NFC with Android

Near Field Communication with Android

Dominik Gruntz
University of Applied Sciences Northwestern Switzerland, Brugg-Windisch
368

JAZOON
INTERNATIONAL CONFERENCE ON THE
MODERN ART OF SOFTWARE, 21-23 JUNE 2011, ZURICH

NFC Experience

NFC at FHNW

> 2005/06 First NFC demonstrator (with Siemens CX70 Emoty)
  - NFC was included in a removable cover

> 2009/10 Mobile Payment project (Nokia 6131 NFC, S40 Phone)
  - *touch'n'pay* Self Service Shop
  - Supported by the Hasler foundation
  - NFC Forum competition: First price in the category "The Best NFC Service of the Year 2010"

> 2010/11 Android Nexus S (with NFC)
  - Tag reading with 2.3.2
  - Tag writing and P2P with 2.3.3
  - Tag emulation with Android Wallet

AGENDA

- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

What is NFC

NFC (Near Field Communication)

- Communication technology based on radio waves at 13.56 MHz frequency
- Short range (<= 10 cm theoretical, 1-4 cm typical)
- Low speed (106 / 216 / 414 kbps)
- Low friction setup (no discovery, no pairing)
  - Setup-time < 0.1 Sec
- Communication roles:
  - Master Device: NFC Initiator (starts communication, typically a device)
  - Slave Device: NFC Target (passive tag or device)
- Standardization: NFC Forum (founded 2004 by NXP, Sony, Nokia)
  - Definition of standards
  - Popularization of NFC
  - Today: More than 150 members

NFC Device Operating Modes

Reader-Writer Mode

- Mobile Device is able to read external tags/smartcards, Device becomes RFID reader/writer (and can launch applications)
  - Tag content: Text, URI (WebLink, Phone Number), SmartPoster
- Like QR-Codes, but faster
  - No need to launch an application
  - With Android, an intent is thrown if a tag is detected
- Tags
  - Different form factors for NFC tags: tags, stickers, key fobs, cards, clocks
  - Supported Technologies:
    - ISO 14443 A/B, Mifare Ultralight, Classic/Standard 1K/4K
    - NXP DESFire, Sony Felica, Innovision Topaz, Jewel tag

    => NFC Forum Specs define how NFC Messages are stored

NFC Device Operating Modes

Peer-To-Peer Mode

- Bidirectional P2P connection to exchange data between devices
  - Proximity triggered interactions
  - Nexus S: Devices have to be placed back-to-back
- Applications
  - Exchange of vCards
  - Hand-over of Tickets & P2P Payment
  - Web-page sharing, Youtube-video-sharing
  - Application sharing

NFC Device Operating Modes

Tag Emulation

> Device emulates a passive tag (typically a smart card)
  - Device can emulate (contain) multiple smartcards
  - Reader can't distinguish between smartcard & tag emulation
  - Android: Emulated tag can be read only if screen is on

> Examples
  - Access to the farm shop (Legic key)
  - Oyster-Card, London
  - Visa payWave Payment System
  - Google Wallet

Android and NFC

Android Gingerbread

- Tag reading (2.3.2)
- Tag writing (2.3.3)
- Limited P2P (NDEF push only, 2.3.3)

Android NFC Devices

- Nexus S *contains PN544 NFC Controller from NXP + SecureMX*
  - Embedded Secure Element
  - Support of SE on SIM (Single Wire Protocol)

- Samsung Galaxy S2
  - SWP (no embedded SE)

AGENDA

- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

NFC Data Exchange Format (NDEF)

NDEF

- Container format to store NFC data in NFC tags
  - Independent from tag type
- Defines a number of specific types
  - URI, TextRecord, SmartPoster
- Standardized by the NFC Forum (http://www.nfcforum.org)
  - Specs are public
  - Specs are free

NFC Data Exchange Format (NDEF)

NdefMessage

> Represents an NDEF (NFC Data Exchange Format) data message
> Contains one or more NdefRecords that represent different sets of data

NdefRecord

> Represents a NDEF record and always contains
  - 3-bit TNF (Type Name Format) field (indicates how to interpret the type field)
  - Variable length type: Describes the record format
  - Variable length ID: A unique identifier for the record
  - Variable length payload: The actual data payload

NFC Data Exchange Format (NDEF)

TNF Types

> EMPTY	(0)	Empty record (without type / id / payload)
> WELL_KNOWN	(1)	Record contains a well-known type according to the RTD definition (Text, URI, SmartPoster, …)
> MIME_MEDIA	(2)	Type of this record is defined with a MIME-type,
> ABSOLUTE_URI	(3)	Type field contains a URI which defines the type of the payload (e.g. a XML schema URI)
> EXTERNAL_TYPE	(4)	Type field contains a NFC-Forum external type, i.e. an application specific type
> UNKNOWN	(5)	Type of payload is unknown (type field is empty), comparable to "application/octet-stream"
> UNCHANGED	(6)	payload is an intermediate or final chunk of a chunked record (type field is empty)
> RESERVED	(7)	to be treated as UNKNOWN

NFC Data Exchange Format (NDEF)

RTD Types (Record Type Definition) for well-known NFC types

- TEXT "T"
  - Record contains plain text
  - Includes a ISO language identifier
- URI "U"
  - Record contains a URI (UTF-8 encoded)
- SMART_POSTER "Sp"
  - "URI with a title" *(key use case for NFC)*
  - Record containing several records
    - URI record (only one)
    - Titles (in different languages)
    - Icon records
    - Action record (what to do with the URI)
      - DO, OPEN (for editing), SAVE (for later use)

NFC Data Exchange Format (NDEF)

NdefMessage

```
class NdefMessage {
   public NdefMessage(NdefRecord[] records);
   public NdefRecord[] getRecords();
   public byte[]       toByteArray();
}
```

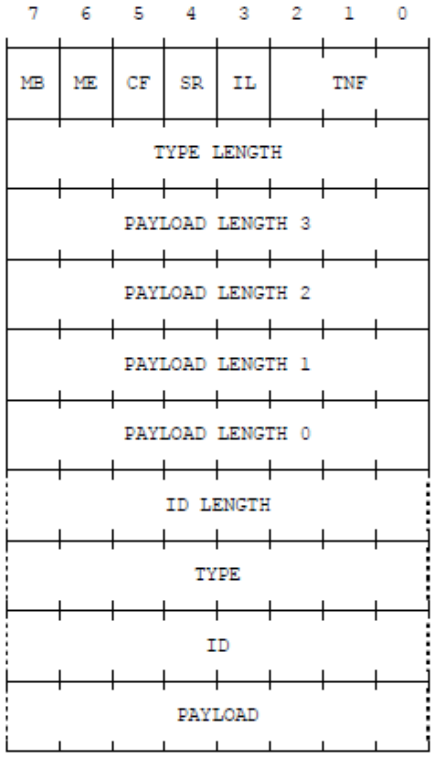
NdefRecord

```
public class NdefRecord {
   public NdefRecord(short tnf, byte[] type, byte[] id, byte[] pl);
   public NdefRecord(byte[] data);
   public short getTnf();
   public byte[] getType();
   public byte[] getId();
   public byte[] getPayload();
   public byte[] toByteArray();
}
```

NFC Data Exchange Format (NDEF)

NDEF Record Layout

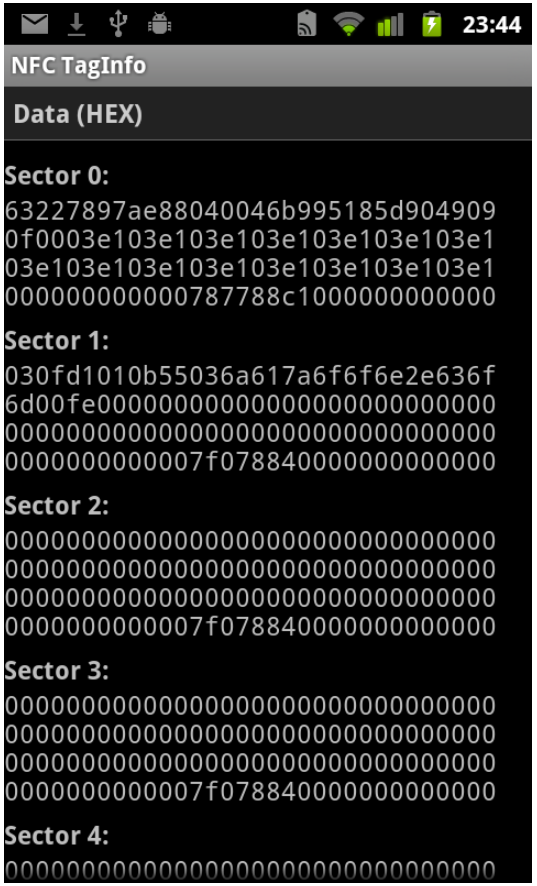
- MB = Message begin
- ME = Message end
- CF = initial or middle chunk of a chunked record
- SR = Short record (payload length = 1 byte)
- IL = ID_Length (and ID) are present



NFC Data Exchange Format (NDEF)

Mifare Tag with NDEF message

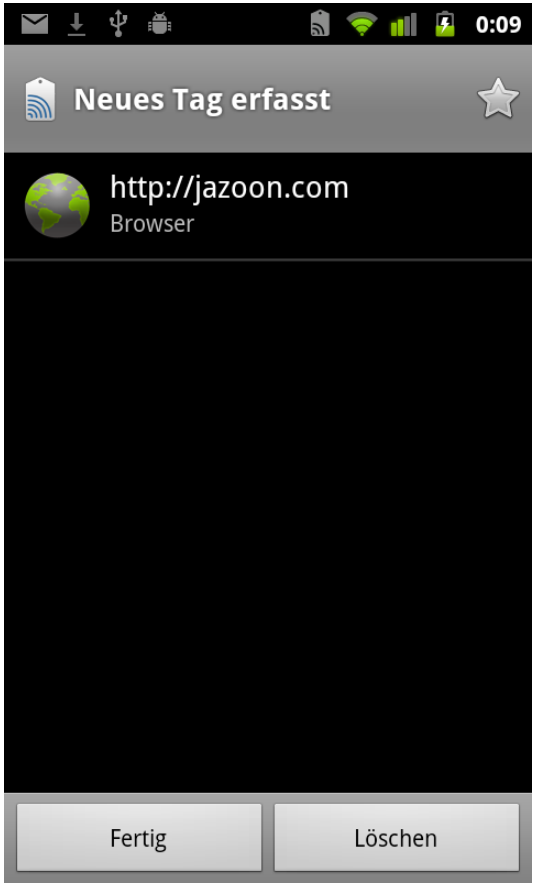
- 03 = NDEF content
- 0F = Length of NDEF message (15 bytes)
- D1 = Status = 1101 0001
  - Short record, no ID
  - TNF = WELL-Known
- 01 = Type length
- 0B = Payload Length
- 55 = Type ("U" => URL)
- 03 = Prefix "http://"
- 6A 61 7A 6F 6F 6E 2E 63 6F 6D = jazoon.com
- 00 = NULL TLV
- FE = Terminator



NFC Data Exchange Format (NDEF)

Mifare Tag with NDEF message

- 03 = NDEF content
- 0F = Length of NDEF message (15 bytes)
- D1 = Status = 1101 0001
  - Short record, no ID
  - TNF = WELL-Known
- 01 = Type length
- 0B = Payload Length
- 55 = Type ("U" => URL)
- 03 = Prefix "http://"
- 6A 61 7A 6F 6F 6E 2E 63 6F 6D = jazoon.com
- 00 = NULL TLV
- FE = Terminator



Reading NDEF Messages

AndroidManifest.xml

- Permission to access the NFC hardware

```
<uses-permission android:name="android.permission.NFC" />
```

- Specify minimum SDK version (2.3.3)

```
<uses-sdk android:minSdkVersion="10"/>
```

- Indication for the market

```
<uses-feature android:name="android.hardware.nfc"
android:required="true"/>
```

- Intent Filter

```
<intent-filter>
   <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
   <data android:mimeType="mime/type" />
   <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

Reading NDEF Messages

Intent Filter and Data Field

- TNF_WELL_KNOWN / RTD_TEXT

```
<data android:mimeType="text/plain" />
```

- TNF_WELL_KNOWN / RTD_URI or RTD_SMART_POSTER

```
<data android:scheme="http" android:host="jazoon.com"
   android:path="/Conference"
/>
```

  - Scheme mandatory
  - Host may be omitted (if present, then exact match necessary, no wildcards)
  - Path may be omitted (if present, then exact match necessary, no wildcards)
    => alternatively use pathPrefix or pathPattern

- TNF_MIME_MEDIA

```
<data android:mimeType="x-urn-nfc-ext/fhnw.ch:selfserviceshop" />
```

  - Wildcards are allowed

Reading NDEF Messages

```
NdefMessage[] getNdefMessages(Intent intent) {
   NdefMessage[] msgs = null; String action = intent.getAction();
   if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)){
      Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
                                       NfcAdapter.EXTRA_NDEF_MESSAGES);
      if (rawMsgs != null) {
         msgs = new NdefMessage[rawMsgs.length];
         for (int i = 0; i < rawMsgs.length; i++)
            msgs[i] = (NdefMessage) rawMsgs[i];
      } else {
         NdefRecord rec = new NdefRecord(NdefRecord.TNF_UNKNOWN,
                              new byte[0], new byte[0], new byte[0]);
         NdefMessage msg = new NdefMessage(new NdefRecord[] {rec});
         msgs = new NdefMessage[] {msg};
      }
   }
   return msgs;
}
```

Writing NDEF Messages

```
void writeUrlToTag(Intent intent, String url)
                                  throws IOException, FormatException  {
   String action = intent.getAction();
   if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
      Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
      Ndef ndefTag = Ndef.get(tag);

      NdefRecord rec = NdefRecordRtdUri.createRtdUriRecord(url);
      NdefMessage msg = new NdefMessage(new NdefRecord[] { rec });
      ndefTag.connect();
      ndefTag.writeNdefMessage(msg);
      ndefTag.close();
   }
}
```

Peer-To-Peer NDEF Messages

Prerequisites

> Pushing activity must be in the foreground
> Data to be send must be encoded as NdefMessage
> Both devices must support the NDEF push protocol

Remarks

> While pushing data, the standard intent dispatch system is disabled
> Pushing is enabled with foreground dispatching (onResume / onPause)

> Specified in Android NDEF Push Protocol Specification (V1, 22.02.2011) is built on top of LLCP
> With Ice Cream Sandwich live pushing is possible (NdefPushCallback)

Peer-To-Peer NDEF Messages

```
private NfcAdapter nfcAdapter;
private NdefMessage pushMessage;

public void onCreate() {
   super.onCreate();
   nfcAdapter = NfcAdapter.getDefaultAdapter(this);
   pushMessage = ...
}
public void onResume() {
    super.onResume();
    if (nfcAdapter != null)
        nfcAdapter.enableForegroundNdefPush(this, pushMessage);
}
public void onPause() {
    super.onPause();
    if (nfcAdapter != null)
        nfcAdapter.disableForegroundNdefPush(this);
}
```

AGENDA

- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

Beyond NDEF

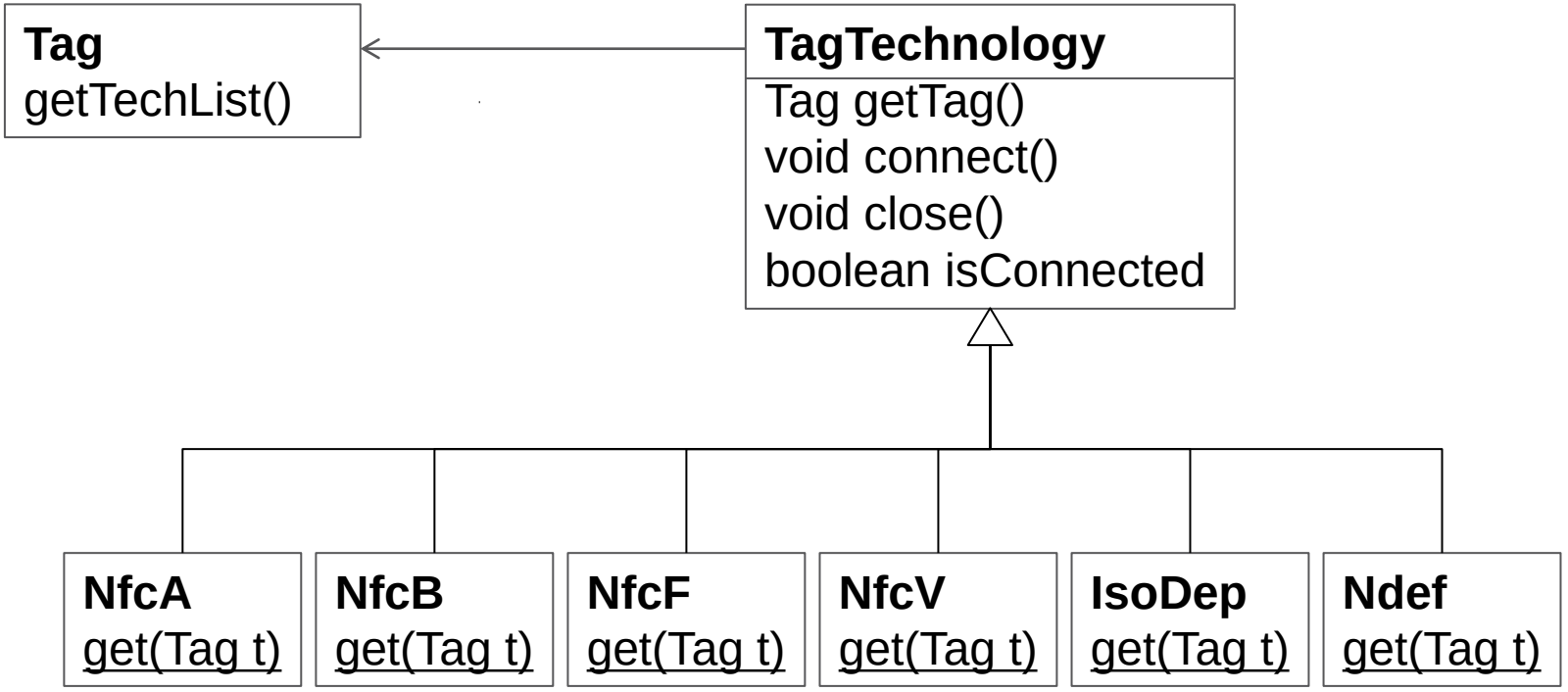
Specifications

> Protocol Level: 14443-3A / B, JIS6319-4 (Felica), ISO-15693 (Vincinity)
> Application Level: 14443-4 (Transmission protocol)
> Proprietary: Mifare Classic/Plus, Mifare Ultralight [C], Mifare DESFire

Tag Technologies

> Classes to expose technology specific functionality (*android.nfc.tech*)
> A tag may have zero or more technologies present
  – NfcA, NfcB, NfcF (Felica), NfcV (Vincinity)
  – IsoDep
  – Ndef
  – NdefFormattable
  – MifareClassic, MifareUltralight

Beyond NDEF



Beyond NDEF

Tag Technology Access

- Method Tag.getTechList() returns a list of supported technologies, as fully qualified class names
- Example: IsoDep: provides access to ISO-DEP (ISO 14443-4) Tags

```
class IsoDep implements TagTechnology {
   static IsoDep get(Tag tag);
   Tag           getTag();
   void          connect();
   boolean       isConnected();
   void          close();

   byte[]        getHiLayerResponse();
   byte[]        getHistoricalBytes();
   void          setTimeout(int timeout);
   byte[]        transceive(byte[] data);
}
```

Beyond NDEF

Tag	Tag Type	Tag Technology	NfcA	NfcB	NfcF	NfcV	IsoDep	Ndef	MifareClassic	MifareUltralight	NdefFormattable
Stoos	Tag-it HF-I Plus Inlay	Type V (ISO 15693 / Vicinity)				X					
Davos-Klosters	EM4x3x	Type V (ISO 15693 / Vicinity)				X					
Nokia NFC 6131	ISO 14443-4 SmartCard, Mifare Classic 4K (emulated)	Type A (ISO 1443 Type A)	X				X		X		
Mifare 1K Tag	Mifare Classic 1K (unformatted)	Type A (ISO 1443 Type A)	X						X		X
Mifare 1K Tag (SelfServiceShop)	Mifare Classic 1K (formatted)	Type A (ISO 1443 Type A)	X					X	X		
MF Ultralight C	Mifare Ultralight (unformatted)	Type A (ISO 1443 Type A)	X							X	X
Mifare4K Tag	Mifre Classic 4K (unformatted)	Type A (ISO 1443 Type A)	X						X		X

Beyond NDEF

Tag Technology Dispatching

> Intent-Filter can also be specified for particular tag technologies

```
<activity android:name="..." android:label="...">

   <intent-filter>
      <action android:name="android.nfc.action.TECH_DISCOVERED"/>
   </intent-filter>

   <meta-data android:name="android.nfc.action.TECH_DISCOVERED"
      android:resource="@xml/filter_nfc"
   />

</activity>
```

Beyond NDEF

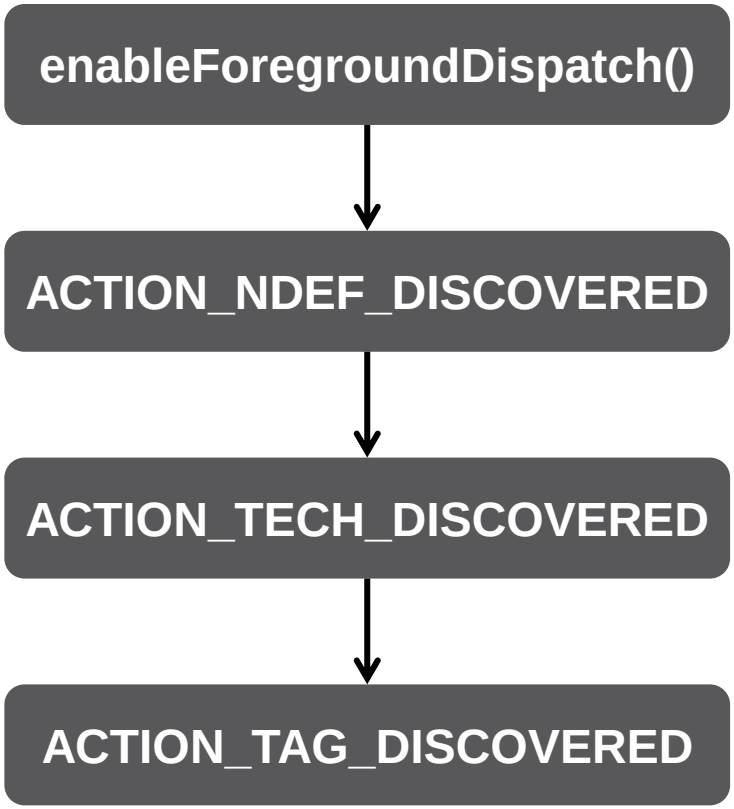
Tag Technology Dispatching

- filter_nfc.xml contains one or more tech-list entries (qualified class names)
- A tag matches if any of the tech-list sets is a subset of Tag.getTechList
- The following list matches Felica or Mifare Classics with NDEF content

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <tech-list>
        <tech>android.nfc.tech.NfcF</tech>
    </tech-list>

    <tech-list>
        <tech>android.nfc.tech.NfcA</tech>
        <tech>android.nfc.tech.MifareClassic</tech>
        <tech>android.nfc.tech.Ndef</tech>
    </tech-list>
</resources>
```

Beyond NDEF

Tag Dispatching



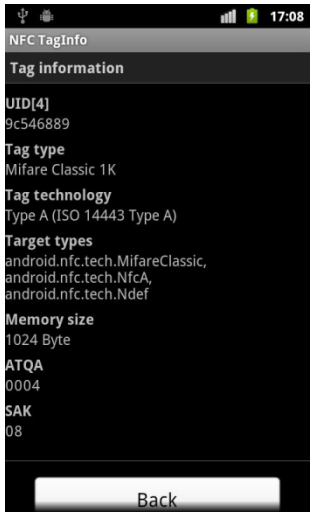
AGENDA

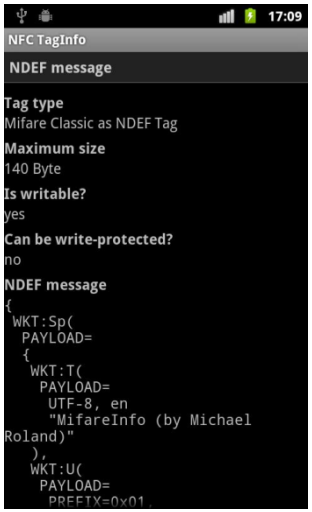
- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

Applications

NFC Tag Info

- Displays card information
- Displays the sectors of a tag (hex / ascii)
- Displays NDEF content

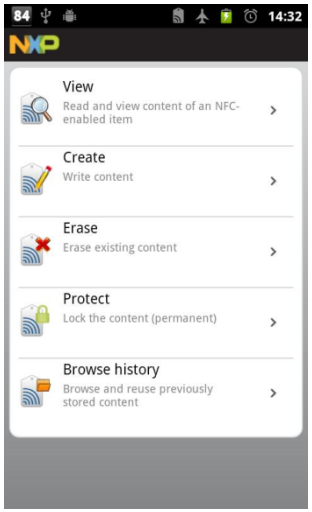


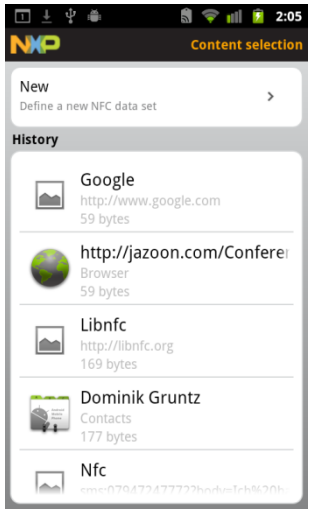


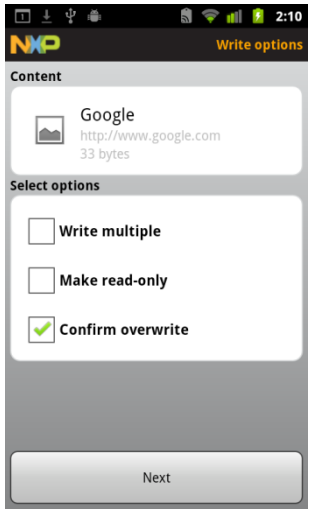
Applications

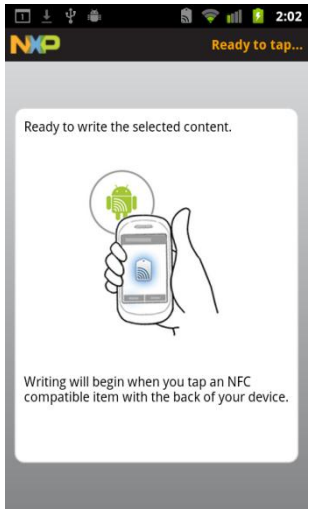
NXP Tag Writer

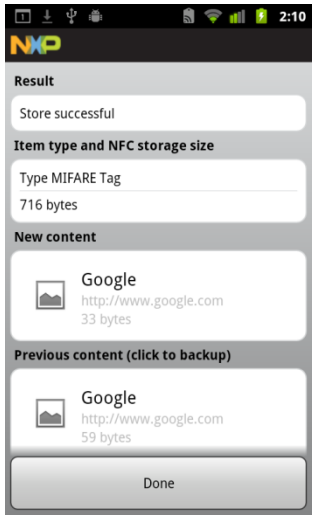
- Supports Reading & Viewing content of a tag
- Supports Creating / Erasing / Protecting content











Applications

WiFiTap
Allows to store & load the WiFi configuration on a tag
(i.e. Name & WPA/WEP password)

NFC TaskLauncher
Use NFC tags to automate tasks (e.g. set volumes, set alarms, etc)

EnableTable
Restaurant couponing & loyalty system
Tag is embedded in the check billfold

NFC Security
Locks Android application; application can only be started if
a NFC tag with the key is read in

TabPats
Real-Time information for Stanford Marguerite bus departures,
simply place the phone against the TapPATS badge at the bus stop

AGENDA

- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

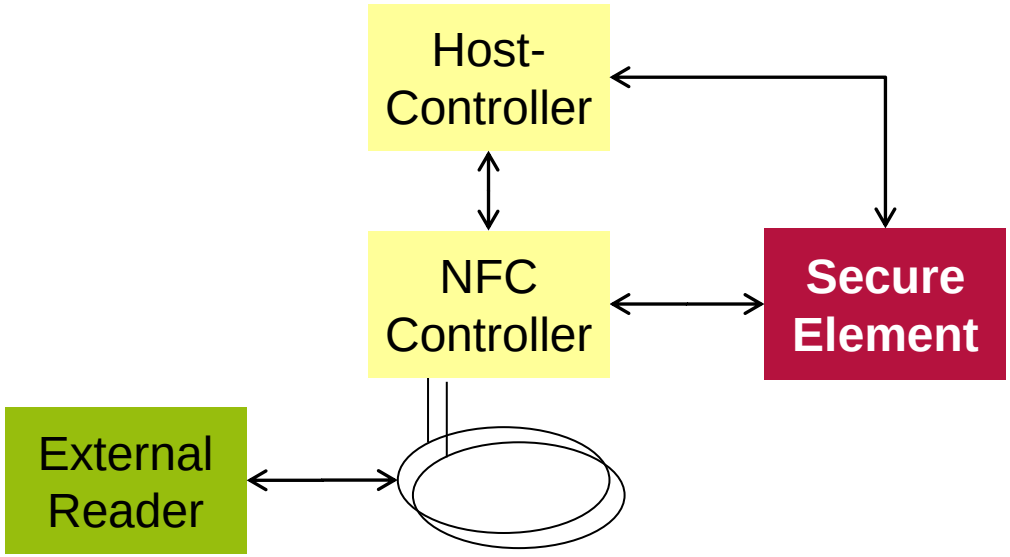
Secure Element

Secure Storage in NFC device

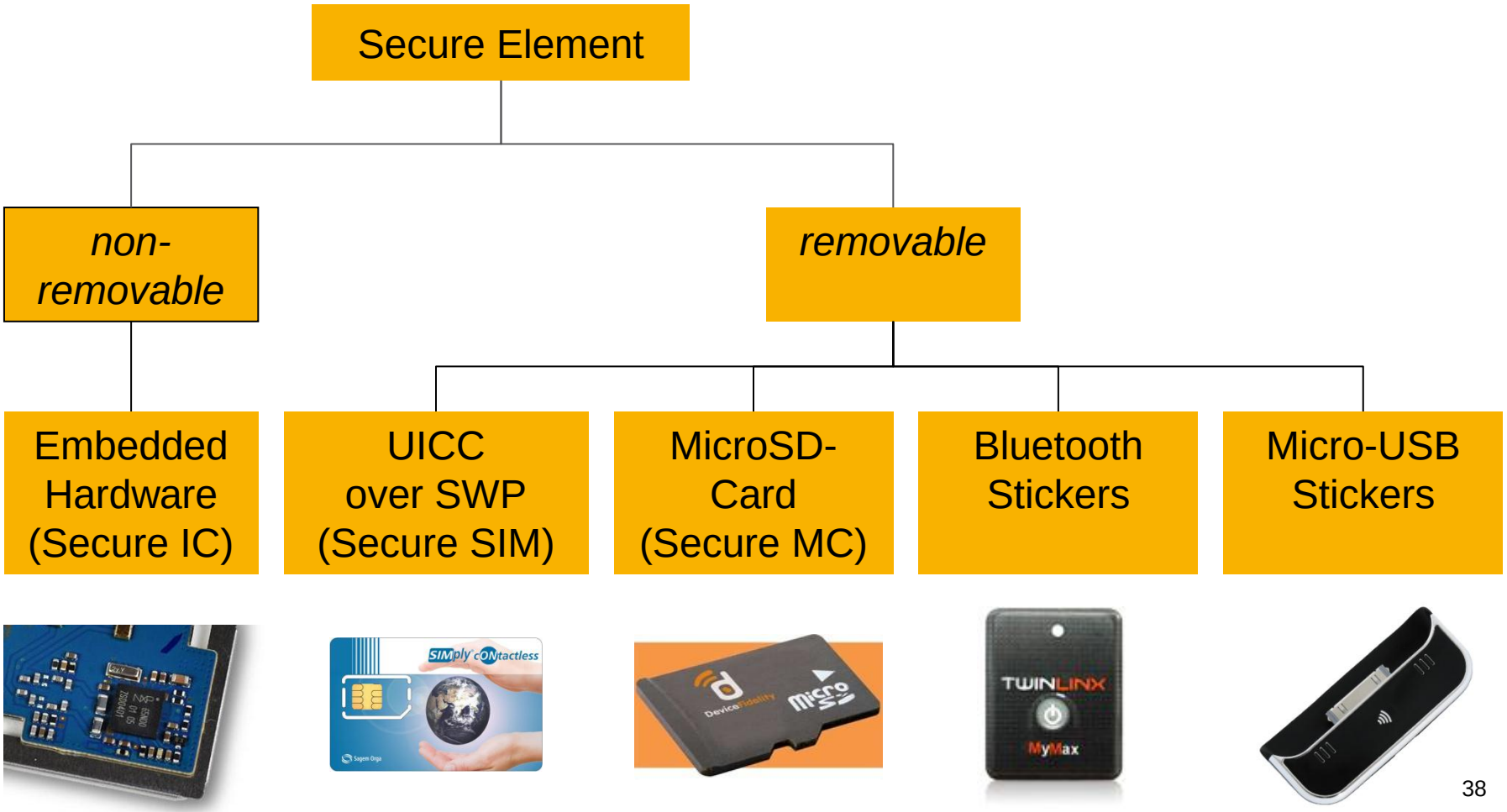
> Tamper-proof storage for sensible data (money, tickets, keys)
> Cryptographic operations
> Secure environment for the execution of program code (sandbox model)

Platforms

> SmartCard (Global Platform)
  – JavaCard system
  – APDU commands



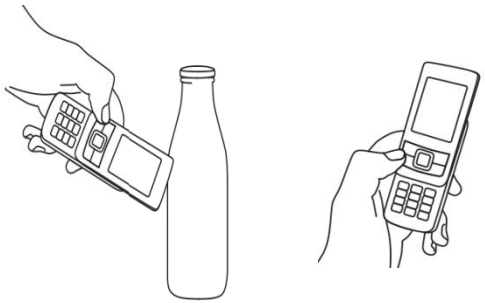
Secure Element



AGENDA

- What is NFC
- NFC with Android: Reading & Writing NDEF Messages
- NFC with Android: Beyond NDEF
- NFC with Android: Applications
- NFC Secure Element
- NFC Use Case: Self Service Shopping

Self Service Shopping



Self Service Shopping

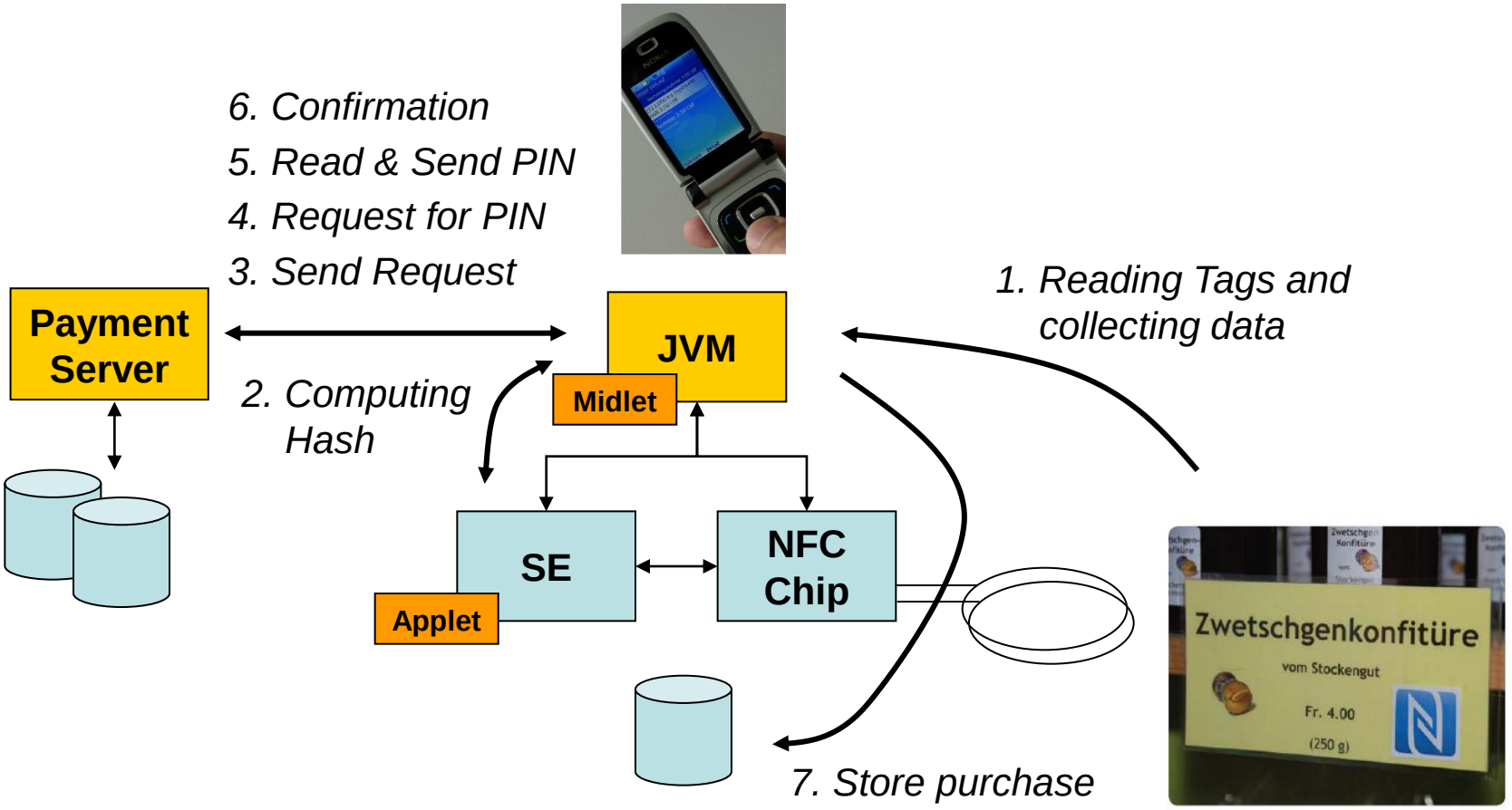
Facts

- Location: Mini-market, Uf-Stocken, Kilchberg
- Pilot start: 12.2009 – 12.2010
- No. of user: 80 consumers
- Devices: Nokia 6131 NFC/ Nokia 6212 Classic

Partners

- e24 — Mobile Payment Solution Provider
  http://www.e-24.ch
- NEXPERTS — NFC Solution Provider
  http://www.nexperts.com
- FHNW — Institute for Mobile and Distributed Systems
  http://www.imvs.ch

Self Service Shopping: Secure Payment



ISO 14443-4 compliant Card Access

Communication with the Applet with APDU commands

```
byte[] SELECT = {
    (byte) 0x00, // CLA Class
    (byte) 0xA4, // INS Instruction
    (byte) 0x04, // P1  Parameter 1
    (byte) 0x00, // P2  Parameter 2
    (byte) 0x0A, // Length
    0x63,0x64,0x63,0x00,0x00,0x00,0x00,0x32,0x32,0x31 // AID
};

Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
IsoDep tag = IsoDep.get(tagFromIntent);
tag.connect();
byte[] result = tag.transceive(SELECT);
if (!(result[0] == (byte)0x90 && result[1] == (byte) 0x00))
   throw new IOException("could not select applet");
```

ISO 14443-4 compliant Card Access

Communication with the Applet with APDU commands

```
byte[] GET_MSISDN = {
    (byte) 0x80, // CLA Class
    (byte) 0x04, // INS Instruction
    (byte) 0x00, // P1  Parameter 1
    (byte) 0x00, // P2  Parameter 2
    (byte) 0x10  // LE  maximal number of bytes expected in result
};

result = tag.transceive(GET_MSISDN);
int len = result.length;
if (!(result[len-2]==(byte)0x90&&result[len-1]==(byte) 0x00))
   throw new IOException("could not retrieve msisdn");
byte[] data = new byte[len-2];
System.arraycopy(result, 0, data, 0, len-2);
String msisdn = new String(data).trim();
tag.close();
```

JavaCard TX Signing Applet

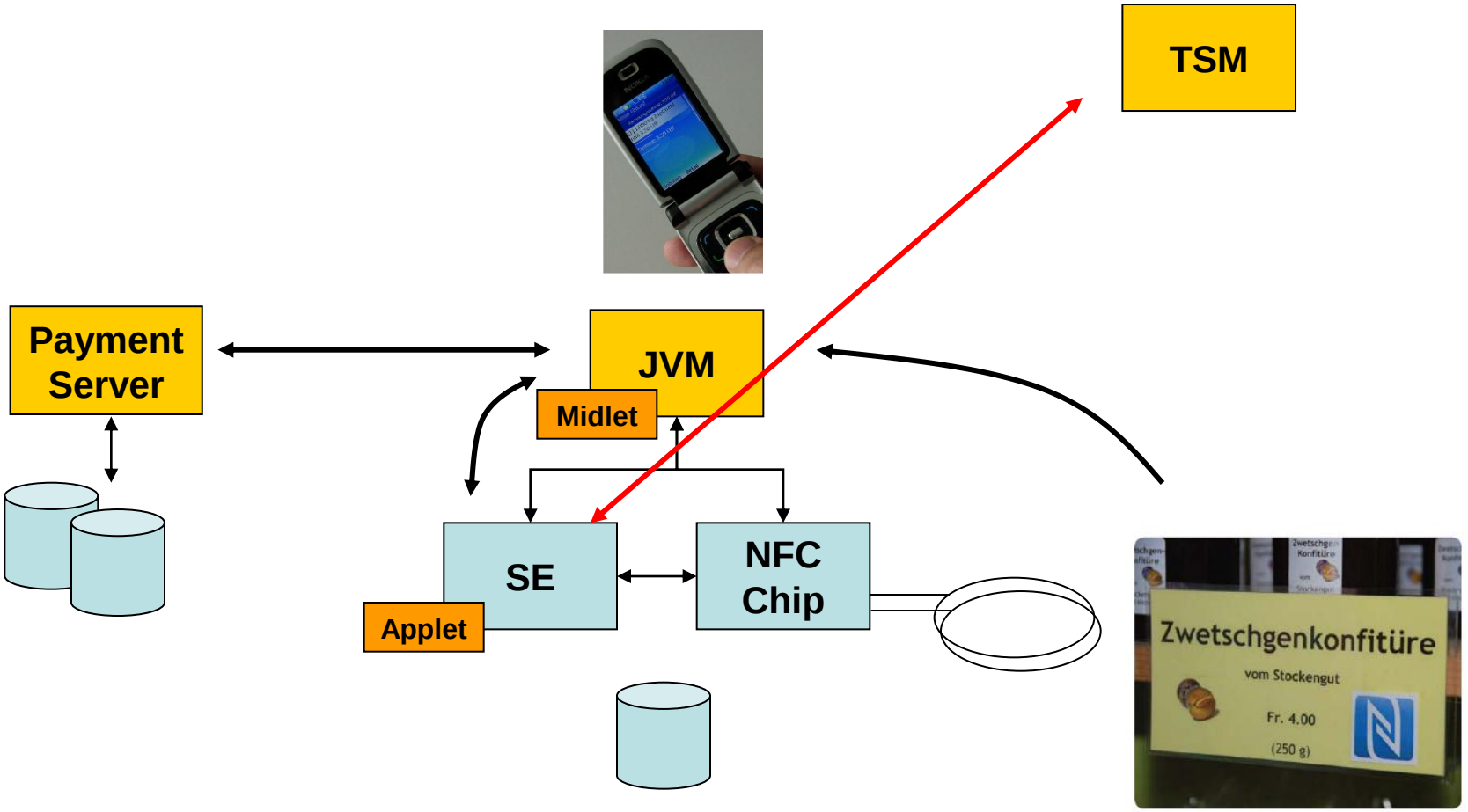
Applet implements APDU commands

```
public class TXSigningApplet extends Applet {
   private final static byte INS_INIT = 0x01;
   private final static byte INS_SIGN = 0x02;
   private final static byte INS_MSISDN = 0x04;

   private byte[] msisdn;
   private byte[] key;
   private boolean initialized = false;

   public static void install(byte[] b, short off, byte len) {
      new TXSigningApplet().register(b, (short)off+1, b[off]);
   }

   public void process(APDU apdu) {
      // Return 9000 on SELECT
      if (selectingApplet()) { return; }y
```

JavaCard TX Signing Applet

```
        byte[] buf = apdu.getBuffer();
        switch (buf[ISO7816.OFFSET_INS]) {
        case INS_MSISDN:
           apdu.setOutgoing();
           apdu.setOutgoingLength((byte) msisdn.length);
           apdu.sendBytesLong(msisdn, (short)0, // offset
                              (byte) msisdn.length);  // length
           break;

        case INS_INIT: cmdInit(apdu); break;
        case INS_SIGN: cmdSign(apdu); break;
        default:
           ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
    }
}
```

OTA Loader



OTA Loader

Proxy between Server and SE

- Proxy reads requests from server and forwards them to secure element
- Proxy may be started by a push SMS
- On server, we use GlobalPlatform (sourceforge project GPShell 1.4.2) which contains a library to convert readable commands to APDUs

- SSL not necessary as APDU commands are encrypted
  - SCP 02 (Secure Channel Protocol), 3DES, 112bit

OTA Loader: Proxy main loop

```
void seCommand() throws IOException, ContactlessException{
   short b0 = (short)( is.read() & 0xFF );
   short b1 = (short)( is.read() & 0xFF );
   short apduLength = (short)((b0 << 8) + b1);
   int n = 0; byte[] apdu = new byte[apduLength];
   while(n < apduLength){
      int read = is.read(temp, n, apduLength-n);
      if(read > -1) n += read; else throw new IOException();
   }
   //send to SE
   byte[] result = seConn.exchangeData(apdu);
   byte[] length = new byte[]{(byte)((result.length>>8)&0xFF),
                              (byte)(result.length&0xFF)};
   os.write(length);
   os.write(result);
   os.flush();
}
```

Google Wallet

Mobile Payment System

- Checkout at MasterCard PayPass-enabled terminals
- Supported Credit Cards
  - Citi MasterCard
  - Google Prepayed
- Partners
  - Citi: Credit Card Issuer
  - FirstData: Accounting / Backend
  - Sprint: Telco Provider

- Android 2.3.4
  - New classes (@hidden) have been provided

Open Questions

Secure Element

- Who controls the keys of the secure element, i.e. which party can enable "card emulation"?
- Will there be a development key to access the SE?
- How are the SE (JavaCard) applets distributed?
- How to revoke applications from a SE?
  - In case that device is stolen
  - In case that device changes ownership
- How to choose emulated card if SE contains several cards?

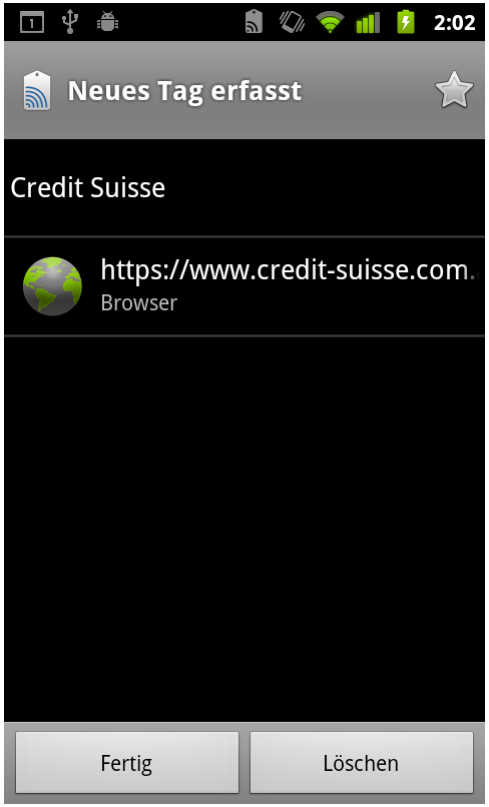
Chicken Egg

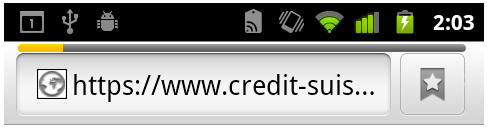
- With Google pushing NFC will it become widespread?
- Will iPhone5 contain NFC

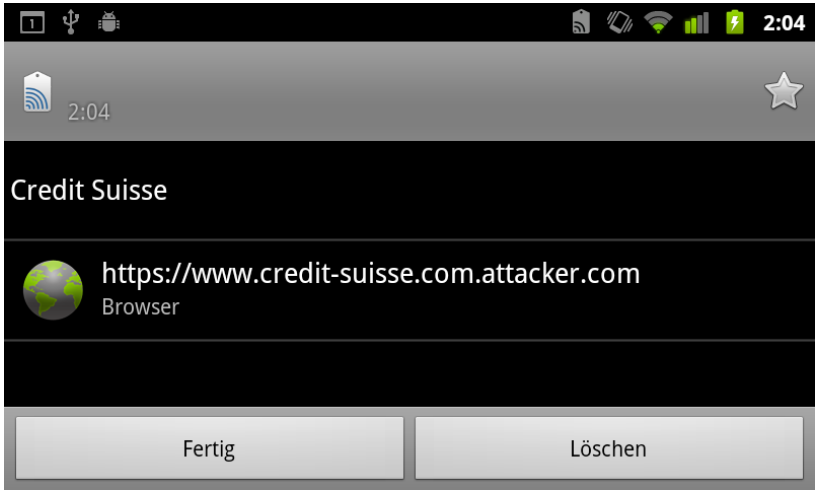
Open Questions

Security

> SmartPoster Spoofing Attack







Source: Collin Mulliner, http://www.mulliner.org/nfc/

NFC Next Steps

Projects & Trials

- Buy Nexus S and upgrade to Android 2.3.4
- Buy NFC Reader & Tags (=> Starter Kits)
- Install NFC Tag Info / NXP Tag Writer Apps
- Read Documentation
  - http://developer.android.com/reference/android/nfc/package-summary.html
- Look at Sample Code (StickyNotes)
  - https://nfc.android.com/StickyNotes.zip

- Contact us for contactless projects – we are interested in applied research

Dominik Gruntz www.imvs.ch

University of Applied Sciences dominik.gruntz@fhnw.ch
Fachhochschule Nordwestschweiz
Institut für Mobile und Verteilte Systeme

Steinackerstrasse 5

5210 Windisch / Switzerland