Reproducing iOS Vote Verification Application Builds for Estonian I-Voting System

Research Seminar in Cryptography

February 23, 2018
I-Voting Protocol

1. Authentication

2. Candidate list $L$

3. $\text{Sig}_v(\text{Enc}_{s_{pub}}(c_v, r))$

4. Vote reference $vr$

5. $r$, $vr$

(8. $c_v$)

6. $vr$ (7. $\text{Enc}_{s_{pub}}(c_v, r)$, $L$)
iOS Application


Description

Kontrollrakendus võimaldab valjel kontrollida oma hääl Eesti Vabariigi valimisel. Hääle kontrollimine võimaldab saada senisest enam teavet häälitamiseks kasutatavate arvutite turvalisuse kohta. Kui arvuti on nakatunud pavarago, mis e-hääletaja e-häält muudab või blokeerib, siis kontrollimine võimaldab selisse olukorra...
ivoteverification

iOS based vote verification application for Estonian i-voting system

The intention behind this repository is to make source code of the official i-vote verification application for Estonian internet-voting system available for public review.

The repository is not used for active development, but will be kept up to date, so the code that can be found here is the code that is used for election. As the voting system used for legally binding elections must strictly follow the legislation, the actual development of Estonian i-voting system and i-vote verification application is supervised by Estonian State Electoral Office (ESEO, www.valimised.ee). Please refer to www.valimised.ee for further information.
Reproducible Builds

How can we know that the binaries distributed are compiled using the source code that is published?

“Reproducible build process allows for independent verification that a binary matches what the source intended to produce.”

Problems:

• How to get the binary which is distributed to verifiers’ mobile devices?

• The same source code compiled using different build environments may produce slightly different binaries
IPA (iOS application archive)

$ unzip "EH kontrollrakendus 2.1.1.ipa"
  inflating: META-INF/com.apple.ZipMetadata.plist
  extracting: META-INF/com.apple.FixedZipMetadata.bin
  inflating: Payload/EH kontrollrakendus.app/_CodeSignature/CodeResources
  inflating: Payload/EH kontrollrakendus.app/Info.plist
  inflating: Payload/EH kontrollrakendus.app/EH kontrollrakendus
  extracting: Payload/EH kontrollrakendus.app/btn_close.png
  inflating: Payload/EH kontrollrakendus.app/btn_close@2x.png
  extracting: Payload/EH kontrollrakendus.app/config.txt
  inflating: Payload/EH kontrollrakendus.app/Default-568h@2x.png
  inflating: Payload/EH kontrollrakendus.app/Default.png
  inflating: Payload/EH kontrollrakendus.app/Default@2x.png
  inflating: Payload/EH kontrollrakendus.app/eh.valimised.ee.der.cer
  inflating: Payload/EH kontrollrakendus.app/ESTEID-SK_2015.pem.crt
  inflating: Payload/EH kontrollrakendus.app/HelpViewController.nib
  inflating: Payload/EH kontrollrakendus.app/PkgInfo
  extrating: Payload/EH kontrollrakendus.app/SC_Info/EH kontrollrakendus.supf
  inflating: Payload/EH kontrollrakendus.app/SC_Info/EH kontrollrakendus.supp
  inflating: Payload/EH kontrollrakendus.app/SC_Info/EH kontrollrakendus.supx
  inflating: Payload/EH kontrollrakendus.app/SC_Info/Manifest.plist
  inflating: Payload/EH kontrollrakendus.app/ScannerViewController.nib
  inflating: Payload/EH kontrollrakendus.app/spinner.png
  inflating: Payload/EH kontrollrakendus.app/spinner@2x.png
  inflating: Payload/EH kontrollrakendus.app/VerificationResultCandidateCell.nib
  inflating: Payload/EH kontrollrakendus.app/VoteVerificationResultsViewController.nib
  inflating: iTunesMetadata.plist
  inflating: Payload/EH kontrollrakendus.app/SC_Info/EH kontrollrakendus.sinf
  inflating: iTunesArtwork

$ file "Payload/EH kontrollrakendus.app/EH kontrollrakendus"
  Mach-O universal binary with 2 architectures: [arm_v7: Mach-O arm_v7 executable] [64-bit architecture=12]
iOS App Binary

- Install iTunes 12.6.3 (newer versions lack the feature) [https://support.apple.com/en-us/HT208079](https://support.apple.com/en-us/HT208079)
- Download “EH kontrollrakendus” from App Store
- Copy .ipa file from:
  ![的形象](image.png)
  
- Run Clutch inside jailbroken iOS device to decrypt binary [https://github.com/KJCracks/Clutch/wiki/Tutorial](https://github.com/KJCracks/Clutch/wiki/Tutorial)
- Alternative: Cybernetica is providing a non-encrypted .xcarchive that was *allegedly* uploaded to iOS App Store
Tasks

- Build application package from the source code
  - Study differences in the binaries
  - Adjust the build process to eliminate differences
  - Repeat until binary package build is reproduced fully
- Decrypt .ipa binary and check if it matches the binary provided by Cybernetica (jailbroken device needed)
- Describe the process in the report
- Provide recommendations for making the build process more reproducible-friendly
- The reproduction of Android VVA has been done:

Challenges of reproducible builds for Signal iOS app:
https://github.com/signalapp/Signal-iOS/issues/641