Implementierung einer Smartphone-Anwendung zum Austausch verschlüsselter Daten mit einer Cloud

10. Juli 2014

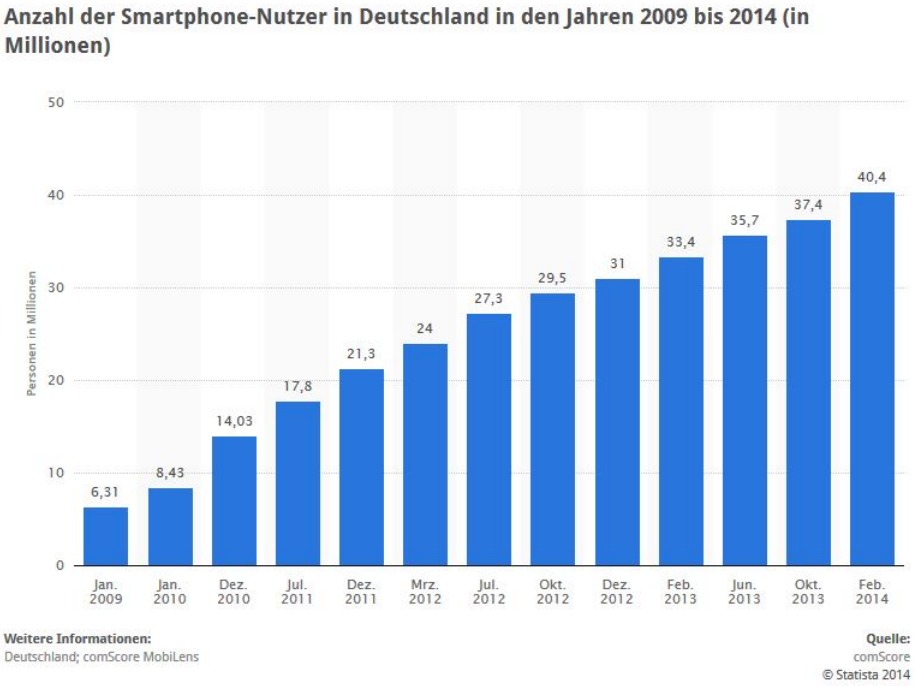
"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology."

Bruce Schneier

Inhaltsverzeichnis

1 Einleitung

"Die Computer- und Internetnutzer in Deutschland setzen seit Bekanntwerden der geheimdienstlichen Abhöraktionen häufiger Verschlüsselungsverfahren ein.Äus der Pressemitteilung der BITKOM geht weiterhin hervor, dass von Juli 2013 auf November 2013 insgesamt 1,1 Millionen mehr Bundesbürger ihre persönlichen Dateien verschlüsseln. Besonders wichtig ist der Aspekt der Sicherheit, wenn es sich bei den Daten um relevante oder firmeninterne Informationen handelt, wie es z. B. am in Hamburg der Fall ist. Auch der Austausch von Daten von mobilen Endgeräten wie oder Tables spielen eine immer größere Rolle wie die Entwicklung der letzten Jahre zeigt (siehe Grafik).



Herkömmliche Verfahren zum Austausch von Daten reichen oftmals nicht mehr aus, wenn man den Aspekt der Sicherheit näher beleuchtet.

1.1 Motivation

Am Deutschen Elektronen Synchrotron, im folgenden DESY, werden bisher wichtige und sensible Dokumente über ein Programm Namens Dropbox gesichert und verwaltet. Dropbox bietet eine plattformunabhänginge Möglichkeit Dokuente Online abzuspeichern und von einem anderen Standort über ein internetfähiges Gerät wieder zu öffnen [https://www.dropbox.com/]. Auch wenn Dropbox nach eigenen Angaben den Advanced Encryption Standard (AES) verwendet, bevor die Daten gespeichert werden, liegen die dafür notwendigen Schlüssel in Händen der Betreiber selbst, die somit vollen Klartextzugriff auf die Nutzerdateien haben. Dropbox begründet diesen Zugriff wie folgt: "Wie die meisten Online-Dienste verfügt auch Dropbox über einen kleinen Mitarbeiterstamm, dem aus in unserer Datenschutzrichtlinie dargelegten Gründen Zugriffsrechte auf Nutzerdaten gewährt werden muss [...]".
Da das DESY über eine eigene Cloud-Infrastruktur verfügt, sollen in Zukunft alle wichtigen Daten nicht nur in dieser Cloud gespeichert werden, sondern auch zusätzlich durch eine Verschlüsselung gesichert werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist es aus diesem Grund einen Prototyp zu entwickeln, der einerseits mit dem Cloud-System des DESY Kommunizieren kann um dort Dateien hoch- und herunter zu laden, andererseits diese Daten auch in angemessener Form (siehe Kapitel Validierung) zu Verschlüsseln.
In der ersten Version dieser Arbeit wird ein Programm entwickelt, welches auf Android-Betriebssystemen zum Einsatz kommen kann. Darüber hinaus ist es wichtig, dass die entsprechenden Schlüssel zum entschlüsseln der Daten nicht zusammen mit den Daten abgelegt werden, sondern ausschließlich den Parteien des Datenaustauschs bekannt sein soll. Dies bedeutet, das selbst die Betreiber am DESY nicht die Möglichkeit haben die abgelegten Daten zu entschlüsseln.
Zum ver- und entschüsseln der Daten sollen Verfahren verwendet werden, die in der heutigen Zeit als sicher angesehen werden und Smartphones im Bezug auf Performance und Akkuverbrauch nicht zu stark belasten. Um diese Faktoren zu Validieren wird eine Testanwendung geschrieben, die mit bestimmten Faktoren die verschiedenen Verfahren untereinander überprüfen (siehe Kapitel Validierung).

1.3 Verwandte Arbeiten

1.4 Verwandte Programme

1.5 Diese Arbeit

1.5.1 Inhaltlicher Aufbau

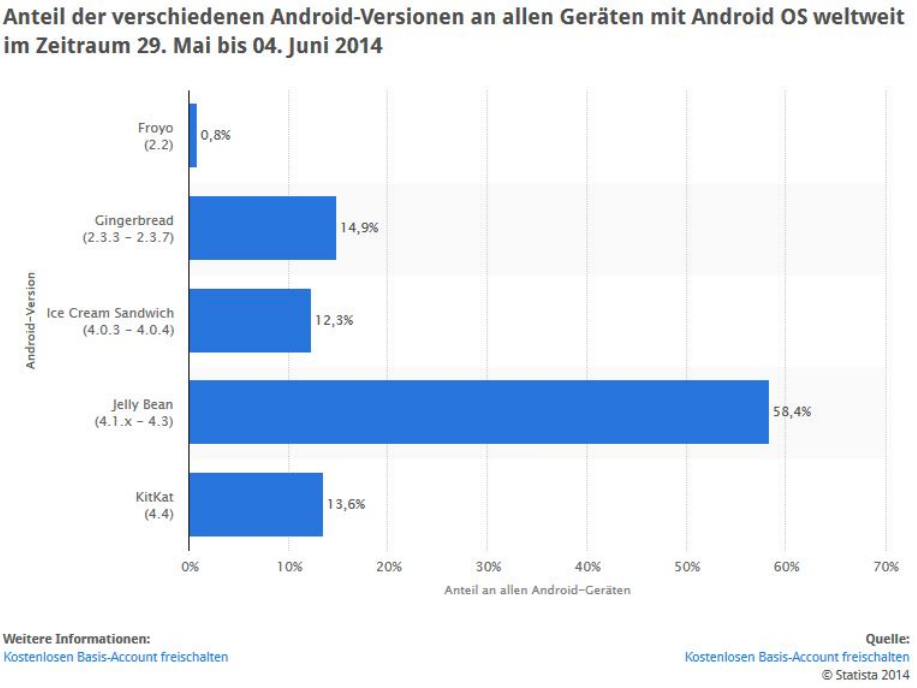
2 Grundlagen Allgemein

2.1 Cloud

2.2 Kryptografische Definitionen

3 Grundlagen Android

Android ist ein Betriebssystem für Smartphones und Tables, welches von der open handset alliance entwickelt wird. Das Konsortium besteht aktuell aus 84 Unternehmen, die an der Entwicklung des Betriebssytemes arbeiten. In diesem Kapitel wird kurz darauf eingegangen, welche Kryptografischen Aspekte Android in den verschiedenen Versionen zur Verfügung stellt um diese im darauffolgendan Kapitel genauer zu Untersuchen. Aufgrund der Tatsache, dass die Android Version Gingerbread (2.3.3) im Juni 2014 noch einen Marktanteil von knapp 15% hält, ist dies auch die niedrigste vom Programm unterstützte Verstion.



Bei der Analyse wird darauf geachtet, dass alle Funktionalitäten die im Programm entwickelt werden, von dieser Version unterstützt werden. Die, während des Schreibens dieser Arbeit, aktuellste Version der Android API ist KitKat (4.4), bei der darauf geachtet wird, dass die eingesetzten Funktionen auch in dieser Version noch zur Verfügung stehen und nicht mit *deprecated (veraltet)* markiert sind.

3.1 Zusammenhang Kryptographie

Beim Thema Sicherheit im Zusammenhang mit Android ist erstmals der Begriff der Sandbox zu nennen. Eine Anwendung wird abgekapselt in einer eigenen Umgebung mit eigenem Prozess, eigenem Betriebssytem-User, eigener Dalvik-VM, eigenem Heap und eigenem Dateisystem ausgeführt. Diese Abgekapselte Konstrukt wird Sandbox bezeichnet. Dadurch ist es dem Betriebssystem möglich unerlaubten Zugriff auf Ressourcen oder andere Programme zu beschränken, hierbei wird das Berechtigungs und Prozess-Management-System von Linux verwendet. Um dennoch verschiedene Zugriffe zu erlauben muss in der sogenannten Manifest-Datei der Anwendung die Berechtigung festgelegt werden und vom Benutzer bei der Erstinstallation bestätigt werden. Auch wenn dieses Konzept Daten zur Laufzeit innerhalb einer Anwendung schützt, ist es möglich Dateien auch auf einer SD-Karte zu speichern, in das Internet zu verschicken oder über andere Wege auszutauschen. Diese Daten sind dann ausserhalb der Anwendung und gegen externe Zugriffe nicht mehr geschützt. Dennoch gibt es die Möglichkeit in Android diese Daten zusätzlich mit einer Verschülsselung zu versehen - hierfür stellt das Paket javax.crypto eine Schnittstelle zur Verfügung,

mit dem der Programmierer die Möglichkeit hat verschiedene Kryptografische Verfahren aufzurufen, ohne die genaue Implementierung kennen zu müssen. Des Weiteren ist die Klasse abstrahiert von der darunter liegenden Umsetzung der Verfahren. In den tieferen Ebenen der Implementierung existieren verschiedene Provider, welche dann die einzelnen Implementierungen der Verfahren zur Verfügung stellen. Mit folgendem Codeabschnitt ist es möglich die einzelnen Provider mit den unterstützten Verfahren auszulesen.

```
Provider[] providers = Security.getProviders();
for (Provider provider : providers) {
    Log.i("CRYPTO","provider: "+provider.getName());
    Set<Provider.Service> services = provider.getServices();
    for (Provider.Service service : services) {
        Log.i("CRYPTO","  algorithm: "+service.getAlgorithm());
    }
}
```

Im Vergleich stehen folgende Android-Versionen:

- 2.3.3 (Gingerbread) : die niedrigste vom Programm unterstützte Version
- 4.1.1 (Jelly Bean) : die Version des Entwickler-Gerätes
- 4.4.4 (KitKat) : aktuellste auf dem Markt verfügbare Android-Version
- "L" : zukünftige Version, welche bereits zu testzwecken als Entwickler-Version freigeschaltet ist. Der CodeName "L" zeigt auf, dass es sich in der Folge der Süßigkeiten (Gingerbread, HoneyComb, Ice Cream Sandwich, Jelly Bean, KitKat) vermutlich alphabetisch fortsetzen wird - die Versionsnummer ist bis dato nicht bekannt.

Im Vergleich der Ausgabe eines Gerätes mit Android 2.3.3 und eines mit 4.1.1 bzw. 4.4.4 und "L" liegt der Hauptunterschied in der Unterstützung von Elliptischen Kurven für das Diffie-Hellmann-Verfahren und den Digital-Signature-Algorithm, welche in der Version 2.3.3 nicht unterstützt werden. Des Weiteren ist ab der Version 4.4.x der Provider OpenSSL und deren Algorithmen genauer aufgesplittet. Folgende Verschlüsselungsverfahren werden sowohl von der Version 2.3.3 als auch von der Version 4.4 unterstützt und werden im nachfolgenden Kapiter näher erläutert: AES, ARC4, Blowfish, DES, Triple DES, PBE, RSA. Des Weiteren wird die Möglichkeit des Key-Wrap für AES und Triple DES beschrieben. Die von Android unterstützten Hash-Funktionen sind: MD5, SHA1, SHA256, SHA384, SHA512. Ausserdem unterstützt wird der Authentifizierungsalgorithmus HMAC und der Standard X509. Die Vollständige Liste aller Unterstützen Algorithmen mit dessen Providern befindet sich im Anhang.

3.1.1 OpenSSL

3.1.2 Bouncy Castle

4 Grundlagen Kryptographie

4.1 Verschlüsselung

4.1.1 Symetrische Verfahren

4.1.2 Asymetrische Verfahren

4.2 Hash-Funktionen

4.3 Digitale Signature

4.3.1 Public Key Infrastruktur

4.4 Schlüsselvereinbarung

4.4.1 Diffie Hellmann

4.4.2 ElGamal

4.5 Schlüsselgenerierung

4.6 Authentifizierung

4.6.1 Zwei-Faktor-Authentifizierung

5 Validierung

5.1 Verschlüsselungsverfahren

5.2 Hashfunktionen

6 Implementierung

6.1 Entwurf

7 Test

7.1 Validierung

7.2 Testverfahren

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

8.2 Ausblick

Index