

Cothority Mobile: a mobile application to perform distributed tasks using the cothority-framework



Bachelor Project – Spring 2017

Romerio Lucio

Index

- Project Overview and Motivation
- Introduction
- Architecture
- Implementation
- Results Analysis
- Future work
- Demo

Project Overview

- Develop a Mobile Application using a cross-platform framework
- Implement a ProtoBuf library
- Implement a Cryptographic library

Project Motivation

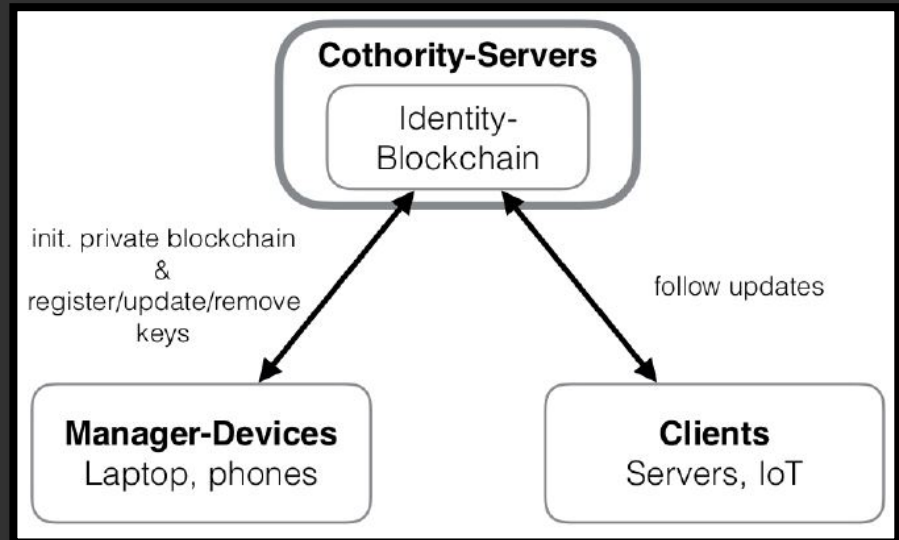
- Existing application uses JSON-interface
- Make application available both for Android and iOS
- Libraries needed to implement Cothority services

Introduction - PhoneGap

- CSS3, HTML5 and JavaScript
- Hybrid application:
 - Access to most of the native APIs
 - Layout rendered via web views
- PhoneGap Build

Introduction - CISC

- Add device to access-control-list
- Make proposition
- Vote proposition

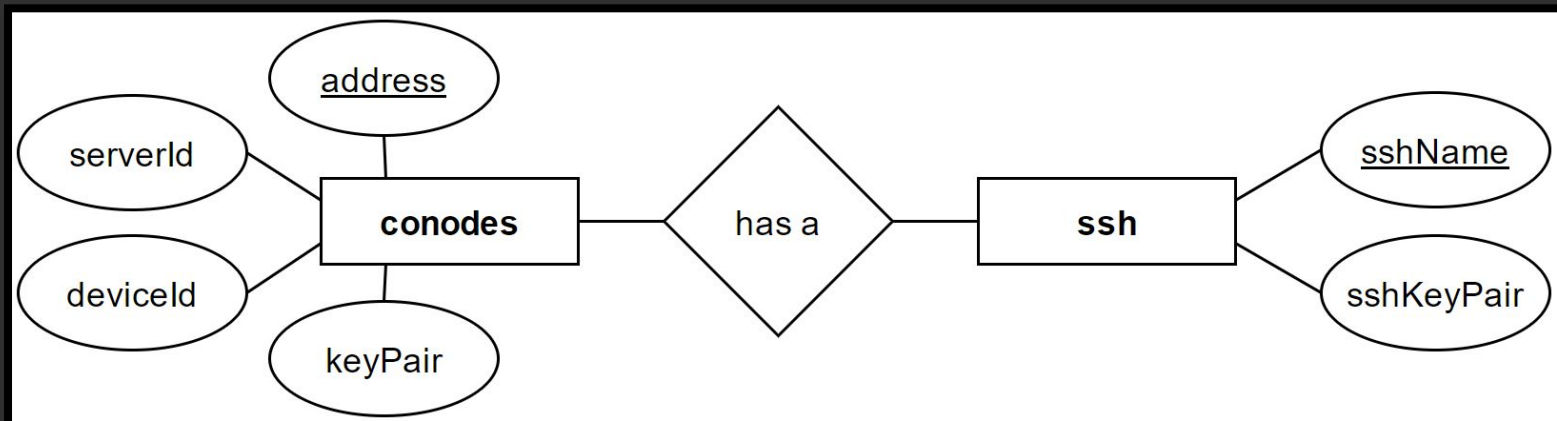


Architecture - Libraries

- Implementation coordinated by Gaylor
 - Cothority ProtoBuf
 - CryptoJS
- Extended to satisfy needs of my application
 - CISC messages
 - hashConfig and SchnorrSign

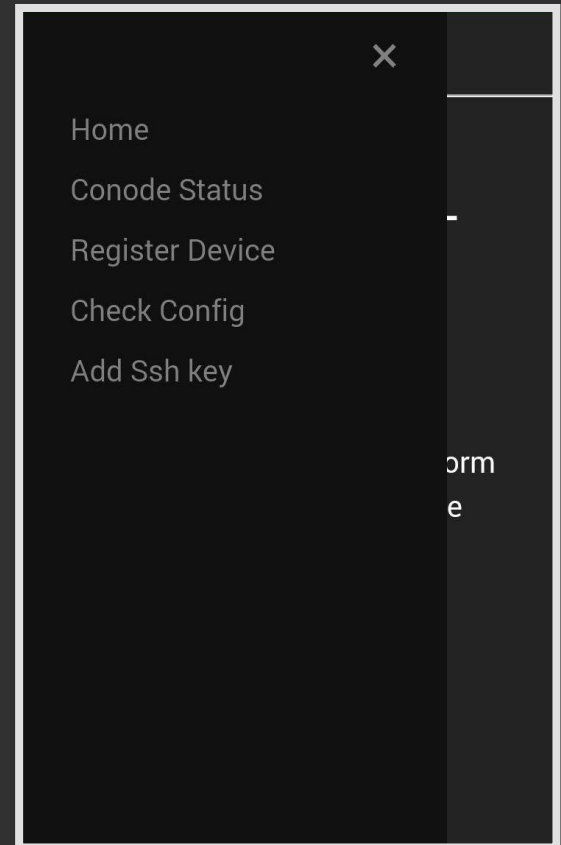
Architecture - Database

- Need to store:
 - Private keys
 - Conode information



Architecture - Setup

- Create PhoneGap project
- Import libraries
- Sidebar

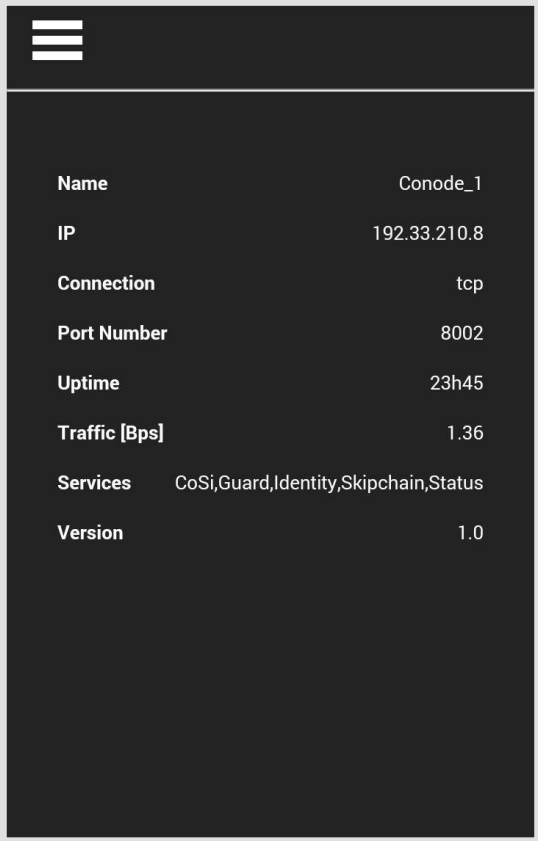


Implementation - Database

- API described into PhoneGap Documentation
- Traditional SQL statements
- Asynchronous task: handlers

Implementation - Conode Status

- Test device - conodes communication
- Procedure:
 1. Send empty message
 2. Show conode status to the user



A screenshot of a terminal window displaying the status of a conode. The window has a dark background with white text. At the top left, there is a hamburger menu icon (three horizontal lines). The status information is presented as a list of key-value pairs:

Name	Conode_1
IP	192.33.210.8
Connection	tcp
Port Number	8002
Uptime	23h45
Traffic [Bps]	1.36
Services	CoSi,Guard,Identity,Skipchain,Status
Version	1.0

Implementation - WebSocket

- Reuse same connection
- Send message: asynchronous task
- Use handlers

Implementation - Register Device

1. Scan QR-code encoding conode ID
2. Parse scanned string
3. Create and send proposition
4. Show informative message

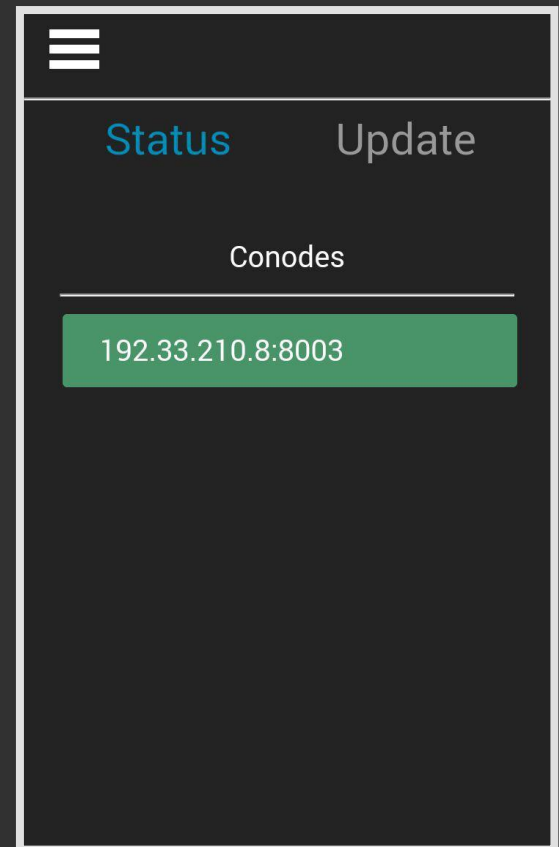
```
"threshold": 2,  
"device": {  
  "icsil1-conode1": {  
    "point": /*public key*/  
  }  
},  
"data": {  
  "ssh:icsil1-conode1:test1": /*ssh key*/  
}
```

Implementation – Key Idea

- One html element for each phase
- Change elements style programmatically
- Wrap related JavaScript in a single file

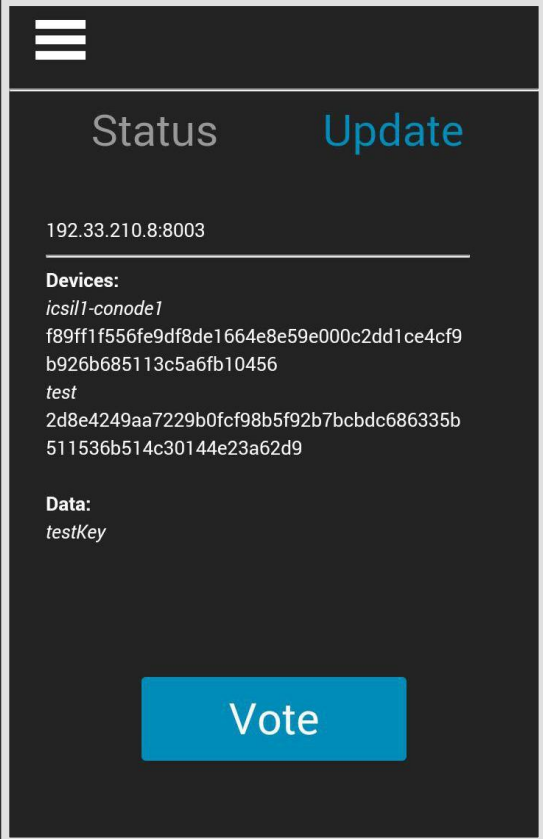
Implementation – Check Config

- Tab menu
- Phases
 - Conode list
 - Send correct message
 - Display Config
 - Vote (if update)



Implementation - Vote

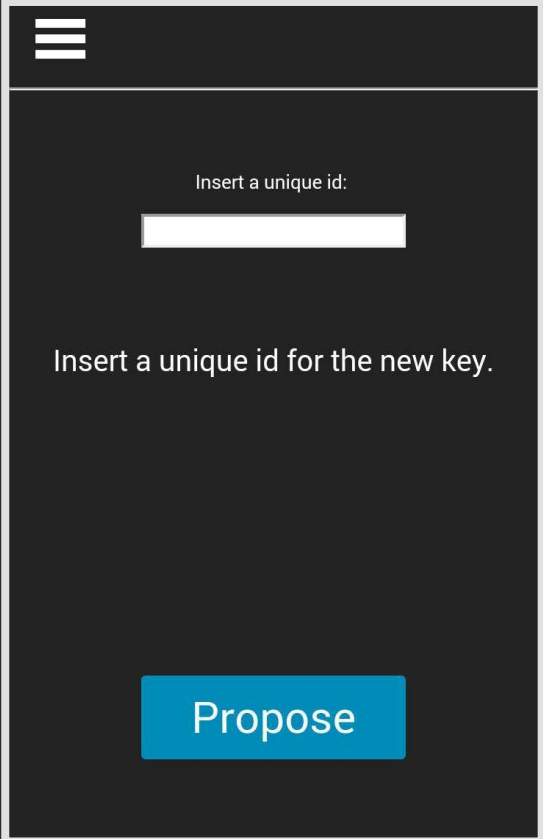
- Procedure:
 1. Compute Config hash
 2. Sign resulting hash
 3. Send ProposeVote message
- Use theCryptoJS library



The screenshot shows a mobile application interface with a dark background. At the top left is a hamburger menu icon. Below it, the words "Status" and "Update" are displayed in a light blue font. A horizontal line separates this header from the main content area. The main content area displays the IP address "192.33.210.8:8003". Below this, the word "Devices:" is followed by three lines of device identifiers: "icsil1-conode1", "f89ff1f556fe9df8de1664e8e59e000c2dd1ce4cf9b926b685113c5a6fb10456", and "test". Below these, the word "Data:" is followed by two lines of data: "2d8e4249aa7229b0fcf98b5f92b7bcbdc686335b" and "511536b514c30144e23a62d9". At the bottom of the screen is a large blue button with the word "Vote" in white text.

Implementation - Register SSH

1. List conodes
2. Send ConfigUpdate
3. Create new key pair
4. Make and vote proposition
5. Show informative message



The image shows a mobile application interface for registering a new SSH key. At the top left, there is a hamburger menu icon. The main content area contains the text "Insert a unique id:" followed by a white text input field. Below the input field, there is a message: "Insert a unique id for the new key." At the bottom of the screen, there is a blue button with the text "Propose".

Results Analysis - Encountered Issues

- React and ES6
- From PoP to CISC
- Server not sending error messages

Results Analysis - Limitations

- Vanilla PhoneGap:
 - Only script tag to import files/libraries
 - No official automated testing tool
- Mobile application:
 - Stuck in case of server error
 - Cannot create SSH keys yet
 - Layout compromise

Future work

- Handle server errors
- Allow user to create SSH keys
- Test PoP messages
- Extend the two libraries
- Extend the application with other services

DEMO