# Securing applications from commit to execution for Android Apps and JAR files

James Tapsell - Supervised by: Jorge Blasco Alis

MSci in Computer Science (YINI)

## Objectives

This project looks to create a proof-of-concept defence against code injection attacks with the following features:

- Developers should not need to install new software on their own devices if possible (although this may be necessary in some cases, this should be avoided if possible).
- The releases are signed in such a way that end user devices can be sure who committed and released an application, and there should be no breaks in the chain of custody of the source or artefacts created from it.
- The releases should not be able to be tampered with in an undetectable way.
- It should not be possible to include an entity (person, group or company) in the list of people who released an application without their permission.

## Introduction

A lot of applications are now signed to prevent tampering with them, this usually consists of a created executable that is signed by the company or group releasing some software, allowing the end user or their system to verify the source of the software. The signature usually covers the executable code, but also the meta-data, and the resource files, allowing prompts to show who signed a piece of software, and stopping manipulating the resource files to, for example, swap out error messages for security issues with benign looking messages.

For some classes of software (device drivers on windows for example), this is a mandatory step, as 64 bit editions need settings changes to allow the use of unsigned drivers[1]. This was introduced to try to prevent malware from getting deep into the system, as once it is at such a layer not only does it have the ability to take over the system, but also hides itself from any software looking for it.

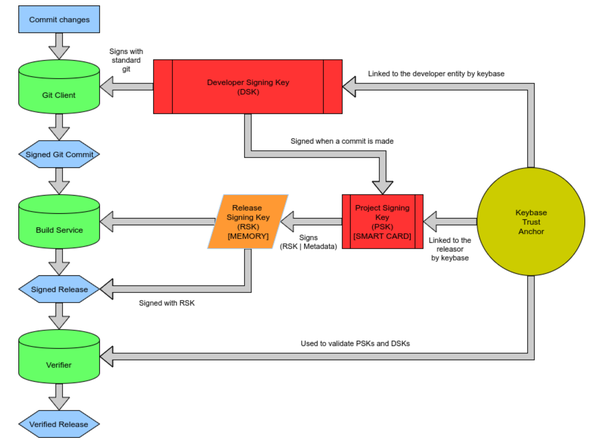
## Issues Faced

- Keybase issue
  The keybase client has a UI issue where users who only have smartcard GPG keys cannot decrypt messages sent to them.
- Keybase API issue | [2]
  The official Keybase API docs had issues getting signers for a given domain.
- Runtime JAR signing
  This is not possible using the JDK tools, so I use jarsigner, but this required making a wrapper for it.
- Git issues
  - Git inconsistencies
    Although there is an unknown key return value, git returns unsigned for unknown keys, which breaks checking.
  - Git only allows 1 signature
    So if the committer and author are different they cannot both sign.
  - JGit limitations
    JGit does not support signatures so a custom version was required.
  - Git format
    Git has many oddities in its disk representation, such as the way that pack files use a mix of big endian, litte endian, and even custom integer representations, which complicated the code and slowed down development.

## Project Outputs

- Kotlin
  At the start of this project, I had only used Kotlin at a basic level. Now I have a better understanding of it.
- Gradle
  I have learned more about build automation and testing. I had used it previously, but I have also learned to use the Kotlin gradle DSL, although I did not use if for this project, as it would require rewriting the build system for the whole project.
- LaTeX
  I have improved my ability to write LaTeX. I had previously only used LaTeXin a very basic capacity, and the documents that I had written were not very well written. Through learning LaTeXI have learnt to properly lay out my documents, and have found a set of tools which support my preffered LaTeXworkflow.
- Git
  I now have an understanding of how the internals of git work. This includes both raw and packed files. This will help not only when I work with Git, but also when I use git, as many of the limitations now make sense, as I can see why they exist, and how to work around them. Some examples of this are the restriction on multple signaturs, and the fact that which branch a commit was made to may not be known.
- Keybase
  There were 2 issues with Keybase that were found during this project:
  - The API issue.[2]
    This caused issues requesting data from the API, and was different to what the documentation said. It was solved, so the issue will no longer affect users.
  - The client issue.[3]
    This causes issues if all of a user's keys are on smartcards, I created a pull request[4] for it, but that is still open.

## Signature Design




## References

[1] Driver signing policy. https://docs.microsoft.com/en-us/windows-hardware/drivers/install/kernel-mode-code-signing-policy--windows-vista-and-later-, 2017.

[2] Api documentation issue - issue #3122 - keybase/keybase-issues. `https://github.com/keybase/keybase-issues/issues/3122`.

[3] Unable decrypt messages from web ui if keys are all stored on smart cards - issue #3126 - keybase/keybase-issues. `https://github.com/keybase/keybase-issues/issues/3126`.

[4] Improved handling of keys where the private key is not accessible by jrtapsell - pull request #10570 Âů keybase/client. `https://github.com/keybase/client/pull/10570`.

## Contact Information

- Web: https://www.jrtapsell.co.uk/project1718.html
- Email: James.Tapsell.2015@rhul.ac.uk