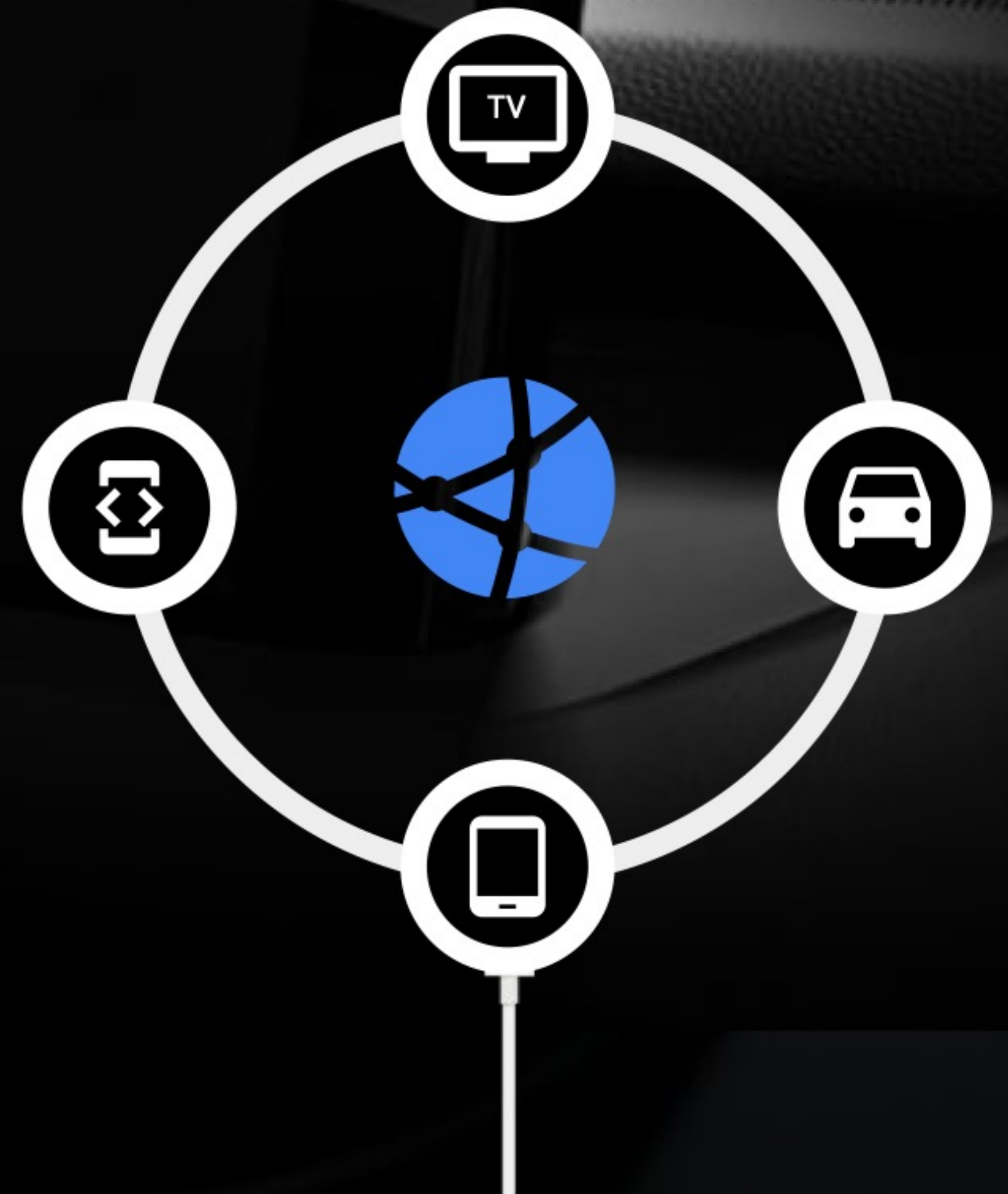
Important lesson: **Clear communication of plans way ahead of schedule**

2. Ratcheting requirements from release to release with a pace that lets hardware keep up (including low-end devices and verticals) or keeping carve-outs

Important lesson: **Let the tail end of the ecosystem keep up**

# 100%

of compatible devices  
launching with Q will  
**encrypt user data**



# Target API version requirements

## Actively maintained apps (forefront) in Play

**August 2019:** New apps are required to target API level 28 (Android 9) or higher.

**November 2019:** Updates to existing apps are required to target API level 28 or higher.

## Apps not getting updates (tail end) on device

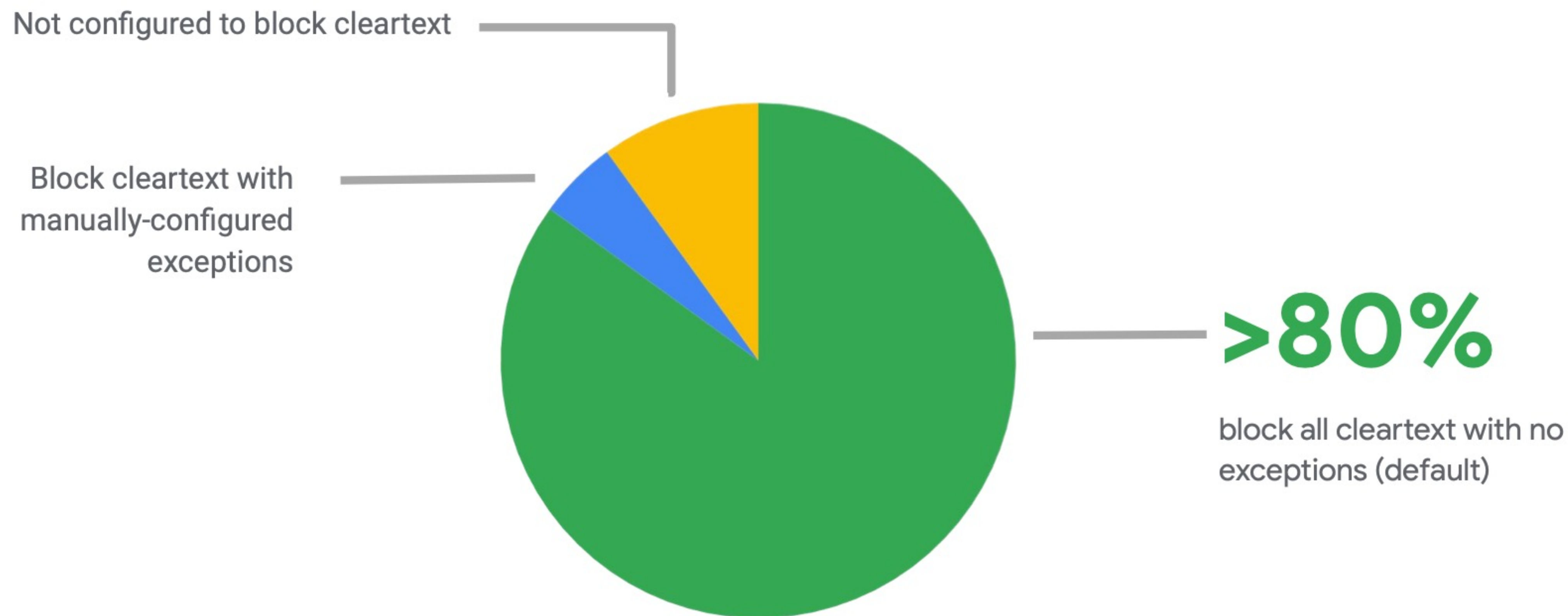
**August 2019:** New apps will receive warnings during installation if they do not target API level 26 or higher.

**November 2019:** New versions of existing apps will receive warnings during installation if they do not target API level 26 or higher.





# Apps targeting Pie, usage of NetworkSecurityConfig



Source: Google Internal Data, 2019-04-01

# Taming complexity in stakeholders

## Tooling

- Compiler/build toolchain ideally used by all stakeholders (e.g. drivers, TrustZone, etc. code)
- Can add new mitigations at this level, but *typically breaks old code*

## Upstream first approach

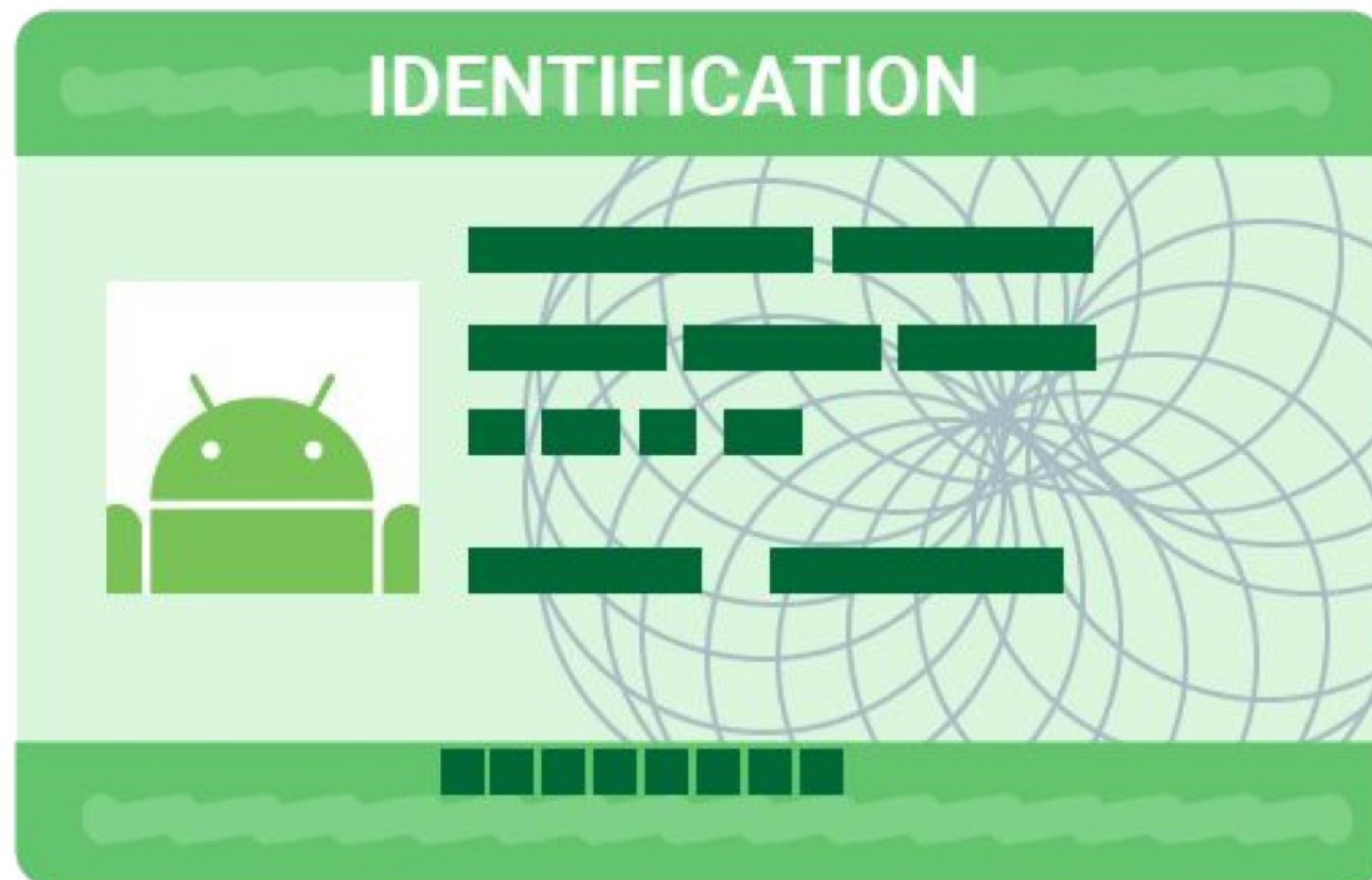
- Importance to commit changes to common upstream code (e.g. Linux kernel, clang, etc.)
- Encouraging other stakeholders to upstream their changes (either to common upstream or to AOSP)

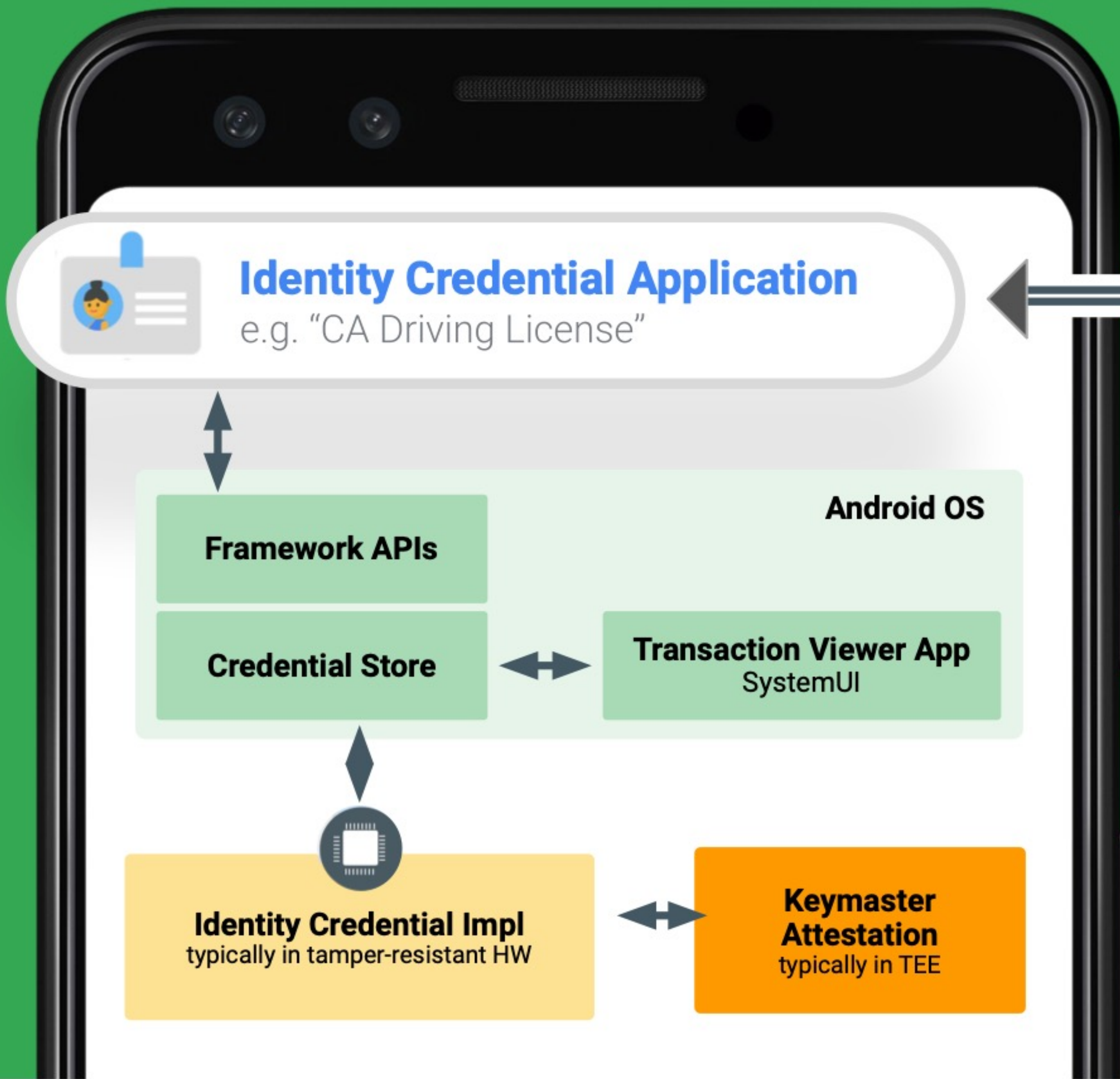
## Open source and common issue trackers

# Where do we go from here?



# Identity Credentials





NFC / Bluetooth /  
Wifi Direct

**Reader / Verifier**  
e.g. "VT State Police"

Internet Protocol

**Issuing Authority**  
e.g. "CA DMV"

# Security and Privacy for draft mDL standard

- **Security properties:**
  - **Anti-forgery:** Identity Credential data is signed by the Issuing Authority
  - **Anti-cloning:** Secure Hardware produces MAC during provisioning using a key derived from a private key specific to the credential and an ephemeral public key from the reader. Public key corresponding to credential private key is signed by the Issuing Authority
  - **Anti-eavesdropping:** Communications between Reader/Verifier and Secure Hardware are encrypted and authenticated
- **Privacy properties:**
  - **Data minimization:** Reader/Verifier only receives data consented to by the holder. Backend infrastructure does not receive information about use
  - **Unlinkability:** Application may provision single-use keys
  - **Auditability:** Every transaction and its data is logged and available only to the Holder (not the application performing the transaction)

# Status

## Android Q

- ✓ No changes to platform itself
- ✓ Software implementation as compatibility library

```
SecurityType = SOFTWARE_ONLY  
CertificationLevel = NONE
```

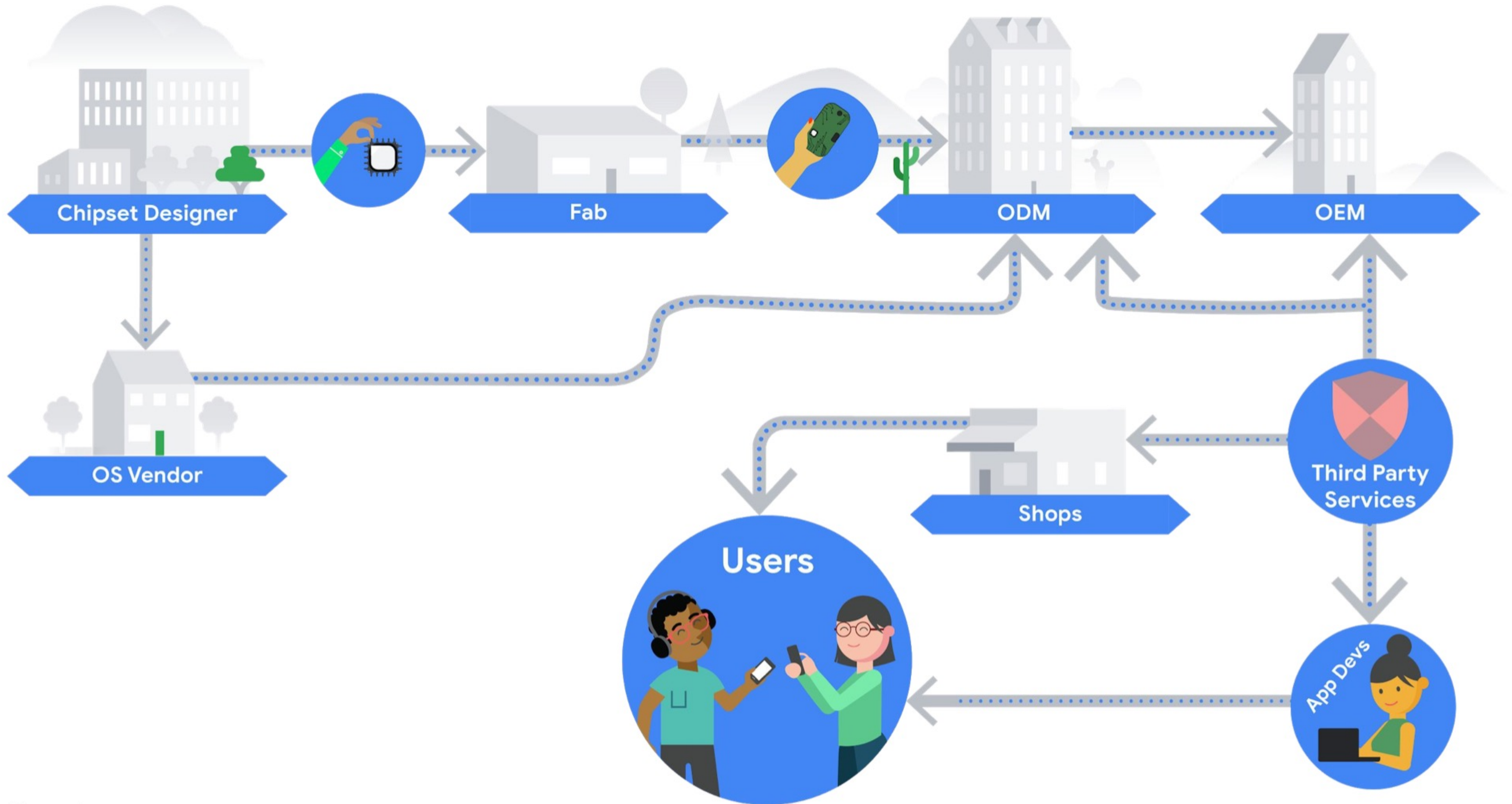
- ✓ Can start developing identity apps, library will be compatible with vast majority of Android devices

## Future versions

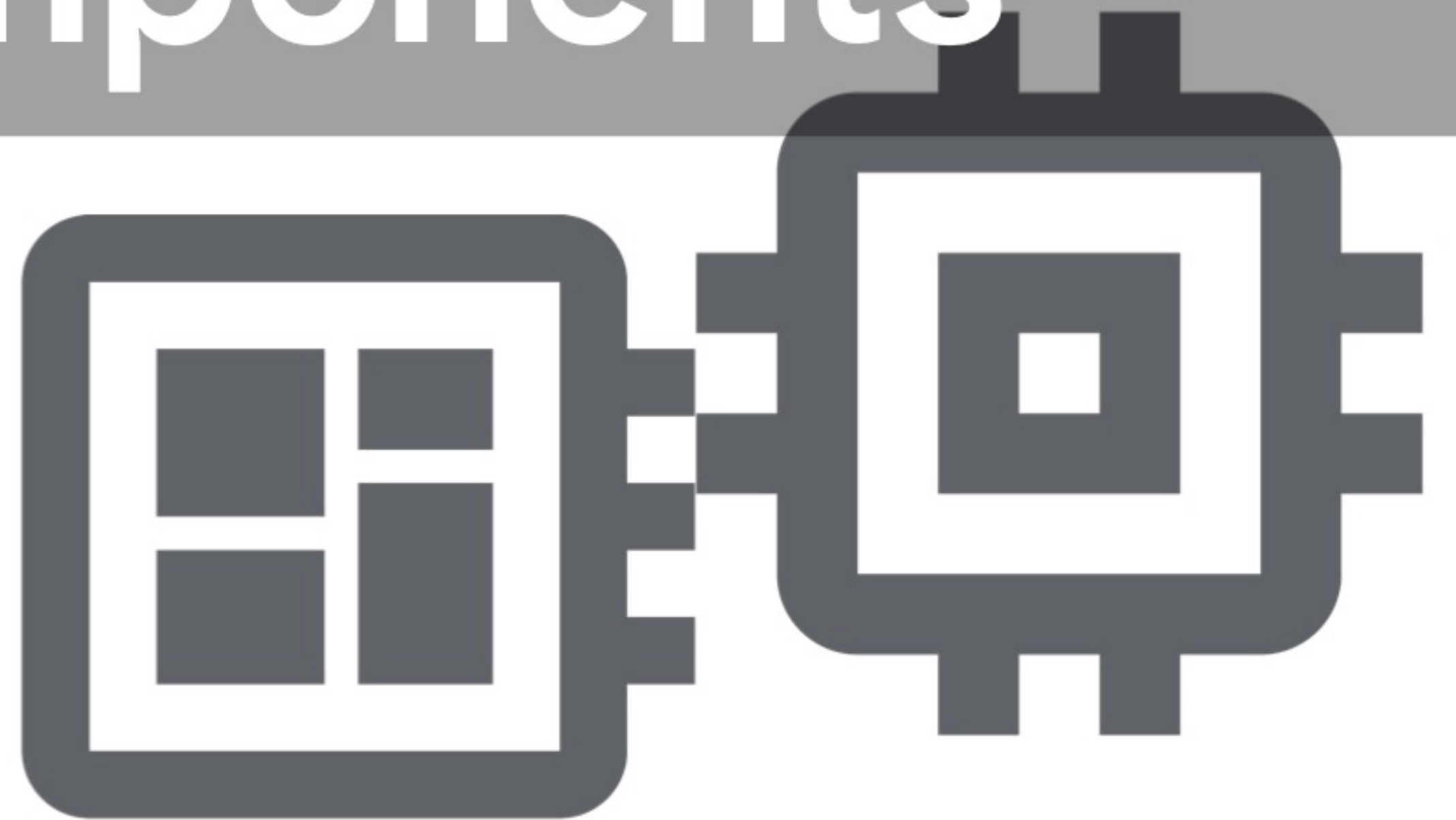
- ✓ HAL implementation based on secure hardware
- ✓ Optional **Direct Access** support
- ✓ Credential Store system daemon
- ✓ Framework APIs

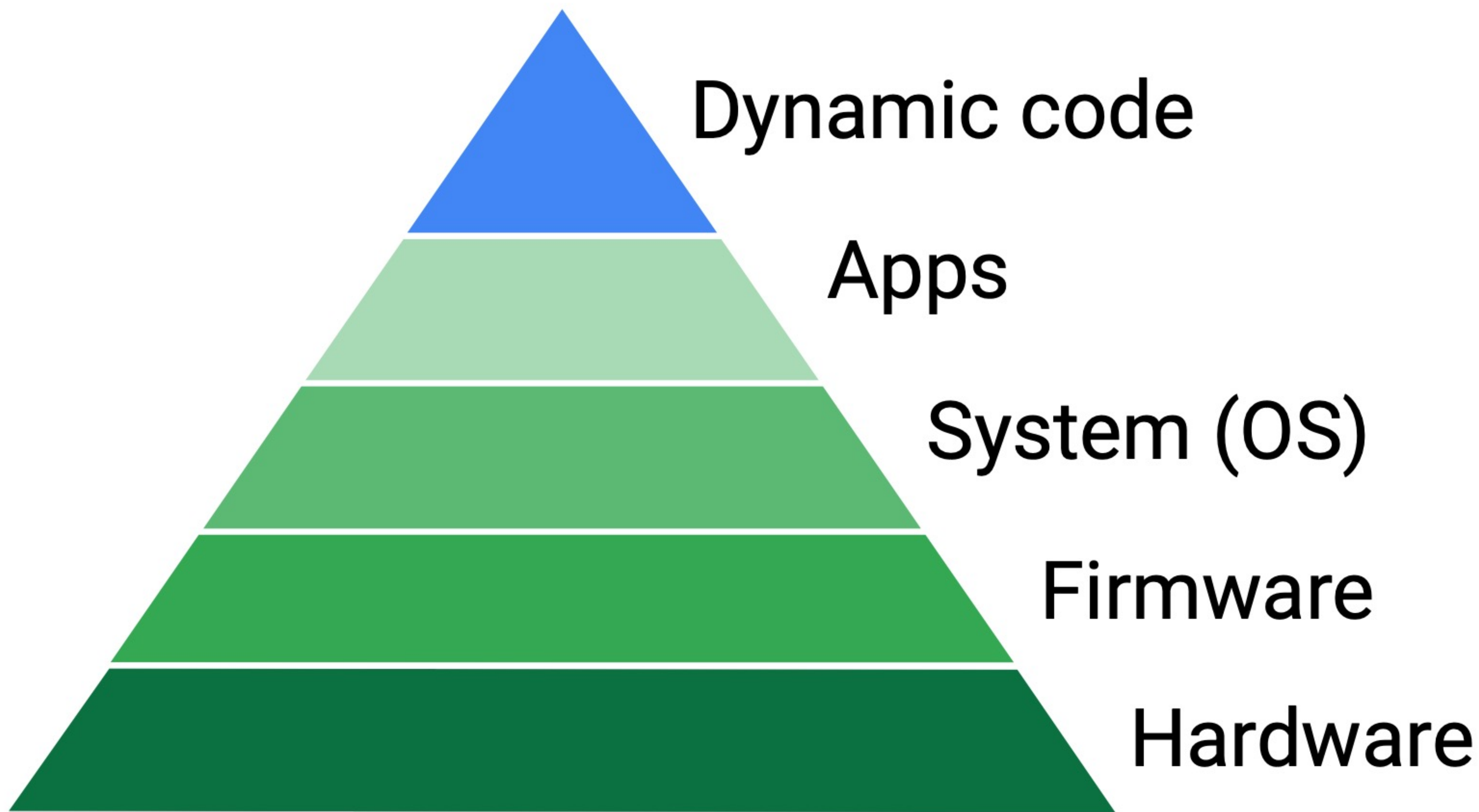


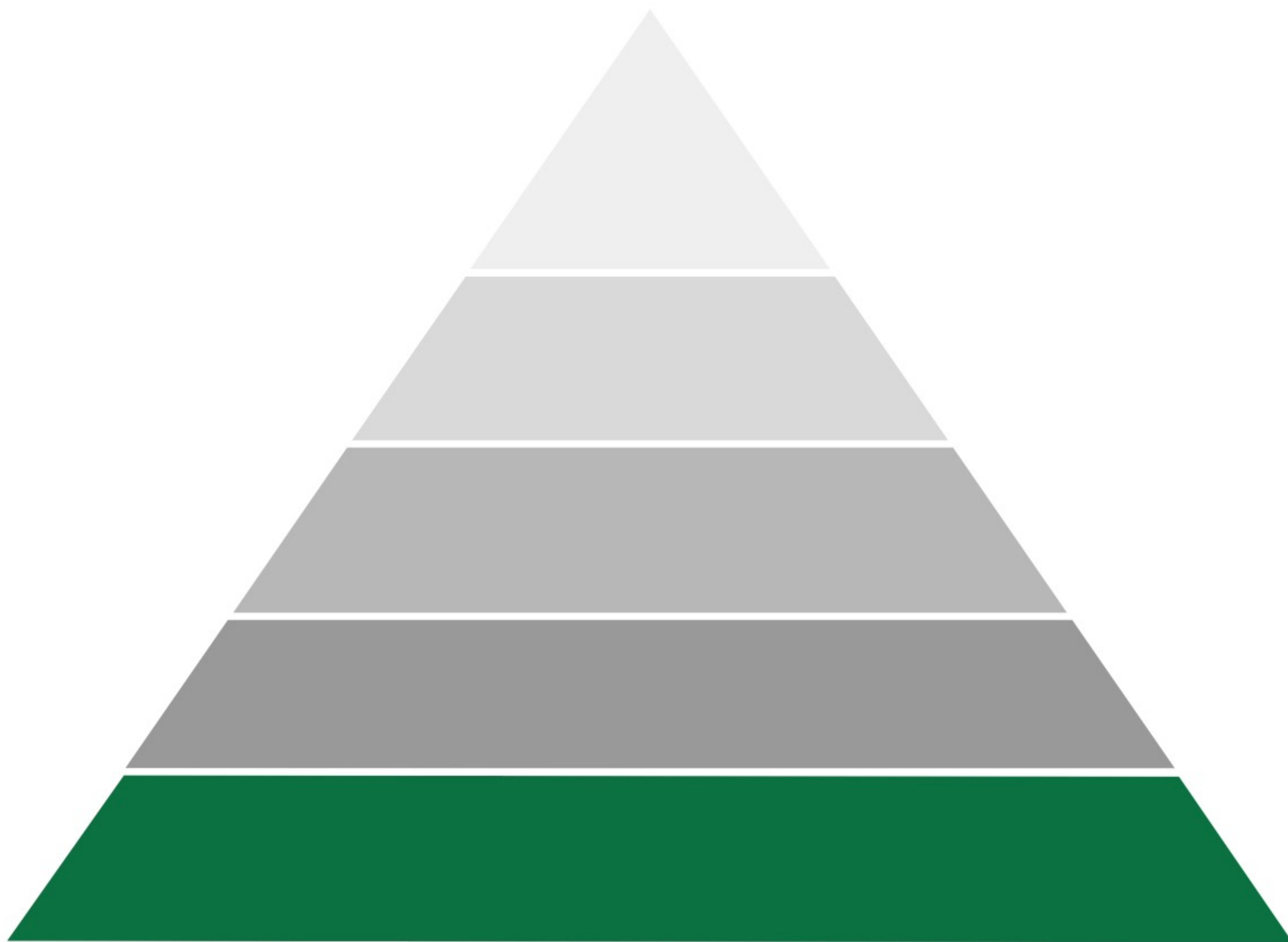




# Simple and few trusted components



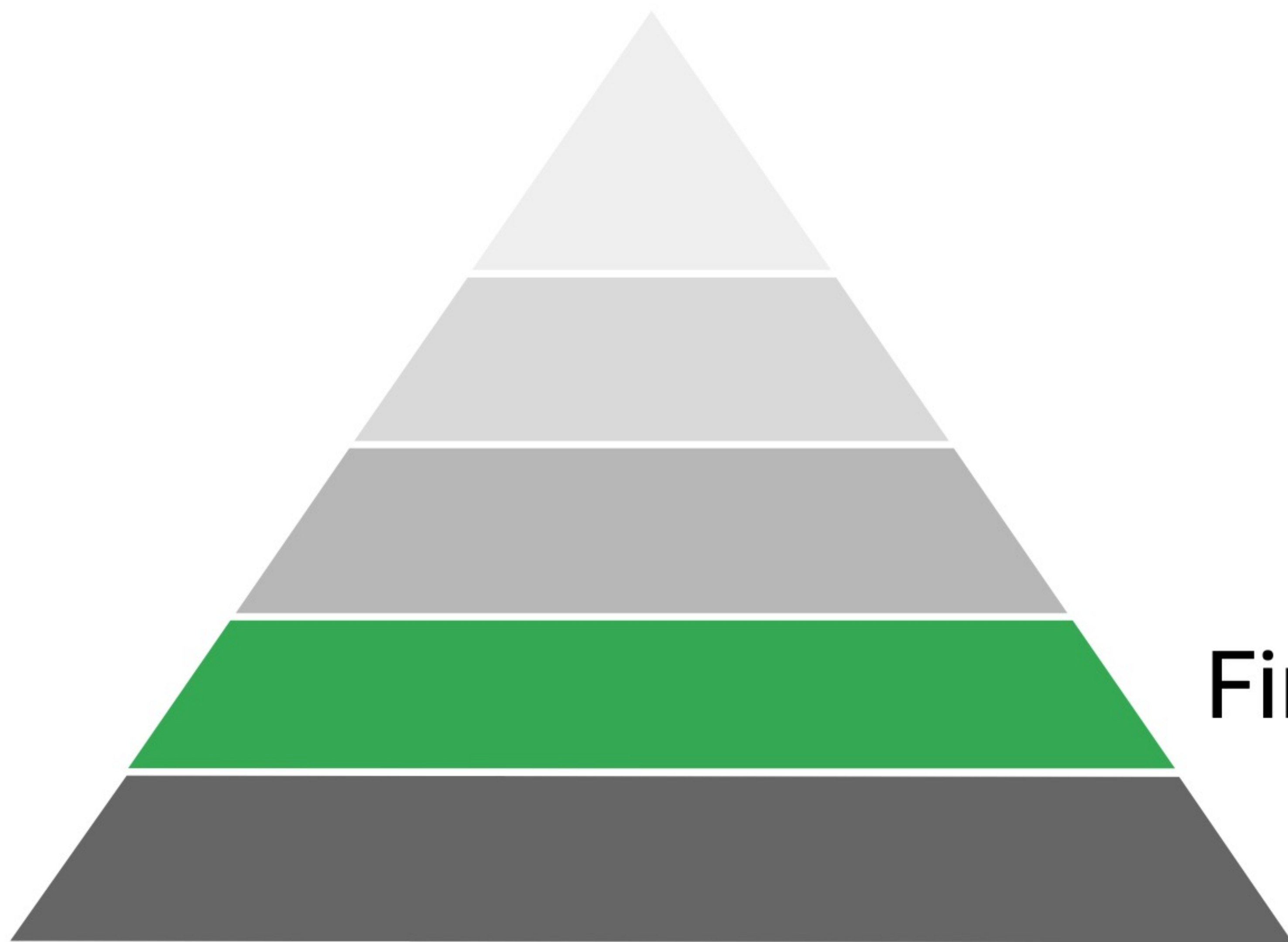




**Hardware**

# Threat models / scenarios for hardware security

- **Basic** assumption for hardware security:
  - Adversary has possession of the hardware
  - Adversary has control over all network channels
  - Adversary can influence sensor readings/input
- **Intermediate** assumptions:
  - Side channel analysis: including power, RF, timing, and potentially others
  - Side channel injection: including power, clock, RF (up to laser), and potentially others
  - Reverse engineering of hardware
  - Modification of hardware on PCB level, but not chip level
- **Advanced** assumptions:  
(AKA nation state adversaries or **insider** threats)
  - Modification of hardware on chip level
  - Access to internal signing keys



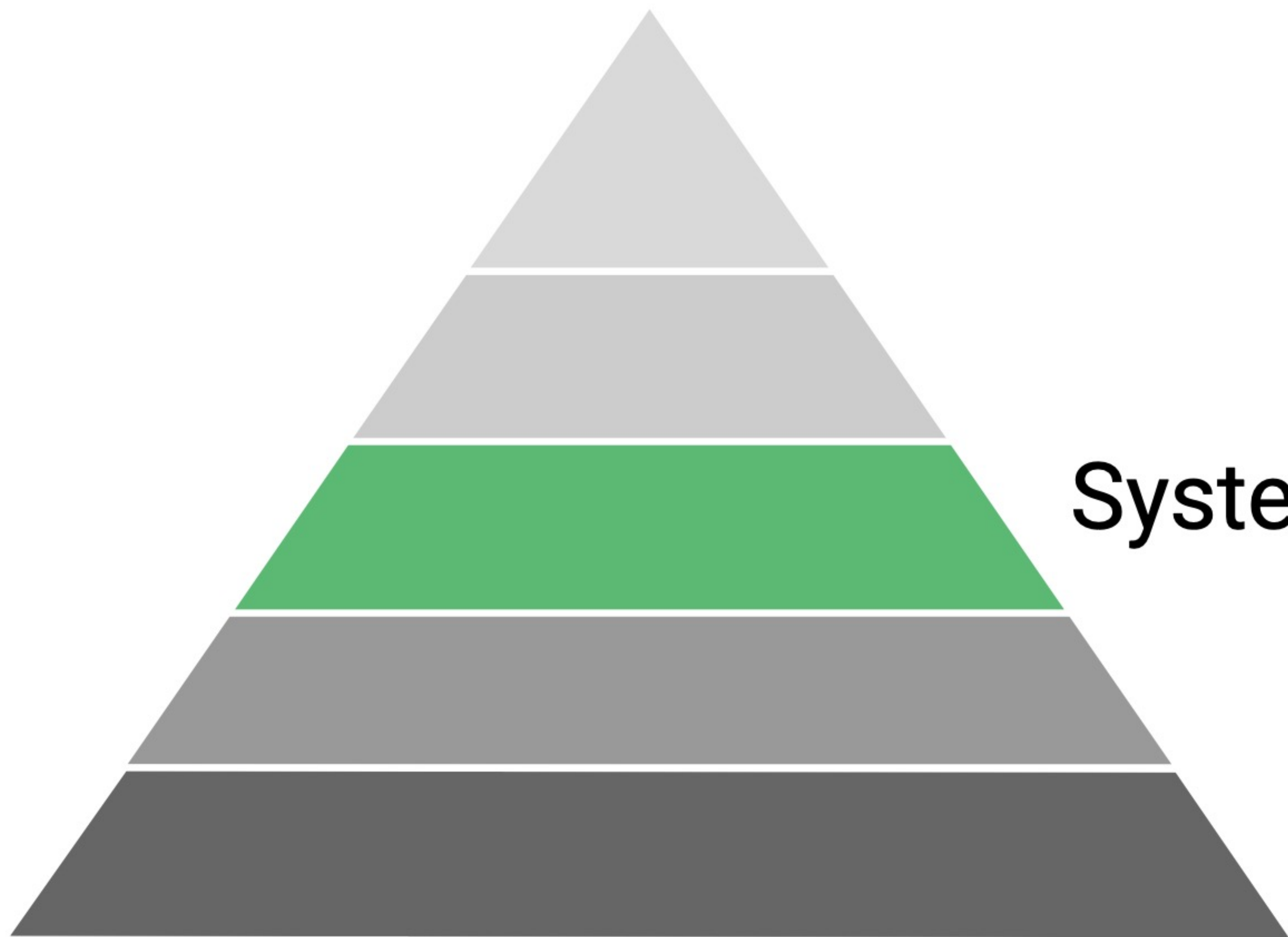
**Firmware**

# Wipe on firmware update without user involvement

[C-SR] are **STRONGLY RECOMMENDED** to provide **insider attack resistance (IAR)**, which means that an insider with access to firmware signing keys cannot produce firmware that causes the StrongBox to leak secrets, to bypass functional security requirements or otherwise enable access to sensitive user data. The recommended way to implement IAR is to **allow firmware updates only when the primary user password is provided** via the IAuthSecret HAL. IAR **will likely become a requirement in a future release**.

<https://android-developers.googleblog.com/2018/05/insider-attack-resistance.html>

<https://source.android.com/compatibility/9.0/android-9.0-cdd> Section 9.11.2. StrongBox



**System (OS)**

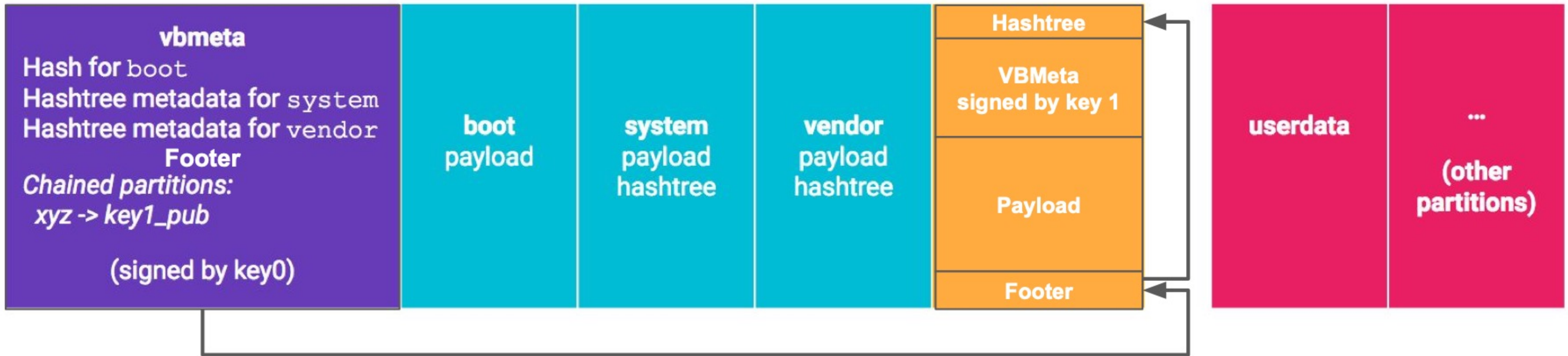




# Transparency for system updates

# Android Verified Boot (AVB) / VBMeta

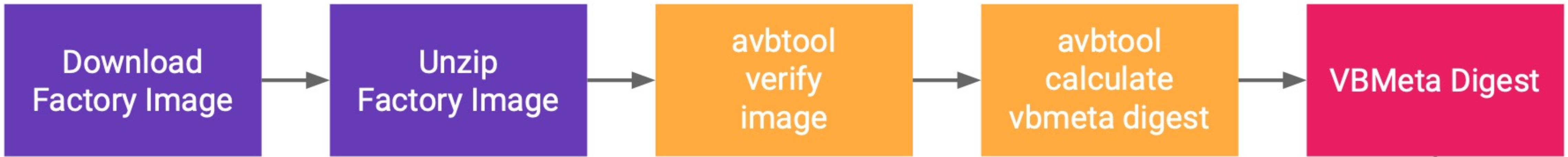
- AVB uses VBMeta structures to describe/verify elements of the boot chain.
- Bootloader stores hash measurement of VBMeta into KeyMaster v4
- VBMeta lives either in its own partition or on chained partitions
- The hash of VBMeta can be remotely attested with Key Attestation



<https://android.googlesource.com/platform/external/avb/>  
<https://developers.google.com/android/images>

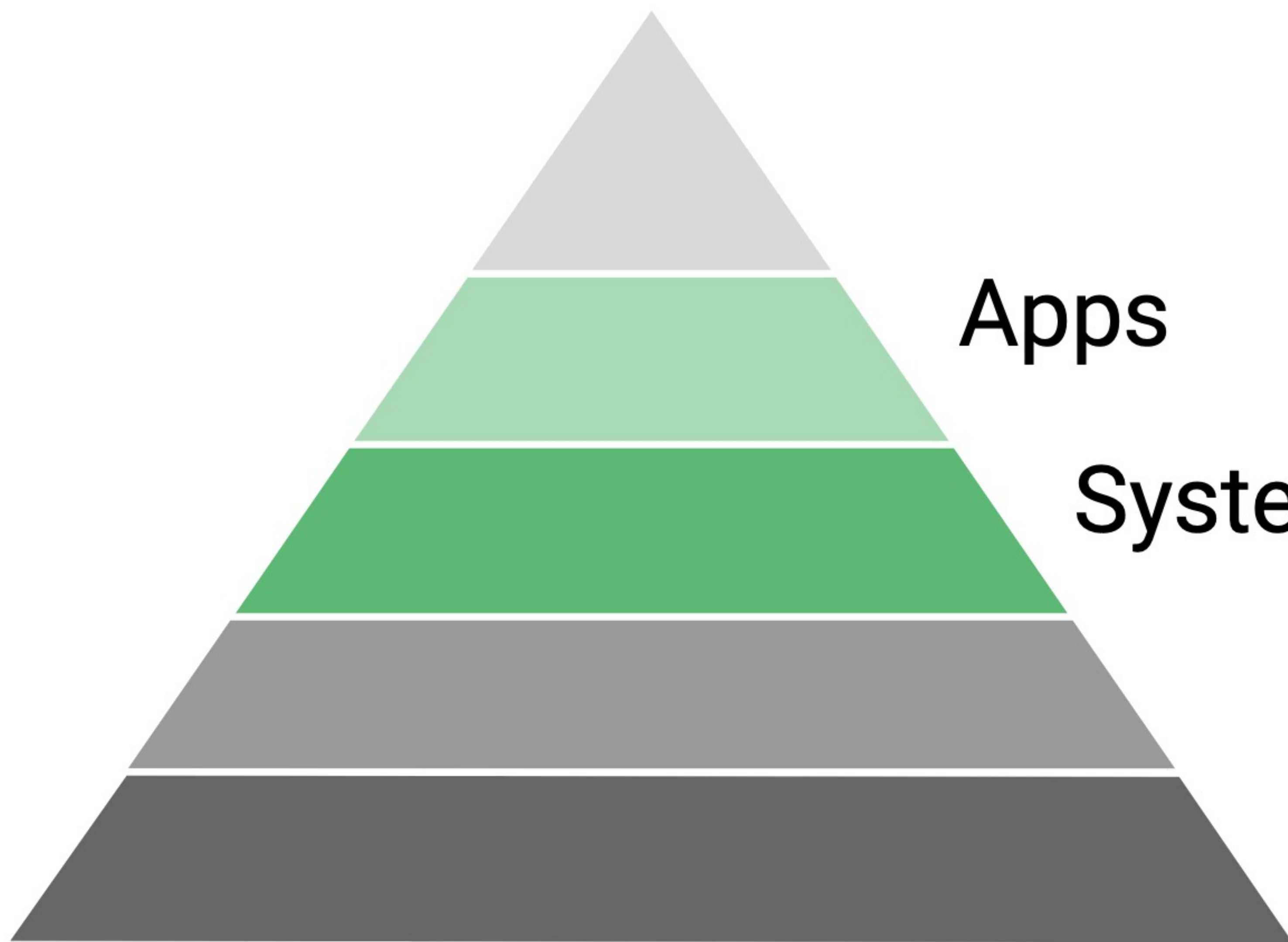
# VBMeta digest verification

## Getting reference VBMeta digest



## Attestation and verification of VBMeta digest





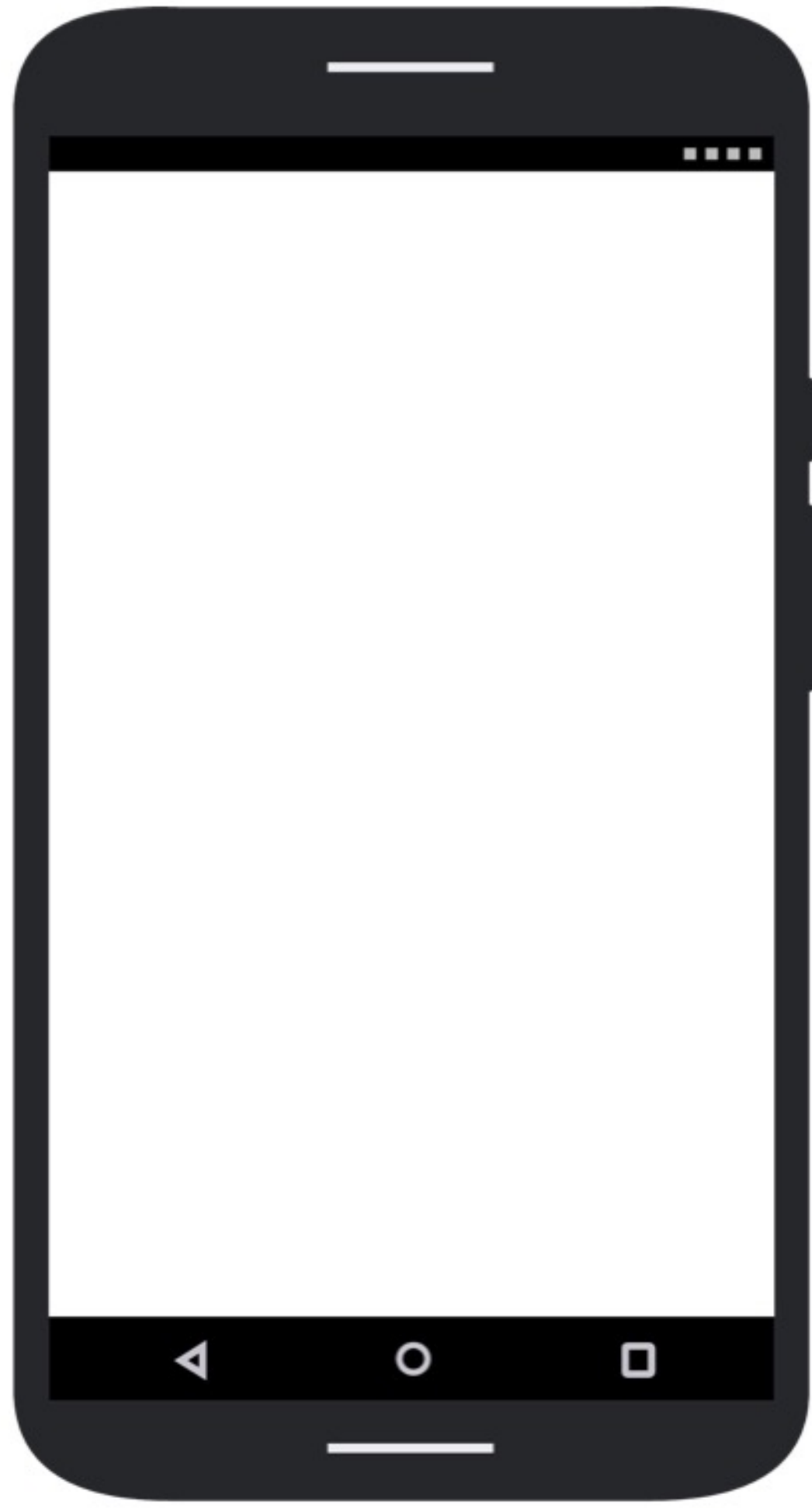
**Apps**

**System (OS)**

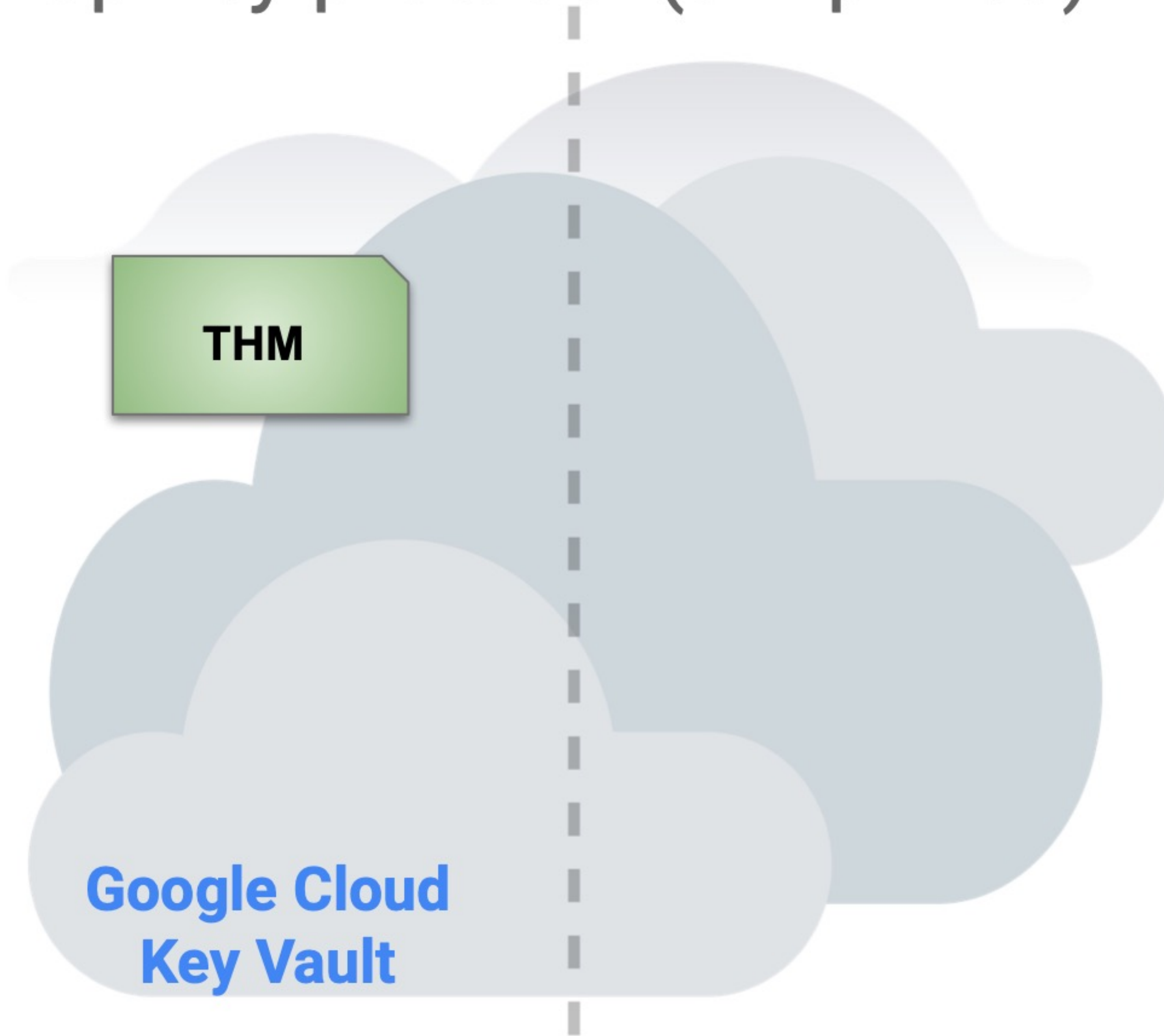
# End-to-end backup encryption

# Encrypted backup key protocol (simplified)

**Backup**

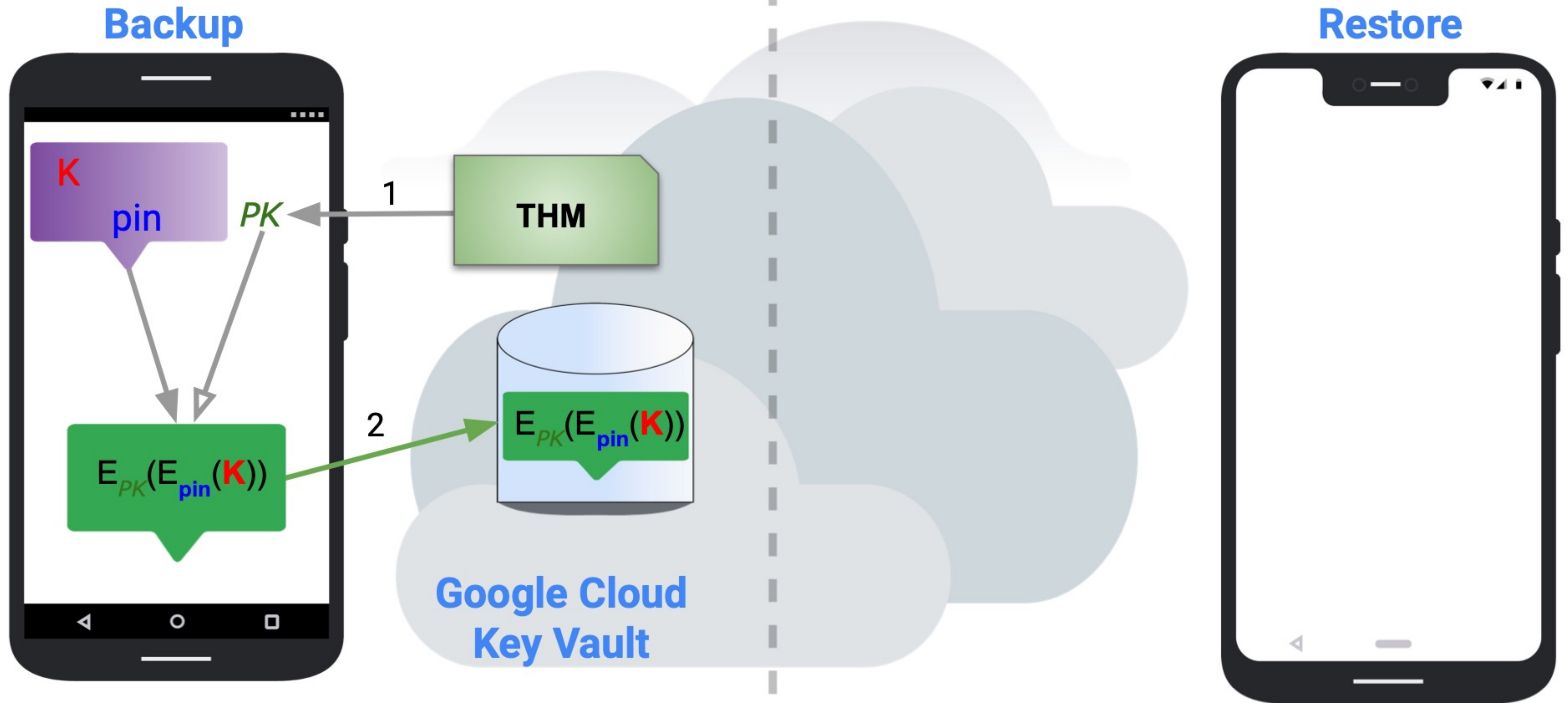


**Restore**



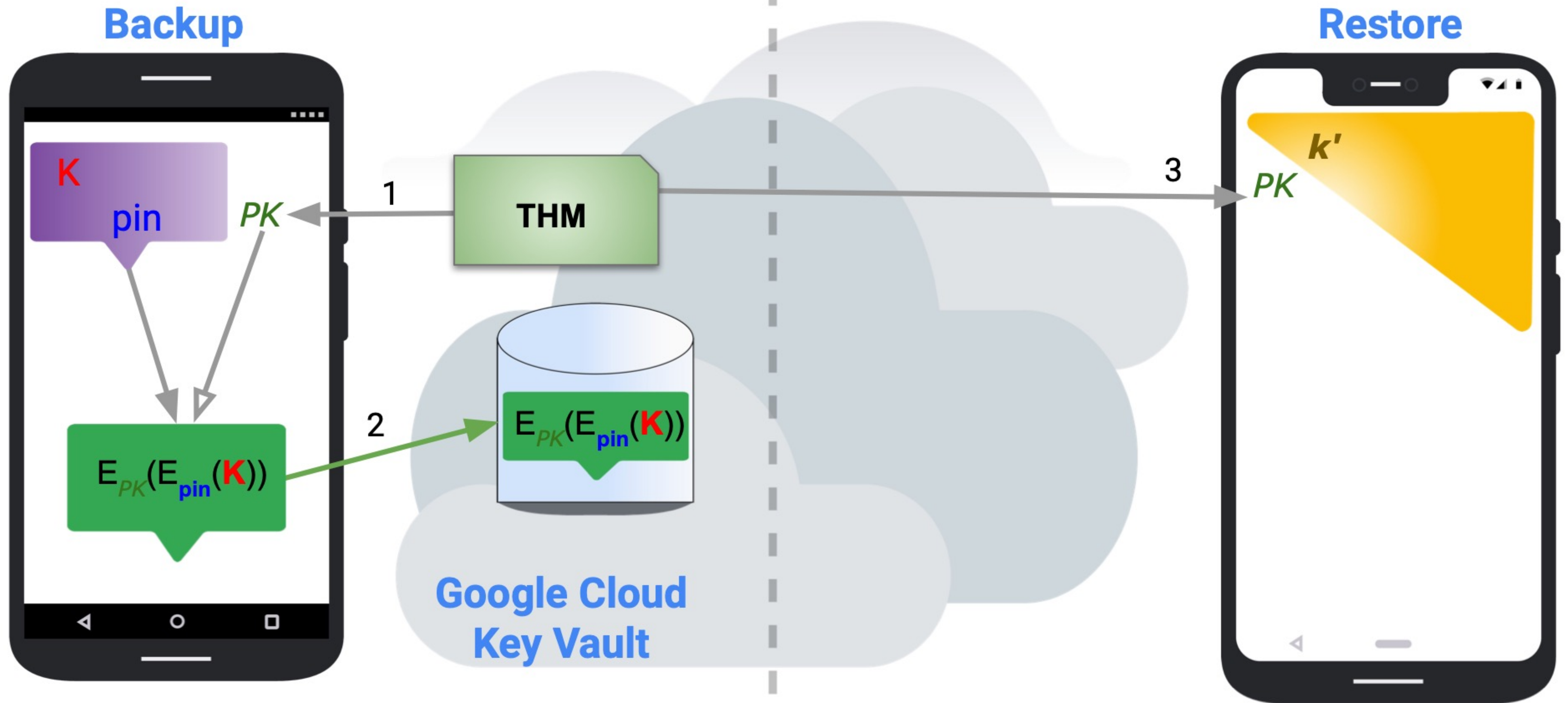
**Google Cloud  
Key Vault**

# Encrypted backup key protocol (simplified)



<https://developer.android.com/about/versions/pie/security/ckv-whitepaper>  
<https://security.googleblog.com/2018/10/google-and-android-have-your-back-by.html>

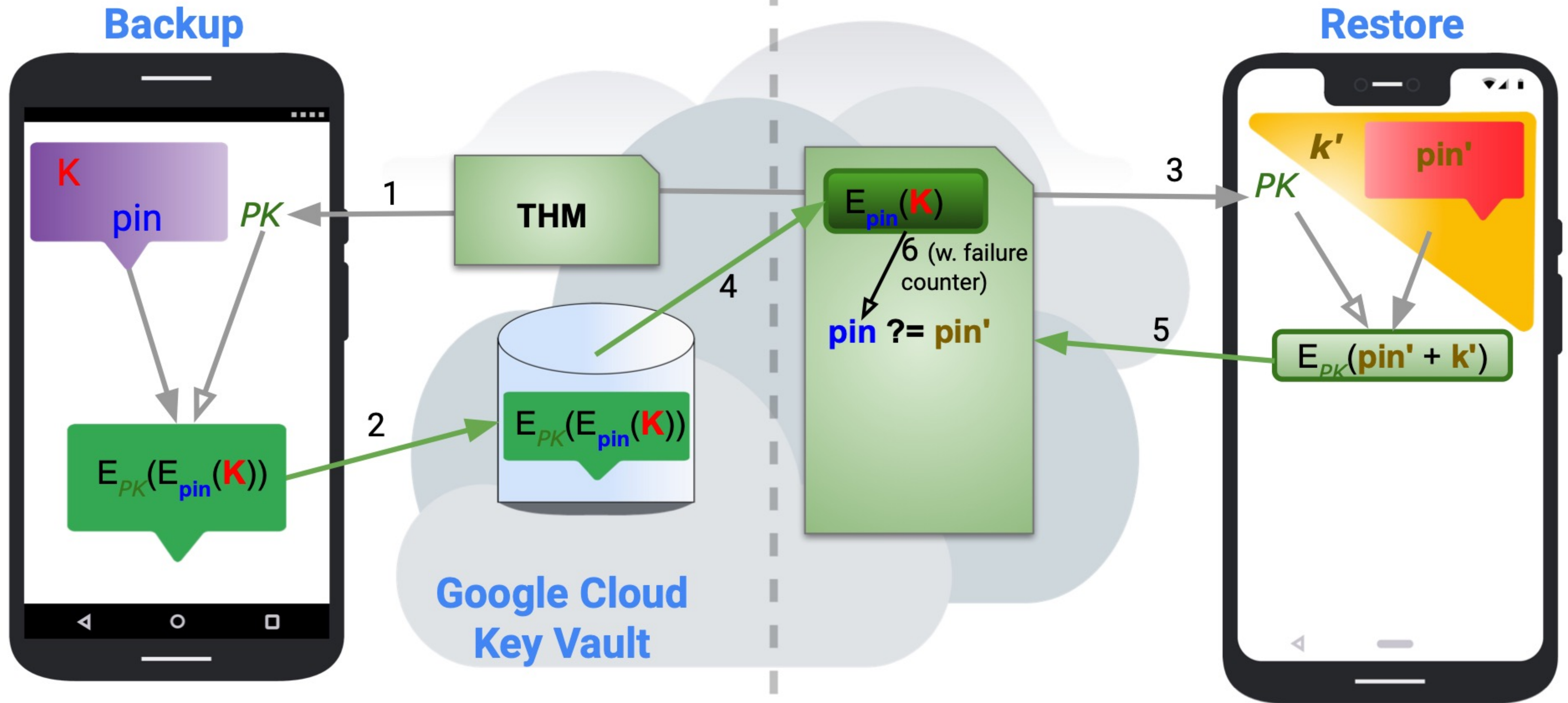
# Encrypted backup key protocol (simplified)



<https://developer.android.com/about/versions/pie/security/ckv-whitepaper>  
<https://security.googleblog.com/2018/10/google-and-android-have-your-back-by.html>

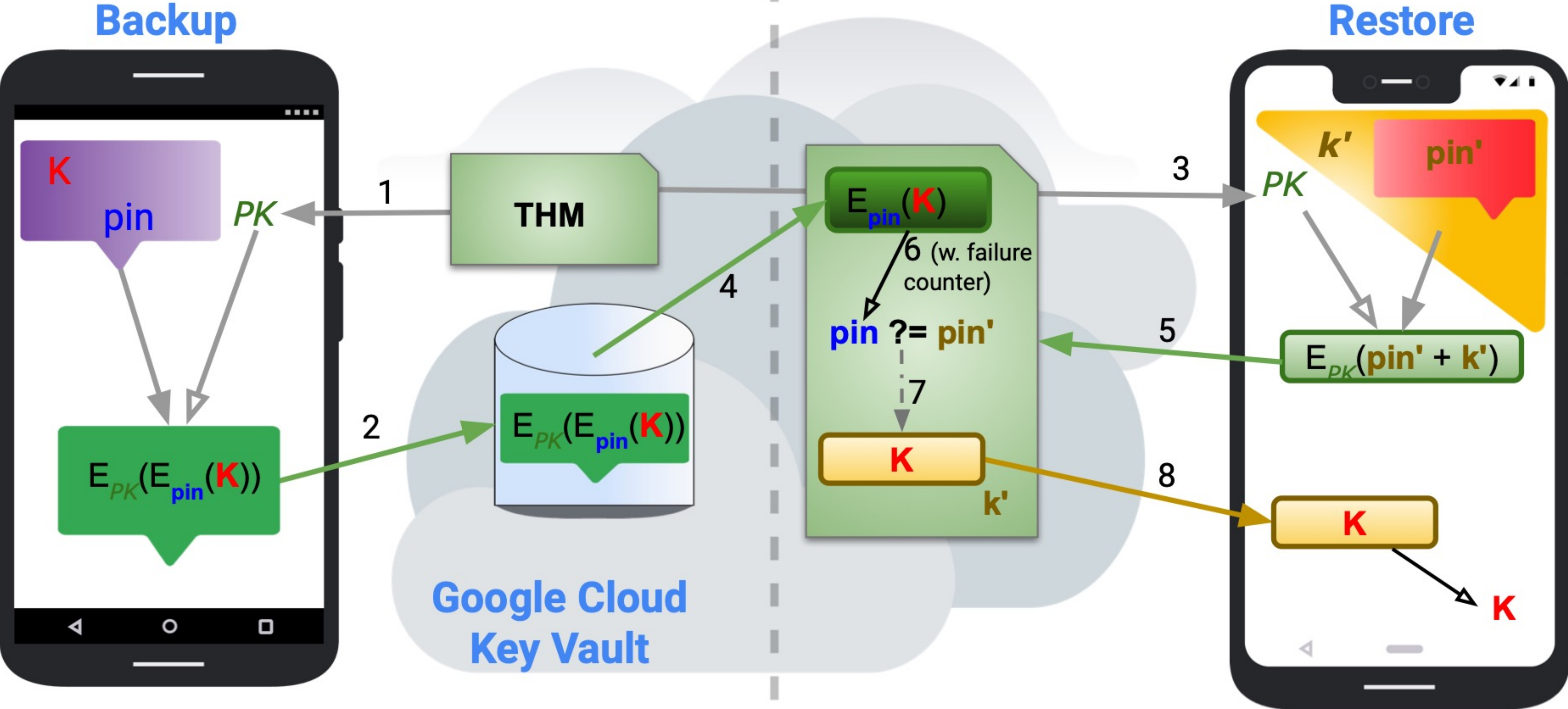


# Encrypted backup key protocol (simplified)



<https://developer.android.com/about/versions/pie/security/ckv-whitepaper>  
<https://security.googleblog.com/2018/10/google-and-android-have-your-back-by.html>

# Encrypted backup key protocol (simplified)



# (Some) Resources

- <https://www.android.com/security-center/>
- <https://source.android.com/security>
- <https://developer.android.com/training/articles/security-tips>
- <https://arxiv.org/abs/1904.05572>
- [https://source.android.com/security/reports/Google\\_Android\\_Enterprise\\_Security\\_Whitepaper\\_2018.pdf](https://source.android.com/security/reports/Google_Android_Enterprise_Security_Whitepaper_2018.pdf)
- <https://android-developers.googleblog.com/search/label/Security>
- <https://android-developers.googleblog.com/2019/05/whats-new-in-android-q-security.html>
- <https://android-developers.googleblog.com/2018/12/android-pie-la-mode-security-privacy.html>
- <https://www.google.com/about/appsecurity/research/presentations/>
- <https://www.mayrhofer.eu.org/post/android-tradeoffs-0-meta/>
- <https://www.mayrhofer.eu.org/post/android-tradeoffs-1-rooting/>
- <https://www.blackhat.com/docs/us-17/thursday/us-17-Krlevich-Honey-I-Shrunk-The-Attack-Surface-Adventures-In-Android-Security-Hardening.pdf>
- <https://www.blackhat.com/docs/us-16/materials/us-16-Krlevich-The-Art-Of-Defense-How-Vulnerabilities-Help-Shape-Security-Features-And-Mitigations-In-Android.pdf>