Provisioning IoT Devices

LF Open Networking Summit ‘19

Cagdas Doner
Senior Software Engineer

cagdas.doner@airties.com
About Provisioning & IoT

Known Facts
Why Provisioning is Needed?
Provisioning Challenges
Known Facts

- IoT devices are mostly headless
  - Lack of I/O peripherals
    - No screen, keyboard etc.
  - No built-in software/UI for user interaction

- Network access needed
  - Wireless connectivity
    - Wi-Fi, LPWAN, ...
  - Cloud connectivity
    - Messaging
    - Data analysis, storage, etc.
Why Provisioning is Needed?

● First-time configuration is a must
  ○ Network credentials
    ■ Wi-Fi SSID and Password
    ■ LoraWAN OTA-Activation data
  ○ User/brand specific data
    ■ Location
      ● Country regulations
    ■ Customer info
      ● Customer ID, contact, etc.
Provisioning Challenges

● Configuration without a UI
  ○ Need a configurator/companion device
  ○ Lack of feedback on operation status

● User experience is crucial on initial setup
  ○ Providing seamless connectivity
  ○ Considering non-technical users

● Secure data transfer

● Standardization
Technical Info Break

Wi-Fi Modes of Operation
Wi-Fi Modes of Operation

- **Station**
  - The client

- **Access Point (AP)**
  - a.k.a Master
  - Creates network with an SSID

- **Monitor**
  - Ability of capturing (sniffing) network packets [1]
  - No need of association
  - Captured data packets might be encrypted
Provisioning in Access Point (AP) Mode

AP Mode Configuration
Captive Portal
Problems with AP Mode
AP Mode Configuration

● IoT device in AP Mode
  ○ Creates a network with public SSID

● Configurator Device as Station
  ○ Mobile phone or PC
  ○ Associates with IoT device’s network
  ○ Custom App for credential transfer
  ○ Raw TCP/UDP or HTTP/Websocket

● Captive portal is applicable
  ○ Serves a customized lightweight web page
  ○ DNS rule manipulation
Captive Portal (Well-known)
Captive Portal (ESP8266)

AutoConnectAP

WiFiManager

Configure WiFi

Configure WiFi (No Scan)

ROBIN HOOD 30%
Tia Network 2 16%

ESP8266 Captive Portal Library [2]
Problems with AP Mode

● Network connectivity change
  ○ Need to switch between networks
  ○ Custom apps may lose data
    ■ Non-user friendly error cases

● Station behaviours
  ○ Disassociation when no Internet access
  ○ Captive portal launch differences

● No password for ease of connection
  ○ Security flaw
Technical Info Break

MAC Address Types
MAC Frame Structure
MAC Address Types

- **Broadcast**
  - FF:FF:FF:FF:FF:FF
- **Multicast**
  - 01:00:5E:XX:XX:XX
- **Unicast**
  - XX:XX:XX:XX:XX:XX
MAC Frame Structure

<table>
<thead>
<tr>
<th>bytes</th>
<th>7</th>
<th>1</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>46 - 1500</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>Destination</td>
<td>Source</td>
<td>Length</td>
<td>LLC</td>
<td>Data Unit</td>
<td>Pad</td>
<td>FCS</td>
</tr>
</tbody>
</table>

IEEE 802.3 Frame
“Smart-ization” on Provisioning

Smart Configuration
Data Encoding
Difficulties
Smart Configuration

● IoT device in Monitor Mode
  ○ Captures wireless packets
● Configurator broadcasts credential data
  ○ Without changing network
  ○ App/UI for target SSID and password
  ○ A clever encoding method based on;
    ■ packet length differences
    ■ destination addresses
● Layer 2 Data frames
● Varying vendors
  ○ TI[3], Espressif[4], MTK, Realtek, MXCHIP
Data Encoding

<table>
<thead>
<tr>
<th>Mode</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:01:49:6f</td>
<td>43</td>
</tr>
<tr>
<td>DMA</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:02:54:36</td>
<td>43</td>
</tr>
<tr>
<td>DMA</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:03:36:36</td>
<td>43</td>
</tr>
<tr>
<td>DMA</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:04:37:38</td>
<td>43</td>
</tr>
<tr>
<td>DMA</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:05:39:cc</td>
<td>43</td>
</tr>
<tr>
<td>DPL</td>
<td>00:90:4c:17:1a:9b</td>
<td>FF:FF:FF:FF:FF:FF</td>
<td>47</td>
</tr>
<tr>
<td>DPL</td>
<td>00:90:4c:17:1a:9b</td>
<td>FF:FF:FF:FF:FF:FF</td>
<td>47</td>
</tr>
<tr>
<td>DPL</td>
<td>00:90:4c:17:1a:9b</td>
<td>FF:FF:FF:FF:FF:FF</td>
<td>96</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:01:01:01</td>
<td>556</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:02:02:02</td>
<td>555</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:03:03:03</td>
<td>554</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:04:04:04</td>
<td>291</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:05:05:05</td>
<td>338</td>
</tr>
<tr>
<td>Hybrid</td>
<td>00:90:4c:17:1a:9b</td>
<td>01:00:5e:06:06:06</td>
<td>198</td>
</tr>
</tbody>
</table>
Difficulties

- Packets are visible but WPA protected
  - Additional shared key inclusion
- Preamble
  - Sync code
  - Obtaining encryption padding on packet size
- Security
  - Weak on eavesdropping
  - Fixed key for encryption [5]
Standard Methods for Provisioning

Wi-Fi Protected Setup (WPS)
Device Provisioning Protocol (DPP)
WPS

- Target AP as Registrar
- IoT device as Enrollee
  - The Magic Button
    - Information elements for discovery (in a timely manner)
    - EAP key exchange
    - IoT Device need to have WPS button
    - Need to have physical access to AP
  - 8-digit PIN
    - No interface on IoT device for PIN input
    - Vulnerabilities were fixed
DPP

- Supported by Wi-Fi Alliance
  - Wi-Fi Easy Connect [6]
- More secure replacement for WPS
- Layer 2 Management frames
- QR code read via Configurator device
  - Public key (256-bit, ECC) [7]
  - MAC Address
- Non-certified methods applicable
  - Cloud supported provisioning
  - Public key and MAC on Information Element
Provisioning on Other Mediums

- Acoustic Configuration
- Optical Configuration
Acoustic Configuration

- Amazon Dash Button
  - Credentials are encoded in acoustic output
    - Mobile phone as configurator
      - Provisioning data modulated as sound wave
      - Outputs via speaker
  - I2S powered microphone on device
  - Multi-level binary FSK
  - Carrier frequency is 19KHz [8]
    - Almost out of human audio spectrum
  - Converting decoded data into ASCII
Optical Configuration

● BlinkUp! by ElectricImpl [9]
  ○ Data encoded on visible light
  ○ IoT device has a light sensor
  ○ Mobile phone as configurator
    ■ Uses brightness changes of screen backlight
    ■ Amplitude based Modulation
    ■ Have several brightness levels
      ● Could be affected by ambient light
  ○ Patent Protected [10]
A New Alternative

AirTies

Mobile App
Device Implementation
Data Encoding
Demo
Problems
A New Alternative

● Based on Visible Light

● Mobile phone as configurator
  ○ Includes a configurator app
  ○ Uses camera flashlight beams for data transfer

● IoT device with a light sensor
  ○ Receives encoded light patterns
  ○ Decodes depended on modulation scheme
Mobile App

● Powered by PhoneGap
  ○ Hybrid, both for IoS and Android
  ○ Uses flashlight native plug-in

● Input fields for Wi-Fi SSID and Pass

● Manipulates flashlight in given time-slots
  ○ Depended on encoded SSID and Pass
  ○ Basically switches on and off
Device Implementation

- Currently supported on 2 devices
  - ESP8266
  - Raspberry (WIP)
- Powered with an analogue light sensor
  - Raspberry with an ADC
  - Sensor readings are collected in timeslots
  - Collected data is converting into ASCII
  - Each time-slots represents 1-bit
Data Encoding

- On-Off Keying (OOK) modulation based
  - Slower but reliable
  - Current flashlight time-slot is 250 msec
    - 1 byte in 2 seconds
    - Need an extra delay to cover sensor/flash delays!
  - Encoding
    - Data payload starts after 1-byte magic preamle: 0xFF
    - SSID and pass is transferred in the same sequence
    - SSID and pass are separated by «null» char
Demo

- Runs on ESP8266
- Android app as configurator
- Led states as status indicator
- Transferring SSID and password
- Network association

Library: https://github.com/cagdasdoner/Modulazy
Problems

● Preemption
  ○ Flashlight might be interrupted by other tasks
  ○ Also sensor readings are likewise
    ■ Bare-metal devices with non-OS will be OK
  ○ Sync bits or preamble usage

● Time Usage
  ○ Totalling SSID and Pass length is 50 + 64 bytes
  ○ To decrease time-slots, need sync preambles
  ○ Pass filter might be applied
## Conclusions

<table>
<thead>
<tr>
<th>Method</th>
<th>User Experience</th>
<th>Security</th>
<th>Ease of Use</th>
<th>Cost</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP Config</td>
<td>Bad</td>
<td>Weak</td>
<td>Hard</td>
<td>Cheapest</td>
<td>No</td>
</tr>
<tr>
<td>Smart Config</td>
<td>Good</td>
<td>Weak</td>
<td>Easy</td>
<td>Cheaper</td>
<td>Vendor specific</td>
</tr>
<tr>
<td>WPS</td>
<td>Best</td>
<td>Strong</td>
<td>Easiest</td>
<td>Cheaper</td>
<td>WFA</td>
</tr>
<tr>
<td>DPP</td>
<td>Best</td>
<td>Strongest</td>
<td>Easier</td>
<td>Cheaper</td>
<td>WFA</td>
</tr>
<tr>
<td>Other Mediums</td>
<td>Good</td>
<td>Unexpectable</td>
<td>Easy</td>
<td>Expensive</td>
<td>No</td>
</tr>
</tbody>
</table>
Thank You!

AirTies

www.airties.com
References

2. https://github.com/tzapu/WiFiManager
9. https://www.youtube.com/watch?v=zbhu7Mwicok