Baton for Android: Key Agility without a Centralized Certificate Infrastructure

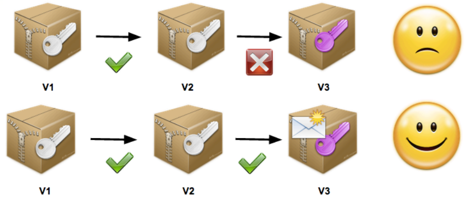
David Barrera, Daniel McCarney, Jeremy Clark, P.C. van Oorschot
Carleton University, Ottawa, Canada

Problem and Background

Android apps are signed by a set of 1 or more private keys

- There is no centralized signing authority/app market
- Signing keys are trusted on first install Subsequent app. updates must be signed by exact same set of keys
- No expiration, revocation or key change supported

Reasons that developers might want to change keys: key compromise, sale of application to another developer, key expiration



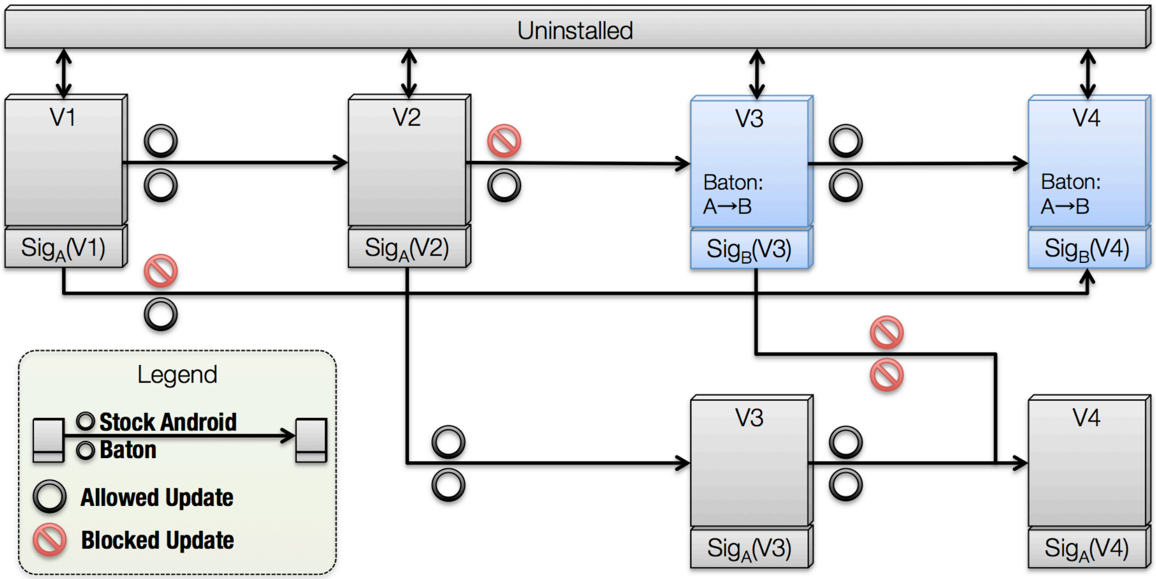
Baton

Baton extends the Android OS to allow developers to delegate signing authority

- To change signing keys, the developer signs a delegation token endorsing the new set of signing keys
- The delegation token is embedded in a certificate chain included in future application packages as part of the app manifest

Token contains app package name, version code, the previous signing certificates, and the newly endorsed signing certificates.

- Each token also includes a hash of the previous token in the chain.



Benefits

- No user involvement: Baton is a system-level mechanism requiring no user action
- Compatible with Android security model: Baton does not break signature protected resources (Shared UIDs, Signature Permissions)
- Incrementally deployable: Users without Baton can still install apps that embed Baton delegation token

Baton Components

Patches to the Android Open Source Project (AOSP) code

- Adds code to represent certificate chain and delegation tokens
- Modifies the *PackageParser* class (to read the certificate chain)
- Modifies the *PackageManageService* to implement the Baton certificate chain verification

Developer tools

- Eclipse plugin to generate delegation tokens
- Walks developer through endorsing new signing keys using their existing keys
- Stand-alone GUI and command line versions also available

ISSNet

Carleton
UNIVERSITY