1 PRIVACY – AVARE: Selbstdatenschutz für Bürger mithilfe von Open-Source-Software

Autor: Sascha Alpers, Enes Erdogan, Stefan Gapp, Andreas Fritsch, Ainara Miller-Askar, Andreas Oberweis, Gunther Schiefer, Manuela Wagner

1.1 Einleitung

Das Kernanliegen des Datenschutzes ist es, natürliche Personen vor nachteiligen Effekten der Speicherung und Verarbeitung der sie betreffenden Daten zu schützen. Aber viele Personen scheinen gar nicht geschützt werden zu wollen. Im Gegenteil, viele Endanwender willigen "freiwillig" – bewusst oder unbewusst – in eine umfassende Verarbeitung ihrer personenbezogenen Daten ein. Warum tun Menschen dies? Es werden verschiedene Ursachen diskutiert (beispielsweise in [GVG17]), hierzu gehören Uninformiertheit, mangelnde Sensibilität, das Gefühl der Hilflosigkeit, mangelnde Zahlungsbereitschaft und mangelnde Alternativen. Auch wenn dies in Einzelfällen zutrifft, so gibt es oft sehr wohl datenschutzfreundliche Alternativen. Beispielsweise existiert zu WhatsApp (als Instant Messaging App) die Alternative Threema. Threema gilt als EU-DS-GVO-konform und funktional durchaus mit WhatsApp vergleichbar [Ell18]. Allerdings ist inzwischen die aktuelle Netzwerkgröße ein entscheidendes Auswahlkriterium: Im Januar 2018 hatte Threema 4,5 Millionen Nutzer [Sta18b], WhatsApp dagegen 1,5 Milliarden [Sta18a]. Dies ist ein Indiz dafür, dass WhatsApp sich quasi zum De-facto-Standard entwickelt hat und es für die einzelne Person nur schwer möglich ist, viele andere "zum Wechsel auf ein anderes Produkt zu bewegen. [...] Bei Diensten mit Nutzerzahlen im Milliardenbereich kann von 'Freiwilligkeit' nur noch bedingt gesprochen werden." [Alt18]

Es ist daher sinnvoll, nach Wegen zu suchen, welche die Teilhabe am sozialen digitalen Leben ermöglichen und dennoch personenbezogene Daten besser schützen. Dazu können und müssen regulatorische Ansätze weiterverfolgt werden [Küh17]; bis zu deren Wirksamkeit verfolgt AVARE dieses Ziel mit technischen Mitteln innerhalb rechtlicher Schranken [APW17]. Dabei möchte AVARE den Endnutzern ermöglichen, ihre persönlichen Präferenzeinstellungen zum Umgang mit den von ihnen bzw. ihrem Smartphone erhobenen Daten einmal zu beschreiben und an vielen Stellen durchzusetzen [AOP+17]. Analog zum bekannten Slogan *"Write once, run anywhere!"* (mit dem Sun Microsystems für die plattformübergreifenden Einsatzmöglichkeiten von Java-Programmen geworben hat) kann man hier als Ziel formulieren: *"Declare once, enforce anywhere!"*. Dabei geht es nicht um die Daten, die der Nutzer selbst und unmittelbar innerhalb einer App eingibt, sondern um die Daten, welche eine App durch Abfragen von Programmierschnittstellen (englisch: "application programming interface", kurz API) oder von Sensoren erhält.

Zu den Zielen von AVARE gehört auch, auf vorhandene Open-Source-Komponenten zurückzugreifen und die eigenen Ergebnisse als Open Source (soweit möglich unter der Apache-2.0-Lizenz [The04]) zu veröffentlichen.

1.2 Herausforderungen

1.2.1 Herausforderung Datenschutz (für den Bürger)

Personenbezogene Daten werden oftmals auf Grundlage sehr weit gefasster Einwilligungserklärungen erhoben und verarbeitet, ohne dass die betroffenen Personen praktisch nutzbare, feingranulare Wahlmöglichkeiten haben [BK17]. Zwar ist es seit Android-Version 6 möglich, einzelnen Apps den Zugriff auf bestimmte Datenkategorien zu entziehen, es verbleiben jedoch zwei wesentliche Probleme: 1. Einige Apps funktionieren danach nicht mehr (obwohl die gesperrten Daten für die Hauptfunktionalität nicht erforderlich wären); 2. Der Zugriff auf einzelne Daten innerhalb einer Kategorie ist für die Nutzung der App gewünscht (beispielsweise Name und Mobilfunknummer der Kontakte mit denen der Nutzer mit einem Messenger kommunizieren möchte), der Nutzer kann aber den Zugriff auf weitere Daten der Kategorie (Geburtsdatum, Kontakte, mit denen der Messenger nicht genutzt werden soll) nicht untersagen.

Auch mit dem Paradigma des Privacy by Design und mit datenschutzfreundlichen Voreinstellungen sollen spätestens seit der EU-DS-GVO betroffene Personen in die Lage versetzt werden, selbst den Grad der Preisgabe persönlicher Daten bestimmen zu können. Hier muss sich jedoch zunächst noch herausbilden, wie feingranular derartige Einstellungsmöglichkeiten bereitgestellt werden sollen.

Eine zentrale Herausforderung ist weiterhin die Bedienbarkeit von Datenschutzlösungen. Viele existierende Lösungen verfehlen grundlegende Bedienbarkeitsziele wie Intuitivität und leichte Erlernbarkeit [AHIC15]. Darüber hinaus verlangen oft schon die Installation und Einrichtung von den Nutzern einen hohen Grad an technischer Expertise. So setzen beispielsweise die vergleichsweise mächtigen Berechtigungsmanager für Android XPrivacy und LBE Security Master ein gerootetes Gerät und die Installation von weiteren Abhängigkeiten voraus [ABF+16].

Für den Bürger wäre es am bequemsten, an einer Stelle (die eine gute Usability bieten muss) die Datenschutzpräferenzen einfach – aber so feingranular wie gewünscht – zu beschreiben. Diese Datenschutzpräferenzen gegenüber verschiedensten Diensten immer wieder neu und meist mit anderen Möglichkeiten erklären zu müssen, ist angesichts der Vielzahl der genutzten Dienste nicht zu leisten. Bis dies möglich ist, ist es sinnvoll, die gebotenen Einstellungsmöglichkeiten so gut wie möglich zu nutzen und – wo diese fehlen – auf Drittlösungen zu setzen. Für Android-Nutzer, welche WhatsApp verwenden möchten, existiert beispielsweise WhatsBox von der Backes SRT GmbH mit der Möglichkeit, einzelne Kontakte vor WhatsApp zu verbergen[1].

[1] Gemäß E-Mail-Kontakt mit dem Hersteller wird dazu der Quellcode von WhatsApp nicht manipuliert. Allerdings ist es gegenwärtig nur möglich, ganze Kontakte zu verbergen (horizontaler Filter). Das Filtern einzelner Attribute ist noch nicht möglich, jedoch wurde als Feature Request in die Weiterentwicklung aufgenommen, nur noch Namen und Nummern weiterzuleiten (E-Mail-Kommunikation mit der Backes SRT GmbH vom 24. und 27. August 2018).

1.2.2 Herausforderungen aus rechtlicher Sicht (Anforderung an das Projekt)

Die Entwicklung einer technischen Selbstdatenschutzlösung bedarf der rechtsgebietsübergreifenden Betrachtung. Die konkrete technische Umsetzung darf insbesondere keine Urheberrechte verletzen, wenn zur Kontrolle des Datenflusses Betriebssystem und/oder Apps anderer Anbieter modifiziert werden [BRRR13]. Da jede Umarbeitung fremden Codes grundsätzlich der Zustimmung des Urhebers bedarf [Bru14], fokussiert das Projekt AVARE auf das Android-Betriebssystem (das unter einer Open-Source-Lizenz lizenziert ist, welche diese Zustimmung enthält).

Daneben sollte die Lösung datengetriebene Geschäftsmodelle nicht in einer Weise behindern, die als unlauter qualifiziert werden könnte [APW17]. Dies betrifft weniger das Blockieren und Filtern der Daten – soweit man nicht der im juristischen Schrifttum postulierten Annahme eines vertraglich bindenden Gegenleistungsverhältnisses zwischen Dienst und Daten bei vermeintlich kostenlosen Angeboten folgt [Met17]. Jedoch ist bei der Bereitstellung generierter Daten (zur Verschleierung der Datenblockade bei andernfalls zu befürchtendem Funktionsausfall) zu bedenken, inwieweit sich Nutzer rechtswidrig verhalten könnten [AOP+17]. Daher wurde ein besonderes Augenmerk auf die Auswahl und Einsatzmöglichkeit derartiger "Ersatzdaten" im Sinne einer präventiven Steuerung gelegt [ABF+18].

Des Weiteren muss eine App zum Privatsphärenschutz selbst datenschutzkonform ausgestaltet sein. Das Projekt folgt hierbei einem stringenten Datenminimierungsgrundsatz. Daneben kommen Verschlüsselungsmethoden nach dem Stand der Technik zum Einsatz. Insbesondere werden nur verschlüsselte Daten an den AVARE-Server übertragen und/oder auf dem Gerät gespeichert, und der Schlüssel wird vor dem Server und dem Betreiber geheim gehalten.

1.3 Client-Implementierung

1.3.1 Konzept

Das Lösungskonzept von AVARE besteht aus einem Client und einem Server; der Client wiederum besteht aus einer Durchsetzungskomponente und einer UI- und Synchronisationskomponente (Abbildung 1.1). Die **Synchronisationskomponente** und der **Server** – selbst auch Open Source – haben die Aufgabe, den Austausch symmetrisch verschlüsselter Präferenzprofile zu ermöglichen. Der Schlüssel zum Entschlüsseln der Präferenzprofile wird direkt mithilfe der **UI** zwischen Endgeräten ausgetauscht. Weder Server noch Betreiber sollen den Schlüssel erfahren. Die UI-Komponente hat auch die Aufgabe, den Bürger zu befähigen, seine Datenschutzpräferenzen zu beschreiben. Bei der Entwicklung dieser Komponente wurde auf eine hohe Usability Wert gelegt.

Den Schwerpunkt des vorliegenden Beitrags bildet die **Durchsetzungskomponente** für Android. Sie besteht aus einer Sandbox und einem Referenzmonitor. Eine Sandbox ist ein isolierter Bereich, innerhalb dessen eine Anwendung keine Auswirkung auf die Umgebung hat [GWTB96]. Die Anwendung wird in einer Sandbox gestartet und sendet ihre Anfragen nicht direkt an das Betriebssystem, sondern über den Referenzmonitor, welcher die Kommunikation zwischen der Anwendung in der Sandbox und dem Betriebssystem übernimmt. Um diese

Kommunikation entsprechend den Datenschutzpräferenzen des Nutzers anzupassen, wird das sogenannte Hooking-Verfahren eingesetzt. Hooking ist eine Technik, welche das Hinzufügen von Quellcode zum Systemaufruf ermöglicht. Dies geschieht typischerweise so, dass der vordefinierte Funktionszeiger durch einen anderen Funktionszeiger ersetzt wird [Ngu04]. Der Referenzmonitor ist mithilfe des Hookings der Betriebssystemaufrufe in der Lage, die Anfragen der App an das Betriebssystem zunächst weiterzuleiten und die Antworten des Betriebssystems entsprechend der Nutzerpräferenzen zu filtern bzw. zu manipulieren und erst anschließend der App zu übermitteln.

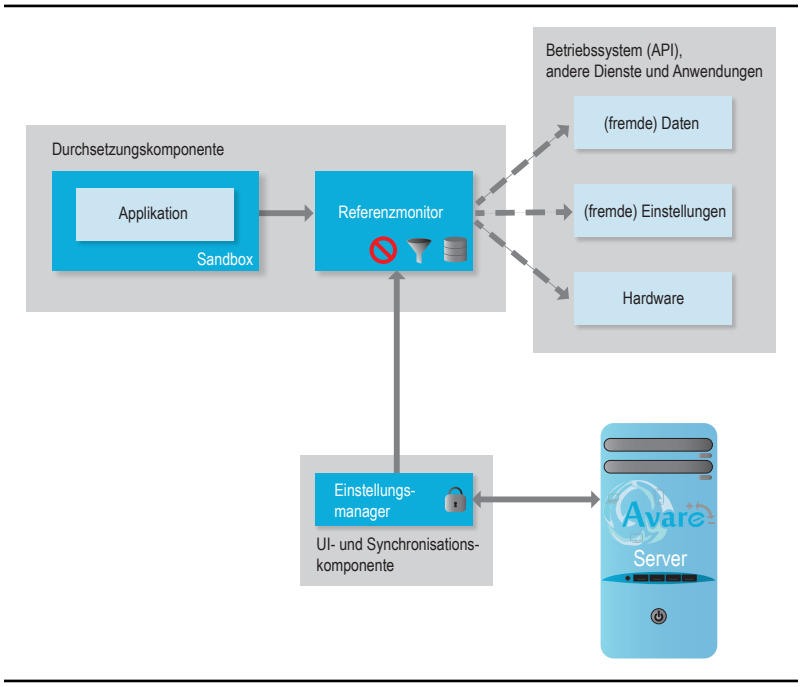


Figure 1.1: Lösungskonzept der Client-Anwendung von AVARE; vgl. [APW17].

Sandbox und Referenzmonitor bilden zusammen die Client-Anwendung mit dem Namen AVARE-Box, welche die Anwendungen intern bei sich installiert, um sie dann in einer kontrollierten Umgebung ausführen zu können. Die AVARE-Box hat Zugriff auf die vom Benutzer definierten Einstellungen für die jeweiligen Daten und Anwendungen. Somit werden die Einstellungen ausgelesen, daraus entsprechende Filterregeln in Form eines AVARE-Box-Plug-ins automatisch generiert und diese Regeln bei den Anfragen von der Anwendung in der AVARE-Box an das Betriebssystem angewendet.

1.3.2 Integrierte existierende Open-Source-Komponenten

Zur Implementierungsstrategie von AVARE gehört es, auf bestehenden Open-Source-Komponenten aufzusetzen, diese weiterzuentwickeln und mit anderen Komponenten und eigenen Entwicklungen zu verknüpfen. Dies soll so geschehen, dass das Gesamtsystem AVARE in einem einzelnen Quellcode-Repository[2] als Open Source veröffentlicht werden kann, um die künftige Nutzung und Weiterentwicklung zu vereinfachen und so zu fördern.

[2] gegenwärtig unter https://github.com/fzi-forschungszentrum-informatik/PRIVACY-AVARE

Sandboxing- und Hooking-Framework

Das Sandboxing- und Hooking-Framework von AVARE basiert auf dem GitHub-Projekt VirtualApp [asL16]. Wir haben dabei auf den Branch des Forks VirtualHook [rk717a] mit Stand vom 21.08.2017 [rk717b] und auf das Hauptprojekt VirtualApp [asL16] mit Stand vom 29.10.2017 [asL17] aufgesetzt. Diese Versionen sind von den Autoren jeweils unter GNU General Public License Version 3 (GPLv3) [Fre07] lizenziert. Die aktuellen Versionen von VirtualApp und VirtualHook haben diese Lizenzangabe nicht mehr. Es kann an dieser Stelle darauf verzichtet werden, zu erörtern, ob die neueren Versionen unter GPLv3 veröffentlicht werden müssten (wenn einer der Urheber der alten Version der Verwendung seines Quellcodes unter einer proprietären Lizenz nicht zugestimmt hätte), weil AVARE auf der alten -– unter GPLv3 lizenzierten – Version aufsetzen konnte. Dazu wurden der Branch des Forks und das Hauptprojekt selbstständig wieder zusammengeführt und dabei auftretende Konflikte gelöst. Zusätzlich wurde das YAHFA-Framework [rk717c] im Projekt auf den Stand vom 03.07.2018 [rk718] aktualisiert. Diese Version des YAHFA-Framework ist ebenfalls unter GPLv3 lizenziert (vgl. Abbildung 1.2).

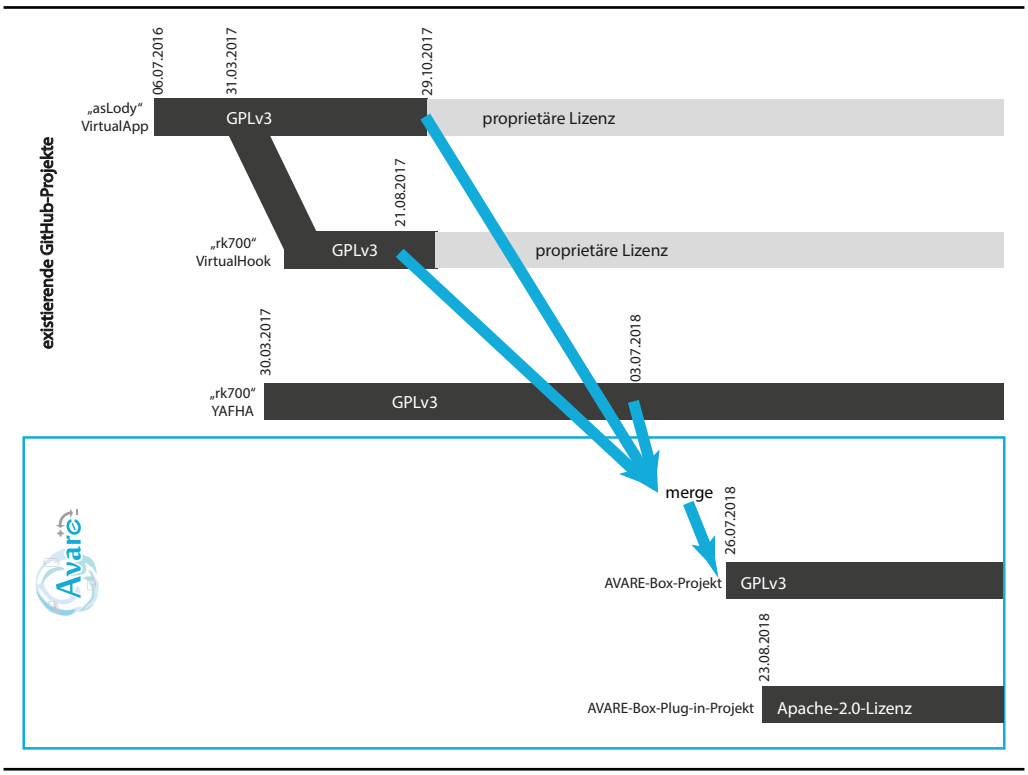


Figure 1.2: Open-Source-Beiträge zur Durchsetzungskomponente von AVARE.

AVARE-Box enthält Java, C++ und C Klassen, die das Ausführen einer Anwendung in einer Sandbox ermöglichen. AVARE-Box hat, wie jede Anwendung in Android, ihren eigenen Adressraum. Eine der wichtigsten Klassen ist die Klasse VirtualCore, welche die Prozesse initialisiert und die Ausführung der Anwendungen steuert. Dazu gehört auch, dass für eine innerhalb der AVARE-Box auszuführende App die Umgebung manipuliert wird. Dazu wird mithilfe des Java Native Interface (JNI) die C-Bibliothek HookZz aufgerufen, welche die Adressraumumleitung und Speicherallokation durchführt. Somit kann AVARE-Box die Kommunikation zwischen den ausgeführten Anwendungen und dem Betriebssystem kontrollieren.

Das YAHFA-Framework ermöglicht das Hooking von den Java-Methoden des Android-Frameworks. Hooking ermöglicht, eine Methode des Android-Frameworks mit einer eigenen Methode zu überschreiben. Dabei kann innerhalb der eigenen Methode die ursprüngliche Methode aufgerufen werden und vor bzw. nach dem Aufruf der Ursprungsmethode die Anfrage bzw. Antwort manipuliert werden. Dabei können aus einer Klasse des Android-Frameworks einzelne Methoden überschrieben werden und andere weiterhin aus der ursprünglichen Klasse aufgerufen werden.

Unter GitHub[3] wurde die neu entstandene Version unter GPLv3 mit Referenz auf die Vorarbeiten veröffentlicht.

1.3.3 Lizenzierung von AVARE

Alle im Rahmen des Projektes AVARE prototypisch neu entwickelten Komponenten werden jeweils unter der "Apache License 2.0" [The04] veröffentlicht. Es wurde bewusst eine Non-Copyleft-Lizenz ausgewählt, um eine Wiederverwendung der neuen Komponenten in verschiedenen Szenarien und unter verschiedenen kompatiblen Lizenzen bzw. auch in kommerziellen Produkten zu ermöglichen. Insbesondere ist auch eine Wiederverwendung in GPLv3-lizenzierten Projekten möglich [GGB09, Smi07], sodass das Gesamtsystem in jedem Fall unter GPLv3 (einer Copyleft-Lizenz) verwendet werden kann.

1.3.4 AVARE-Box

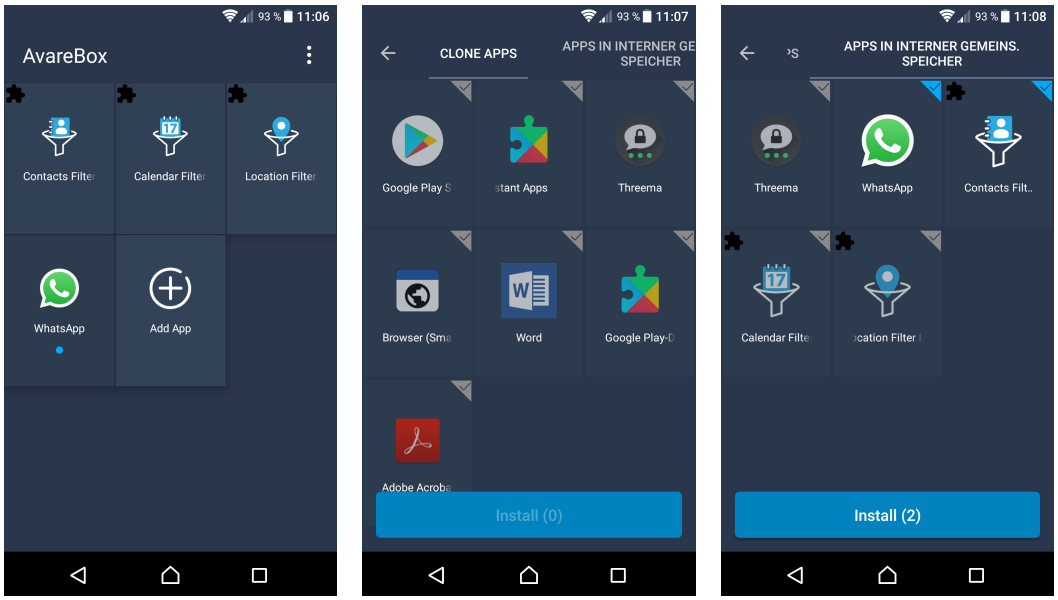
Die AVARE-Box ist eine Anwendung, in welcher man die gewünschten Apps installieren und auf einer virtuellen Ebene ausführen kann. In Abbildung 1.3 ist die grafische Oberfläche der AVARE-Box dargestellt. Nach dem Start der AVARE-Box ist zunächst eine Liste der innerhalb der AVARE-Box installierten Plug-ins (in der Abbildung 1.3a drei Stück) und Anwendungen (hier WhatsApp) dargestellt. AVARE-Box steuert die Kommunikation zwischen den Anwendungen und dem Betriebssystem. Um die Datenfilterung für unterschiedliche Datenkategorien durchzuführen, werden Regeln innerhalb von Plug-ins definiert. Durch Drücken des Plus-Zeichens werden alle auf dem Endgerät installierten bzw. als APK-Datei verfügbaren Anwendungen aufgelistet, wie in Abbildungen 1.3b und 1.3c zu sehen ist. Man kann eine oder mehrere Anwendungen auswählen und in die AVARE-Box installieren.

Zur AVARE-Box wurden bereits drei Plug-ins implementiert, welche die Daten entsprechend den vom Benutzer eingegebenen Einstellungen filtern können: Contacts-Filter-Plug-in, Calendar-Filter-Plug-in und Location-Filter-Plug-in. Diese Plug-ins können ebenfalls aus der Liste nach dem Drücken des Plus-Zeichens ausgewählt und installiert werden.

1.3.5 Plug-in: Adressbuchfilter

Um Zugriffe auf die Kontaktliste des Geräts zu manipulieren, benutzt AVARE-Box die Hooking-Funktionalität des YAHFA-Frameworks, um den Zugriff auf Methoden abzufangen, die auf die Kontaktliste zugreifen. Genauer gesagt wird hier die Query-Methode der Klasse android.content.ContentResolver überschrieben.

[3] https://github.com/fzi-forschungszentrum-informatik/PRIVACY-AVARE/tree/master/AvareBoxProject



(a) Liste aller in AVARE-Box installierten Anwendungen.

(b) Liste aller auf dem Gerät installierten Anwendungen.

(c) Liste aller APK-Dateien auf dem internen gemeinsamen Speicher.

Figure 1.3: Grafische Oberfläche der AVARE-Box.

Dazu wurde eine Unterklasse der Klasse Cursor, welche von der Query-Methode zurückgegeben wird, erstellt, die exakt wie die Klasse Cursor des Ergebnisses des eigentlichen Query-Aufrufs funktioniert und nahezu dieselben Daten enthält – mit dem Unterschied, dass gewisse Informationen gemäß den Datenschutzpräferenzen verändert zurückgegeben werden. Diese Veränderungen entstehen durch das Überschreiben der jeweiligen Methoden in der durch das Adressbuchfilter-Plug-in definierten Cursor-Klasse; konkret wird die GetString-Methode von Cursor überschrieben, da die ursprüngliche Klasse Cursor ebendiese Methode verwendet, um auf Daten wie Kontaktname und Telefonnummer zuzugreifen.

Um der aufrufenden Anwendung nun das Ergebnis des Cursors zu liefern, wird zunächst die Query-Methode des ursprünglichen Cursors ausgeführt; das Ergebnis wird der aufrufenden Anwendung jedoch noch nicht zurückgegeben. Stattdessen wird, wie oben beschrieben, ein neuer Cursor mit dem Ergebnis des ursprünglichen Cursors erstellt, welcher jedoch eine überschriebene GetString- bzw. MoveToNext-Methode besitzt. Dies hat den Effekt, dass jedes Mal, wenn der Cursor auf die Kontaktdaten mithilfe der entsprechenden Methode zugreift, nicht die ursprüngliche Methode ausgeführt wird (und demnach alle Kontaktdaten aus der Liste zurückgegeben werden), sondern die überschriebene GetString-/MoveToNext-Methode, die nur diejenigen Daten aus der Liste zurückgibt, die gemäß den Datenschutzpräferenzen des AVARE-Nutzers mit der Anwendung geteilt werden sollen. Somit entsteht eine weitere Schicht zwischen der aufrufenden Anwendung und der Kontaktliste mithilfe des selbst erstellen Cursors.

In AVARE-Box werden zwei Arten der Kontaktdatenfilterung unterschieden:

1. Vertikale Filterung: Die vertikale Filterung manipuliert die Rückgabe einzelner Attribute der Kontakte auf dem Gerät. So können einem Instant Messenger wie WhatsApp beispielsweise

nur Vorname und Mobilfunknummer, aber nicht das Geburtsdatum oder die E-Mail-Adresse der Kontakte übergeben werden. Die vertikale Filterung wird dadurch realisiert, dass die zu verbergenden Attribute auf eine leere Zeichenkette gesetzt werden. Zum Beispiel wird für das Herausfiltern der Nachnamen der Kontakte das Attribut FAMILY_NAME bei jedem Kontakt auf eine leere Zeichenkette gesetzt. Im Falle des Kontaktes "Jonas Weingaertner" würde also nur "Jonas" zurückgegeben werden.

2. Horizontale Filterung: Die horizontale Filterung versteckt einzelne Kontakte komplett vor der aufrufenden Anwendung. Statt eines Adressbuchs mit allen Einträgen (beispielsweise "Sophia-Marie Koch", "Jonas Weingaertner" und "Daniel Schuster") werden nur freigegebene Einträge (beispielsweise nur "Sophia-Marie Koch" und "Jonas Weingaertner") zurückgeliefert. Hierzu wird nicht wie bei der vertikalen Filterung die GetString-Methode überschrieben, sondern die MoveToNext-Methode des Cursors. Hier wird jedes Mal, wenn der Cursor über die Ergebnisse der Abfrage iteriert, der Kontaktname des aktuellen Kontakts abgerufen (mithilfe der GetString-Methode des ursprünglichen Cursors) und dann überprüft, ob der zurückgegebene Kontakt in dieser Zeile des Cursors zu verbergen ist. Falls nein, wird die Zeile weitergegeben. Falls ja, wird die Zeile übersprungen und für die nächste Zeile erneut geprüft, ob der Kontakt weitergegeben werden darf.

Der horizontale Filter und der vertikale Filter können auch miteinander kombiniert werden.

1.3.6 Plug-in: Positionsfilter

Mithilfe des Positionsfilter-Plug-ins ist es möglich, die aktuelle Geoposition des Benutzers zu verbergen und stattdessen eine andere Position an Apps zu übergeben. Die von AVARE neu erzeugte Position liegt dabei in einem konfigurierbaren Umkreis zur tatsächlichen Position des Benutzers. So kann beispielsweise eine App zur Wettervorhersage weiter genutzt werden, hier ist es in der Regel nicht erheblich, wo genau sich ein Nutzer befindet, da für eine Position in der Nähe (beispielsweise 5 km entfernt) die gleiche Wetterprognose ausgegeben wird.

Um diese Funktionalität zu implementieren, werden die Funktionsaufrufe android.location. Location.getLatitude und android.location.Location.getLongitude mit dem oben beschriebenen Verfahren gehookt. In der gehookten Methode wird dann zunächst festgestellt, ob bereits zuvor eine verschleierte Position berechnet wurde. Falls nicht, wird diese initialisiert. Zur Initialisierung wird zunächst die reale Position ermittelt, ohne dass diese an die Endanwendung übermittelt wird. Von dieser Position aus wird nun zufällig ein Winkel im Intervall [0,360) Grad gewählt. Anschließend wird zufällig eine Entfernung zwischen dem vom Nutzer konfigurierbaren minimalen und maximalen Radius gezogen. Die initial bestimmte verschleierte Position ergibt sich dann aus diesen Werten, indem man von der realen Position aus die gezogene Entfernung in die gewählte Richtung geht. Falls die so gewählte Position in einem anderen Land liegt, werden neue Zufallswerte für den Winkel und die Entfernung gezogen.

Wenn schon zuvor eine verschleierte Position berechnet wurde, wird ein modifiziertes Vorgehen angewandt: Zunächst wird wieder zufällig ein Winkel im Intervall [0,360) Grad gewählt. Dann wird die real zurückgelegte Distanz zwischen diesem und dem letzten Funktionsaufruf berechnet. Diese wird verwendet, um eine neue Position in der Richtung des gewählten Winkels zu erzeugen. Dieses Vorgehen ermöglicht eine Erhaltung der Fortbewegungsgeschwindigkeit: Wenn der Nutzer sich also zwischen zwei Funktionsaufrufen um beispielsweise drei Meter bewegt hat, wird sich auch die verschleierte Position um drei Meter ändern. Hierbei wird wieder

darauf geachtet, dass sich die neue verschleierte Position innerhalb der definierten Radien und im selben Land befindet.

Im Rahmen der Antwort kann auch der Unschärferadius mit übermittelt werden, um offenzulegen, dass es keine genaue Position ist. Diese Möglichkeit sieht die Positions-API von Android bereits vor, weil es auch bei der normalen Positionsbestimmung zu Unsicherheiten kommt und daher zu jeder Position ein solcher Radius mit angegeben wird.

1.3.7 Plug-in: Kalenderfilter

Das Kalenderfilter-Plug-in ermöglicht die Filterung der Kalenderdaten, die wiederum in zwei Arten zu unterscheiden ist. Bei einer horizontalen Filterung werden Kalendereinträge ausgewählt, die nicht weitergegeben werden sollen. Diese Kalendereinträge können beispielsweise alle Kalendereinträge des privaten Kalenders sein. Bei einer vertikalen Filterung werden einzelne Attribute nicht übermittelt. Dies kann auch bedeuten, nur den Tag eines Termins (nicht aber die Uhrzeit) oder nur die Stadt, nicht aber die genaue Anschrift eines Terminortes zurückzugeben.

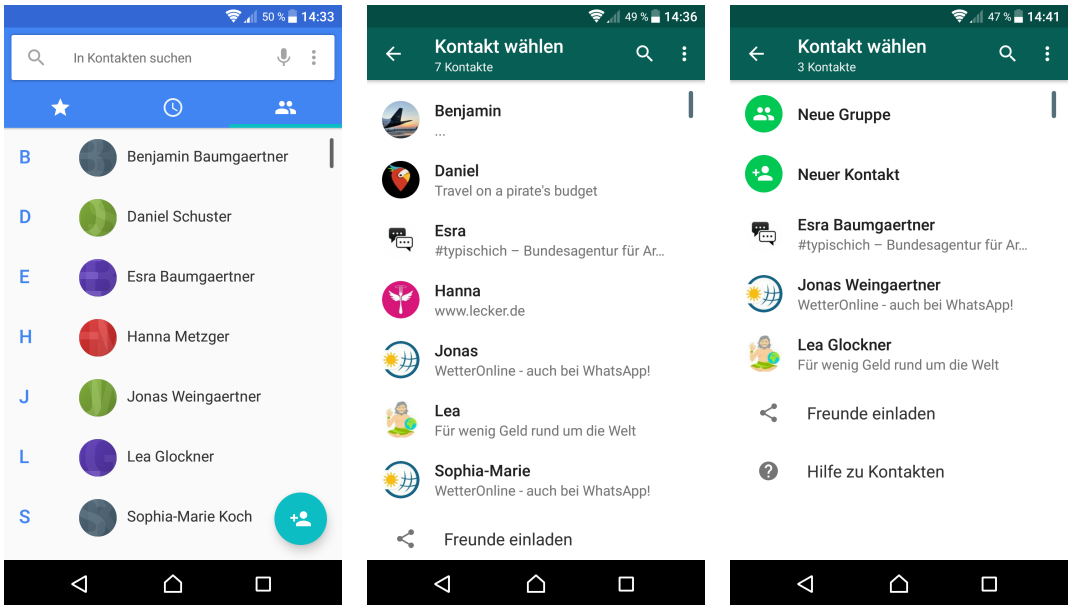
1.4 Beispiel

In diesem Anwendungsfall wird die Nutzung von AVARE am Beispiel der Adressbuchdatenfilterung vorgestellt.

In Abbildung 1.4a ist das Adressbuch in der vorinstallierten Kontakte-Anwendung auf Android dargestellt. Dafür muss das entsprechende Plug-in "Contacts Filter Plug-in" installiert und konfiguriert werden. Der WhatsApp-Anwendung sollen als vertikale Filterung nur die Vornamen der Kontakte weitergegeben werden. Das Ergebnis ist in Abbildung 1.4 zu sehen.

Wenn eine horizontale Filterung gewünscht ist, werden bestimmte Dateneinträge nicht weitergegeben. Zum Beispiel wird die Weitergabe der Adressbucheinträge mit den Namen "Benjamin Baumgaertner", "Daniel Schuster", "Hanna Metzger" und "Sophia-Marie Koch" nicht gewünscht. In Abbildung 1.4c ist das Ergebnis zu sehen: Nur die von den Nutzerdatenschutzpräferenzen freigegebenen Kontakte sind für die WhatsApp-Anwendung im Rahmen eines API-Aufrufs verfügbar.

In Abbildung 1.5 ist als UML-Sequenzdiagramm der Anwendungsfall der vertikalen Adressbuchdatenfilterung darstellt. Nach der Installation der AVARE-Box-Anwendung muss die entsprechende Messenger-App in AVARE-Box installiert werden. AVARE-Box fragt die vom Benutzer eingestellten Präferenzen bei der Datenbank ab und bekommt eine Antwort zurück. Nun müssen die Plug-ins entsprechend den Einstellungen definiert und installiert werden. Um die Adressbuchdaten auszulesen, wird die Query-Methode der Klasse ContentResolver des Android-Frameworks aufgerufen. Diese Methode liefert einen Cursor auf die Ergebnisse anhand der übergebenen URI zurück. Die Methode wird mithilfe vom YAHFA-Framework gehookt, sodass ein anderer Cursor bereitgestellt wird. Bei dem neuen Cursor wird die GetString-Methode so überschrieben, dass nur die Vornamen der Kontaktliste übergeben werden. Beim Starten des Messengers wird die Anwendung im Adressraum der AVARE-Box ausgeführt, und AVARE-Box kontrolliert die Anfragen, die vom Messenger an das Betriebssystem gesendet



(a) Beispieladressbuch in der Kontakte-Anwendung.

(b) Kontaktliste in WhatsApp nach der vertikalen Filterung.

(c) Kontaktliste in WhatsApp nach der horizontalen Filterung.

Figure 1.4: Anwendung der vertikalen und horizontalen Adressbuchdatenfilterung am Beispiel der WhatsApp-Anwendung.

werden. Der Messenger fragt die Adressbuchdaten an, die zwischengeschaltete AVARE-Box holt die gesamten Adressbuchdaten von Android ab, filtert diese entsprechend den im Plug-in definierten Filterregeln und leitet die gefilterten Adressbuchdaten an den Messenger weiter.

1.5 Fazit

AVARE hilft Anwendern, die Preisgabe von personenbezogenen Daten besser zu kontrollieren. Dadurch wird ein Beitrag zu einem selbstbestimmten Datenschutz geleistet. Die Anwender werden dadurch aber nicht vollständig vor nachteiligen Folgen der Speicherung und Verarbeitung ihrer Daten geschützt, es bedarf beispielsweise weiterer Regulierung und Transparenz, um beispielsweise Diskriminierung aufgrund von Entscheidungen mithilfe von Datenbeständen zu verhindern [Sch17a], und Maßnahmen gegen die Zentralisierung und damit verbundene Monopolisierung von großen Datenbeständen [Sch17b].

Der technische Durchsetzungsansatz von AVARE stößt auch an Grenzen. Für jede Betriebssystemplattform ist eine eigene Implementierungsstrategie zu finden, teils müssen sogar innerhalb einer Plattform für unterschiedliche Versionen unterschiedliche Strategien gewählt werden. Wünschenswert wäre es, wenn künftig die Betriebssysteme Schnittstellen bieten würden, um -– auf Wunsch des Nutzers -– feingranulare Einstellungen vornehmen zu können. Damit wäre eine plattformübergreifende Durchsetzung von Präferenzen unabhängig vom Gerät (Smartphone, Smart-TV, Smartwatch, Smart Car,...) leichter zu realisieren. Wenn zusätzliche Dienste ebenfalls entsprechende APIs anbieten würden, wäre es deutlich leichter, einen zentralen

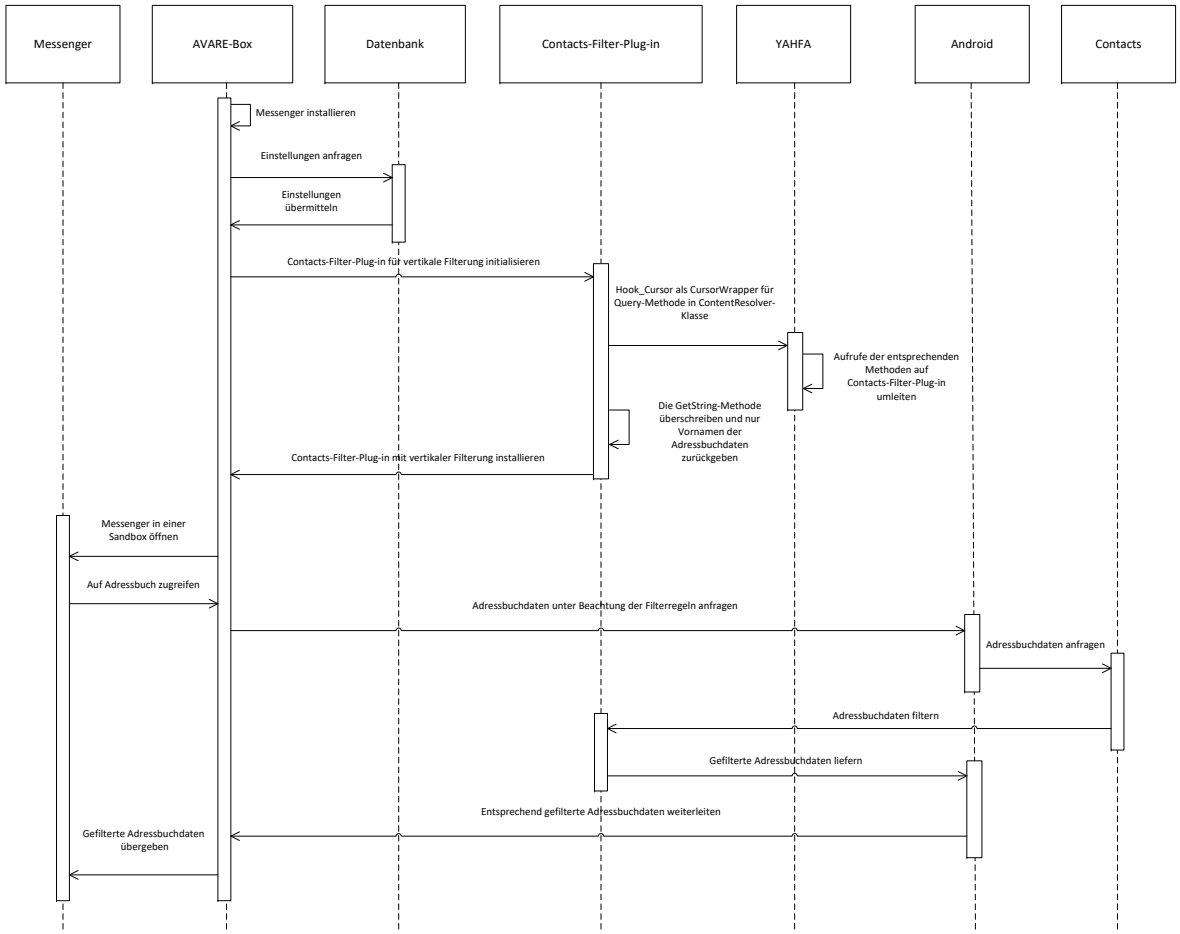


Figure 1.5: Sequenzdiagramm zur Darstellung des Adressbuchdatenanwendungsfalls mit der vertikalen Filterung.

Privatsphärenmanager zu entwickeln. Solche Schnittstellen wird es jedoch ohne regulatorische Bemühungen wahrscheinlich nicht geben, weil sie den gegenwärtigen Geschäftsmodellen der Betriebssystemhersteller und Dienstanbieter entgegenstehen.

Danksagung

Die Veröffentlichung ist im Projekt AVARE ("Anwendung zur Verteilung und Auswahl rechtskonformer Datenschutzeinstellungen") entstanden.

Das Projekt wird von der Baden-Württemberg Stiftung gGmbH (www.bwstiftung.de) im Rahmen des Forschungsprogramms "IKT-Sicherheit" finanziert. Projektträger ist das DLR. Beteiligte Einrichtungen sind das Karlsruher Institut für Technologie (www.kit.edu) und das FZI Forschungszentrum Informatik (www.fzi.de).

Bibliographie

[ABF+16] Alpers, Sascha ; Betz, Stefanie ; Fritsch, Andreas ; Oberweis, Andreas ; Pieper, Maria ; Schiefer, Gunther ; Wagner, Manuela: AVARE-Projektbericht, 1. Meilenstein / Karlsruher Institut für Technologie (KIT). 2016. (KIT Scientific Working Papers). – Forschungsbericht. – ISSN 2194–1629

[ABF+18] Alpers, Sascha ; Betz, Stefanie ; Fritsch, Andreas ; Oberweis, Andreas ; Schiefer, Gunther ; Wagner, Manuela: Citizen Empowerment by a Technical Approach for Privacy Enforcement. In: CLOSER, 2018, S. 589–595

[AHIC15] Assal, Hala ; Hurtado, Stephanie ; Imran, Ahsan ; Chiasson, Sonia: What's the deal with privacy apps? A comprehensive exploration of user perception and usability. In: Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia ACM, 2015, S. 25–36

[Alt18] Althammer, Thomas: Datenschutz und IT-Sicherheit in Zeiten der Digitalisierung. In: Kreidenweis, Helmut (Hrsg.): Digitaler Wandel in der Sozialwirtschaft : Grundlagen – Strategien – Praxis. Nomos Verlagsgesellschaft, Baden-Baden, 2018, S. 223–240

[AOP+17] Alpers, Sascha ; Oberweis, Andreas ; Pieper, Maria ; Betz, Stefanie ; Fritsch, Andreas ; Schiefer, Gunther ; Wagner, Manuela: PRIVACY-AVARE: An approach to manage and distribute privacy settings. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, CHN, December 13-16, 2017, IEEE, Piscataway, NJ, 2017. – ISBN 978–1–5090–6352–9, S. 1460–1468

[APW17] Alpers, Sascha ; Pieper, Maria ; Wagner, Manuela: Herausforderungen bei der Entwicklung von Anwendungen zum Selbstdatenschutz. In: Eibl, Maximilian (Hrsg.) ; Gaedke, Martin (Hrsg.): INFORMATIK 2017, Gesellschaft für Informatik, Bonn, 2017, S. 1061–1072

[asL16] asLody: VirtualApp. https://github.com/asLody/VirtualApp, 2016. – letzter Zugriff am 10. August 2018

[asL17] asLody: VirtualApp: Commit fd29f19410caaf56060bc941a19299723b550970. https://github.com/asLody/VirtualApp/commit/fd29f19410caaf56060bc941a19299723b550970, 2017. – letzter Zugriff 10. August 2018

[BK17] Buchner, Benedikt ; Kühling, Jürgen: Die Einwilligung in der Datenschutzordnung 2018. In: Datenschutz und Datensicherheit – DuD 41 (2017), Nr. 9, S. 544–548

[BRRR13] Bodden, Eric ; Rasthofer, Siegfried ; Richter, Philipp ; Rossnagel, Alexander: Schutzmaßnahmen gegen datenschutz-unfreundliche Smartphone-Apps. In: Datenschutz und Datensicherheit – DuD 37 (2013), Nr. 11, S. 720–725

[Bru14] Brummund, Anke: Smartphones und Apps: Datenschutzrechtliche Risiken und deren Begrenzung. In: Plödereder, E. (Hrsg.) ; Grunske, L. (Hrsg.) ; Schneider, E. (Hrsg.) ; Ull, D. (Hrsg.): INFORMATIK 2014, Gesellschaft für Informatik, Bonn, 2014, S. 539–550

[Ell18] Ellena: Messenger im Vergleich – Signal-Telegram-Threema-WhatsApp-Wire. https://ebblogs.com/apps/messenger-im-vergleich/, 2018. – letzter Zugriff am 10. August 2018

[Fre07] Free Software Foundation, Inc.: GNU General Public License, Version 3. https://www.gnu.org/licenses/gpl-3.0.en.html, 2007. – letzter Zugriff am 10. August 2018

[GGB09] German, Daniel M. ; González-Barahona, J. M.: An Empirical Study of the Reuse of Software Licensed under the GNU General Public License. In: Boldyreff, Cornelia (Hrsg.) ; Crowston, Kevin (Hrsg.) ; Lundell, Björn (Hrsg.) ; Wasserman, Anthony I. (Hrsg.): Open Source Ecosystems: Diverse Communities Interacting, Springer Berlin Heidelberg, 2009. – ISBN 978–3–642–02032–2, S. 185–198

[GVG17] Gerber, Paul ; Volkamer, Melanie ; Gerber, Nina: Das Privacy-Paradoxon – Ein Erklärungsversuch und Handlungsempfehlungen. In: DDV Deutscher Dialogmarketing Verband (Hrsg.): Dialogmarketing Perspektiven 2016/2017: Tagungsband 11. wissenschaftlicher interdisziplinärer Kongress für Dialogmarketing. Wiesbaden : Springer Fachmedien Wiesbaden, 2017, S. 139–167

[GWTB96] Goldberg, Ian ; Wagner, David ; Thomas, Randi ; Brewer, Eric A.: A Secure Environment for Untrusted Helper Applications Confining the Wily Hacker. In: Proceedings of the 6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography, USENIX Association, 1996 (SSYM'96)

[Küh17] Kühling, Jürgen: What to do with OTT? - Die Regulierung von Gmail, WhatsApp & Co. de lege ferenda. In: Körber, Torsten (Hrsg.) ; Kühling, Jürgen (Hrsg.): Regulierung – Wettbewerb – Innovation Bd. 3. Nomos Verlagsgesellschaft, Baden-Baden, 2017, S. 165–184

[Met17] Metzger, Axel: Data as Counter-Performance: What Rights and Duties do Parties Have. In: Jipitec 8 (2017), S. 2

[Ngu04] Nguyen, Binh: Linux Dictionary. http://www.tldp.org/LDP/Linux-Dictionary/html/index.html, 2004. – letzter Zugriff am 10. August 2018

[rk717a] rk700: VirtualHook. https://github.com/rk700/VirtualHook, 2017. – letzter Zugriff am 10. August 2018

[rk717b] rk700: VirtualHook: Commit fca5dcbc37b3e3d39cff62e59134b5544a4d4261. https://github.com/rk700/VirtualHook/commit/fca5dcbc37b3e3d39cff62e59134b5544a4d4261, 2017. – letzter Zugriff am 10. August 2018

[rk717c] RK700: Yet Another Hook Framework for ART. https://github.com/rk700/YAHFA, 2017. – letzter Zugriff am 10. August 2018

[rk718] RK700: Yet Another Hook Framework for ART: Commit b57672dbd95ca6d78d1d56badb5c60322c4d0807. https://github.com/rk700/YAHFA/commit/b57672dbd95ca6d78d1d56badb5c60322c4d0807, 2018. – letzter Zugriff am 10. August 2018

[SAB+18] Stach, Christoph ; Alpers, Sascha ; Betz, Stefanie ; Dürr, Frank ; Fritsch, Andreas ; Mindermann, Kai ; Palanisamy, Saravana M. ; Schiefer, Gunther ; Wagner, Manuela ; Mitschang, Bernhard ; Oberweis, Andreas ; Wagner, Stefan: The AVARE PATRON: A Holistic Privacy Approach for the Internet of Things. In: Proceedings of the 15th International Conference on Security and Cryptography (SECRYPT '18), INSTICC Press, Juli 2018, S. 1–8

[Sch17a] Schaar, Peter: Wie die Digitalisierung unsere Gesellschaft verändert. In: Big Data – In den Fängen der Datenkraken. Nomos Verlagsgesellschaft, 2017, S. 105–122

[Sch17b] Schwarke, Christian: Ungleichheit und Freiheit. Ethische Fragen der Digitalisierung. In: Zeitschrift für Evangelische Ethik 61 (2017), Nr. 3, S. 210–221

[Smi07] Smith, Brett: A Quick Guide to GPLv3. https://www.gnu.org/licenses/quick-guide-gplv3.pdf, 2007. – letzter Zugriff am 10. August 2018

[Sta18a] Statista: Anzahl der monatlich aktiven Nutzer von WhatsApp weltweit in ausgewählten Monaten von April 2013 bis Januar 2018 (in Millionen). https://de.statista.com/statistik/daten/studie/285230/umfrage/aktive-nutzer-von-whatsapp-weltweit/, 2018. – letzter Zugriff am 10. August 2018

[Sta18b] Statista: Anzahl der Nutzer des Schweizer Messengers Threema von Februar 2014 bis Januar 2018 (in Millionen). https://de.statista.com/statistik/daten/studie/445619/umfrage/nutzer-des-schweizer-messaging-dienstes-threema/, 2018. – letzter Zugriff am 10. August 2018

[The04] The Apache Software Foundation: Apache License, Version 2.0. https://www.apache.org/licenses/LICENSE-2.0, 2004. – letzter Zugriff am 10. August 2018