Automated embedding of dynamic libraries into iOS applications from GNU/Linux

Marwin Baumann[1] & Leandro Velasco[1]

[1]Systems and Network Engineering MSc.
University of Amsterdam

Research Project 2, 2017

Introduction

Dynamic library embedding:

- Deploy debugging mechanisms
- Monitor the invocation of functions
- Tracking how data is propagated through the application
- Modify the behavior of Apps (on non-jailbroken devices)

Introduction

Dynamic library embedding:

- Deploy debugging mechanisms
- Monitor the invocation of functions
- Tracking how data is propagated through the application
- Modify the behavior of Apps (on non-jailbroken devices)

Common Use-case:

- Frida Instrumentation

Introduction

Problem:

- Only on MacOS
- MacOS in Virtual Machine not legal [1]
- Cumbersome process

Introduction

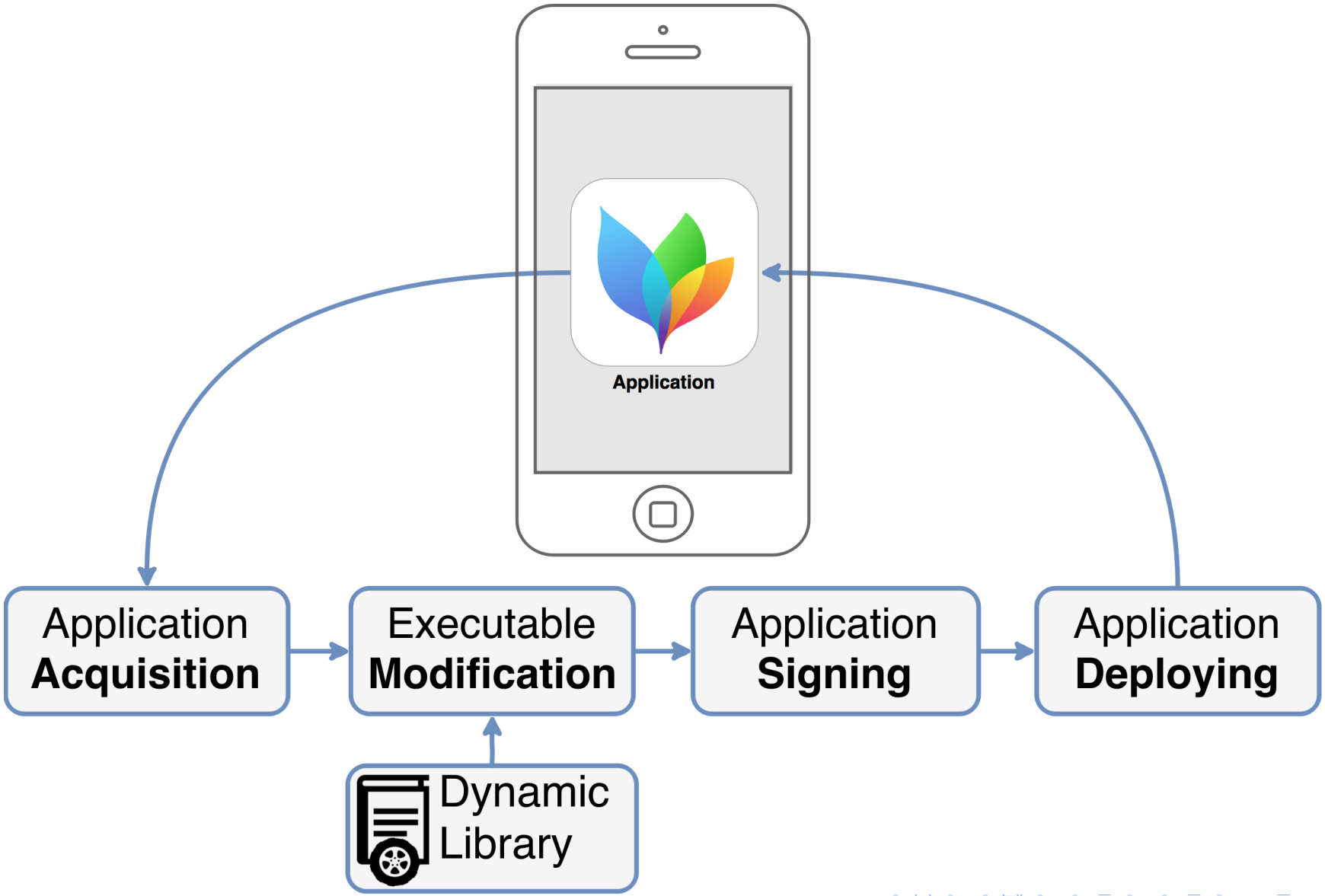
Problem:

- Only on MacOS
- MacOS in Virtual Machine not legal [1]
- Cumbersome process

Motivation:

- More apps released every day [2]
- Increase in need for mobile app security assessments
- Need for automation and free publicly available tools

Procedure Overview



Research Question

Is it possible from GNU/Linux to automate the process of embedding dynamic libraries into iOS applications?

Methodology

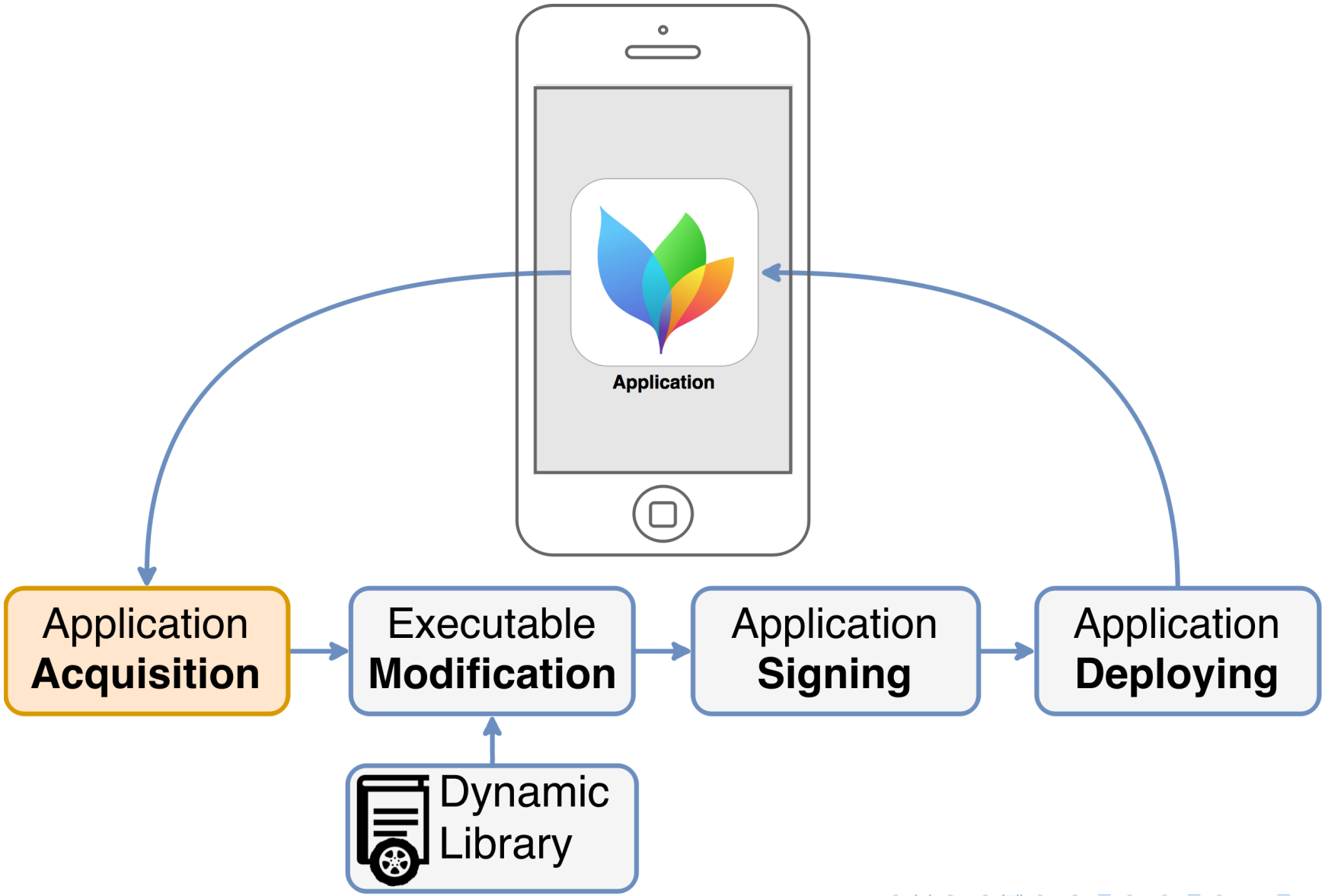
Study procedure internals:

- Analyze iOS application format
- Analyze internals of dynamic library embedding
- Investigate Xcode signing procedure

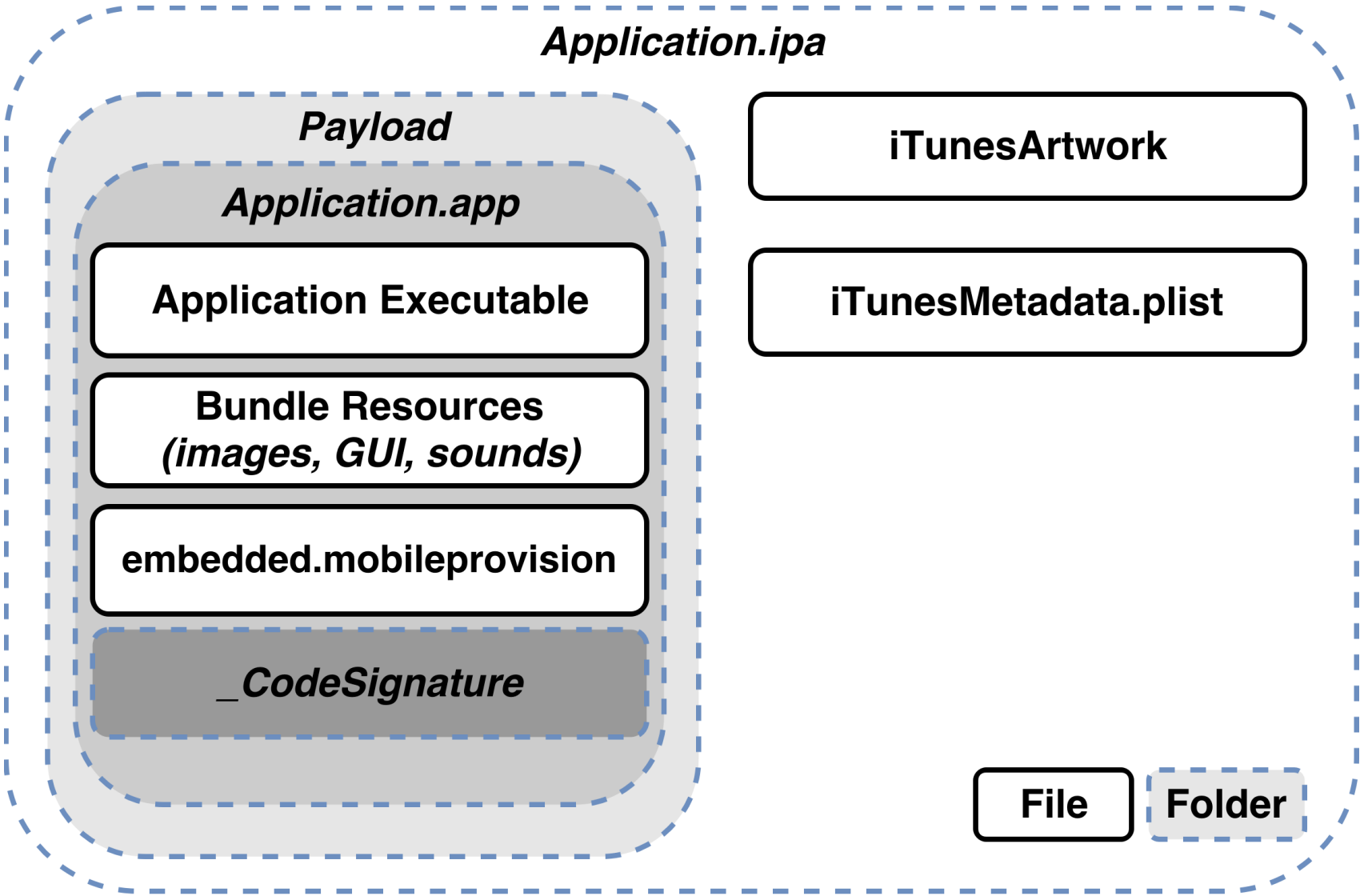
Implement procedure in GNU/Linux:

- Explore tools already ported
- Write/port new tools

Procedure Overview



iOS App Store Package (.ipa)



Application Acquisition

Pre iOS 9:

- Get IPA from backup

iOS 9 and later:

- iTunes redownload (Fairplay)
- Clutch

Application Acquisition

Pre iOS 9:

- Get IPA from backup

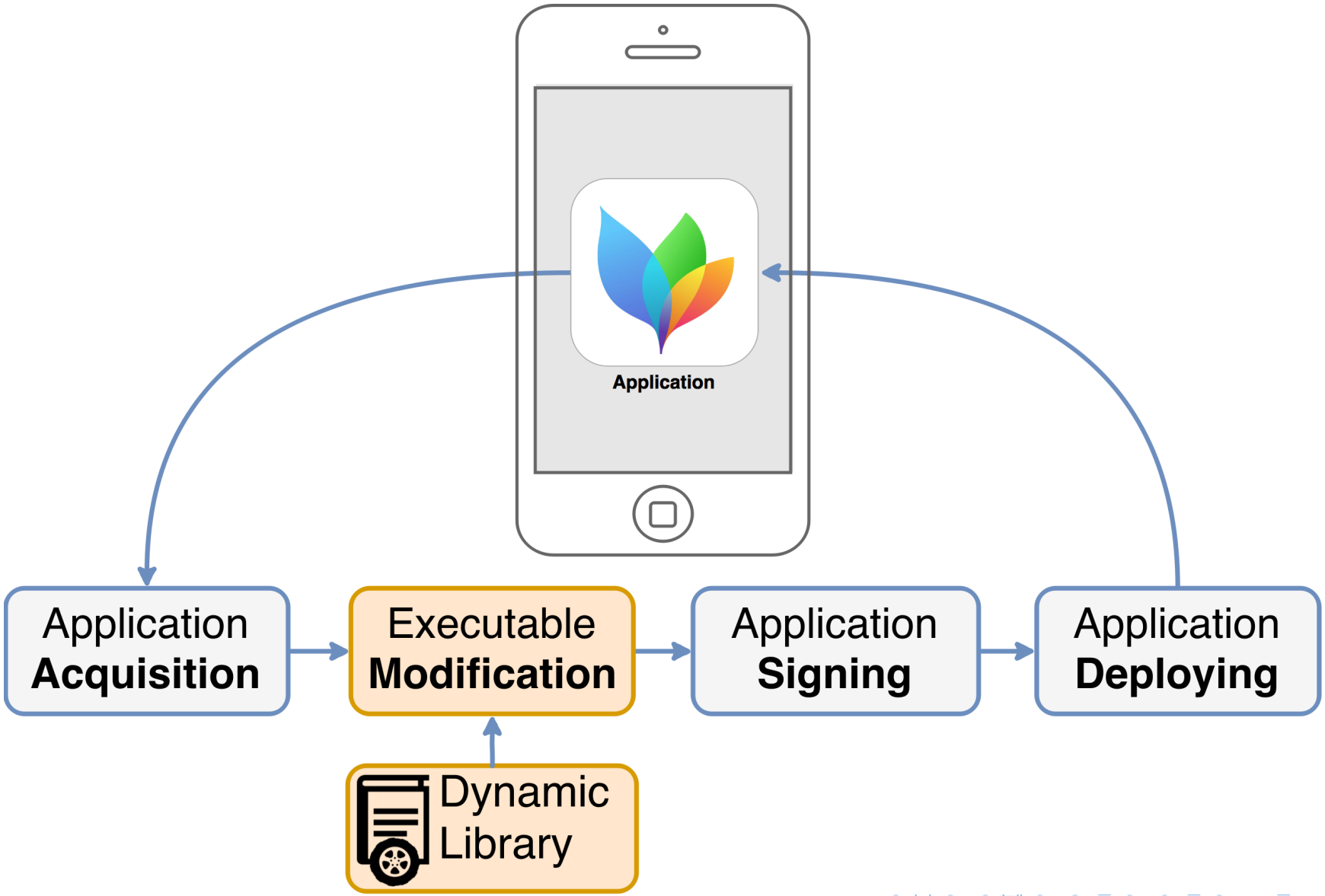
iOS 9 and later:

- iTunes redownload (Fairplay)
- Clutch

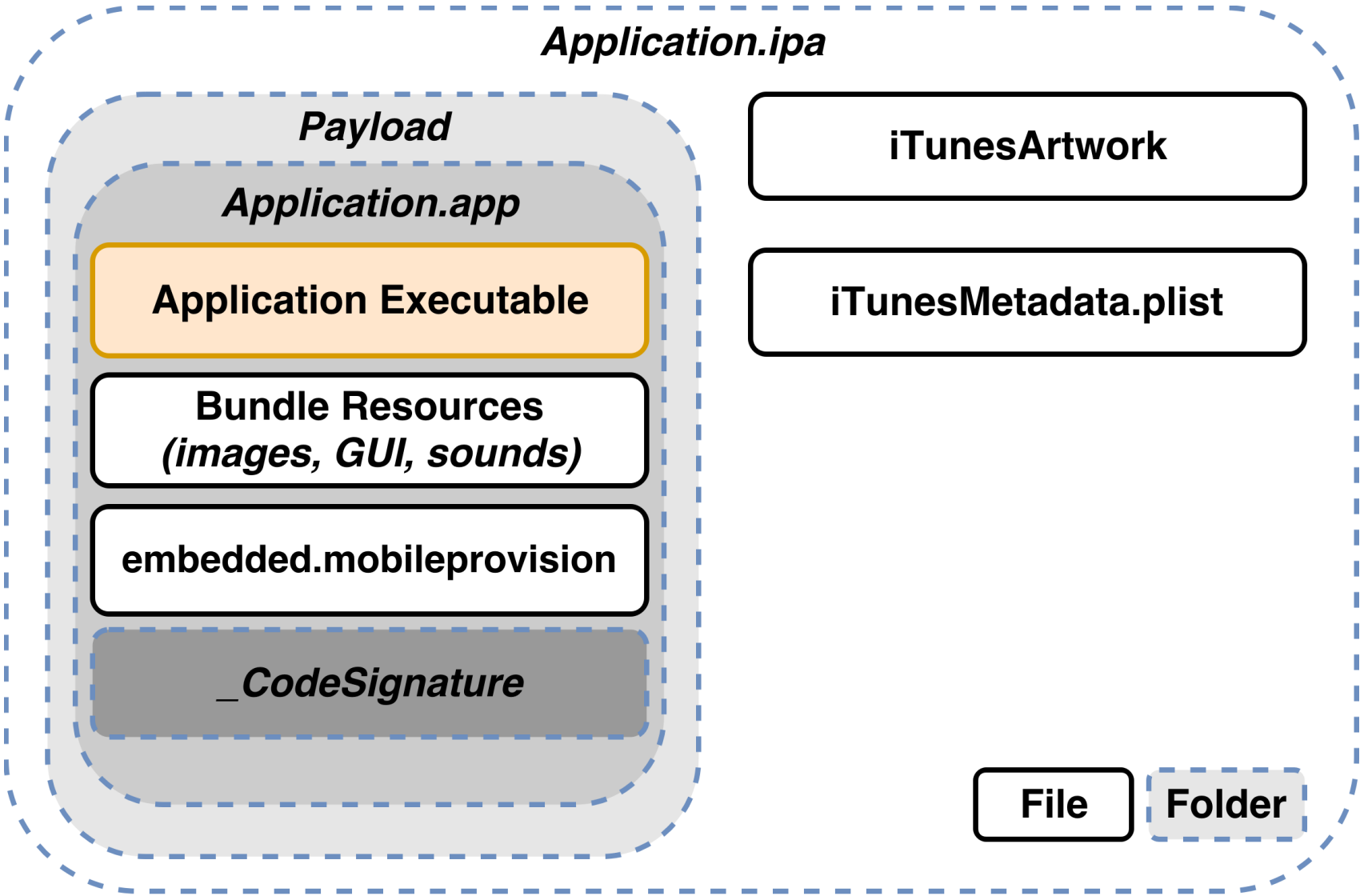
Requirements Clutch:

- Jailbroken iDevice running iOS 9+

Procedure Overview



iOS App Store Package (.ipa)



Mach-O File Format

Header

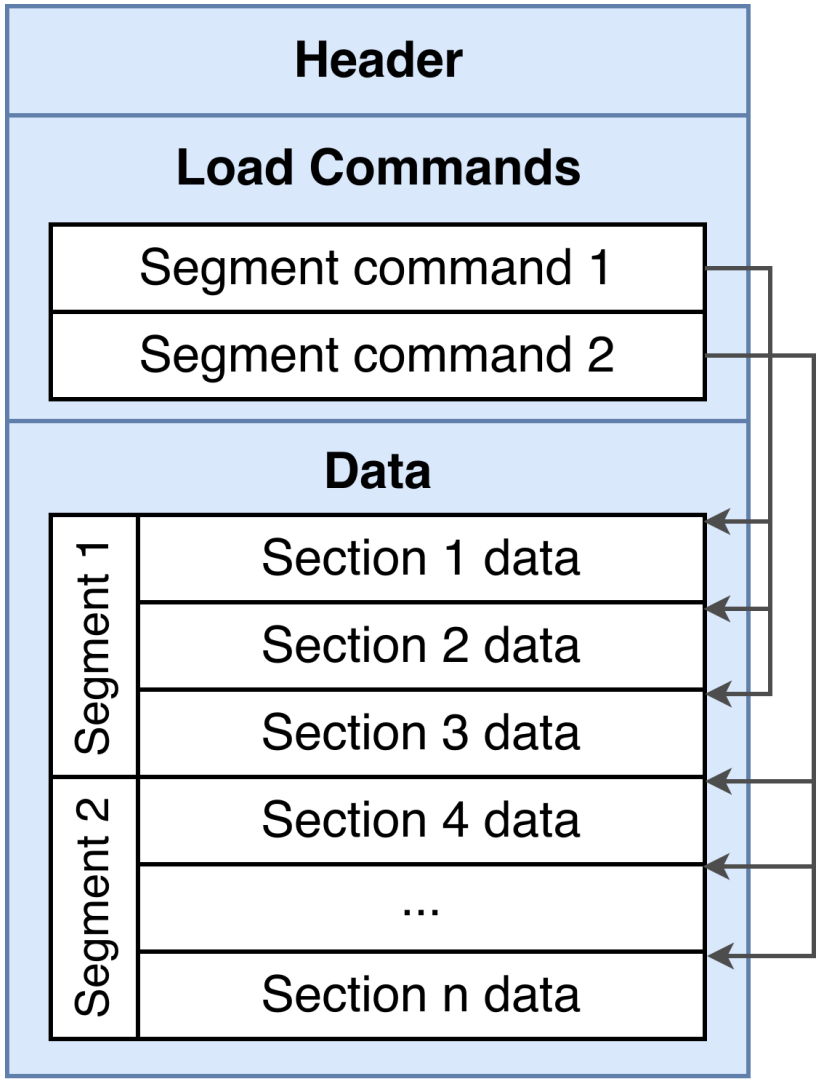
- Identifier
- Architecture
- Number of load commands
- Size of load commands
- ...

Load Command region

- Layout and linkage properties

Data region

- Data stored in segments which contain sections



Mach-O File Format

Header

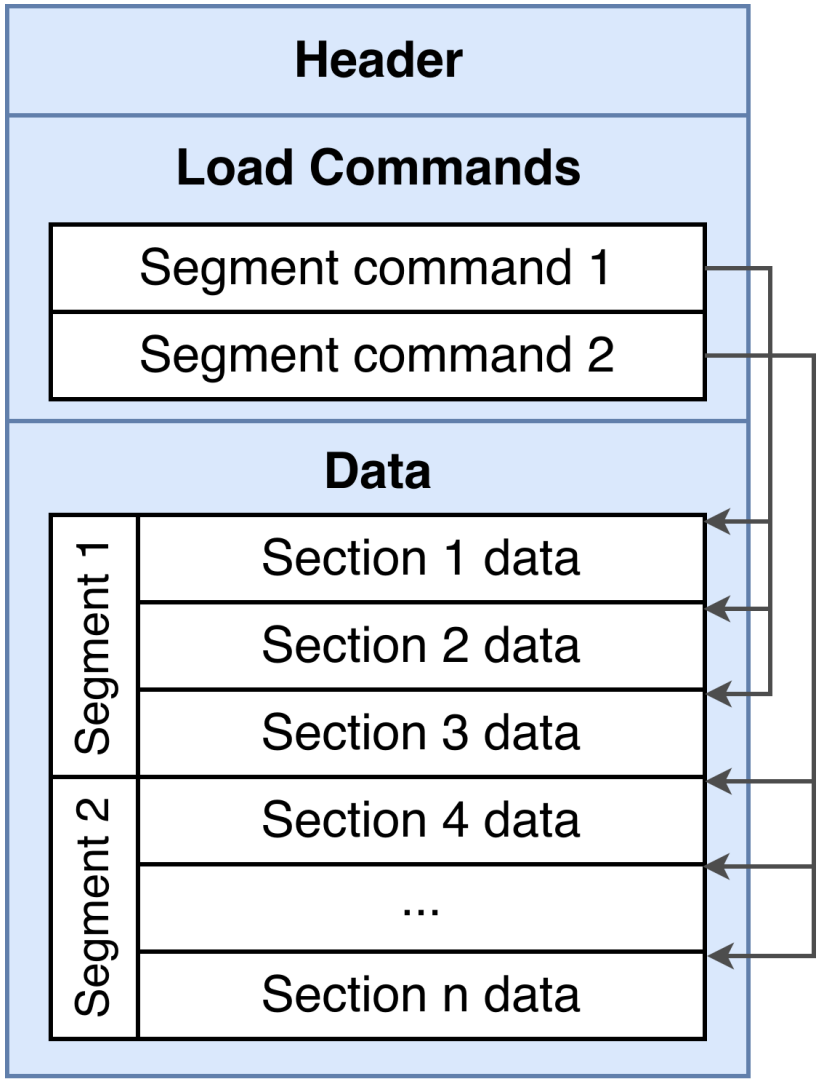
- Identifier
- Architecture
- Number of load commands
- Size of load commands
- ...

Load Command region

- Inserting a `LC_LOAD_DYLIB` command

Data region

- Data stored in segments which contain sections



Executable Modification

Open Source Tools (all MacOS):

- Node_applesign
- Optool
- Insert_dylib

Executable Modification

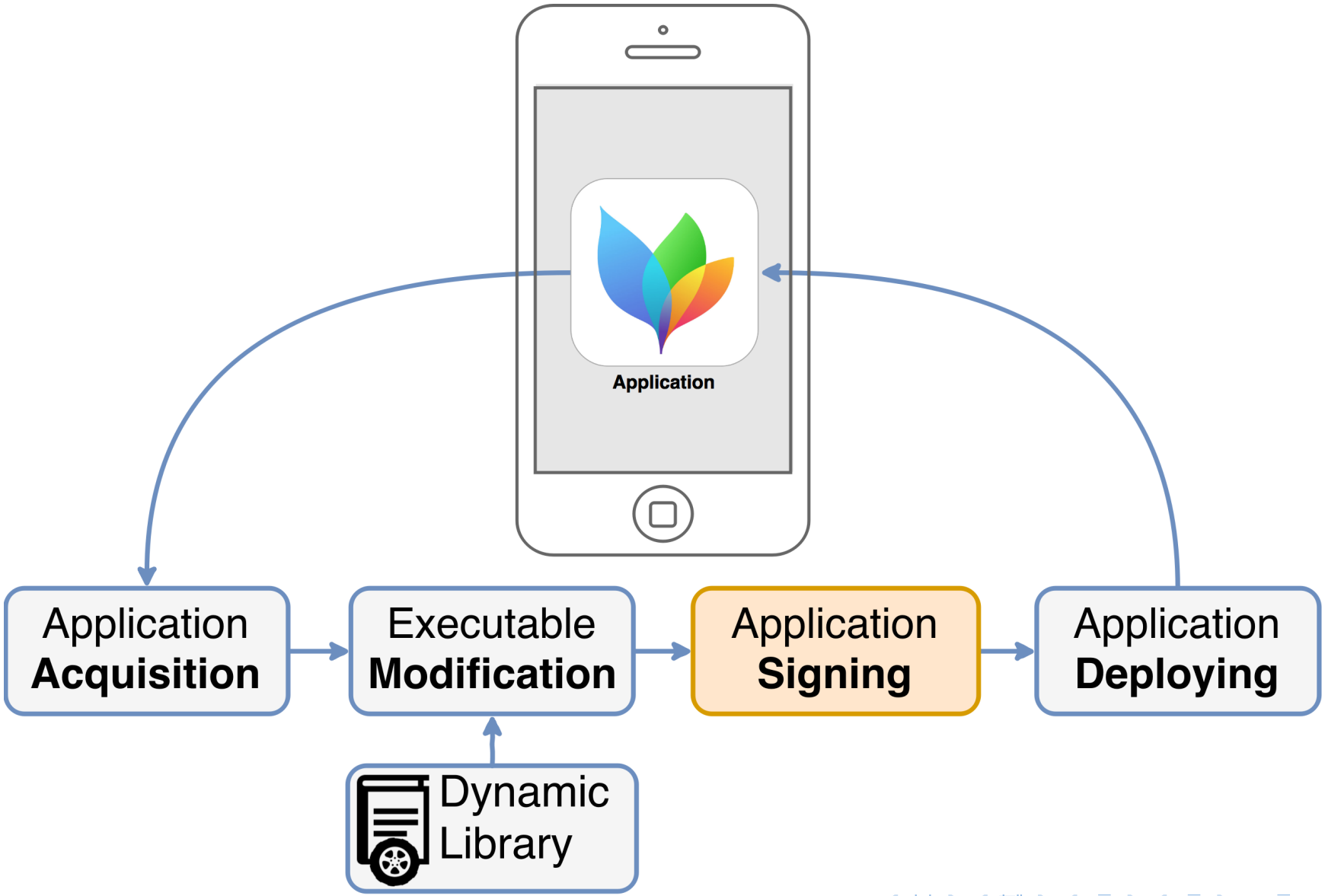
Open Source Tools (all MacOS):

- Node_applesign
- Optool
- Insert_dylib

Port Insert_dylib to GNU/Linux:

- Mach-O headers are Open Sourced by Apple
- Header files from `hogliux/cctools` project used
- Changed code to avoid usage of `copyfile.h`

Procedure Overview



Application Signing - Background

Mandatory Code Signing

- Integrity of the code
- Identify code source (developer / signer)
- For Apps not signed by Apple, Mobile Provisioning is needed

Application Signing - Background

Mandatory Code Signing

- Integrity of the code
- Identify code source (developer / signer)
- For Apps not signed by Apple, Mobile Provisioning is needed

Mobile Provisioning

- Free Apple Account
- Individual Developer Account
- Enterprise Developer Account

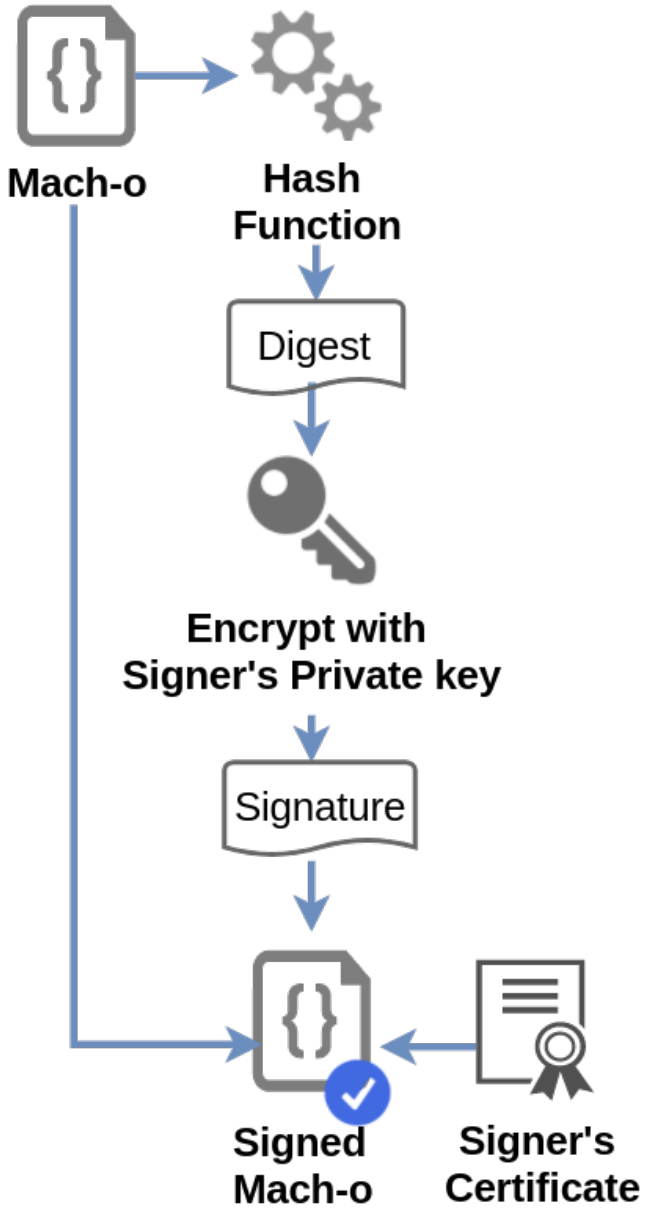
Application Signing - Procedure

Resources files :

- Signature stored in the file `_CodeSignature/CodeResources`

Mach-o files :

- Signature stored in the file via `LC_CODE_SIGNATURE` load command



Application Signing - Software

Jtool

- Only signs mach-o files
- Does not include Code Requirements in signature
- Close Source

iSign

- Signs complete IPA or app bundle
- Experimental branch needed to sign binaries from scratch
- Open Source

Application Signing - Software

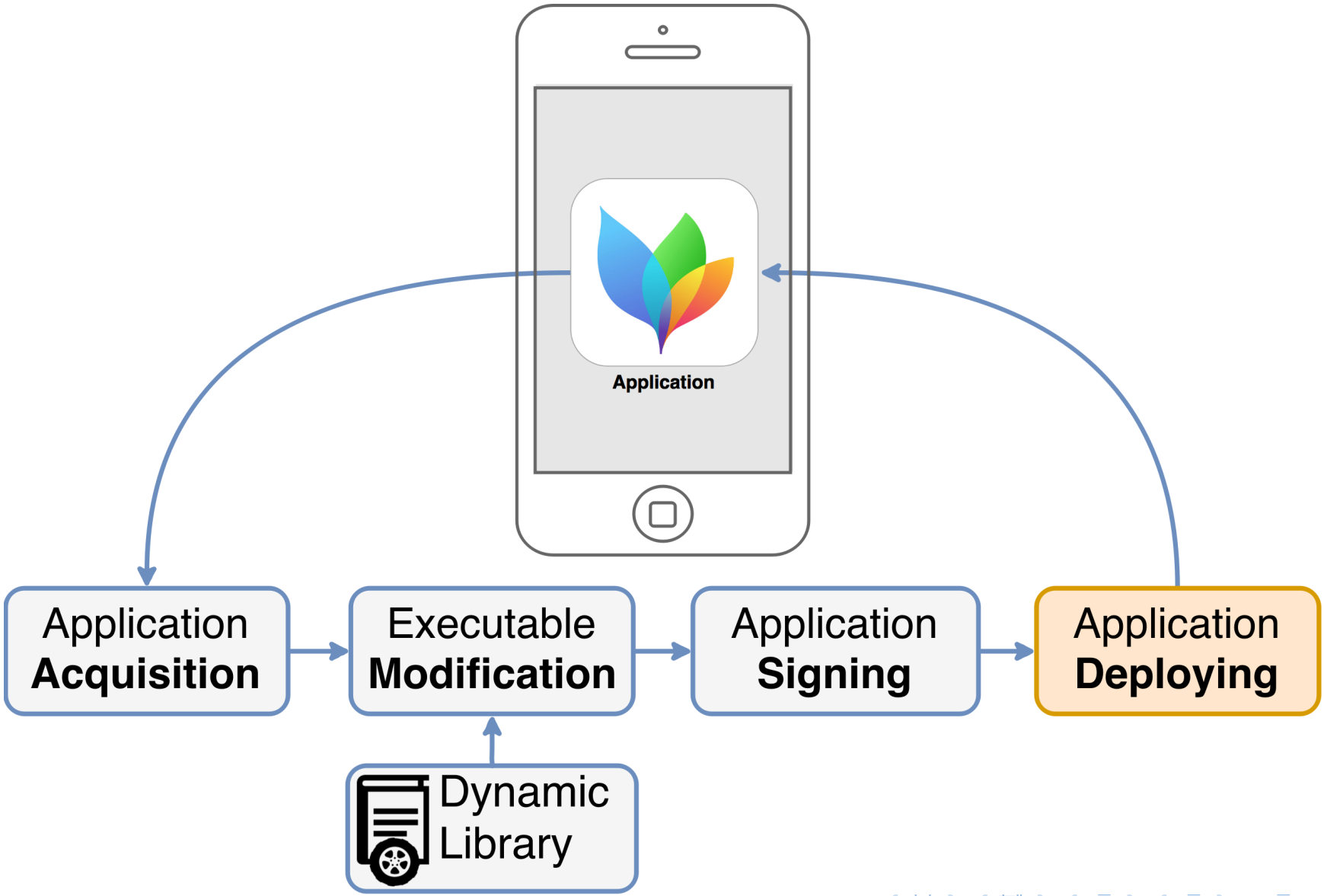
Jtool

- Only signs mach-o files
- Does not include Code Requirements in signature
- Close Source

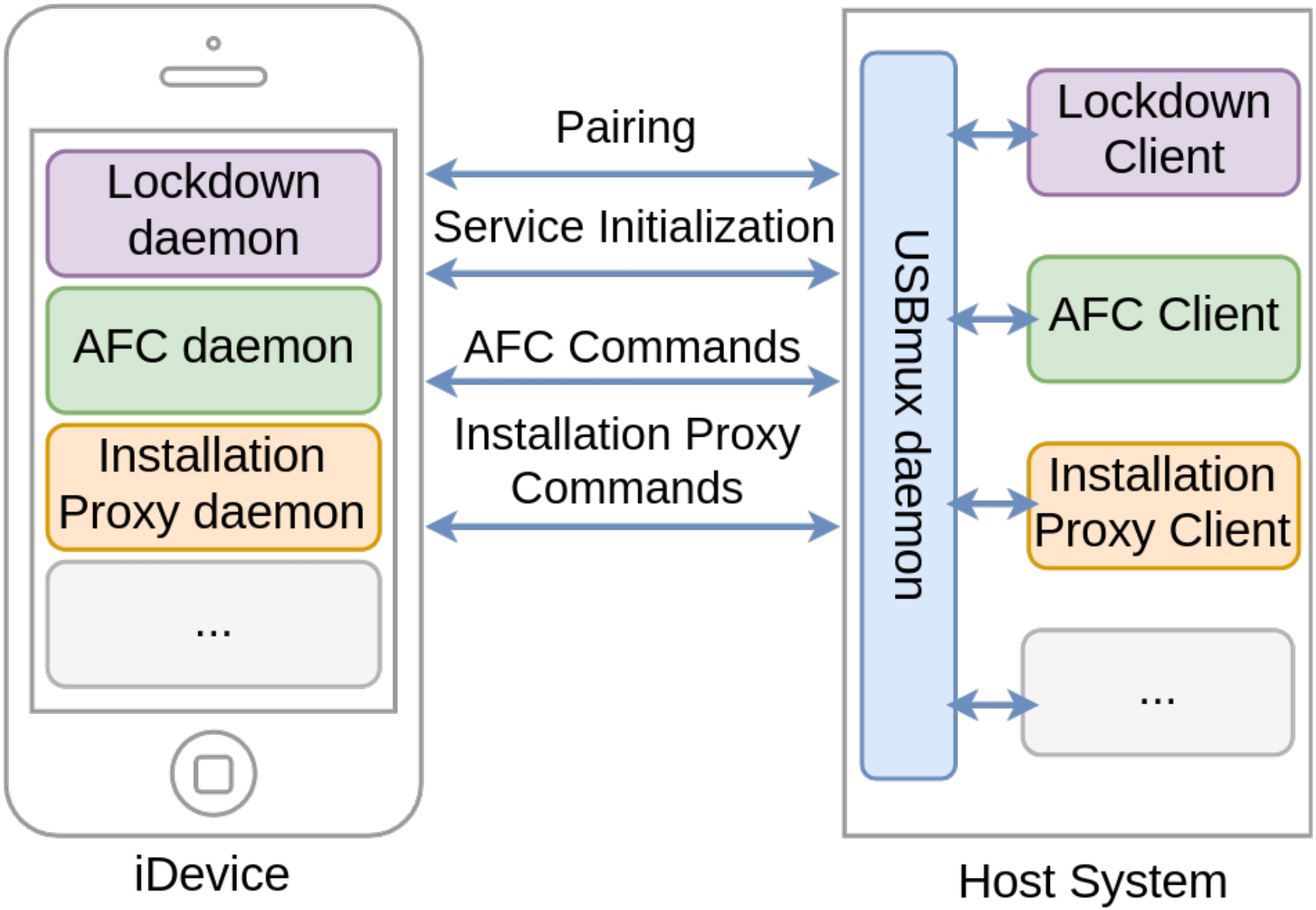
iSign

- Signs complete IPA or app bundle
- Experimental branch needed to sign binaries from scratch
- Open Source

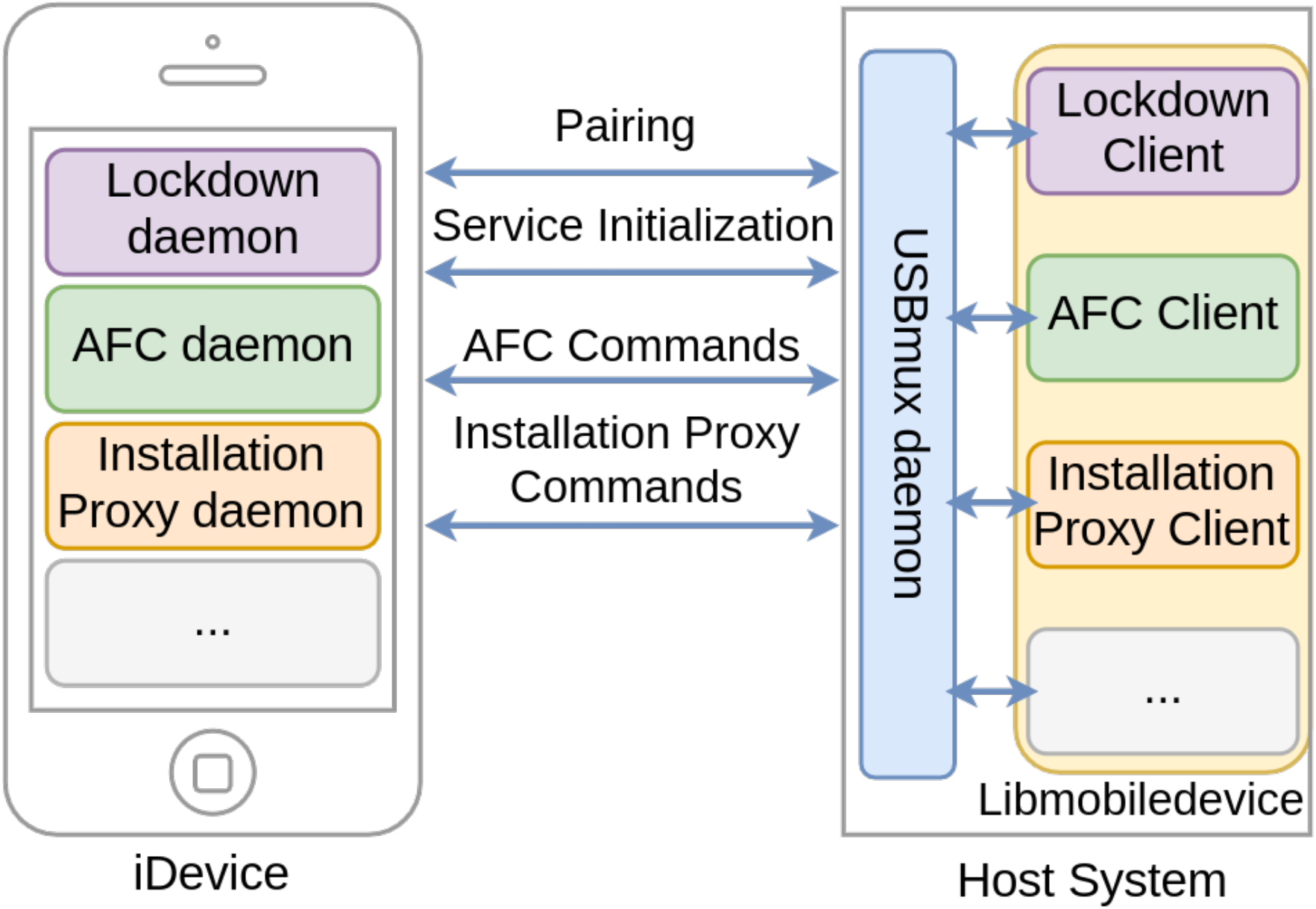
Procedure Overview



Application Deploying - Background



Application Deploying - GNU/Linux



Application Deploying - Software

Cydia Impactor

- Signs & Install IPA's
- Close Source
- GUI tool
- Entitlements do not allow app debugging

iDeviceinstaller

- Libmobiledevice Utility
- Open Source
- Command line tool

Application Deploying - Software

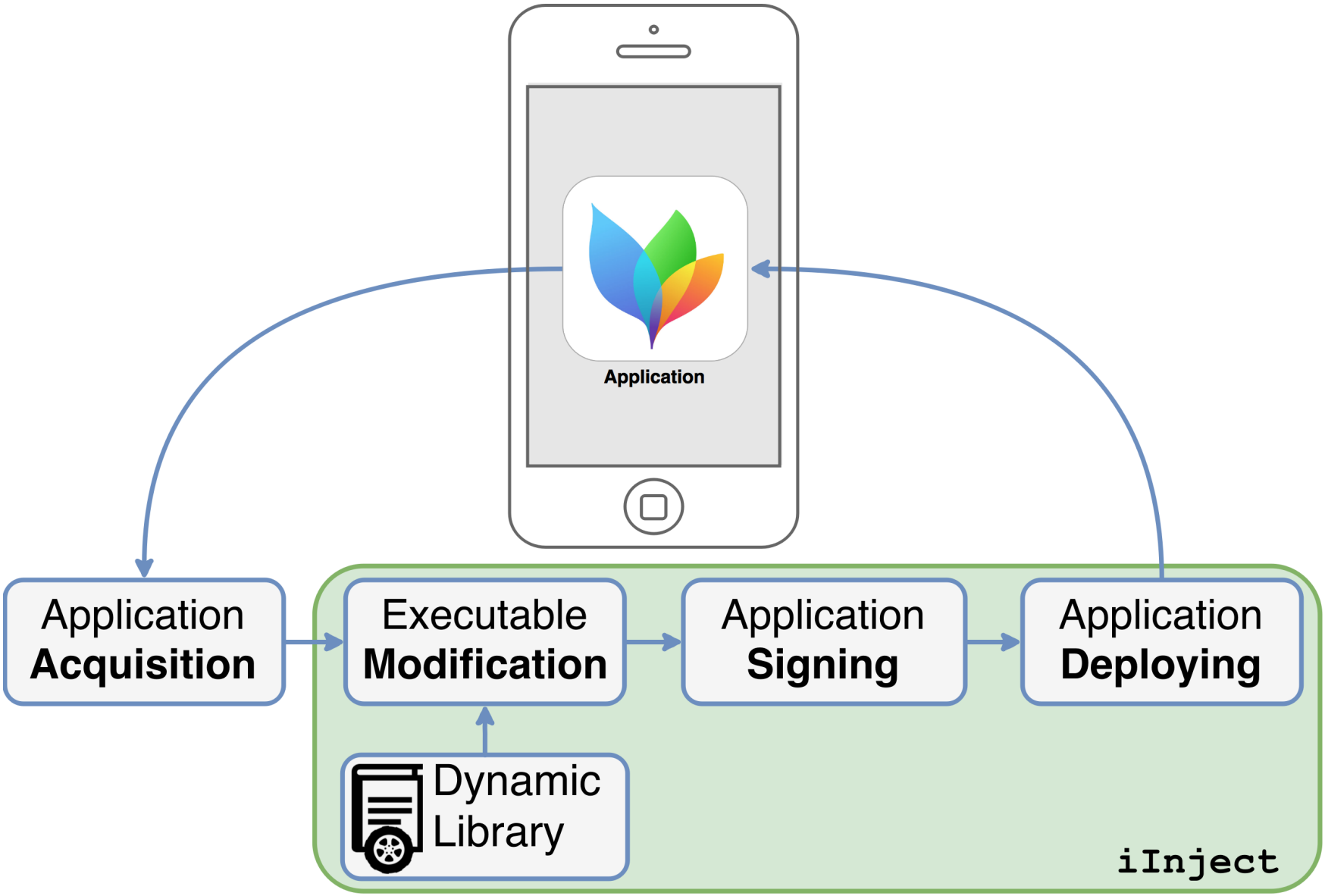
Cydia Impactor

- Signs & Install IPA's
- Close Source
- GUI tool
- Entitlements do not allow app debugging

iDeviceinstaller

- Libmobiledevice Utility
- Open Source
- Command line tool

Automation


Application
Application Acquisition
Executable Modification
Application Signing
Application Deploying
Dynamic Library
iInject

Roadmap

Application acquisition :

- Clutch usage could be automated ⇒ little value added

Provision profile generation :

- Free Apple account ⇒ automation possible, but requires deep analysis of `Xcode / Cydia`
- Paid Apple Developer account ⇒ automation possible with `Fastlane/Spaceship`

Conclusion / Discussion

It is possible to automate the embedding process in GNU/Linux using a paid Developer Account, however:

Conclusion / Discussion

It is possible to automate the embedding process in GNU/Linux using a paid Developer Account, however:

- For free Apple accounts, Xcode access is needed once per week to renew the provisioning profile
- For IPA acquisition jailbroken device needed

Conclusion / Discussion

It is possible to automate the embedding process in GNU/Linux using a paid Developer Account, however:

- For free Apple accounts, Xcode access is needed once per week to renew the provisioning profile
- For IPA acquisition jailbroken device needed
- iInject is still a proof of concept
  - iInject was tested against iOS 10.2.1 and iOS 10.3.2 (non-jailbroken)
  - iInject was tested against 9 diferent IPA's

Questions?

Try it out yourself:

https://github.com/LeanVel/iInject

Bibliography

Apple Support Community.
Macintosh virtual machine hosted by Windows.
https://discussions.apple.com/thread/5785112?tstart=0, 2014.
[Online; accessed 8-June-2017].

Android Open Source Project.
Android Security 2015 Year In Review.
https://source.android.com/security/reports/Google_Android_Security_2015_Report_Final.pdf, 2016.
[Online; accessed 7-June-2017].