Datenbank-Spektrum: Data Management for Mobility
DOI 10.1007/s13222-015-0189-y

Der Secure Data Container (SDC)

Sicheres Datenmanagement für mobile Anwendungen

Christoph Stach · Bernhard Mitschang

Received: 28. Februar 2015 / Accepted: 12. Mai 2015 / Published: 23. Juni 2015

Zusammenfassung Mobile Endgeräte wurden zu Marc Weisers *Computer des* 21. *Jahrhunderts*, da sie als dauerhaft verfügbare Informationsquelle Einzug in unseren Alltag gehalten haben. Auf ihnen treffen private Daten (z. B. Fotos) auf Kontextdaten (z. B. Standortdaten); verknüpft stellen diese ein immenses Sicherheitsrisiko dar. Wie eine Vielzahl an Datendiebstählen belegt, reichen die existierenden Datensicherheitssysteme für Mobilplattformen bei weitem nicht aus. Daher bedarf es einer Identifikation möglicher Angriffsvektoren sowie einer Analyse der speziellen Schutzziele eines solchen Systems. Darauf basierend wird die *Privacy Management Platform*, ein Berechtigungssystem, mithilfe des neu eingeführten *Secure Data Containers* zu einem ganzheitlichen Datensicherheitssystem erweitert. Dabei zeigt sich, dass diese Kombination alle Schutzziele erfüllt und dennoch hochperformant ist. Obwohl die vorgestellten Prototypen auf Android basieren, ist das Konzept auch auf andere App-Plattformen anwendbar.

Schlüsselwörter Datenschutz · Schutzziele · PMP-Erweiterung · Datencontainer · Evaluation

1 Einleitung

Marc Weisers Vision vom *Computer des* 21. *Jahrhunderts* [Weiser(1999)] wurde, Dank der großen technischen Fortschritte im Bereich der mobilen Endgeräte, Wirklichkeit [Schmidt and Cohen(2013)]. Da die Leistung heutiger mobiler Endgeräte (häufig als *Smart Devices* bezeichnet) stetig ansteigt, sie immer längere Stand-by-Zeiten haben und sie dennoch in die Hosentasche passen, steht uns überall und jederzeit ein vollwertiger PC zur Verfügung. Mithilfe eines Smart Devices haben wir Zugriff auf eine nahezu unbegrenzte Datenquelle – das Internet. Smart Devices bieten aber noch mehr Funktionen: Da die darin integrierten Sensoren (z. B. der GPS-Empfänger) unentwegt Daten sammeln, ist ein Smart Device nicht nur eine Informationssenke, sondern gleichzeitig eine Datenquelle. Der Kontext des Anwenders kann so möglichst exakt erfasst werden, um daraus Rückschlüsse auf die gegenwärtige Situation, in der der Anwender sich befindet, zu ziehen. Dieses Wissen steht allen Anwendungen, die auf dem mobilen Endgerät ausgeführt werden, (den sogenannten *Apps*) zur Verfügung. Dadurch können sich diese bestmöglich auf die Bedürfnisse des Anwenders anpassen [Shilton(2009)]. Somit treffen auf einem Smart Device die beiden größten Bedrohungen für den Datenschutz aufeinander: Eine übersteigerte Datensammelwut seitens der Apps [Leontiadis et al.(2012)] und die Vermischung und Verknüpfung von Daten aus den unterschiedlichsten Domänen [Gaff et al.(2014)].

Diese Informationen können Apps untereinander nahezu unbeschränkt austauschen. Die Servicequalität sowie der Leistungsumfang einer App werden erheblich gesteigert, wenn diese sich autonom mit allen benötigten Daten versorgen können. Dieser technologische Fortschritt stellt allerdings gleichzeitig eine neue Gefahrenquelle dar: Die große Anzahl an privaten Daten aus den unterschiedlichsten Anwendungsbereichen, die jeder App beinahe uneingeschränkt zur Verfügung stehen, machen diese Geräte zu einem Angriffsziel. Seit 2012 steigt die Anzahl an bösartiger Schadsoftware kontinuierlich an: Die Angreifer zielen dabei auf unterschiedliche Daten ab, angefangen bei (scheinbar) harmlosen Informationen

Diese Arbeit wurde durch Google unterstützt.

C. Stach, B. Mitschang
IPVS / AS Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart
E-Mail: Christoph.Stach@ipvs.uni-stuttgart.de
E-Mail: Bernhard.Mitschang@ipvs.uni-stuttgart.de

über das System, über die Standortdaten bis hin zu persönlichen Daten [Svajcer(2014)]. Studien belegen, dass 20 von 30 populären Apps aus Google Play sensiblen Daten missbrauchen, z. B. indem sie sie an Werbeanbieter weiterleiten [Enck et al.(2014)]. Darüber hinaus kommen Smart Devices in immer mehr Anwendungsbereichen zum Einsatz, wie dem Gesundheitssektor (z. B. mittels *Smart Watches* [Felizardo et al.(2014)]) oder in der *Industrie* 4.0 [Gorecky et al.(2014)]. Die dabei entstehenden Daten sind in besonderem Maße schützenswert.

Mobilplattformanbieter sehen sich daher verstärkt mit der Problematik konfrontiert, wie der Datenschutz technisch gewährleistet werden kann. Die Lösungen der Keyplayer Apple und Google sind jedoch nicht zufriedenstellend, da sie entweder den (häufig) unerfahrenen Anwender überfordern (Android OS) oder ihm jedwede Entscheidungsfreiheit und Kontrolle über seine Daten nehmen (iOS) [Posegga and Schreckling(2011)].

Daher beschäftigen sich viele externe Entwicklerteams damit, wie ein Datensicherheitssystem für Mobilplattformen aussehen sollte. Die bisherigen Untersuchungen der Autoren haben ergeben, dass neben u. a. einer feingranularen und kontextsensitiven Zugriffskontrolle der Anwender in besonderem Maße berücksichtigt werden muss. Da dieser in der Regel kein IT Experte ist, darf ein Datensicherheitssystem nicht zu kompliziert sein und es muss ihn jederzeit umfassend über die Auswirkungen seiner Einstellungen auf das Verhalten der Apps informieren [Stach and Mitschang(2013)]. Basierend auf diesen Anforderungen entstand die **P**rivacy **M**anagement **P**latform (**PMP**), ein alternatives Berechtigungssystem für Android [Stach(2013a)]. Dieses reine Berechtigungssystem soll nun zu einem Datensicherheitssystem erweitert werden, um somit den Schutz der privaten Daten weiter zu verbessern. Die wichtigsten Ergebnisse dieses Artikels lassen sich mit den folgenden drei Punkten zusammenfassen:

(*I*) Initial wird eine Definition von „Datenschutz“ gegeben und daraus Schutzziele für ein Datensicherheitssystem abgeleitet und Angriffsvektoren identifiziert.

(*II*) Ausgehend von diesen Schutzzielen wird mit dem **S**ecured **D**ata **C**ontainer (**SDC**) eine Komponente zur sicheren Speicherung sensibler Daten als Erweiterung für die PMP eingeführt. Der SDC erlaubt allerdings nicht nur eine sichere Speicherung von sensiblen Daten, sondern ermöglicht zusätzlich einen einfachen, aber dennoch sicheren Austausch von Daten zwischen mehreren Apps. Durch eine vollständige Verschlüsselung der Daten kann sichergestellt werden, dass nur solche Apps Zugriff darauf erhalten, die zuvor vom Anwender dazu legitimiert wurden. Der SDC besitzt weitere Sicherheitsmaßnahmen, wie beispielsweise einen sogenannten *Kill Switch* zur sicheren Löschung der gespeicherten Daten für den Fall einer Kompromittierung oder eine aggregierte (und damit eingeschränkte) Sicht auf die Daten.

(*III*) Abschließend werden PMP und SDC evaluiert und die Messergebnisse mit den Messungen auf einer unveränderten Android-Mobilplattform verglichen.

Der weitere Beitrag ist wie folgt gegliedert: In Abschnitt 2 wird eine Definition des Begriffs „Datenschutz“ gegeben. Durch die Identifikation von relevanten Schutzzielen sowie möglichen Angriffsvektoren wird aufgezeigt, inwiefern diese Arbeit über den Datenschutz im engeren Sinne hinausgeht. Abschnitt 3 beschreibt die PMP kurz. Bedingt durch die auf eine einfache Erweiterbarkeit ausgelegte Architektur der PMP, lässt sich der SDC sehr einfach in diese integrieren. Details hierzu sowie zu den Funktionen, die von SDC unterstützt werden, werden in Abschnitt 4 beschrieben. Im Anschluss daran folgt eine Evaluation der PMP und des SDCs in Abschnitt 5. Abschließend wird in Abschnitt 6 ein Überblick über Datensicherheitssysteme für Mobilplattformen gegeben und die mit dem SDC erweiterte PMP mit dem Stand der Forschung verglichen, bevor Abschnitt 7 eine Zusammenfassung dieses Beitrags liefert.

2 Datenschutz – rechtliche und technische Sicht

Es gab einen großen Aufschrei in der Bevölkerung, als Edward Snowden aufdeckte, dass sogar staatliche Behörden (u. a. die NSA) Apps zu Spionagezwecken verwenden. Hierzu konnten (scheinbar) harmlose Apps, wie beispielsweise das Spiel *Angry Birds*, dazu missbraucht werden, um äußerst private Informationen über die Anwender aufzusammeln. Zu diesen Daten gehören beispielsweise deren politische Ansichten oder deren sexuelle Orientierung [Larson et al.(2014)]. Aufgewühlt von diesen beunruhigenden Fakten, forderten viele Anwender bessere Datenschutzmaßnahmen [Balebako et al.(2013)].

„Datenschutz“ wird (zumindest für die Europäische Union) in Artikel 8 der *Charta der Grundrechte der Europäischen Union*[1] definiert. So muss jede Person die Möglichkeit haben, alle personenbezogenen Daten, die sie betreffen, zu *schützen*. Unter dem Schutz versteht die Charta, dass eine Vereinbarung zwischen dem Anwender und der App getroffen wird, in der geregelt ist, welche persönlichen Daten zu welchem Zweck von der App verarbeitet werden. Artikel 2 der *Europäischen Datenschutzkonvention*[2] verfeinert dabei, was unter Verarbeitung verstanden wird: Das Speichern von Daten, das Durchführen logischer und / oder rechnerischer Operationen mit diesen Daten sowie das Verändern, Löschen, Wiedergewinnen oder Bekanntgeben von Daten.

[1] http://goo.gl/L8aKeE letzter Zugriff am 05.06.2015
[2] http://goo.gl/yW8hm3 letzter Zugriff am 05.06.2015

Anhand dieser Definition wird klar, dass das Vorgehen von Angry Birds nicht gegen die Charta verstößt: Der Anwender wird bei der Installation einer App aus Google Play darüber informiert, auf welche Daten diese zugreifen kann. Stimmt der Anwender diesen Zugriffen nicht zu, so wird die App nicht installiert und infolgedessen sind die persönlichen Daten weiterhin geschützt. Somit wird der Charta vollständig entsprochen. Allerdings ist dies aus Anwendersicht keine zufriedenstellende Lösung. Wie das Angry Birds Beispiel eindrucksvoll zeigt, kann der Anwender häufig nicht absehen, welche potentielle Gefahr aus der Kombination von scheinbar harmlosen Informationen ausgehen kann. Außerdem wird der Anwender mit einer Flut von Berechtigungsanfragen überschüttet, so dass er nicht in der Lage ist, die wirklich relevanten Anforderungen herauszufiltern [Felt et al.(2012a)]. So bedarf es beispielsweise einer Genehmigung, dass eine App das Gerät vibrieren lassen darf, wodurch keine Gefährdung für den Anwender ausgeht.

Ferner wird der Anwender nur sehr allgemein über die Verwendung der Daten informiert. Er erfährt weder, welche Informationen eine App genau anfragt, noch was sie damit macht. Nur eine Übersicht, welche Gruppen von Daten von einer App verarbeitet werden können, wird generiert. Die *READ_PHONE_STATE*-Berechtigung erlaubt beispielsweise, dass eine App Zugriff auf diverse Informationen über das mobile Gerät bekommt. Dies kann dazu verwendet werden, um beispielsweise zu erkennen, ob gerade ein Telefonat geführt wird und gegebenenfalls die Audioausgabe der App vorübergehend einzustellen. Die gleiche Berechtigung ermöglicht es einer App aber auch, dass sie die IMEI-Nummer (International Mobile Station Equipment Identity) oder die Telefonnummer auslesen kann, was wiederum eine erheblich größere Gefährdung für die Privatheit des Anwenders darstellt. Die Berechtigungen, denen ein Anwender zustimmen muss, um eine App verwenden zu können, stellen also in der Regel nur eine Obermenge der tatsächlich benötigten Rechte dar. Außerdem führt die Tatsache, dass der Anwender ohnehin allen Berechtigungen zustimmen muss, wenn er eine App verwenden will, dazu, dass er sich nicht spezifisch damit auseinandersetzt, sondern blind allen Zugriffen zustimmt [Felt et al.(2012b)]. Im Folgenden sollen daher relevante Schutzziele, die über den reinen Datenschutz hinausgehen, identifiziert und mögliche Angriffsvektoren beschrieben werden.

2.1 Schutzziele

Einerseits steigt die Anzahl an aggressiver *Adware* auf mobilen Plattformen, d. h. Apps, die ihre vom Anwender erteilten Berechtigungen dazu missbrauchen, um private Daten unbemerkt an Werbenetzwerke weiterleiten (z. B. für personenbezogene Werbeeinblendungen); andererseits nimmt auf diesen Platformen ebenfalls die Anzahl an *Malware* zu, d. h. Schadsoftware, die illegal auf Benutzerdaten zugreift [Lookout(2014)]. Daher stellt die Datensicherheit ein multidisziplinäres Problem dar, das nur gelöst werden kann, wenn juristische Regulierungen und technische Maßnahmen nahtlos ineinander greifen. Die technische Umsetzung muss vollständig anwenderorientiert sein, damit dieser das System richtig verstehen und nutzen kann [Gerber et al.(2015)]. Die ISACA[3] identifiziert drei wesentliche Schutzziele:

- (a) **Vertraulichkeit** (*confidentiality*) stellt sicher, dass nur autorisierte Instanzen Zugriff auf Daten haben.
- (b) **Intaktheit** (*integrity*) garantiert, dass die Daten nicht von Dritten kompromittiert werden können.
- (c) **Verfügbarkeit** (*availability*) gewährleistet, dass die Daten jederzeit verfügbar sind.

Diese drei Grundschutzziele werden von Cherdantseva und Hilton um weitere ergänzt, von denen im Bereich der mobilen Plattformen die folgenden von Relevanz sind [Cherdantseva and Hilton(2013)]:

- (d) **Nachvollziehbarkeit** (*auditability*) regelt, dass alle Aktionen von Apps persistent und unüberwindbar überwacht werden.
- (e) **Authentizität** (*authenticity*) überprüft die Identität der Instanzen, die Daten anfordern.
- (f) **Nachweisbarkeit** (*non-repudiation*) belegt, dass eine Instanz bestimmte Daten verarbeitet hat (bzw. garantiert, dass dies nicht der Fall war).
- (g) **Privatheit** (*privacy*) gibt dem Anwender die Kontrolle über seine Daten; dies sollte nutzerzentrisch sein, d. h. Konsequenzen, die sich aus bestimmten Einstellungen ergeben, müssen nachvollziehbar sein.

Nachdem die Schutzziele identifiziert wurden, die für Datensicherheit durch technische Maßnahmen unterstützt werden müssen, wird im Folgenden besprochen, wie Angreifer diese Schutzziele kompromittieren können. Die aufgeführten Angriffe stellen dabei jeweils nur ein mögliches Angriffsszenario dar.

2.2 Angriffsvektoren

Ein Angriff auf die Vertraulichkeit (a) und Intaktheit (b) von Daten kann dadurch erfolgen, dass diese Daten direkt auf einer tieferen Schicht des Betriebssystems angefragt werden. Wie in Abb. 1 zu sehen ist, sind Apps klar von den Systemfunktionen getrennt. Dazwischen existiert eine Schicht, die Schnittstellen für kontrollierte Zugriffe auf die darunterliegenden Schichten (und damit auch auf die

[3] http://goo.gl/1c1iN letzter Zugriff am 05.06.2015

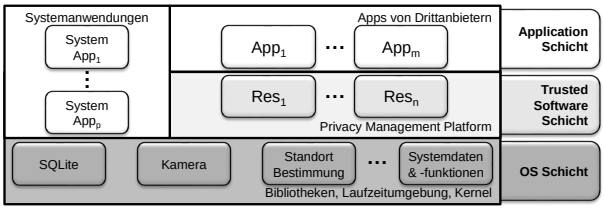


Abb. 1 Einbettung der PMP in die Systemarchitektur

Daten) zur Verfügung stellt. Berechtigungssysteme greifen auf dieser Schicht ein und reglementieren hier den Zugriff auf die Schnittstellen. Erlangt ein Angreifer hingegen direkten Zugriff auf den Kernel des Systems, so ist das Berechtigungssystem ausgehebelt und sämtliche Daten können uneingeschränkt gelesen und verändert werden. Eine andere Möglichkeit, um Zugriff auf geschützte Daten zu erlangen, stellt ein Angriff auf die Authentizität (e) dar. Ein Angreifer gibt vor, dass er eine vertrauensvolle Instanz ist, um so das System zu täuschen. Dazu müssen die Identifikationsmerkmale einer anderen Instanz übernommen werden.

Die Nachvollziehbarkeit (d) und die Nachweisbarkeit (f) können durch beide oben beschriebenen Angriffsvektoren ausgehebelt werden. Wenn die App eines Angreifers auf einer tieferen Schicht ansetzt, so kann ein Berechtigungssystem diese App weder überwachen, noch deren Handlungen protokollieren. Gibt sich die App eines Angreifers als eine andere Instanz aus, so kann ein Berechtigungssystem die App zwar überwachen, jedoch werden die falschen Berechtigungen (d. h. die Berechtigungen der anderen App) angewandt. Auch können sämtliche Handlungen der App zwar nachvollzogen werden; diese werden aber der anderen App zugeschrieben.

Die Verfügbarkeit (c) kann durch Denial of Service Angriffe beeinträchtigt werden, d. h. ein Angreifer fragt wiederholt Daten an, um das System so zu überlasten. Hierdurch sind zwar keine Daten in Gefahr, aber die Nutzbarkeit des Systems wird stark eingeschränkt.

Die Kombination aus den oben beschriebenen Angriffsvektoren schränken die Privatheit (g) ein. Da es bei der Privatheit in erster Linie darum geht, dem Anwender die volle Kontrolle über seine Daten zu geben, könnte ein Angreifer diese zusätzlich schädigen, indem er das Berechtigungssystem direkt außer Betrieb setzt. Wenn das Berechtigungssystem nicht mehr arbeitet, so kann der Anwender keine Einstellungen mehr vornehmen oder (schlimmer noch) er kann zwar Einstellungen tätigen, diese werden aber vom System nicht mehr umgesetzt.

Abschnitt 3 beschreibt, wie die **P**rivacy **M**anagement **P**latform (PMP) [Stach and Mitschang(2014)] die Schutzziele erfüllt. Anschließend wird in Abschnitt 4 die Arbeitsweise des **S**ecure **D**ata **C**ontainers (SDC) beschrieben und erörtert, wie dieser Angriffe verhindert.

3 Die Privacy Management Platform

Das Regelwerk der PMP basiert auf der Überlegung, dass *Apps* an sich keine Berechtigungen benötigen, sondern nur bestimmte Funktionen innerhalb einer App. Diese Funktionen werden logisch zu sogenannten *Service Features* zusammengefasst. Sämtliche Daten, auf die diese Service Features zugreifen können, werden von Informationsvermittlern, den *Resources*, angeboten. Thematisch zusammengehörige Ressourcen können in Gruppen, den *Resource Groups*, gebündelt werden. Der Anwender kann in der PMP Regeln aufstellen (*Policy Rule*), die beschreiben, welches Service Feature unter welchem Kontext Daten von einer Ressource erhalten soll. Zusätzlich kann er Einstellungen vornehmen (*Privacy Settings*), um die Daten, die die Apps von den Ressourcen erhalten zu anonymisieren, zu verfremden oder gänzlich zu randomisieren [Stach(2013b)]. Damit die PMP eine App in dieser Weise vollständig reglementieren kann, muss die App zusätzliche Metadaten bereitstellen, z. B. eine Spezifikation der Service Features. Wie die PMP mit *Legacy Apps* verfährt, d. h. Apps, die diese Metadaten nicht zur Verfügung stellen, ist in [Stach(2015)] beschrieben.

Abb. 2 zeigt die Funktionsweise der PMP. Installiert ein Anwender eine neue App (N) & (1), so registriert sich diese automatisch bei der PMP (2). Die PMP informiert den Anwender darüber, welche Service Features die App anbietet und welche Berechtigungen (Zugriffe auf Ressourcen) diese benötigen (F). Der Anwender wählt aus, welche der Service Features initial aktiviert werden sollen (3). Eine Besonderheit der PMP ist es, dass neue Ressourcen bei Bedarf sogar zur Laufzeit nachinstalliert werden können (4). Dadurch kann der Funktionsumfang der PMP jederzeit erweitert werden (z. B. durch den SDC), wie in Abschnitt 4 gezeigt. Benötigt eine App geschützte Daten, so fragt sie diese bei der PMP an (5), die darauf in dem Regelwerk nachschaut, ob es eine entsprechende Regel dafür gibt (6). Anschließend wird der Zugriff auf die Daten erlaubt (respektive verboten) (7). Wird der Datenzugriff untersagt, wird der Anwender darüber informiert, so dass er Regelanpassungen vornehmen kann, wenn er das Service Feature dennoch verwenden will (F). Erhält eine App Zugriff auf die Daten, so sammelt die Ressource die Daten (8), nimmt, abhängig von den Privacy Settings, nötige Aggregationen oder Verfremdungen vor und schickt die Daten anschließend an die App (9). Für weitere Informationen zur PMP sei an dieser Stelle auf die entsprechende Literatur verwiesen [Stach and Mitschang(2013), Stach(2013a)].

Die PMP ist in erster Linie ein stark erweitertes Berechtigungssystem. Der Anwender kann Apps feingranular an seine Bedürfnisse anpassen und so die Menge der preisgegebenen Daten auf ein Minimum reduzieren. Die

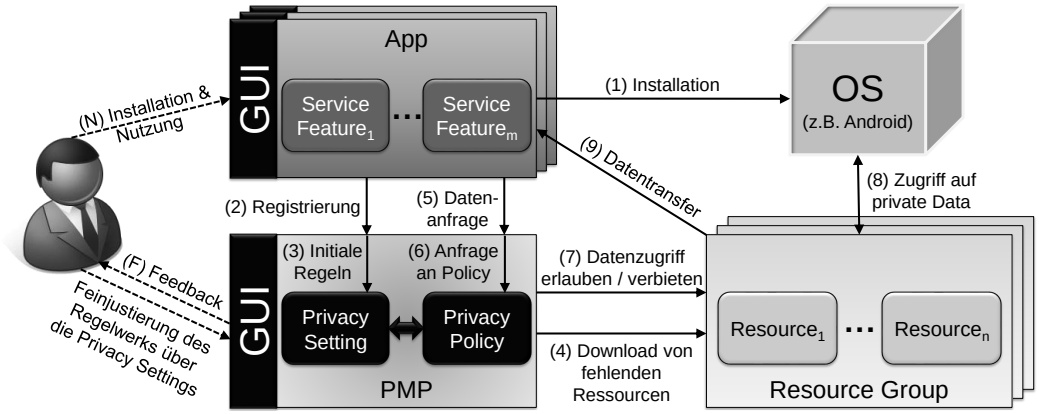


Abb. 2 Funktionsweise der PMP

Privatheit (g) wird demnach vollständig gewährleistet. Die *Authentizität* (e) einer App wird mittels der Android-Signatur, die bei der Registrierung überprüft wird, sichergestellt. Nur registrierte Apps können die PMP nutzen und somit Zugriff auf die Daten der Ressourcen erlangen (*Vertraulichkeit* (a) und *Intaktheit* (b)). Die *Nachvollziehbarkeit* (d) und die *Nachweisbarkeit* (f) wird dadurch realisiert, dass jede Anfrage an eine Ressource vom System erfasst und in einer Log-Datei hinterlegt wird. Die *Verfügbarkeit* (c) hängt von der jeweiligen Ressource ab. Da sämtliche Daten i. d. R. direkt aus dem Android-System stammen, muss dieses deren Verfügbarkeit garantieren.

Auch wenn die unterschiedlichen Prototypen der PMP für Android umgesetzt worden sind [Stach(2013b)], so handelt es sich bei der PMP um eine zusätzliche Schicht in der Systemarchitektur, die zwischen den Apps und dem eigentlichen System sitzt (siehe Abb. 1); durch diese generische Struktur ist es möglich, die PMP auch für andere App-Plattform, wie etwa die *Facebook Platform*, zu realisieren. Da Systemanwendungen vom Plattformanbieter aus Performanz- und Stabilitätsgründen grundsätzlich nicht reglementiert werden sollten, beschränkt sich die PMP auf die Kontrolle von Drittanbieter-Apps. Somit kann die PMP insbesondere die Schutzziele (a), (b) und (f) nur dann erfüllen, wenn das darunterliegende System nicht kompromittiert werden kann und es technisch nicht möglich ist, über den Kernel Zugriff auf die Daten der Ressourcen zu erlangen.

4 Der Secure Data Container

Für ein sicheres Datenmanagement in mobilen Anwendungen reicht ein reines Berechtigungssystem, wie es die PMP darstellt, allerdings nicht aus. Aus diesem Grund wird mit dem SDC ein verschlüsselter Datencontainer als Erweiterung für die PMP eingeführt. In diesem generischen Container kann jede App ihre Daten speichern. Zusätzlich bietet der SDC die Möglichkeit, dass die Daten mit allen, nur mit bestimmten oder mit keinen Apps geteilt werden. Intern speichert der SDC alle Daten als Key-Value-Paare in einer SQLite-Datenbank ab. Durch dieses generische Datenmodell wird größtmögliche Flexibilität garantiert. Nach außen lässt sich der SDC über eine Schnittstelle ähnlich der originalen Android SQLite API ansprechen. Allerdings muss beim SDC eine App nicht extra einen Content Provider (mit allen damit verbundenen Sicherheitsrisiken, siehe [Ongtang et al.(2012)]) für den Datenaustausch implementieren, sondern kann direkt bei der Erzeugung das Datum im SDC hinterlegen und angeben, welche Apps Zugriff auf diesen Datenbankeintrag erhalten sollen. Die Vorgehensweise bei einer Anfrage ist in Abb. 3 abgebildet und wird im Folgenden detailliert:

Sobald eine App eine Verbindung zu dem SDC aufbaut, prüft der SDC, ob diese App bereits bei ihm registriert ist. Wenn das nicht der Fall ist, vergibt er der App eine eindeutige interne ID und legt einen neuen Eintrag in der *sdc_maintenance_apps*-Tabelle an. Diese ID wird dazu verwendet, um den Besitzer eines Datensatzes zu identifizieren (*owner*-Eintrag). Standardmäßig ist nur der Besitzer eines Datensatzes dazu berechtigt, auf diese Daten zuzugreifen. Da ein unkontrollierter Informationsfluss zwischen verschiedenen Apps, wie es bei Android Usus ist, eine Bedrohung hinsichtlich Information Security darstellt, unterstützt der SDC einen sichereren Modus Operandi für den Datenaustausch. Der Besitzer der Daten kann diese mit einer *sharable*-Flag versehen, wenn ein Datensatz mit anderen Apps geteilt werden soll. Daraufhin erzeugt der SDC in der *sdc_maintenance_share*-Tabelle einen Eintrag, welcher Datensatz mit welcher App (oder welchen Apps) geteilt werden kann. Danach können alle Apps, die vom Datenbesitzer angegeben wurden, auf die geflaggten Daten lesend zugreifen. Aus Sicherheitsgründen hat nur der Besitzer Schreibrechte auf den Daten. Das interne Datenmodell des SDCs ist in Abb. 4 abgebildet.

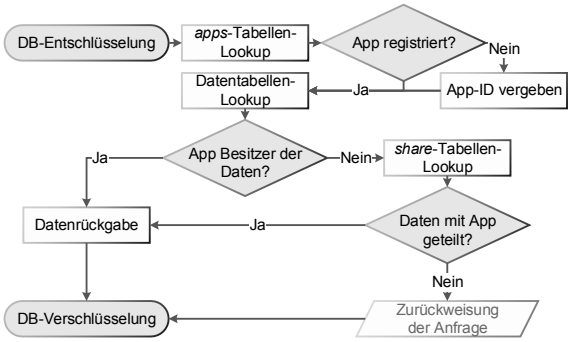


Abb. 3 SDC-Anfragealgorithmus als Aktivitätsdiagramm

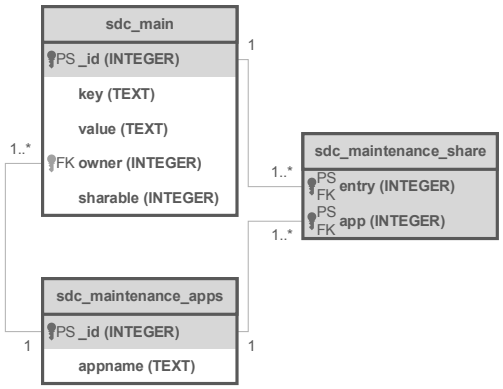


Abb. 4 SDC-Datenmodell in UML-Notation

Um illegale Datenzugriffe ausschließen zu können, verschlüsselt der SDC alle gespeicherten Daten, wohingegen Android-Datenbanken standardmäßig vollständig unverschlüsselt als Klartext gespeichert sind! Hierzu wird eine AES-256-Verschlüsselung verwendet, wobei auch jeder andere Verschlüsselungsalgorithmus möglich wäre. Einzig der SDC besitzt den Schlüssel; die Daten werden nur dechiffriert, wenn eine App die benötigten Zugriffsrechte hat. Der SDC stellt über die beiden Hilfstabellen zusätzlich sicher, dass eine App nur auf die eigenen oder mit ihr geteilten Daten zugreifen kann. Die Daten sind nur zum Zeitpunkt eines Zugriffs unverschlüsselt. Als weitere Sicherheitsmaßnahme verfügt der SDC über einen sogenannten Kill Switch. Im Falle einer Kompromittierung kann der Anwender damit den Schlüssel löschen, wodurch die im SDC gespeicherten Daten unlesbar werden (vgl. [Geambasu et al.(2011)]). Da mittels forensischer Analyse einer Datenbank selbst gelöschte Datensätze wiederhergestellt werden können [Stahlberg et al.(2007)], benötigt man zum Löschen des Schlüssels ein Verfahren, wie es in [Reardon et al.(2012)] vorgestellt wird. Dieses eignet sich aber nur für kleinere Dateien; für ganze Datenbanken kann es nicht verwendet werden.

Der SDC unterstützt die PMP in mehreren Schutzzielen: Selbst wenn eine App über OS-Exploits Zugriff auf die SDC-Datenbank erhält, so sind darin alle Daten verschlüsselt und somit ohne den Schlüssel, der nur für den SDC zugänglich ist, weder lesbar (a) noch veränderbar (b). Da damit Daten nur über die PMP erlangt werden können, kann diese auch alle Zugriffe erfassen (d) und protokollieren (f). Da mehrere SDC-Instanzen gleichzeitig verwendet werden können, können die Daten einer App beispielsweise auf mehrere SDC aufgeteilt werden oder einer App ein eigener SDC zugeteilt werden, um die dauerhafte Verfügbarkeit zu garantieren (c). Die Schutzziele (e) und (g) werden bereits durch die PMP ausreichend erfüllt, wie in Abschnitt 3 dargestellt, und bedürfen somit keiner weiteren Unterstützung durch den SDC. Darüber hinaus ist es möglich, für unterschiedliche SDC-Instanzen unterschiedliche Verschlüsselungsalgorithmen anzuwenden. Dadurch kann, abhängig davon, wie sicherheitskritisch die gespeicherten Daten sind, die Performanz des SDCs aufwandsabhängig angepasst werden. Wie die Ergebnisse aus Abschnitt 5 zeigen, überzeugt der SDC hinsichtlich Performanz selbst mit AES-256-Verschlüsselung und stellt, trotz der erhöhten Sicherheit, das klassische Android-System in den Schatten, wenn Daten mit anderen Apps geteilt werden.

5 Evaluation des SDCs

Die Messungen wurden auf einem unmodifizierten NexusOne Dev Phone (1 GHz, 512 MB RAM, Android 2.3.6) durchgeführt. Die Test Suite besteht aus voneinander getrennten Schreib- und Lesetests. Beide Tests arbeiten auf unterschiedlich großen Datenbeständen, angefangen bei kleinen Datenbeständen (K) mit 500 Datenbankeinträgen geht es über 1.000, 2.000, 4.000 und 8.000 Einträgen zu großen Datenbeständen (G) mit 16.000 Einträgen in Anlehnung an den TestIndex Benchmark von McObject[4]. Zunächst erzeugt der Schreibtest die jeweilige Anzahl an zufälligen Datensätzen und speichert diese. Ein Datensatz besteht dabei jeweils aus einem Schlüssel-Wert-Paar. Da der Overhead des SDCs hauptsächlich durch Berechtigungsanfragen und Verschlüsselungsaufgaben erzeugt wird, würde ein komplexeres Datenmodell dem SDC eher zu Gute kommen, da dann diese Vorlaufkosten im Vergleich zu den Abfragekosten vernachlässigbar werden würden. Anschließend greift der Lesetest auf alle Datensätze in einer zufälligen Reihenfolge zu. Beide Tests wurden auf vier unterschiedliche Arten implementiert: als **Android-App ohne Verwendung der PMP** (*Android*), als **PMP-App ohne Verwendung der SDC-Ressource** (*PMP*), als **PMP-App unter Verwendung einer SDC-Ressource ohne Verschlüsselung** (SDC_{UV}) und als **PMP-App unter Verwendung**

[4] http://www.mcobject.com/android letzter Zugriff am 05.06.2015

Tabelle 1 Benchmark Ergebnisse für sehr kleine (K) repektive große (G) Datenbestände (siehe Abb. 5 für Zwischenschritte)

		Schreibtest				Lesetest			
		Android	**PMP**	**SDC$_{UV}$**	**SDC$_{AES}$**	**Android**	**PMP**	**SDC$_{UV}$**	**SDC$_{AES}$**
Laufzeit	**K**	25,7 s	25,3 s	25,3 s	37,3 s	17,8 s	19,3 s	16,4 s	76,4 s
	G	790,0 s	784,6 s	795,4 s	1.206,6 s	1176,6 s	581,5 s	440,7 s	532,9 s
CPU-Auslastung	**K**	13,78 %	13,05 %	14,01 %	15,00 %	6,95 %	9,07 %	16,10 %	38,50 %
	G	15,01 %	14,56 %	14,63 %	15,46 %	34,64 %	22,05 %	18,37 %	22,02 %
Speicherverbrauch	**K**	9,69 MB	8,70 MB	8,48 MB	8,77 MB	13,08 MB	13,70 MB	13,76 MB	13,53 MB
	G	10,07 MB	10,61 MB	11,39 MB	11,02 MB	38,87 MB	21,48 MB	18,91 MB	18,54 MB
DB-Größe	**K**	29 kB		39 kB		29 kB		39 kB	
	G	813 kB		1.054 kB		813 kB		1.054 kB	
App-Größe		248,51 kB	345,06 kB	344,24 kB	381,82 kB	247,06 kB	344,78 kB	344,27 kB	344,42 kB

einer SDC-Ressource mit AES-256-Verschlüsselung (*SDC$_{AES}$*). Beide Tests wurden für jede App 10-mal wiederholt. Nach jedem Durchlauf wurden die Datenbank sowie der Cache vollständig gelöscht, um eine Beeinflussung durch warme Caches ausschließen zu können. Die Ergebnisse für die kleinen (500 Einträge) und großen Datenbestände (16.000 Einträge) sind für den Schreib- sowie den Lesetest in Tabelle 1 sowie Abb. 5 abgebildet. Dabei wurden die Laufzeit der App, die mittlere CPU-Auslastung, der maximale Speicherverbrauch, die Größe der Datenbank und die Größe der jeweiligen Test-App betrachtet. Die Referenzwerte in den Tabellen sind die Mediane aus den Testläufen, um eine Verzerrung der Messergebnisse durch Ausreißer auszuschließen. Die Messungen liefern zahlreiche Erkenntnisse:

1. Für die **Laufzeit** des Schreibtests spielt es nahezu keine Rolle, ob dieser unter Android, der PMP oder der PMP mit einem SDC ausgeführt wird. Erst wenn bei dem SDC die Verschlüsselung aktiviert wird, entsteht hierbei ein Overhead. Allerdings ist AES-256 auch eines der sichersten (und rechenintensivsten) Kryptographieverfahren und sollte daher auch nur für sehr vertrauliche Daten genutzt werden. Beim Lesetest können die PMP und ganz besonders der SDC ihre Stärken ausspielen. Werden Daten mit anderen Apps geteilt, so ist der Einsatz von Content Providern sehr kostenintensiv. Während man dies mit der PMP bereits auf ein Minimum reduzieren kann, können Apps ihre Daten in einem SDC direkt untereinander austauschen; trotz der höheren Sicherheit, erzielt man einen Performanzgewinn.

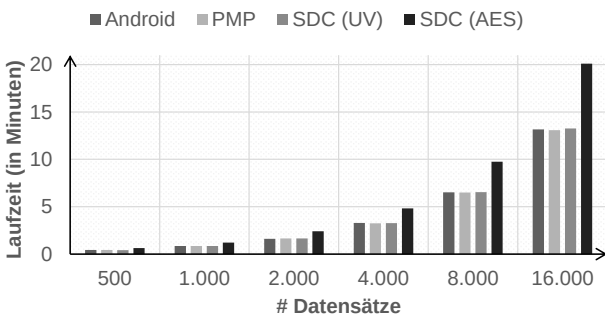
2. Die mittlere **CPU-Auslastung** (berücksichtigt wurden jeweils sämtliche Prozesse, die mit den Tests in Verbindung stehen) beim Schreibtest ist auf allen Systemen nahezu gleich. Beim Lesetest zeigt sich, dass für sehr wenige Anfragen die Kosten für Berechtigungsanfragen und Dechiffrierung erheblich bemerkbar machen. Diese relativieren sich allerdings für sehr viele Anfragen. Aufgrund der Kosten für die Content Provider bei Android, ist die mittlere CPU-Auslastung bei der PMP sowie bei dem SDC in diesem Fall deutlich günstiger.

3. Betrachtet man den maximalen **Speicherverbrauch** zur Laufzeit, so liegen beim Schreibtest alle Systeme dicht beieinander. Beim Lesetest machen sich bei steigender Anzahl an Anfragen die kostenintensiven Content Provider bemerkbar, wodurch die PMP und der SDC etwa 50 % weniger Speicher benötigen. Dabei werden im Fall der PMP und des SDCs jeweils die kumulierten Kosten betrachtet, d. h. die Kosten für die App, die PMP und die verwendeten Ressourcen. Etwa 70 % davon werden von der PMP und den Ressourcen verursacht. Eine Instanz der PMP kann allerdings beliebig viele Apps verwalten, wodurch sich im Realbetrieb dieser Speicher-Overhead auf alle diese Apps aufteilt.

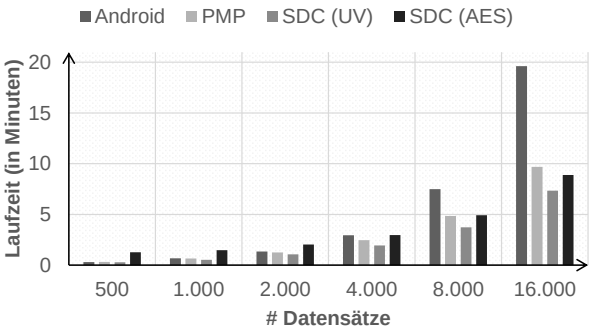
4. Die **Größe der Datenbank** ist unter Android und der PMP identisch; bei dem SDC entsteht durch die zusätzlichen Spalten in der eigentlichen Datentabelle sowie durch die benötigten Verwaltungstabellen (sdc_maintenance_apps und sdc_maintenance_share) hingegen ein Overhead. Dieser Verwaltungs-Overhead liegt im Mittel bei ca. 30 % und fällt, je mehr Daten einer App in der Datenbank gespeichert werden. Jedoch selbst bei einem für eine App bereits sehr großen Datenbestand von 16.000 Einträgen, liegt der Speicherbedarf für diese Hilfstabellen bei unter 0,25 MB.

5. Auch der Overhead der **App-Größe** ist mit unter 150 kB gering. Dieser Overhead wird in erster Linie durch die Definition der Service Features erzeugt. Da bei einer „richtigen“ App sich das Verhältnis App-Code zu Service Feature-Definition im Vergleich zu diesen Test-Apps erheblich zu Gunsten des App-Codes entwickeln würde, wird dieser Overhead weiter relativiert.

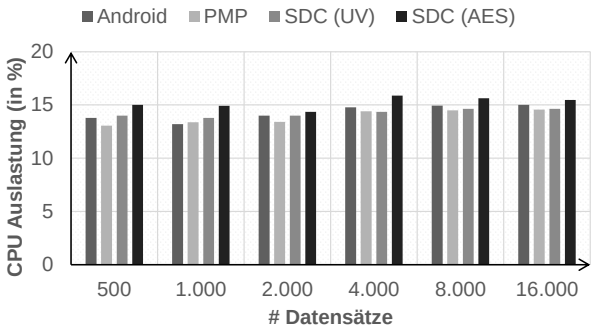
Die Kennzahlen für exponentiell wachsende Datensätze sind in Abb. 5 dargestellt. Hierbei werden die Erkenntnisse 1. bis 3. weiter gestützt. Insgesamt lässt sich festhalten, dass sich der Einsatz eines SDCs besonders dann eignet, wenn Daten mit anderen Apps geteilt werden müssen. In diesem Fall erzielt man einen erheblichen



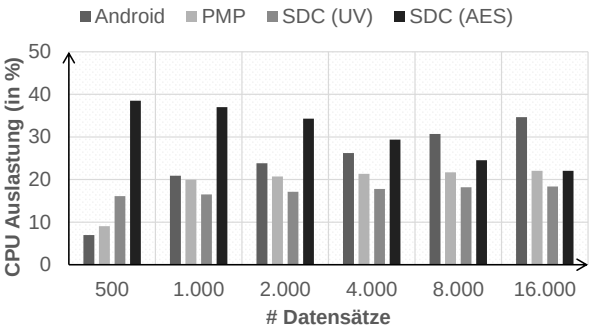
(a) Laufzeit (Schreibtest)



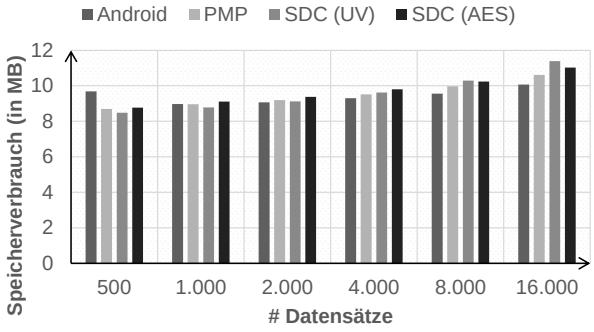
(b) Laufzeit (Lesetest)



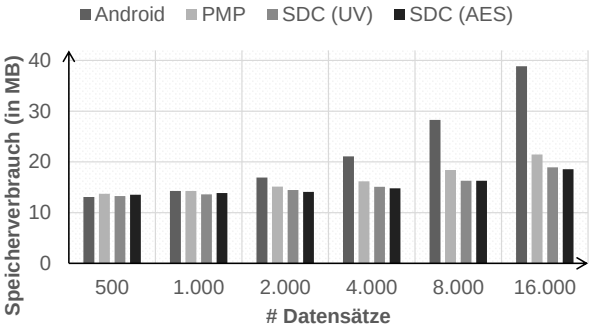
(c) Mittlere CPU-Auslastung (Schreibtest)



(d) Mittlere CPU-Auslastung (Lesetest)



(e) Maximaler Speicherverbrauch (Schreibtest)



(f) Maximaler Speicherverbrauch (Lesetest)

Abb. 5 Grafische Übersicht der Messergebnisse aus Tabelle 1 für eine exponentiell wachsende Anzahl an Datensätzen

Vorteil hinsichtlich Laufzeit, CPU-Auslastung und Speicherverbrauch. Außerdem zeigt der Vergleich zwischen dem unverschlüsselten SDC und dem SDC, der AES-256 verwendet, dass es sinnvoll ist, SDCs mit unterschiedlichen Kryptographiealgorithmen zu verwenden, um Daten je nach Geheimhaltungsstufe und Verwendungszweck adäquat zu schützen. Moderne Mehrprozessorsysteme unterstützen die Ver- und Entschlüsselung.

Neben dieser rein performanzorientierten Evaluation muss in zukünftigen Arbeiten geprüft werden, ob das Modell des SDCs sämtliche Angriffsvektoren ausreichend abgedeckt (z. B. mittels risikobasierten Sicherheitstests [McGraw and Potter(2004)]) sowie, ob die Implementierung realen Angriffen standhält (z. B. mittels Penetrationstests [Arkin et al.(2005)]). Obwohl erste Nutzertests der PMP positiv ausfielen und der SDC für den Anwender transparent agiert, muss dieser Eindruck durch kommende Usability-Tests verifiziert werden.

6 Verwandte Arbeiten

Im Folgenden wird ein Überblick über repräsentative Datensicherheitssysteme für Mobilplattformen gegeben. Insbesondere wird dabei auf die Schutzziele aus Abschnitt 2.1 geachtet: (a) *Vertraulichkeit*, (b) *Intaktheit*, (c) *Verfügbarkeit* (d) *Nachvollziehbarkeit*, (e) *Authentizität*, (f) *Nachweisbarkeit* sowie (g) *Privatheit*.

Backes et al. [Backes et al.(2013)] stellen mit *AppGuard* ein Berechtigungssystem für Android vor. Der AppGuard App-Konverter baut in jede App eine Sicherheitsbibliothek ein. Potentiell gefährliche Datenzugriffe sind nach der Konvertierung nur über diese AppGuard-Bibliothek möglich. Allerdings ist die Manipulation von Bytecode aus zwei Gründen als kritisch zu betrachten: Einerseits muss man sich dabei auf Heuristiken und Annahmen verlassen, um die relevanten Codefragmente identifizieren zu können. Eine falsche Annahme kann allerdings unvorhersehbare Effekte nach sich ziehen. Andererseits bewegt sich ein solcher Konverter in einer juristischen Grauzone, da u. a. § 14 UrhG (Entstellung des Werkes) oder § 303a (Datenveränderung) durch solch tiefgreifende Eingriffe in den Programmfluss verletzt werden können. Aus diesem Grund wurde die App SRT AppGuard Pro aus Google Play entfernt. Daher wählt *Aurasium* [Xu et al.(2012)] eine andere Herangehensweise. Hierbei wird nicht die App selbst verändert, sondern sie wird in eine sichere Sandbox eingebettet und darin ausgeführt. Zur Laufzeit überwacht die Sandbox jeden Datenzugriff und greift gegebenenfalls ein. Im Gegensatz dazu erweitern Conti et al. [Conti et al.(2012)] das existierende Android Berechtigungssystem um kontextbasierte Regeln. Mit ihrem System *CRêPE* kann der Anwender Situationen definieren, in denen bestimmte Regeln gelten sollen (z. B. dass auf Standortdaten nur in der Freizeit zugegriffen werden darf). Während diese Systeme versuchen, die Zugriffsrechte von Apps zu reglementieren, sehen Russello et al. [Russello et al.(2011)] unter Android die unkontrollierbare Weitergabe von Berechtigungen, die sogenannte *Permission Escalation*, als größtes Problem an. In deren System *YAASE* wird daher ein neues Regelsystem eingeführt, das Apps daran hindert, Berechtigungen untereinander auszutauschen.

Während diese Systeme reine Berechtigungssysteme darstellen, verstärkt *Samsung KNOX* [Samsung(2015)] die Sicherheitsfunktionen auf jeder Schicht des Android Software Stacks, angefangen bei einem sicheren Bootloader, über Erweiterungen im Linux Kernel bis hin zu getrennten Speicherbereichen in der Anwendungsschicht. *Virtual Fort Knox* [Holtewert et al.(2013)] befasst sich mit den Sicherheitsproblemen von Firmen, die Mobilgeräte in deren Unternehmen einsetzen. Virtual Fort Knox bietet eine Middleware, die sämtlichen Datenverkehr zwischen einem Mobilgerät und dem Unternehmensnetzwerk absichert. Damit stellt Virtual Fort Knox einen wichtigen Schritt in Richtung Industrie 4.0 dar. In erster Linie definiert es allerdings eine sichere Infrastruktur für Unternehmen und lässt den Anwender weitestgehend außen vor. Tabelle 2 beinhaltet die wesentlichen Erkenntnisse dieser empirischen Studie der verwandte Arbeiten. Dieser Vergleich ergibt, dass die PMP in Kombination mit dem SDC das einzige ganzheitliche Datensicherheitssystem ist.

Alternative Ansätze befassen sich mit einer Analyse von Apps, da Google Apps vor der Veröffentlichung nicht umfassend prüft wodurch Schadsoftware in Google Play gelangt. Aus Anwendersicht ist eine Vorabprüfung unbefriedigend: 1. Aufgrund der großen Anzahl an täglich erscheinenden (und damit zu prüfenden) Apps, verzögert sich deren Veröffentlichung. 2. Eine Vorselektion schränkt die Entscheidungsfreiheit der Anwender ein. Wird die Prüfung hingegen auf die Endgeräte verlagert, bleibt der Anwender in Kontrolle. Mittels einer Echtzeitanalyse des Kontroll- [Chan et al.(2012)] sowie Informationsflusses [Kodeswaran et al.(2012)], kann das Verhalten einer App überwacht und das Gefährdungspotential eingeschätzt werden. Ebenfalls kann auf Crowd-basierte Ratingsysteme [Jin et al.(2013)] zurückgegriffen werden, um das Vertrauen in eine App zu bestimmen. Neisse et al. [Neisse et al.(2013)] stellen ein Rahmenwerk vor, mit dem die Verarbeitung von Daten kontrolliert werden kann, nachdem diese das Gerät verlassen haben. Dies liegt allerdings außerhalb des Fokus dieser Arbeit.

7 Zusammenfassung

Den Besitzern von *Smart Devices* wird durch die steigende Anzahl an Datendiebstählen mehr und mehr bewusst, dass sie Schutzmaßnahmen treffen müssen. Betrachtet man allerdings bestehende Datensicherheitssysteme für Mobilplattformen, so lassen sich zwei Trends beobachten: Entweder werden die Anwender vollständig bevormundet oder völlig überfordert. Auch tragen die Anbieter der Mobilplattformen nur wenig dazu bei, etwas an dieser Situation zu ändern, da deren Lösungsansätze die Anforderungen der Anwender nur ungenügend erfüllen.

Aus diesem Grund verfolgt dieser Beitrag drei wichtige Ziele: (I) Es wird eine Definition von „Datenschutz“ gegeben und daraus Schutzziele für ein Datensicherheitssystem abgeleitet sowie Angriffsvektoren identifiziert. (II) Mit dem **S**ecured **D**ata **C**ontainer (**SDC**) wird eine Erweiterung für die **P**rivacy **M**anagement **P**latform (**PMP**) eingeführt, mit der alle Anforderungen an ein Datensicherheitssystem erfüllt werden. (III) Abschließend wird gezeigt, dass die PMP mit dem SDC alle Anforderungen an ein Datensicherheitssystem erfüllt und auch hinsichtlich Performanz sehr gute Ergebnisse liefert. Obwohl der vorgestellte Prototyp der PMP und des SDCs für Android umgesetzt wurden, lässt sich das zugrundeliegende Konzept leicht auf andere Umgebungen, wie die *Facebook Platform*, übertragen.

Danksagung Die PMP entstand initial in Zusammenarbeit mit Google München. Dafür möchten wir Google danken.

Tabelle 2 Gegenüberstellung von Datensicherheitssystemen für Mobilplattformen anhand der Schutzziele aus Abschnitt 2.1

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
AppGuard	×	×	√	×	√	×	√
Aurasium	√	√	√	×	√	×	√
CRêPE	√	√	×	×	√	×	√
Samsung KNOX	√	√	√	×	√	×	×
Virtual Fort Knox	√	√	√	√	√	√	×
YAASE	√	√	×	×	√	√	√
PMP	×	×	√	√	√	×	√
PMP & SDC	√	√	√	√	√	√	√

Literatur

Arkin et al.(2005). Arkin B, et al. (2005) Software Penetration Testing. IEEE Security & Privacy 3(1):84–87

Backes et al.(2013). Backes M, et al. (2013) AppGuard - Enforcing User Requirements on Android Apps. In: TACAS '13

Balebako et al.(2013). Balebako R, et al. (2013) "Little Brothers Watching You": Raising Awareness of Data Leaks on Smartphones. In: SOUPS '13

Chan et al.(2012). Chan PP, et al. (2012) DroidChecker: Analyzing Android Applications for Capability Leak. In: WISEC '12

Cherdantseva and Hilton(2013). Cherdantseva Y, Hilton J (2013) A Reference Model of Information Assurance & Security. In: ARES '13

Conti et al.(2012). Conti M, et al. (2012) CRêPE: A System for Enforcing Fine-Grained Context-Related Policies on Android. IEEE Information Forensics and Security 7(5):1426–1438

Enck et al.(2014). Enck W, et al. (2014) TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. ACM Computer Systems 32(2):5:1–5:29

Felizardo et al.(2014). Felizardo V, et al. (2014) E-Health: Current Status and Future Trends. In: Handbook of Research on Democratic Strategies and Citizen-Centered E-Government Services, IGI Global, pp 302–326

Felt et al.(2012a). Felt AP, et al. (2012a) Android Permissions: User Attention, Comprehension, and Behavior. In: SOUPS '12

Felt et al.(2012b). Felt AP, et al. (2012b) How to Ask for Permission. In: HotSec '12

Gaff et al.(2014). Gaff BM, et al. (2014) Privacy and Big Data. Computer 47(6):7–9

Geambasu et al.(2011). Geambasu R, et al. (2011) Keypad: An Auditing File System for Theft-prone Devices. In: EuroSys '11

Gerber et al.(2015). Gerber P, et al. (2015) Usability Versus Privacy Instead of Usable Privacy. ACM SIGCAS Computers and Society 45(1):16–21

Gorecky et al.(2014). Gorecky D, et al. (2014) Mensch-Maschine-Interaktion im Industrie 4.0-Zeitalter. In: Industrie 4.0 in Produktion, Automatisierung und Logistik, Springer Fachmedien Wiesbaden, pp 525–542

Holtewert et al.(2013). Holtewert P, et al. (2013) Virtual Fort Knox Federative, Secure and Cloud-based Platform for Manufacturing. Procedia CIRP 7:527–532

Jin et al.(2013). Jin H, et al. (2013) Recommendations-based Location Privacy Control. In: PerCom Workshops '13

Kodeswaran et al.(2012). Kodeswaran P, et al. (2012) Securing Enterprise Data on Smartphones Using Run Time Information Flow Control. In: MDM '12

Larson et al.(2014). Larson J, et al. (2014) Spy Agencies Probe Angry Birds and Other Apps for Personal Data. ProPublica

Leontiadis et al.(2012). Leontiadis I, et al. (2012) Don't Kill My Ads!: Balancing Privacy in an Ad-supported Mobile Application Market. In: HotMobile '12

Lookout(2014). Lookout (2014) 2014 Mobile Thereat Report. White Paper, URL `http://goo.gl/bnt10q`

McGraw and Potter(2004). McGraw G, Potter B (2004) Software Security Testing. IEEE Security & Privacy 2(5):81–85

Neisse et al.(2013). Neisse R, et al. (2013) A Trustworthy Usage Control Enforcement Framework. Intl J Mobile Computing and Multimedia Communications 5(3):34–49

Ongtang et al.(2012). Ongtang M, et al. (2012) Semantically Rich Application-Centric Security in Android. Security and Communication Networks 5(6):658–673

Posegga and Schreckling(2011). Posegga J, Schreckling D (2011) Next Generation Mobile Application Security. In: IT-Sicherheit zwischen Regulierung und Innovation, Vieweg+Teubner Verlag, pp 181–199

Reardon et al.(2012). Reardon J, et al. (2012) Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory. In: Security '12

Russello et al.(2011). Russello G, et al. (2011) YAASE: Yet Another Android Security Extension. In: PASSAT '11

Samsung(2015). Samsung (2015) An Overview of the Samsung KNOX Platform. White Paper

Schmidt and Cohen(2013). Schmidt E, Cohen J (2013) The New Digital Age: Reshaping the Future of People, Nations and Business. Alfred A. Knopf, Inc.

Shilton(2009). Shilton K (2009) Four Billion Little Brothers?: Privacy, Mobile Phones, and Ubiquitous Data Collection. Communications of the ACM 52(11):48–53

Stach(2013a). Stach C (2013a) How to Assure Privacy on Android Phones and Devices? In: MDM '13

Stach(2013b). Stach C (2013b) Wie funktioniert Datenschutz auf Mobilplattformen? In: 43. GI-Jahrestagung

Stach(2015). Stach C (2015) How to Deal with Third Party Apps in a Privacy System—The PMP Gatekeeper. In: MDM '15

Stach and Mitschang(2013). Stach C, Mitschang B (2013) Privacy Management for Mobile Platforms—A Review of Concepts and Approaches. In: MDM '13

Stach and Mitschang(2014). Stach C, Mitschang B (2014) Design and Implementation of the Privacy Management Platform. In: MDM '14

Stahlberg et al.(2007). Stahlberg P, et al. (2007) Threats to Privacy in the Forensic Analysis of Database Systems. In: SIGMOD '07

Svajcer(2014). Svajcer V (2014) Sophos Mobile Security Threat Report. Sophos, URL `http://goo.gl/iIhj5e`

Weiser(1999). Weiser M (1999) The Computer for the 21st Century. ACM SIGMOBILE Mobile Computing and Communications Review 3(3):3–11

Xu et al.(2012). Xu R, et al. (2012) Aurasium: Practical Policy Enforcement for Android Applications. In: Security '12