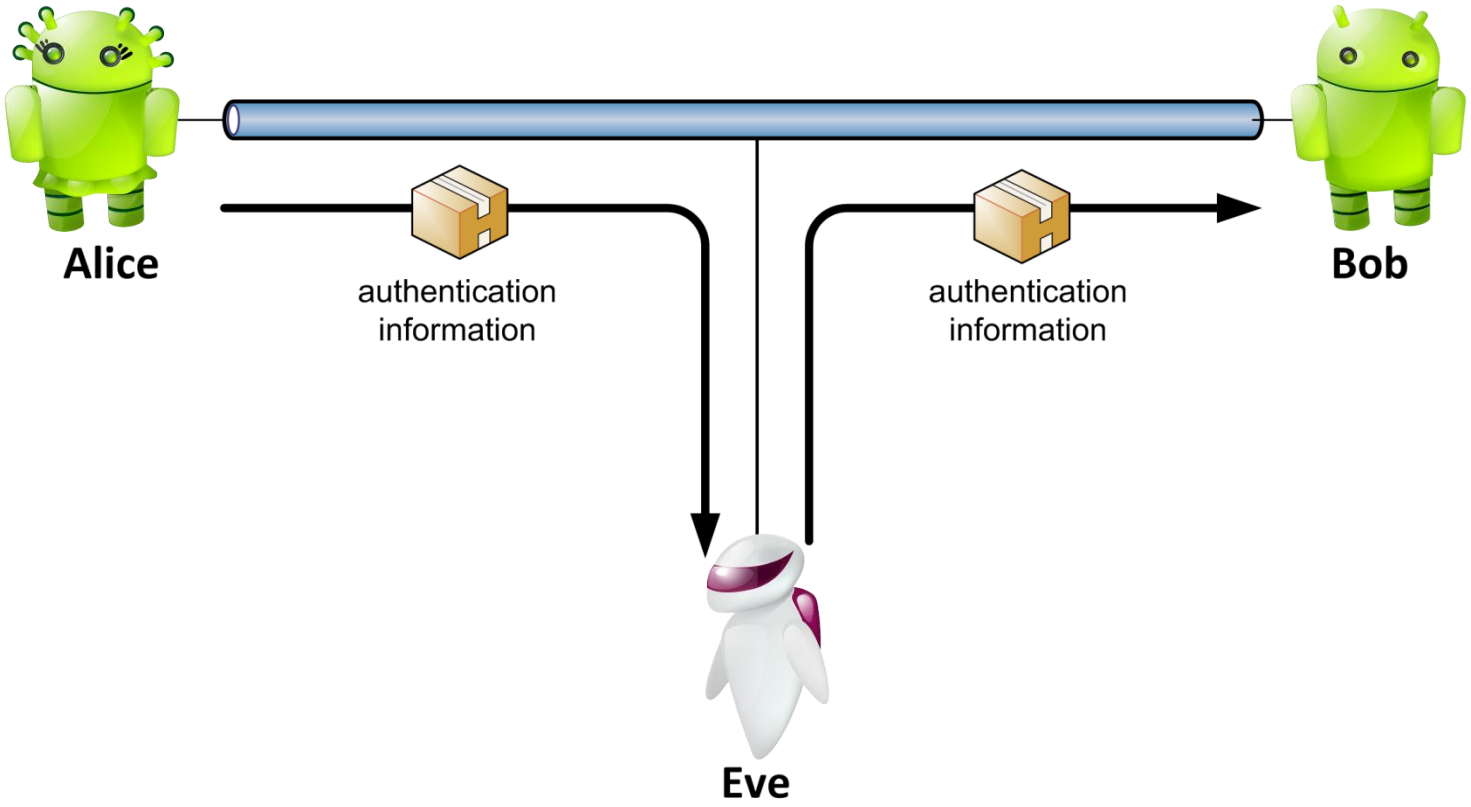
Entwicklung einer Android App zur One-Time Password Generierung & Verwaltung

Bachelor Thesis
Michael Barth

Agenda

1. One-Time Passwords
2. OTPW
3. Android Plattform
4. App Entwicklung
5. OTPManager App

Replay Attack



Folgen einer Replay Attack

- Umgeht Sicherheitsversprechen durch verschlüsselte Passwörter komplett
  - Password muss nicht entschlüsselt werden
- System kann Angreifer nicht von einem validen Benutzer unterscheiden
  - System kann nicht erkennen, dass ein Angriff überhaupt stattfand
- Angreifer kann abgefangene Packete wiederholt senden um Schaden anzurichten

Problemstellung

- Problem:

  Angreifer macht sich Tatsache zunutze, dass sich statische PWs nicht ändern

- Lösung:

  Ändere diese Tatsache durch einführen einer variablen Komponente, dem OTP

One-Time Passwords

- One-Time Password (OTP) ist ein Passwort, welches nach einmaliger Benutzung nicht mehr gültig/verwendbar ist
- Typische Formate:
  - Six-word format:
    `OUST COAT FOAL MUG BEAK TOTE`
  - Random string:
    `Kxkp8LW9`
- Es existieren viele Implementationen. Für die Thesis wurde OTPW verwendet.

Funktionsweise von OTP-Systemen

3 generelle Ansätze basierend auf

1. Zeit-Synchronisation
2. Vorhergehendem OTP & einer mathe-matischen Funktion (Lamport Schema)
3. Challenges

Zeit-Synchronisation

- OTP wird generiert basierend auf der aktuellen Zeit
  - Benutzer benötigt ein Gerät um ein OTP vor Ort generieren zu können
  - Gerätezeit muss mit Serverzeit synchron sein
- OTP wird nach Ablauf einer gewissen Zeit ungültig
  - Diese Eigenschaft ermöglicht das Erkennen und Abblocken von absichtlich verzögerten Nachrichten

Lamport Schema

- Eine Einwegfunktion f wird benutzt um OTPs zu generieren:

$$f(s),\ f(f(s)),\ f(f(f(s))),\ \ldots$$
$$p_0 = f(s),\ p_1 = f(p_0),\ p_2 = f(p_1),\ \ldots$$

- OTPs werden in umgekehrter Reihenfolge benutzt. Nächstes OTP basiert auf Vorgänger.
- Sicherheit der OTPs ist stark abhängig von der Einwegfunktion. Kann diese invertiert werden gilt:

$$p_{i-1} = f^{-1}(p_i)$$

Challenges

- Server schickt eine Challenge an Benutzer, auf welche dieser Antworten muss
  - Challenge fragt in der Regel nach einer persönlichen Information (z.B. einer PIN)
- Antwort beeinflusst Generierung des OTPs
- OTPs werden deterministisch generiert
  - OTP ist reproduzierbar sofern man den Algorithmus und den Inhalt der Antwort kennt
- Benutzer benötigt ein Gerät womit sich OTPs generieren lassen

OTPW

- Erzeugt OTP aus zwei Komponenten:
  - Konstantes Geheimnis (Präfix Passwort)
  - One-Time Token
- Optimiert für Einsatz von ausgedruckten One-Time Password Listen
- Erzeugt beim Generieren zwei Dateien
  - Die OTP Liste mit den One-Time Tokens
  - Eine Server Konfigurationsdatei, welche das gehashte Ergebnis von Präfix Passwort + One-Time Token für alle Tokens der aktuellen Liste beinhaltet

Login Vorgang mit OTPW

- Benutzer hat eine Liste mit typischerweise 280 Tokens, identifizierbar über Indexe
- Man gibt Benutzername & Präfix Passwort ein, zusätzlich noch ein One-Time Token:

```
username: mbarth
password 019: babylon1234AbCd
```

- Race Attack möglich
  - Angreifer belauscht Eingaben des Benutzers. Er versucht den Rest des Passwortes zu erraten und sich anzumelden vor dem Benutzer.
  - Erkennt OTPW dies muss man sich erneut anmelden unter der Verwendung von 3 Tokens

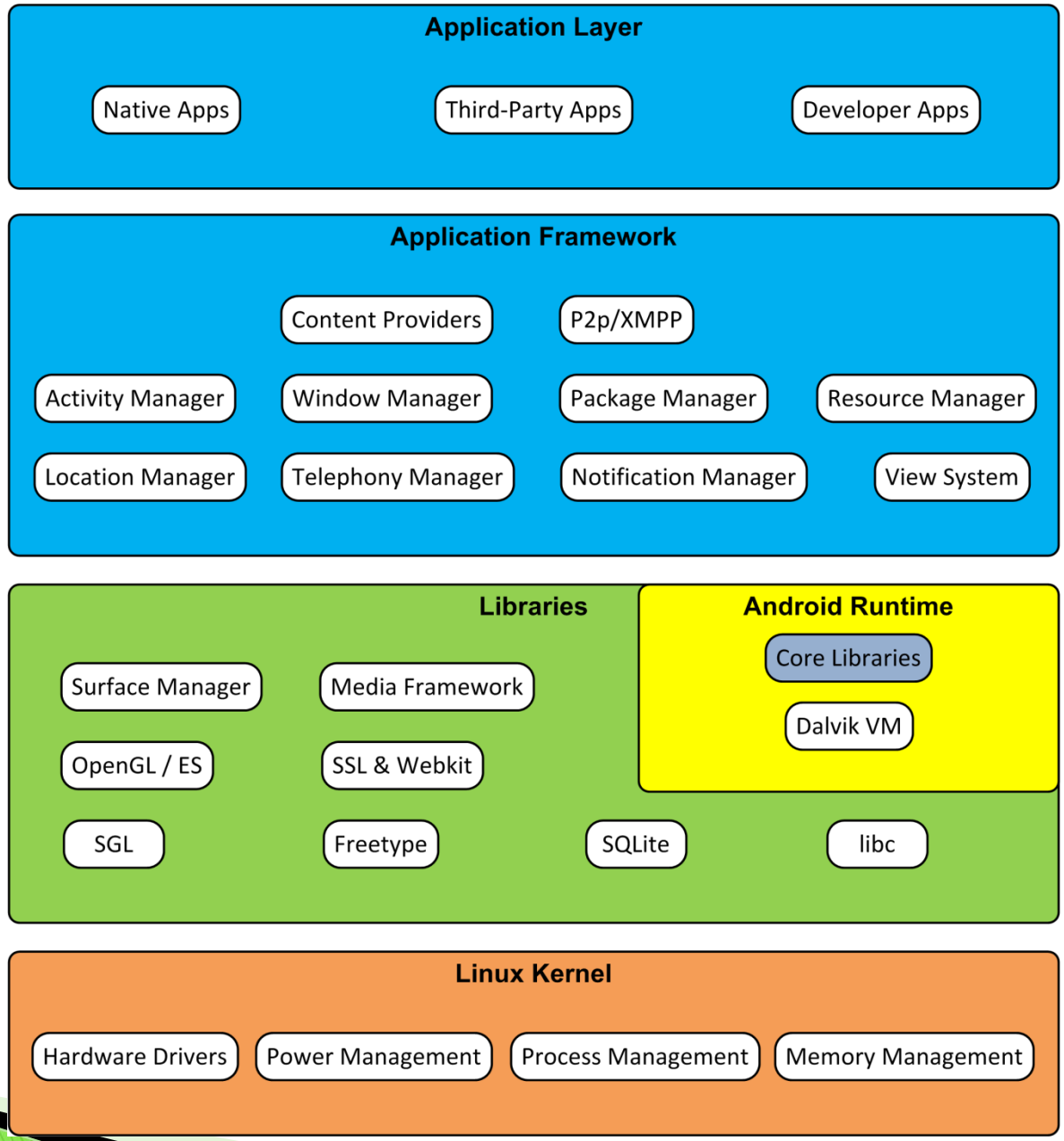
Grenzen von OTP-Systemen

- Verhindern nicht das Nachrichten abgefangen werden können
- Verhindern nicht das Verzögern von Nachrichten
- Schützen nicht die Vertraulichkeit von Daten
- ...

Ein OTPS sollte nicht als alleinige Sicherheits-maßnahme verstanden werden!

Die Android Plattform

- Android ist ein Software-Stack bestehend aus mehreren Schichten (engl. layers)
- Basiert auf einem modifizierten Linux Kernel (aktuell Kernel Version 2.6.32 in Froyo)
- Wird gewartet und weiterentwickelt von der *Open Handset Alliance*
- Verwendet Java für App Entwicklung, setzt jedoch auf eine eigene VM (Dalvik)
- Betriebssystem zum Großteil Open-Source


Application Layer
Native Apps
Third-Party Apps
Developer Apps
Application Framework
Content Providers
P2p/XMPP
Activity Manager
Window Manager
Package Manager
Resource Manager
Location Manager
Telephony Manager
Notification Manager
View System
Libraries
Android Runtime
Core Libraries
Dalvik VM
Surface Manager
Media Framework
OpenGL / ES
SSL & Webkit
SGL
Freetype
SQLite
libc
Linux Kernel
Hardware Drivers
Power Management
Process Management
Memory Management

Dalvik Virtual Machine

- Register-basierte VM
- Keine Java VM
  - Kann **keinen** Java Bytecode ausführen
  - Eigenes Format: Dalvik Executable (DEX) optimiert für minimalen Speicherbedarf
  - `dx` Tool generiert `.dex` Datei aus `.class` Dateien
- Jede App läuft in einem eigenen Prozess mit einer eigenen Instanz der Dalvik VM
  - Verwendet Prozess-, Thread- und Speicher-management des Linux Kernels

Android Packages

- Dateiendung `.apk`
- Ein Android Package enthält genau eine App
- Beinhaltet `.dex` Datei sowie alle Ressourcen, welche die App benötigt
- Resourcen sind ansprechbar im Code über automatisch generierte statische Klasse `R`
- Arten von Ressourcen:
  - XML Dateien zur Definition von Layouts, Styles, Themes, Strings, Menüs, ...
  - Animationen (Tweened, Frame-by-frame)
  - Drawables (BMP, PNG, JPG, GIF, ...)

Layout XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">
6
7      <TextView
8          android:id="@+id/someLabel"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="@string/some_text"
12     />
13     <EditText
14         android:id="@+id/someTextField"
15         android:layout_width="fill_parent"
16         android:layout_height="wrap_content"
17     />
18
19 </LinearLayout>
```

App Entwicklung

- Android Framework ist Komponenten-basiert
  - Jede App kann jede andere App oder Komponenten einer anderen App wiederverwenden, vorausgesetzt diese erlaubt es
  - Beispiele: Kontaktlisten-Ansicht kann man für seine eigene SMS App wiederverwenden
- Android Apps haben keine `main()`-Funktion
  - Um App zu starten muss man eine Nachricht (`Intent`) an das Application Framework senden
  - Framework lokalisiert, instanziert (falls nötig) und startet die Komponente

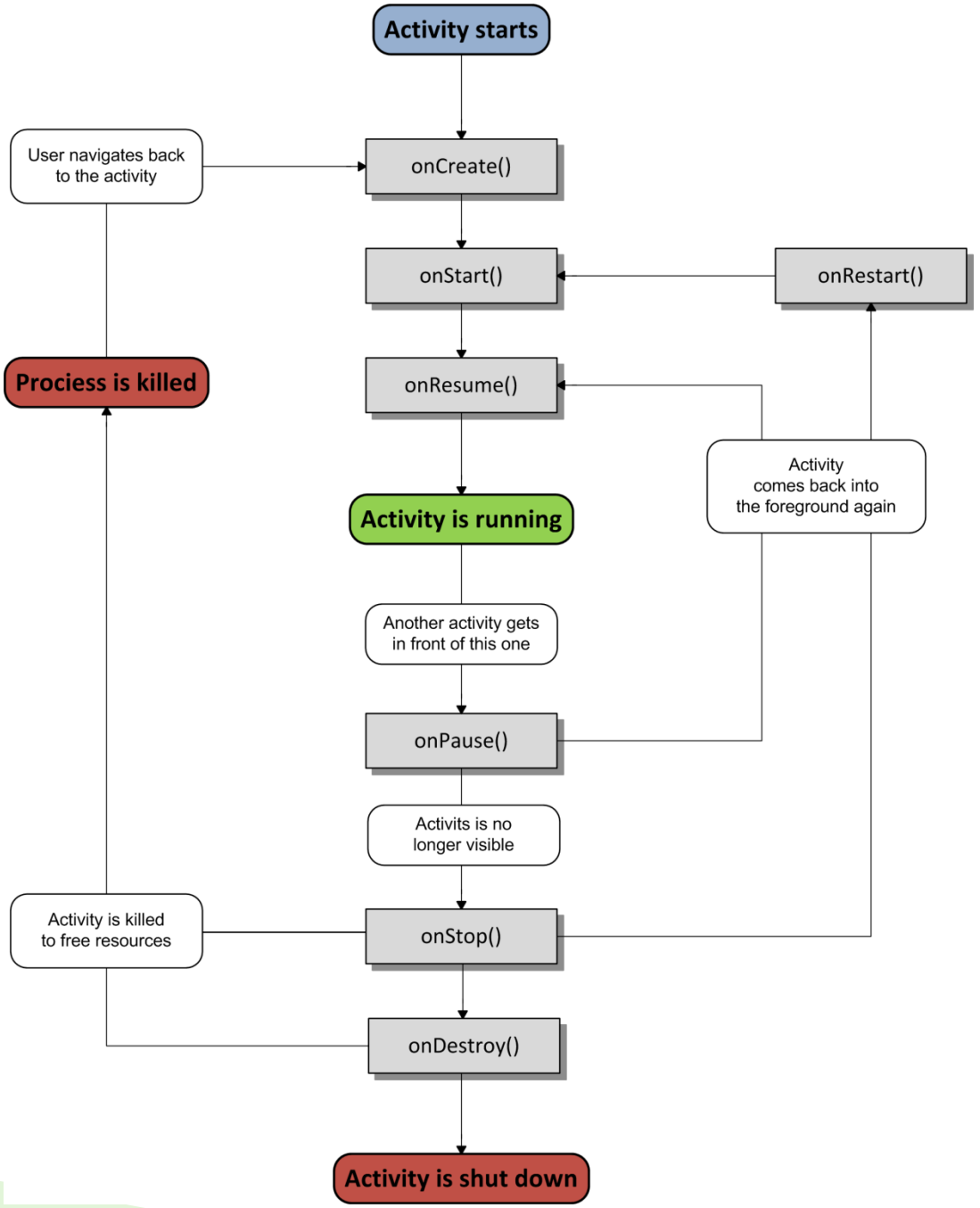
Komponenten

- Es gibt 4 Basiskomponenten:
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers
- Eigene Implementation lassen sich über Vererbung einbinden
- Jede Komponente stellt Lifecycle-Methoden zur Verfügung, die sich überschreiben lassen
  - Method-Hooks in das Framework (*Template method pattern*)

Activities

- Eine Activity ist eine GUI Komponente, welche dem Benutzer die Ausführung einer Aktion ermöglicht
- Erweitert die Basisklasse `Activity`
- Eine App besteht üblicherweise aus mehreren Activities, je nach Design und Komplexität
- Visueller Inhalt wird über eine Hierarchie von `View` Komponenten erzeugt
  - View ist Basistyp, kann von einem Layout(-Manager) bis zu einem Button jede visuelle Komponente sein

Lifecycle


Activity starts
User navigates back to the activity
onCreate()
onStart()
onRestart()
Prociess is killed
onResume()
Activity comes back into the foreground again
Activity is running
Another activity gets in front of this one
onPause()
Activits is no longer visible
Activity is killed to free resources
onStop()
onDestroy()
Activity is shut down

Code Beispiel

```
1  package de.htwaalen.helloworld;
2
3  // Imports...
4
5  public class HelloWorldActivity extends Activity {
6
7      private TextView myTextView;
8
9      /** Called when the activity is first created. */
10     @Override
11     public void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.main);
14
15         // Get reference to view element defined in layout
16         myTextView = (TextView)findViewById(R.id.someTextField);
17         myTextView.setText("Hello Android!");
18     }
19 }
```

Restliche Komponenten

- Services
  - Laufen im Hintergrund, besitzen keine GUI
  - Für rechenaufwändige Aufgaben oder Aufgaben die nicht die Aufmerksamkeit des Benutzers benötigen
- Broadcast Receivers
  - Empfangen bestimmte Broadcasts/Events
  - Erlauben es der App auf die Broadcasts/Events zu reagieren durch starten anderer Komponenten
- Content Providers
  - Erlauben das Teilen von Daten mit anderen Apps
  - Z.B. Kontaktdaten, Bilder, Musik, etc.

Android Security

- Apps haben standardmäßig wenig Rechte und dürfen nichts was andere Apps beeinflusst
- Jede App läuft isoliert in eigenem Prozess mit eigener Linux User ID
- Wenn App erweiterte Rechte benötigt muss sie die *Permission* dafür einfordern
  - Beispiele: Zugriff auf Speicherkarte, Standortdaten, Bluetooth, Wifi, SMS senden, Telefonie, etc. ...
  - Die benötigten Rechte werden bei der Installation aufgelistet und müssen akzeptiert werden

Probleme mit Permissions

- Benutzer erteilt App bei Installation die Erlaubnis für Ihre benötigten Rechte
  - Wenig Transparent: Benutzer weiß nicht wann/wie genau die Rechte genutzt werden von der App
  - Permissions sind eher grob (Zugriff auf Speicherkarte)
  - Bösartige Apps können als normale Apps getarnt die nötigen Rechte erlangen und dann schädlich agieren
- Beispiel:
  - SMS App verlangt `SEND_SMS` und `RECEIVE_SMS` Rechte
  - App lässt sich wie erwartet zum Versenden und Empfangen von SMS einsetzen
  - App sendet eine Kopie jeder SMS an Angreifer

Android Manifest

- XML Datei namens `AndroidManifest.xml` im Root Verzeichnis der App
- Beinhaltet Meta-Informationen über App:
  - Auflistung aller Komponenten
  - Definiert Eintrittspunkt (Activity) der App
  - Alle Permissions welche die App benötigt
  - Anforderungen an Hardware/Plattform
- Android liest und hält sich die Informationen des Manifests im Speicher. Komponenten werden über Manifest lokalisiert.

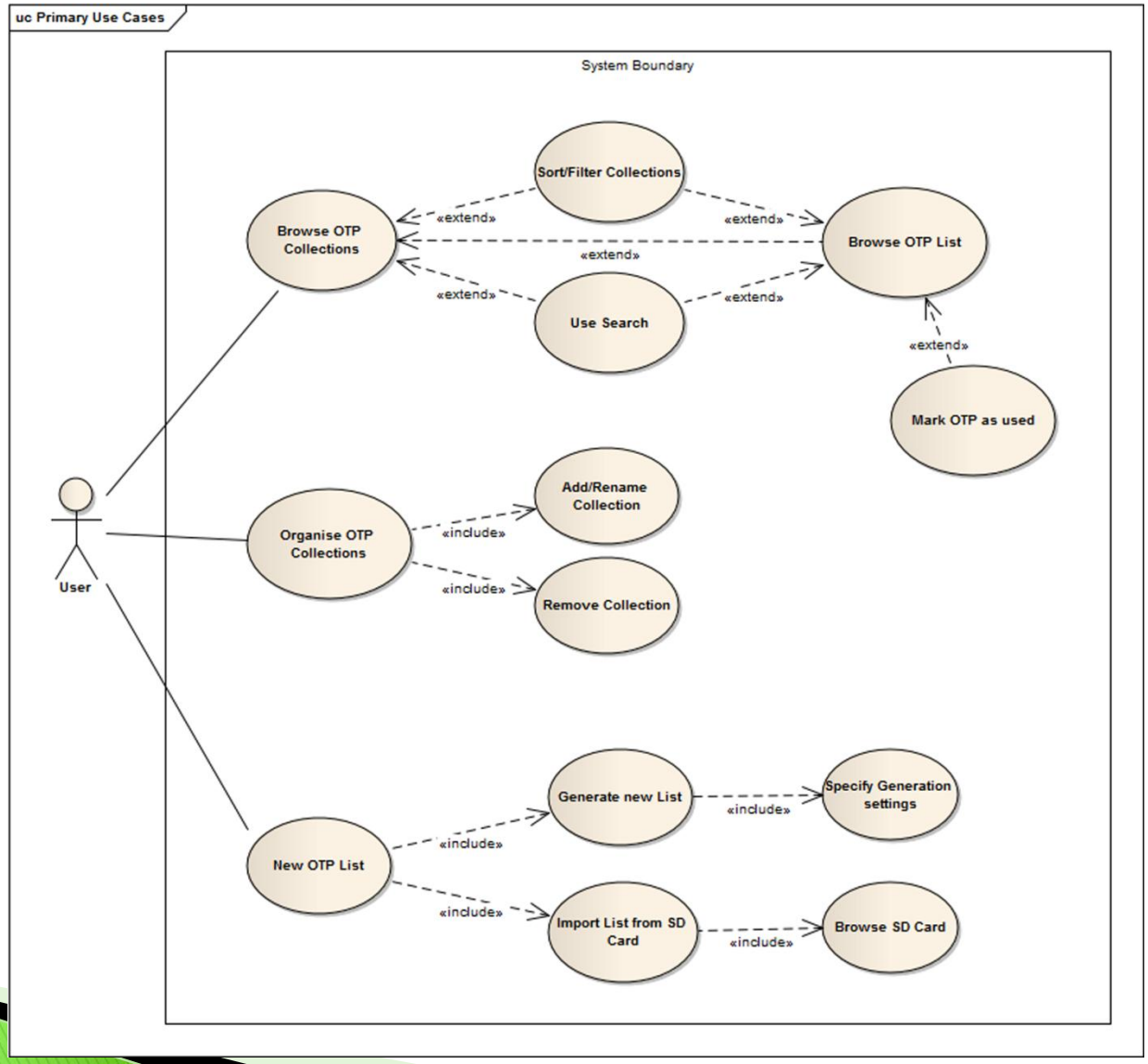
AndroidManifest.xml

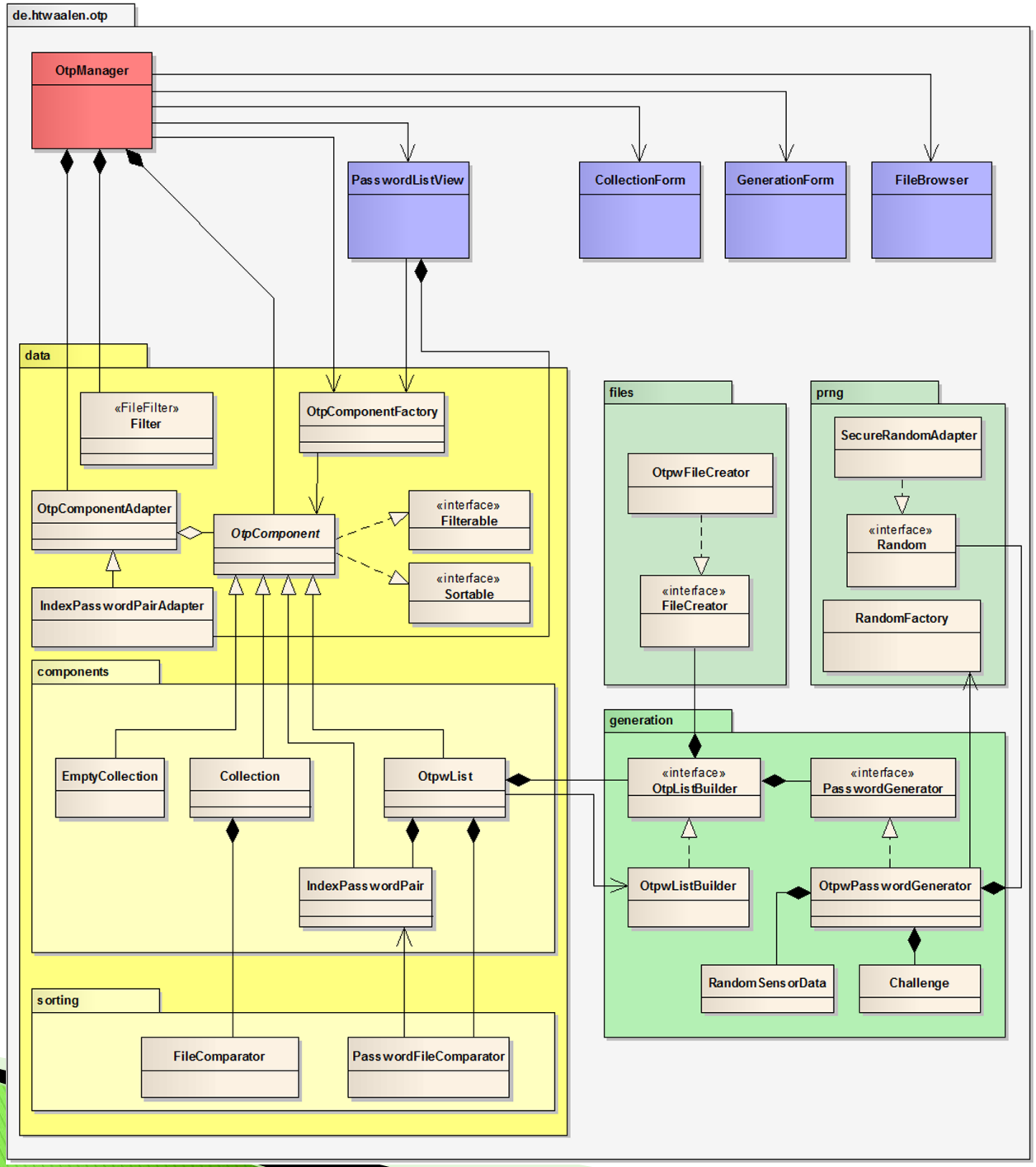
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.htwaalen.helloworld"
    android:versionCode="1"
    android:versionName="1.0">

    <application
            android:icon="@drawable/icon"
            android:label="@string/app_name">

        <activity android:name=".HelloWorldActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-sdk android:minSdkVersion="4" />

</manifest>
```

OTPManager App 

- Ausdrucken und mitführen von OTP Listen wird schnell unpraktikabel bei vielen Listen
- Idee: App zu Verwaltung und Generierung der OTP Listen
  - Erlaubt einfachen und schnellen Zugriff auf beliebig viele OTPW Listen
  - Architektur wurde offen für Erweiterung um weitere OTP-Systeme gehalten
  - Geringe Gefahr des vergessens, da man sein Handy meistens sowieso dabei hat

Anforderungen




de.htwaalen.otp
OtpManager
PasswordListView
CollectionForm
GenerationForm
FileBrowser
data
«FileFilter»
Filter
OtpComponentFactory
OtpComponentAdapter
OtpComponent
«interface»
Filterable
«interface»
Sortable
IndexPasswordPairAdapter
components
EmptyCollection
Collection
OtpwList
IndexPasswordPair
sorting
FileComparator
PasswordFileComparator
files
OtpwFileCreator
«interface»
FileCreator
prng
SecureRandomAdapter
«interface»
Random
RandomFactory
generation
«interface»
OtpListBuilder
«interface»
PasswordGenerator
OtpwListBuilder
OtpwPasswordGenerator
RandomSensorData
Challenge

Demo

Fragen?

Quellen

- Android Developers
  http://developer.android.com
- OTPW – A one-time password login package
  http://www.cl.cam.ac.uk/mgk25/otpw.html
- Icons der Präsentation
  http://www.large-icons.com/stock-icons/free-large-android-icons.htm