Karl Koscher – @supersat
Eric Butler – @codebutler

The Secret Life of SIM Cards

Rev: July 8, 2013
Updated slides available from defcon.org and http://simhacks.github.io/
In an alternate reality...

- Applications live on your SIM card
- Phones are dumb hosts – UI and connectivity only
- Telcos own the SIMs, so they control the applications
Actually...

- These are called SIM Toolkit (STK) applications
- Used widely in the developing world
  - Mobile banking, etc.
- Very little documentation on how they work, or how to develop them
An opportunity!

- Toorcamp 2012 had its own GSM network
- SIM cards supported SIM Toolkit, so why not explore it?
- After lots of research, finally figured out how to program the *#$!ing things
- Learn from our misery
Why should you care?

- SIM cards are a lot more powerful than you might have imagined
- Development of STK apps is the same as Java Cards – you can use unlocked STK SIMs as cheap smart card development platforms
- Maybe if people care enough, SIMs can be better utilized (e.g. secure storage of SSH keys, BitCoins, phone decryption keys)
Okay, what can an STK app do?

- Rudimentary UI – display text, menus, play tones, read input
- Send SMSes, initiate calls, initiate and use data services
- Receive and act on events, such as call connected, call disconnected, etc.
- Interact with the rest of the SIM card
- Run arbitrary AT commands on the phone
Technologies involved

- Working our way up the technology stack:
  - Smart cards
  - Java Card
  - GlobalPlatform
  - SMS
  - SIM Toolkit API
Smart Card Standards

- ISO 7816-1: Physical characteristics
- ISO 7816-2: Electrical contacts
- ISO 7816-3: Electrical interface, Transmission Protocol Data Units (TPDUs)
  - T=0: Byte-oriented protocol
  - T=1: Block-oriented protocol
- ISO 7816-4: Standard commands, Application Protocol Data Units (APDUs)
- ISO 14443-4: “T=CL”: APDUs over RFID

## Smart Card Protocols: ISO 7816-4

### Command APDU

<table>
<thead>
<tr>
<th><strong>CLS</strong></th>
<th><strong>INS</strong></th>
<th><strong>P1</strong></th>
<th><strong>P2</strong></th>
<th><strong>LC</strong></th>
<th><strong>DATA</strong></th>
<th><strong>LE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Instruction</td>
<td>Param 1</td>
<td>Param 2</td>
<td>Data Length</td>
<td>Length Expected</td>
<td></td>
</tr>
<tr>
<td>MSB</td>
<td>LSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **8**  Standard structure and CLS/INS
- **1-7** RFU
- **8-9** Standard structure, custom INS, standard CLS
- **A**  Standard unless spec’d by context
- **B-C** Standard structure, custom CLS/INS
- **D-F** Custom

### Optional Command Dependent

<table>
<thead>
<tr>
<th>B&lt;sub&gt;3&lt;/sub&gt;</th>
<th>B&lt;sub&gt;2&lt;/sub&gt;</th>
<th>B&lt;sub&gt;1&lt;/sub&gt;</th>
<th>B&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Meaning when CLS=0X,8X,9X,AX</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Secure Messaging Format</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td>No 7816-4 SM</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>No SM</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>Proprietary SM</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td>7816-4 SM</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>Command header not auth’d</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Command header auth’d</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>Logical chan num</td>
</tr>
</tbody>
</table>

[http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx](http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx)
Smart Card Protocols: ISO 7816-4

Command APDU

<table>
<thead>
<tr>
<th>INS</th>
<th>(CLS=0X, AX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0E</td>
<td>ERASE BINARY</td>
</tr>
<tr>
<td>20</td>
<td>VERIFY</td>
</tr>
<tr>
<td>70</td>
<td>MANAGE CHANNEL</td>
</tr>
<tr>
<td>82</td>
<td>EXTERNAL AUTHENTICATE</td>
</tr>
<tr>
<td>84</td>
<td>GET CHALLENGE</td>
</tr>
<tr>
<td>88</td>
<td>INTERNAL AUTHENTICATE</td>
</tr>
<tr>
<td>A4</td>
<td>SELECT FILE</td>
</tr>
<tr>
<td>B0</td>
<td>READ BINARY</td>
</tr>
<tr>
<td>B2</td>
<td>READ RECORD(S)</td>
</tr>
<tr>
<td>C0</td>
<td>GET RESPONSE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INS</th>
<th>(CLS=0X, AX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>ENVELOPE</td>
</tr>
<tr>
<td>CA</td>
<td>GET DATA</td>
</tr>
<tr>
<td>D0</td>
<td>WRITE BINARY</td>
</tr>
<tr>
<td>D2</td>
<td>WRITE RECORD</td>
</tr>
<tr>
<td>D6</td>
<td>UPDATE BINARY</td>
</tr>
<tr>
<td>DA</td>
<td>PUT DATA</td>
</tr>
<tr>
<td>DC</td>
<td>UPDATE DATA</td>
</tr>
<tr>
<td>E2</td>
<td>APPEND RECORD</td>
</tr>
</tbody>
</table>

Optional Command Dependent

INS (Any class)

<table>
<thead>
<tr>
<th>INS</th>
<th>(Any class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6X</td>
<td>Prohibited</td>
</tr>
<tr>
<td>9X</td>
<td>Prohibited</td>
</tr>
</tbody>
</table>

http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx
## Response APDU

<table>
<thead>
<tr>
<th>SW1</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>9X</td>
<td>Success</td>
</tr>
<tr>
<td>90</td>
<td>Success – no additional info</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SW1</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>SW2 bytes still available</td>
</tr>
<tr>
<td>62</td>
<td>NVM unchanged</td>
</tr>
<tr>
<td>63</td>
<td>NVM changed</td>
</tr>
<tr>
<td>64</td>
<td>NVM unchanged</td>
</tr>
<tr>
<td>65</td>
<td>NVM changed</td>
</tr>
<tr>
<td>66</td>
<td>Security issue</td>
</tr>
<tr>
<td>67</td>
<td>Wrong length</td>
</tr>
<tr>
<td>68</td>
<td>Functions in CLS not supported</td>
</tr>
<tr>
<td>69</td>
<td>Command not allowed</td>
</tr>
<tr>
<td>6A</td>
<td>Wrong parameter</td>
</tr>
<tr>
<td>6B</td>
<td>Wrong parameter</td>
</tr>
<tr>
<td>6C</td>
<td>Wrong Le: SW2 is exact length</td>
</tr>
<tr>
<td>6D</td>
<td>INS not supported or invalid</td>
</tr>
<tr>
<td>6E</td>
<td>CLS not supported</td>
</tr>
<tr>
<td>6F</td>
<td>Other/unknown</td>
</tr>
</tbody>
</table>
### Smart Card Protocols

- **Example:** SELECT 1234

<table>
<thead>
<tr>
<th>CLS</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>LC</th>
<th>DATA</th>
<th>LE</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>A4</td>
<td>08</td>
<td>08</td>
<td>02</td>
<td>12 34</td>
<td></td>
</tr>
</tbody>
</table>

- **Response:**

<table>
<thead>
<tr>
<th>DATA</th>
<th>SW 1</th>
<th>SW 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional</td>
<td>Status Word 1</td>
<td>Status Word 2</td>
</tr>
<tr>
<td>02 12 34</td>
<td>98</td>
<td>88</td>
</tr>
</tbody>
</table>
Java Card

- It’s Java!
- ... not really.
  - No garbage collection
  - No chars, no strings, no floats, no multi-dim arrays
  - ints are optional
  - No standard API, no threads, etc.
  - But there are Exceptions!

- Instance and class variables are saved in EEPROM, which has limited write cycles
Two mandatory methods:

- `install (static)` – creates your application object and registers it with the card manager

- `process` – handles APDUs sent to the card
  - Byte array in, byte array out
  - Not all bytes are immediately available due to packing of APDUs in TPDUs!
package com.degdeg.HelloCard;

import javacard.framework.*;

public class HelloApplet extends Applet {
    private static byte[] msg = { 'H', 'e', 'l', 'l', 'o' };

    public static void install(byte[] bArray, short bOffset, byte bLength) {
        HelloApplet applet = new HelloApplet();
        applet.register();
    }

    public void process(APDU apdu) throws ISOException {
        byte[] buf = apdu.getBuffer();

        if (buf[ISO7816.OFFSET_CLA] != 0x80)
            ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

        if (buf[ISO7816.OFFSET_CLA] != 0xA5)
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);

        Util.arraycopyNonAtomic(msg, (short)0, buf, ISO7816.OFFSET_CDATA, (short)msg.length);
        apdu.setOutgoingAndSend(ISO7816.OFFSET_CDATA, (short)msg.length);
    }
}
Building Java Card Apps

- There are specialized IDEs for this, but you can do without.
- Download the Java Card Development Kit from Oracle (it’s free)
- If you’re using Eclipse, remove the JRE system library and add the Java Card library
You must target Java 1.1 bytecode! 1.3 source code compatibility is okay.

After you have your .class files, you need to convert them to Java Card bytecode
  ▪ Use the converter tool in the SDK
  ▪ Need to specify application ID (more on this in a minute), API export directory, etc.

Now you have a CAP file, which is a ZIP archive of CAP components, which define your app
Building Java Card Apps

- Eclipse settings:

![Eclipse settings for STKChecker](image1)

![Eclipse settings for STKChecker](image2)
Building Java Card Apps


- java -jar ../sim-tools/javacard/bin/converter.jar -exportpath ../sim-tools/javacard/api21_export_files -applet 0xde:0xfc:0x09:0x20:0x13:0x01 com.degdeg.HelloCard.HelloApplet com.degdeg.HelloCard 0xde:0xfc:0x09:0x20:0x13 1.0

- We also have Makefiles for your convenience!
  - http://simhacks.github.io
Cards support multiple applications

Applications are selected by their AID
  - These are officially allocated, but you can make one up

There is an app on the card that is the *card manager* – use it to load your app

The card manager is defined by the GlobalPlatform spec
At this point, you might be able to use a tool like GPShell to load your app

- No such luck on our SIM cards

- Time to dig in to the GlobalPlatform specs!
GlobalPlatform

- All apps are loaded and authorized by the *Issuer Security Domain* – in practice this means that you can’t load apps onto a card you didn’t issue yourself :(

- On pure GlobalPlatform cards, the ISD is the default app on pre-personalized cards
  - Accessing it on our SIM cards is a lot harder
Installing an app is a two-step process:
- Load the binary
- Instantiate the app

Loading an app first requires authorization through the INSTALL for load command

The individual CAP components are concatenated together and sent in blocks with LOAD
**GlobalPlatform**

- **INSTALL for load:**
  
  80 E6 02 00 12 07 F0 F1 F2 F3 F4 F5  
  01 00 00 06 EF 04 C6 02 05 00 00

- **LOAD:**
  
  80 E8 00 00 6C C4 81 F7 01 00 11 DE  
  CA FF ED 01 02 04 00 01 07 F0 F1 F2  
  F3 F4 F5 01 . . .

- **LOAD:**
  
  80 E8 00 01 6C . . .

- **LOAD:**
  
  80 E8 80 02 20 . . .
To instantiate an app, issue the INSTALL for install command

There are THREE AIDs involved:

- Application AID – associated with the load file
- Module AID – associated with the main class
- Instance AID – used to select a particular instance
INSTALL for install and make selectable:

```
80 E6 0C 00 36 07 F0 F1 F2 F3 F4 F5
01 08 F0 F1 F2 F3 F4 F5 01 01 01 08 F0
F1 F2 F3 F4 F5 01 01 01 00 18 EF 14
C8 02 05 00 C7 02 00 00 CA 0A 01 00
FF 00 10 01 00 00 00 00 00 C9 00 00
```
At install time, you can specify:

- Initialization data
- App parameters
  - STK uses these extensively
    - Privileges
    - Priority
    - Number of menu items
    - Max menu item size
GlobalPlatform

- Other fun commands:
  - List AIDs, including both modules and instances
  - Delete AIDs
    - You MUST delete instances before deleting the executable!
    - You MUST delete old AIDs before reusing them!

- The spec is freely available
The SIMalliance has a free tool called the SIM Alliance Loader that can be used to program SIMs:

- It didn’t work out of the box on our SIM cards
- It is clearly designed for experts
- If you know how to configure it, and you use Windows, it will work with our SIM cards
Dealing with $#&!ing SIM cards

- The only way to talk to the SIM’s ISD is through the over-the-air update mechanism
  - i.e. SMS packets
  - Can also be used to send arbitrary APDUs!

- We don’t have to actually send SMSes, but we need to generate commands to the card with SMS packets
Turtles all the way down (GSM 03.48)

- CAT ENVELOPE (A0 C2)
  - SMS-PP Download (D1)
    - Device Identities
    - SMS-TPDU (GSM 03.40)
      - Header
      - User Data
        - Header
        - Command Packet
          - Header (Security parameters, app selection)
            - Uses a 3 byte TAR ID
              - Holy shit powerpoint supports this much nesting
                - This is the actual limit
          - APDU

- [http://adywicaksono.wordpress.com/2008/05/21/understanding-gsm-0348/](http://adywicaksono.wordpress.com/2008/05/21/understanding-gsm-0348/)
Turtles all the way down (GSM 03.48)

- Remember this INSTALL command?

<table>
<thead>
<tr>
<th></th>
<th>A0</th>
<th>C2</th>
<th>00</th>
<th>00</th>
<th>43</th>
<th>D1</th>
<th>41</th>
<th>82</th>
<th>02</th>
<th>83</th>
<th>81</th>
<th>8B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3B</td>
<td>40</td>
<td>08</td>
<td>81</td>
<td>55</td>
<td>66</td>
<td>77</td>
<td>88</td>
<td>7F</td>
<td>F6</td>
<td>00</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>12</td>
<td>00</td>
<td>00</td>
<td>04</td>
<td>2A</td>
<td>02</td>
<td>70</td>
<td>00</td>
<td>00</td>
<td>25</td>
<td>0D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>80</td>
<td>E6</td>
<td>02</td>
<td>00</td>
<td>12</td>
<td>07</td>
<td>F0</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
<td>F4</td>
</tr>
<tr>
<td>F5</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>06</td>
<td>EF</td>
<td>04</td>
<td>C6</td>
<td>02</td>
<td>05</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>
In case you missed it, you can use this exact mechanism to remotely send APDUs to a SIM card(!!!)
The simhacks toolset

- Open-source Python script to manage apps on SIM cards (plus other SIM card tools)

- [http://simhacks.github.io](http://simhacks.github.io)

- DEMO!
Back to STK Apps!

Now that we’ve figured out how to build and load apps, let’s make some!
Life of an STK app

- App is loaded onto the card
- App registers itself with the SIM Toolkit API
- Phone informs STK of its capabilities
- STK informs the phone about registered apps
- Selection of an app will trigger an event to be delivered to the app
- App can then send UI requests back to phone
Anatomy of an STK app

- An STK app is also a Java Card app
  - install
  - process – not normally used

- `processToolkit` method:
  - Handles STK events
Example STK Apps

- Hello STK

- Toorcamp 2012 Crypto Challenge App
  - Provided hints to the challenge
    - Set tamper-proof hint flags
    - We could remotely query these flags to find out if someone viewed a hint

- DEMOS!
Future Directions

- STK apps are pretty limited, but there is potential for awesomeness
  - SIM card botnet?
- If phones provide an API to send APDUs to SIM apps, things will get really interesting
  - SSH private keys secured on your SIM?
  - Secure BitCoin transactions?
  - What else?
    - Of course, we need carriers to get on board
Side-stepping the carriers:
Android’s Secure Element

Yes, it ALSO supports JavaCard/GlobalPlatform!
- Retail phones have non-default ISD keys :(
- Come on Google, give us access!
Tools and more!

Learn more and get the tools at:

http://simhacks.github.io/
References

- Java Card 2.1.1 Virtual Machine Specification
- GlobalPlatform card specification 2.1/2.2
- GSM 03.48 – Secure remote SIM access
- GSM 03.40 – SMS standard
- ETSI TS 101 220 – Assigned numbers
- ETSI TS 102 221 – UICC/(U)SIM spec
- ETSI TS 102 223 – Card Application Toolkit
- ETSI TS 102 226 – Remote APDUs
- ETSI TS 102 241 – UICC/SIM API for JavaCard
References

- http://adywicaksono.wordpress.com/2008/05/21/understanding-gsm-0348/

- http://wiki.thc.org/gsm/simtoolkit