Zum sicheren Löschen auf Smartphones am Beispiel von Android

Björn Roos · Harald Baier · Peter Lyko

Hochschule Darmstadt
{bjoern.roos | harald.baier}@h-da.de
PeterLyko@t-online.de

Zusammenfassung

Die Verbreitung von Smartphones nimmt rasant zu. Durch kurze Innovationszyklen der Hard- und Software von Smartphones wächst der Markt für gebrauchte Geräte. Vor der Weitergabe eines Smartphones möchte der Verkäufer typischerweise seine persönlichen Daten und oft auch die individuell installierten Anwendungen löschen. Gängige mobile Betriebssysteme löschen dabei im Dateisystem nur die Metadaten oder den Verweis auf den Speicherort der Inhaltsdaten, nicht aber den Payload selbst. Ausgangspunkt der vorliegenden Arbeit ist daher die Untersuchung, inwiefern Daten auf einem Smartphone durch einen Benutzer mit Bordmitteln des Betriebssystems „sicher“ gelöscht werden können. Die vorliegende Arbeit untersucht dabei zwei Szenarien: Löschen einer App samt aller individueller Daten sowie Löschen aller durch den Benutzer verursachten Daten auf einem Smartphone. Weil Android-basierte Geräte die mit Abstand höchste Verbreitung besitzen, führen wir unsere Untersuchungen für solche Geräte durch. Es zeigt sich, dass abhängig von Gerät und Betriebssystemversion Daten sicher gelöscht werden können oder nicht. Da also eine sichere Löschung nicht garantiert werden kann, entwickeln wir ein Konzept zum sicheren Löschen für mobile Endgeräte und setzen das Konzept prototypisch als App SLamE (Sicheres Löschen auf mobilen Endgeräten) für das Betriebssystem Android um. Ein Funktionstest zeigt, dass die App wie gewünscht arbeitet.

1 Einleitung

Smartphones erfreuen sich seit einigen Jahren großer Beliebtheit und haben einen Siegeszug in unser privates wie berufliches Leben vollzogen. Heute besitzt in Deutschland 39,8% der Bevölkerung ein solches mobiles Endgerät [Oste13]. Im Unterschied zu klassischen Telefonen können wir mit einem Smartphone unter anderem unsere Kontakte verwalten, E-Mails schreiben/empfangen sowie Bankgeschäfte erledigen. Dadurch finden sich viele persönliche oder vertrauliche Daten auf einem Smartphone.

Smartphones können im Laufe ihres Lebenszyklus ihren Besitzer wechseln - und zwar autorisiert oder nicht autorisiert. Durch kurze Innovationszyklen der Hard- und Software von Smartphones wächst zum Beispiel der Markt für gebrauchte Geräte [Press12]. Vor Verkauf möchte der bisherige Besitzer typischerweise seine persönlichen Daten und oft auch die individuell installierten Anwendungen deinstallieren. Oder ein Gerät kann verloren oder gestohlen werden. Um dieses Risiko zu begrenzen, kann der Besitzer nicht mehr benötigte Daten oder Dateien löschen. Gängige mobile Betriebssysteme löschen dabei im Dateisystem nur die Metadaten oder den Verweis auf den Speicherort der Inhaltsdaten, nicht aber den Payload selbst [Carr05]. Danach kann der Anwender die Inhaltsdaten nicht mehr mithilfe des Betriebssystems finden,

P. Schartner · P. Lipp (Hrsg.) · D•A•CH Security 2014 · syssec (2014) 273-286.

unter Zuhilfenahme von IT-forensischen Tools ist dies aber oft möglich (z.B. mittels des Digital Forensic Framework [DFFr13], File Carving oder einer Stringsuche). Ähnliches gilt über die Aussage zu Aktivitäten auf dem Smartphone (sogenanntes Timelining), etwas Aussagen zu der Frage: Auf welche Dateien wurde wann wie zugegriffen?

Ausgangspunkt der vorliegenden Arbeit ist daher die Untersuchung, inwiefern Daten auf einem Smartphone durch einen Benutzer mit Bordmitteln des Betriebssystems „sicher" gelöscht werden können. Unter „sicher" verstehen wir dabei, dass ein fachkundiger Informatiker/IT-Forensiker die Daten im benutzbaren[1] Speicher mit vertretbarem Aufwand nicht wiederherstellen kann (z. B. weil Datenblöcke auf dem Datenträger überschrieben werden, wenn spezielle Flasher Tools eingesetzt werden müssen oder ein Zugriff nur mittels JTAG[2] Testzugangsport möglich ist). Die vorliegende Arbeit untersucht dabei zwei Szenarien:

1. Löschen einer App samt allen individuellen Daten: Möglicher Anwendungsfall ist auch hier Verlust oder Diebstahl des Gerätes.
2. Löschen aller durch den Benutzer verursachten Daten auf einem Smartphone: Ein typisches Szenario ist der Verkauf des Smartphones.

Es existieren zwar Apps, wie der von Google zur Verfügung gestellte Android Reset, Autowipe, Shredroid oder Secure wipe. Diese Applikationen arbeiten aber nicht auf allen Geräten und Android Versionen zuverlässig. Weil Android-basierte Geräte die mit Abstand höchste Verbreitung besitzen, führen wir unsere Untersuchungen für solche Geräte durch. Die Untersuchungen erfolgen auf zwei Gerätetypen, einem Samsung Galaxy S3 GT-I9300 und einem Samsung Galaxy S2 GT-I9100 mit je einer Android Version 4.x. Die Untersuchungen zeigten, dass es Szenarien gibt, in denen die Daten nicht sicher von den Geräten gelöscht werden können. Deshalb entwickelten wir ein Konzept zum sicheren Löschen für mobile Endgeräte und setzen das Konzept für die Szenarien prototypisch als App SLamE (Sicheres Löschen auf mobilen Endgeräten) für das Betriebssystem Android um. Ein Funktionstest auf beiden Geräten zeigt, dass die App wie gewünscht arbeitet.

Der Rest dieser Arbeit ist wie folgt aufgebaut: Das Kapitel 2 gibt einen Überblick über verwandte Arbeiten auf dem Gebiet des sicheren Löschens auf NAND-Datenträgern. Im darauf folgenden Kapitel werden benötigte Grundlagen vermittelt. Im Kapitel 4 werden die Anwendungsfälle zum Entfernen von Benutzerdaten erläutert, um im anschließenden Kapitel die Sicherheit der vom System bereitgestellten Löschmöglichkeiten zu analysieren. Im Kapitel 6 wird das Konzept zum sicheren Löschen vorgestellt. Die Implementierung und Evaluation des Konzeptes erfolgt im Kapitel 7. Das letzte Kapitel gibt eine Zusammenfassung über die Ergebnisse und ein Ausblick.

2 Related Work

Reardon, Basin und Capkun lieferten 2013 ein System of Knowledge in dem sie Arbeiten auf dem Gebiet des sicheren Löschens beschreiben [ReBC13]. Diese werden dabei nach dem Interface und dem Datenträgertyp sortiert dargestellt. Ein Abschnitt beschäftigt sich mit dem si-

[1] Ein NAND-Flashspeicher beinhaltet ein Spare Area, auf den nicht ohne weiteres und nur mit höherem Aufwand zugegriffen werden kann. In diesem können Benutzerdaten gespeichert sein (Kapitel 4.1).

[2] Joint Test Action Group: Hard-, und Softwaretest nach dem IEEE-Standard um Schaltungen im laufenden Betrieb zu testen ohne die Funktion des Bauteils zu beeinträchtigen.

cheren Löschen auf Benutzerebene und liefert dabei fünf Ansätze zum sicheren Löschen. Davon können zwei als sicher für die Voraussetzungen in einem NAND-Datenträger betrachtet werden [ReBC13]. Das secure erase Verfahren umfasst eine Software, die über die Kommunikation mit dem Hardware-Controller ein sicheres Löschen aller Daten auslöst [ReBC13]. Auf einer tieferen Ebene ist also eine Möglichkeit gegeben, alle Daten auf dem physischen Datenträger zu löschen, welche von der Benutzerschicht aus initiiert wird [ReBC13]. Vergleichbar ist dies mit dem Ausführen eines Werksrests unter Android. Durch das freespace filling wird der nicht belegte Speicherbereich beschrieben, um die darauf unter Umständen noch gesicherten Daten zu löschen. Die Anwendung SlamE ist eine Adaption dieses Prinzips für Android. Da bei NAND-Datenträgern ein hoher Aufwand durch Wear Leveling[3] entsteht, existieren Ansätze, die darauf abzielen, den freien Speicherbereich permanent zu warten. Hierbei wird nur eine bestimmte Menge an gelöschten Daten zugelassen.

In [KJDN08] wurden bereits 2008 zwei generelle Möglichkeiten zum Löschen von Daten auf NAND-Datenträgern vorgestellt, welche zum einen das sogenannte Block Cleaning und zum anderen das Zero Overwriting sind, bei dem alle Daten eines Speicherbereiches mit Nullen überschrieben werden. Die zweite Methode ist allerdings nicht ohne weiteres umsetzbar. Hier wird ein bereits beschriebener Block, der als gelöscht markiert wurde, mit Nullen überschrieben [KJDN08]. Beim Schreiben von Seiten ist eine Änderung der Ladung in den Speicherzellen nur in eine Richtung möglich. Das heißt, der Wert einer Speicherzelle kann von einer Eins zu einer Null geändert werden, nicht umgekehrt. [KJDN08] zeigen auf, dass es auf die Menge der Daten ankommt, welches Prinzip effektiver ist. Bei großen Datenmengen macht es Sinn, das Block Cleaning zu nutzen, bei kleinen das überschreiben mit Nullen, wobei dieses zu Fehlern führen kann und nicht von jedem NAND-Datenträger unterstützt wird. Diese Arbeit zeigt die grundlegenden Prinzipien auf, wie auf NAND-Datenträger gelöscht werden kann, sie ist jedoch nicht aus der Benutzerebene anwendbar. SLamE liefert diese Möglichkeit. Keine der beiden Prinzipien werden direkt angewandt, eher indirekt, falls der Datenträgertreiber während des Füllens, Blöcke zum Wiederbeschreiben löscht.

Shuba [Subh09] präsentierte 2009 eine Lösung, welche Dateien nicht direkt löscht, jedoch unlesbar macht. In jedem Block auf einem NAND-Datenträger befinden sich einige Bits zur Fehlerkorrektur [Subh09]. Dieser Bereich wird als Error Correcting Code (ECC) bezeichnet. Er umfasst einen redundanten Anteil an Daten, der dafür genutzt wird, Fehler im entsprechenden Block zu korrigieren, falls solche auftreten [Subh09]. Um Dateien im entsprechenden Block unlesbar zu machen, werden drei Bits im ECC Speicherbereich gekippt [Subh09]. Des Weiteren werden per Zufall weitere drei Bits im Speicherbereich des Blocks gekippt [Subh09]. Durch die zweite Änderung entsteht ein Fehler, der mittels ECC korrigiert werden soll. Durch die Änderung im ECC ist das allerdings nicht möglich. Somit sind die Daten nicht mehr lesbar und können als gelöscht bezeichnet werden [Subh09]. Shuba gibt eine Performanzsteigerung von 92.88% im Vergleich zu Algorithmen unter der Methoden des Zero Overwriting und eine Steigerung von 92,98% im Vergleich zur Block Cleaning Methode an. Diese Methode ist sehr hardwarenahe und nicht aus der Benutzerschicht durchführbar. Für ein sicheres Löschen ist es ausreichend, wenn beim Zugriff auf die Daten der herkömmliche Weg verwendet wird. Des Weiteren stehen die Daten theoretisch noch zur Verfügung.

---

[3] Durch das Wear-Leveling werden Flashzellen gleichmäßig abgenutzt, um die Lebensdauer zu erhöhen. Grundsätzlich existieren zwei Methoden, dynamisch und statisch. Bei der dynamischen Methode werden durch den Controller, die am wenigsten benutzten freien Blöcke zum Schreiben verwendet. Bei der statischen Methode wird durch umlagern der Daten auf wenig benutzte Bereiche des gesamten Speichers, die gesamte Lebensdauer erhöht.

In [Ilho2] wurde 2012 eine Lösung für das Problem von nicht überschreibbaren Seiten geliefert. Damit ist das sogenannte out-of-place Update[4] gemeint. Mittels Änderungen am internen Buffer und einer Manipulation im Flash Translation Layer[5] wird in [Ilho12] ein sicheres Löschen gewährleistet, in dem Informationen bezüglich der Datenverarbeitung gesichert werden. Es wird festgehalten, wie viele Versionen einer Datei bestehen, gesucht und entfernt werden müssen.

3 Grundlagen

3.1 Funktionsweise von NAND-Flashspeicher

Ein NAND-Flashspeicher, besteht aus einer großen Menge von Zellen, die Ladungen über einen langen Zeitraum speichern können [MiCM10]. Die Zellen haben eine begrenzte Lebensdauer, die sich auf ihre Programm- und Löschzyklen bezieht und wird vom NAND-Hersteller festgelegt [King14]. Jede durchgeführte Programm- und Löschfunktion reduziert die Fähigkeit, eine elektrische Ladung zuverlässig zu speichern. Die Lebensdauer eines NAND-Flashspeichers ist abhängig von der Anzahl der Programm-/Löschbeständigkeit, auf die Geometrie bezogene Lese-/Programm-/Löschkomplexität, Flashspeicher Kapazität, FSP Funktion und Effizienz [King14].

Die Anordnung und Verschaltung dieser Zellen miteinander macht den resultierenden Speicher zu einem NAND- oder NOR-Datenträger [MiCM10]. Die Zellen können 1 Bit (SLC – Single-Level Cell) 2 Bit (MLC – Multi-Level Cell) oder 3 Bit (TLC – Triple-Level Cell) an Daten speichern. Allerdings wird es mit zunehmender Anzahl der speicherbaren Bits schwieriger zwischen den Zuständen zu unterscheiden. Die Zuverlässigkeit wird aber durch eine fortschrittliche Fehlerkorrekturcode und anderen Techniken stark erhöht, so dass man aus Kosten und Kapazität auf die MLC oder TLC Technik setzt [Sams13].

Die Zellen bilden bei einem NAND-Speicher in Reihenschaltung einen String [MiCM10], der als Spalte bezeichnet wird. Die einzelnen Zellen einer Spalte werden jeweils parallel mit den Zellen der benachbarten Spalten in einer Wordline angesteuert [MiCM10] und als Zeile bezeichnet. Eine Zeile wird im Zusammenhang mit NAND-Datenträgern allgemein als Seite bezeichnet, wobei mehrere dieser Seiten einen Block bilden [MiCM10]. Die Seiten haben typischerweise eine Größe zwischen 512 und 8192 Bytes und sind die die kleinste lesbare-/ schreibbare Einheit auf dem Flash-Speicher. Die Blöcke haben typischerweise eine Größe zwischen 512 und 2048 KB und sind die kleinste löschbare Einheit auf dem Flash-Speicher.

Auf einem NAND-Flashspeicher wird auf Seitenebene gelesen/geschrieben und auf Blockebne gelöscht. Soll nun eine Seite innerhalb des Blocks beschrieben oder gelöscht werden, so muss zuerst der gesamte Inhalt aller Seiten des Blocks in einen Zwischenspeicher kopiert und gelöscht werden, bevor der Inhalt auf dieselbe Blockadresse gespeichert werden kann. Ist die zu beschreibende Seite leer, dann kann sie direkt in den Block geschrieben werden [King14].

---

[4] Anstelle eines kostspieligen Lösch- und Schreibzyklus wird bei einer Aktualisierung der Seite, diese an einer anderen freien bereits gelöschte Stelle gespeichert. Die alte Version der Daten werden logisch als ungültig markiert und später durch den Garbage Collection Mechanismus gelöscht.

[5] Das Flash Translation Layer ist eine zusätzliche Softwareschicht, die ein blockorientiertes Gerät auf einem Flash-Speicher emuliert, so dass herkömmliche Dateisysteme verwendet werden können. Das FTL wandelt die Zugriffe auf den Speicher um. Die Schreibanforderungen können so gleichmäßig auf alle Bereiche des Flash-Speichers verteilt werden.

Um die Lebensdauer eines NAND-Flashspeichers zu erhöhen, werden zum einen sogenannte Wear-Leveling Algorithmen verwendet, um eine gleichmäßige Verteilung der Lesen-Löschen-Modifizieren-Schreiben-Zyklen für alle Blöcke zu gewährleisten und zum anderen Reserve Speicherzellen eingesetzt, die zusammen das Spare Area ergeben. Die Blöcke im Spare Area stehen als Ersatz für defekte Blöcke, zum Wear Leveling und als Zwischenspeicher für die Read-Modify-Write Zyklen zur Verfügung. Normalerweise werden vom Hersteller bis zu 7% der Gesamtkapazität für Flashspeicher bis zu einer Größe von 128GB als Spare Area verwendet [King14]. Durch Over-Provisioning (d.h. Vergrößerung des Spare Areas) kann dieser Bereich bei SSD-Festplatten vergrößert werden.

3.2 Das Löschen von Daten

Auf dem NAND-Flashspeicher existieren unterschiedliche Bereiche, wo Daten gespeichert werden können. Normalerweise werden die Daten im benutzbaren Speicher abgespeichert. Durch das Wear-Leveling oder durch das Bad Block Management können Daten aber auch in anderen Bereichen, wie z. B. dem Spare Area oder in als defekt markierte Blöcke gespeichert werden, die nur dem Controller zugänglich sind [Wilt13]. Auf diese Bereiche kann normalerweise nur mit Hilfe von Flasher Tools, einem JTAG[6] Testzugangsport oder mittels einer Extraktion des physischen Speichers zugegriffen werden [BMK+07], bis sie durch die Garbage Collection Routine während der Leerlaufzeiten gelöscht werden [Wilt13]. Ein Löschen dieser Bereiche mit Bordmitteln kann selbst bei einem Werksreset mit Secure Erase Kommando nicht garantiert werden [Wilt13].

Wird mithilfe eines Betriebssystems eine Datei im benutzbaren Speicher gelöscht, so wird nur der Verweis auf den Speicherort der Datei gelöscht, nicht aber die Datei selbst [Fox09]. Um ein tatsächliches Löschen durchzuführen, ist zusätzliche Software nötig, welche die Speicherbereiche der Dateien überschreibt. Das sichere Löschen umfasst ein mehrfaches Überschreiben mit speziellen Bytemustern [Gutm96]. Ein sicheres Löschen gemäß Gutmann ist hier jedoch nicht anwendbar, da in einem mobilen Endgerät kein magnetischer Datenträger verbaut ist, sondern ein Flash Datenträger [Hoog12]. Dieser führt grundsätzlich andere Operationen zum Beschreiben, Lesen und Löschen aus. Hier werden Daten nicht durch magnetische, sondern durch elektrische Ladung dauerhaft gespeichert. In aktuellen mobilen Endgeräten sind Flash-Datenträger mit NAND Technologie verbaut [MiCM10]. In diesen Speichern können Daten auf einer Seite nur gelöscht werden, wenn der gesamte entsprechende Block gelöscht wird. Befinden sich noch benötigte Daten auf den anderen Seiten des Blocks, erschwert dies das „sichere“ Löschen auf einem NAND-Datenträger. Durch das Speichern von noch benötigten Daten eines zu löschenden Blocks entsteht eine Verteilung von Daten. Beim Löschen solcher verteilten Daten müssen alle Fragmente berücksichtigt werden. Des Weiteren entspricht das Löschen einer Änderung. Der entsprechende Speicherbereich wird auf dem NAND-Datenträger nicht überschrieben. Vielmehr wird einfach ein anderer freier Block mit den geänderten Daten beschrieben. Die alte Version verbleibt auf dem Datenträger und kann wiederhergestellt werden solange sie nicht überschrieben wurde. Das Überschreiben eines Blockes auf einem NAND-Datenträger reicht aus, um ein Wiederherstellen der dort gesicherten Daten auszuschließen. Jedoch muss sichergestellt sein, dass alle Kopien der zu löschenden Daten berücksichtigt werden.

---

[6] Oftmals ist nicht bekannt, ob es ein JTAG-Testzugangsport existiert. Viele Hersteller geben diese Information nicht Preis, so dass der Forensiker unter Umständen versuchen muss den Port zu finden.

3.3 Benutzerdaten auf einem Android-System

Eine Unterscheidung von Benutzerdaten ist vor allem wichtig, wenn mittels File Carving der Datenträger analysiert wird. Sowohl die Applikations-Datei, als auch eine mögliche Datei des Nutzers würden in diesem Fall als Ergebnis anfallen und es gilt diese richtig einzuordnen. Dies führt zur folgenden Definition von Benutzerdaten:

Benutzerdaten sind alle Daten, die abgesehen von den Systemdaten des Herstellers auf dem Endgerät gespeichert werden.

Allerdings sind nicht alle Daten, die durch die Benutzung entstehen hinsichtlich des Datenschutzes als wichtig einzuordnen. Eine durch den Benutzer installierte Applikation ist aus forensischer Sicht nicht unbedingt interessant [Hoog12]. Möchte der Benutzer jedoch seine Daten sicher löschen, wird er auch installierte Software entfernen. Die Daten, die durch die Benutzung einer solchen Applikation entstehen, sind zu einem großen Teil nach § 3 Abs.1 und Abs. 9 BDSG als schützenswert einzustufen. Stellt ein Benutzer Daten öffentlich zur Verfügung, sind diese für einen Forensiker uninteressant, da sie ohnehin zugänglich sind [Hoog12]. Ein Nutzer jedoch, der sein Handy weitergeben möchte, will auch solche Daten löschen.

4 Anwendungsfälle zum Entfernen von Daten

4.1 Zurücksetzen des Systems

Heutige Smartphones bieten die Möglichkeit das System zurückzusetzen. Dies wird als Werksreset bzw. „Wipe“ bezeichnet. Bei einem Werksreset werden Einstellungen, installierte Anwendungen und Benutzerdaten gelöscht. Das Gerät wird in einen Zustand der Werksauslieferung zurückgesetzt.

4.2 Löschen von Datensätzen innerhalb einer Applikation

Innerhalb einer Applikation stellt der Entwickler Möglichkeiten bereit, gesicherte Daten wieder zu löschen. In diesem Fall werden also Inhalte einer Datei entfernt, nicht aber komplette Dateien. Meistens sind SQLite Datenbanken bei dieser Art von Löschvorgang betroffen. Benutzerdaten können bspw. E-Mails, Kontakte, Browser-Links, Kalendereinträge, Anruflisten oder sogar Routen einer Navigations-Applikation sein. In der Android-Version 4.1.2 können solche Benutzerdaten innerhalb der standardmäßig installierten Applikationen anfallen.

4.3 Löschen über den Android Anwendungs-Manager

Der Anwendungs-Manager liefert neben der Möglichkeit die Anwendung zu deinstallieren, auch die Möglichkeit Benutzerdaten zu löschen. Es ist möglich nur den Cache zu löschen oder alle Daten der Anwendung.

4.4 Manuelles Löschen von Dateien

Das manuelle Löschen von Dateien erfolgt unter Android über einen Dateimanager. Unter Android gibt es dafür die native Applikation "Eigene Dateien". Damit lassen sich Dateien auf der externen Speicherkarte und auf der emulierten externen Speicherkarte öffnen, verschieben und auch löschen. Der Löschvorgang ist vergleichbar mit dem Löschen über den Android Anwendungs-Manager. Aus diesem Grund wird im folgenden Kapitel darauf nicht näher eingegangen.

5 Sicherheit der Löschmöglichkeiten unter Android

5.1 Zurücksetzen des Systems

Der Werksreset oder unter Android 4.1.2 auf dem Samsung Galaxy S3 „Sichern und zurücksetzen“ genannt, wird von Nutzern eines Smartphones vor dem Verkauf häufig verwendet. Beim Aufrufen dieser Funktion warnt das System davor, dass alle Benutzerdaten auf dem Gerät unwiederbringlich gelöscht werden. Um die Sicherheit des Werksresets hinsichtlich gelöschter Benutzerdaten zu überprüfen, wurde eine Datei *EmailProvider.db* über den Reset hinweg beobachtet. Die ausgewählte Datei befindet sich in der Benutzerdaten-Partition (data) und ist auf jedem Smartphone, dass die Android Version 4 enthält zu finden. Nach dem Werksreset ist die Datei an der ursprünglichen Stelle nicht mehr vorhanden. Stattdessen befindet sich an dieser Stelle eine SQLite Datei ohne jeglichen Inhalt. Der Inode, welcher zuvor die Datei referenzierte, ist in der dazugehörigen Blockgruppe keiner Datei mehr zugeordnet. Entweder wurde *EmailProvider.db* komplett gelöscht und durch die Installation der Android Applikationen während des Zurücksetzens überschrieben oder als freier Speicherbereich markiert und ebenfalls überschrieben. Neben einem kleinen Teil wird nun der gesamte Speicherbereich betrachtet. Dazu wurde sowohl beim Modell S3 als auch beim Modell S2 eine Datei, die den String „Peter“ enthält, im Data-Bereich gesichert. Nahezu jeder nicht vom System belegte Speicherbereich wurde mit diesem String beschrieben. Eine String-Analyse bestätigt, dass nach dem Zurücksetzen des Systems der komplette Speicherbereich beim Galaxy S3 überschrieben wurde. Der String „Peter“ kommt nicht ein einziges Mal vor. Beim Samsung Galaxy S2 ergibt derselbe Test ein anderes Ergebnis. Nach dem Zurücksetzen des Systems wurde, wie zuvor beim Modell S3, die Ausgabe der Strings Analyse vor und nach der „Befüllung“ verglichen. Der Test-String konnte in beiden gefunden werden. Fraglich ist nun, ob bei einem Reset nur allozierte Bereiche des Datenträgers berücksichtigt werden. Wäre das der Fall, würde das System nur Benutzerdaten löschen, die es kennt. Anderenfalls kann man inklusive der bereits gemachten Erkenntnisse beweisen, dass der komplette Speicherbereich zurückgesetzt wird. Dieser Versuch wurde auf dem Samsung Galaxy S3 vorgenommen. An das Ende der Cache Partition wurde der String „Peter“ in den nicht allozierten und zuvor mit ausschließlich 00 beschriebenen Block eingefügt. Der dafür benötigte Platz von 5 Byte wurde zuvor gelöscht, damit die Größe des Images gleich bleibt. Nach dem Reset ist der String „Peter“ in dieser Partition nicht zu finden. Der letzte Block ist wie zuvor, ausschließlich mit dem Byte 00 beschrieben und nicht alloziert.

5.2 Löschen von Datensätzen innerhalb einer Applikation

Deinstallation einer Applikation

Die Applikation *Web.de Mail* speichert wie die vergleichbare Standard Software von Android Benutzerdaten aus dem E-Mail Konto des Nutzers. Wobei die Umsetzung etwas anders ist. Dateien etwa, die als Anhang heruntergeladen werden, sichert das Programm in einem separaten Ordner im Database Verzeichnis. Im Cache Ordner, den die Android Software für Dateien aus dem Anhang verwendet, sichert *Web.de Mail* bei diesem Test keine Benutzerdaten. Eine Datenbank, welche die geladenen Nachrichten in einer Datei enthält, existiert hier. Diese soll im Folgenden, vor und nach der Installation von Web.de Mail betrachtet werden. Betrachtet wird dies zunächst am Samsung Galaxy S3. Es wurde der Inode und der Speicherbereich der untersuchten Datenbankdatei gesichert. Die Fragmentierung dieser Datei beläuft sich auf drei Teile bei einer Dateigröße von 90112 Byte. Die tatsächliche Größe beläuft sich allerdings auf

90113 Byte. Dieses eine Byte wird nach dem Deinstallieren wichtig sein. Nachdem die Deinstallation durchgeführt wurde, findet sich im Pfad *data/data* das Verzeichnis *de.web.mobile.android.mail* nicht mehr, welches zuvor alle Benutzerdaten enthielt. Der referenzierende Inode verweist noch immer auf die FS-Blöcke (Filesystem Blöcke), welche vor der Deinstallation ausgewiesen wurden. Die Dateigröße jedoch ist auf null gesetzt worden. Es wurde ein Löschvorgang durchgeführt, wodurch bis auf den letzten Block der Datei jeglicher Inhalt davon mit dem Byte 00 überschrieben wurde. Zu beachten ist, dass im fünften FS-Block nur ein einziges Byte gesichert wurde. Exakt dieses Byte ist nach dem Deinstallieren der Applikation auf dem Datenträger zurückgeblieben. Beim Samsung Galaxy S2 unterscheidet sich das Ergebnis zu dem des Samsung Galaxy S3. Im Inode der untersuchten Datenbank ändert sich nach dem Deinstallieren die Dateigröße auf null. Der Verweis zu den Blöcken, welche die Dateifragmente beinhalten, bleibt unverändert. Der Inode selbst, wird nach dem deinstallieren der Applikation, als frei ausgewiesen. Die komplette Datei und ihr Inhalt sind in den zuvor referenzierten Fragmenten noch vorhanden.

Löschen des App-Caches

Die Android *E-Mail* Applikation enthält in ihrem *Cache* Verzeichnis Benutzerdaten. Diese Dateien speichert die Software beim Laden einer Datei aus dem Anhang einer E-Mail. Das *Cache* Verzeichnis der *E-Mail* Applikation beinhaltet drei Dateien. Nach Ausführung der Option „Cache löschen“ wurde erneut eine Kopie des gleichen Bereiches erstellt. Beim Samsung Galaxy S3 sind nach dem Löschen keine Dateien mehr über das Betriebssystem zu finden. Die Blöcke, die die Dateien belegten, wurden ausschließlich mit dem Byte 00 beschrieben und wurden somit gelöscht. Beim Samsung Galaxy S2 ist nach dem Löschen des Caches im Inode der betrachteten Datei sowohl die Dateigröße als der Verweis zu den genannten Blöcken gelöscht worden. Auf dem Datenträger befindet sich allerdings der Dateiinhalt, der von diesem Inode referenziert wurde, in unveränderter Form. Dies gilt ebenso für alle anderen Dateien, welche zuvor im *Cache* Verzeichnis abgelegt waren.

Löschen aller Datensätze

Es wird nun die Umsetzung beim Löschen aller Datensätze betrachtet. Es wird der Inode und der Speicherort der *EmailProviderBody.db* Datei untersucht. Diese Datei existiert immer, unabhängig davon, ob Benutzerdaten gesichert wurden oder die Applikation verwendet wurde. Dadurch kann ein guter Vergleich, vor und nach dem Vorgang des Löschens durchgeführt werden. Nachdem die Option „Daten Löschen“ beim Samsung Galaxy S3 durchgeführt wurde, referenziert der verwendete Inode noch immer die beobachtete Datenbankdatei. Eine Untersuchung des Inodes zeigt jedoch, dass dieser nun auf andere Bereiche des Datenträgers verweist. Die Dateigröße hat sich von zuvor 53248 Byte auf 24576 Byte nahezu halbiert. Der Speicherbereich, der zuvor von diesem Inode referenziert wurde, ist in allen Fragmenten unverändert geblieben. Das heißt, es wurde eine neue Datenbankdatei erstellt, die alte Datenbank blieb unverändert, wird jedoch nicht mehr von einem Inode referenziert und kann somit nicht mehr über das Dateisystem gefunden werden. Eine Wiederherstellung ist möglich, bis die Fragmente der Datei durch andere Daten überschrieben wurden. Beim Samsung Galaxy S2 sind, nachdem die Option „Daten löschen“ ausgeführt wurde, wie zuvor beim Ausführen der “Cache löschen” Option die Inodes der Dateien verändert worden. Die Dateien selbst liegen unverändert auf dem Datenträger vor. Ein Ausführen des Befehls *ls* zeigt, dass der Inhalt des Benutzerdaten Verzeichnisses auf den Stand einer Neuinstallation zurückgesetzt wurde. Teilweise wurden für

diese Dateien Inodes verwendet, welche vor dem Löschen der Daten auf die nun verwaisten Benutzerdaten referenzierten.

6 Konzept zum sicheren Löschen

Das Konzept lässt sich in zwei Schritte unterteilen, das Löschen und das Füllen. Das Löschen der Dateien wird über eine Bibliothek festgelegt. Diese Bibliothek enthält Listen, die bestimmten Benutzerdaten zugeordnet werden. Das heißt, es können entweder alle Dateien, die von einer Applikation erstellt wurden, gelöscht werden oder es existiert eine spezielle Liste. Um vom Benutzer erstellte Dateien zu berücksichtigen, beinhaltet die Bibliothek dafür den Pfad der emulierten externen SD Karte. Dadurch ist es möglich, alle der Bibliothek bekannten Benutzerdaten zu löschen. Die Inodes und die Dateinamen aller ausgewählten Dateien werden gesichert. So kann der Speicherbereich der entsprechenden Dateien sowohl vor als auch nach dem Löschen ermittelt werden. Einerseits kann damit das Löschen selbst vorgenommen werden, andererseits kann im Nachhinein überprüft werden, ob der Speicherbereich tatsächlich gelöscht wurde. Jedes Objekt dieser Liste wird folgendermaßen verwendet. Mittels Dateinamen kann eine Referenz auf die zu löschende Datei erzeugt werden. Über diese wird geprüft, ob eine Datei mit diesem Namen existiert und welche Speicherkapazität diese besitzt. Im Anschluss kann diese mithilfe des Dateisystems überschrieben werden. Daraufhin wird die nun unbrauchbare und nicht mehr lesbare Datei gelöscht. Bei einem NAND-Datenträger kann das Überschreiben der Datei weggelassen werden. Ein bereits beschriebener Block kann nicht überschrieben werden. Somit würde der Versuch, eine Datei zu überschreiben, dem Erstellen einer neuen Datei gleichen. Das Löschen ändert die Meta-Informationen der betroffenen Datei. Dadurch wird aus dem Dateiinhalt freier Speicherplatz. Beim Überschreiben geschieht das ebenfalls, jedoch auf einem anderen Weg. Der Inode, welcher zuvor auf die zu löschende Datei wies, referenziert nun eine neue Datei an einer zuvor freien Stelle des Speichers. Die bisher durchgeführten Schritte führen im Falle einer vom Benutzer erstellten Datei dazu, dass ihr Speicherbereich wieder freigegeben wird. Bei einer Applikation wird diese in den Zustand vor der ersten Nutzung versetzt, da keine gesicherten Daten mehr vorliegen. Benutzerdaten einer Applikation sind ausschließlich auf dem persistenten Speicher des Endgerätes gesichert. Bei einem Neustart muss die Applikation diese zunächst wieder laden. Liegen keine Benutzerdaten vor, wird die App wie bei der ersten Nutzung ausgeführt.

6.1 Füllen – Methode 1

Um alle gelöschten Dateien unwiderruflich zu entfernen, wird der gesamte freie Speicher der Partition beschrieben. Damit kann sichergestellt werden, dass keine ältere Version einer gelöschten Datei oder die gelöschte Datei selbst, zurückbleibt. Zunächst werden der vorhandene freie Speicherplatz der Benutzerdaten-Partition und das verwendete Dateisystem ermittelt. Es kann für die Erfassung aller wichtigen Benutzerdaten nur die Benutzerdaten-Partition betrachtet werden. Durch den ermittelten Wert kann eine Einschätzung der zu erzeugenden Dateien gemacht werden, welche den Speicherbereich füllen. Der Inhalt dieser Dateien besteht ausschließlich aus Nullen. Die Größe der Dateien wird über den Divisor der Gesamtkapazität bestimmt. Bei den Dateisystemen ist zu beachten, dass unter FAT die Dateigröße unter vier Gigabyte bleibt, da das die maximal zu speichernde Dateigröße ist [Carr05]. Unter EXT4 liegt sie bei 2 Terabyte [MaCB07] und ist daher vernachlässigbar. Um die Dateien möglichst groß werden zu lassen, wird der Divisor zunächst klein gewählt und anschließend nach oben hin korrigiert.

Nachdem die Größe feststeht, werden alle Dateien in einem expliziten Verzeichnis erstellt. Somit kann im Anschluss dieses Verzeichnis komplett gelöscht werden um den Speicherbereich, der nun ausschließlich mit Nullen beschrieben ist wieder freizugeben. Damit wurde erreicht, dass ein sicheres Löschen durchgeführt wurde. Allerdings erzeugt dieses Vorgehen eine hohe Abnutzung des NAND-Datenträgers. Um dies zu verhindern, wird durch das folgende aufwendigere Verfahren nur der nötige Speicherbereich beschrieben.

6.2 Füllen – Methode 2

Dieses Verfahren bezieht sich auf das bei aktuellen Android Smartphones übliche Dateisystem EXT4. Es wird zunächst eine Datei erstellt, die der Blockgröße des Dateisystems entspricht. Anschließend wird für jede Blockgruppe der Benutzerdaten-Partition folgende Prozedur durchgeführt. Aus der Databitmap wird ermittelt, welche Blöcke nicht alloziert sind. Diese Blöcke werden nacheinander mit der erstellten Datei verglichen. Sind diese gleich, wird der Block als alloziert markiert und es wird fortgefahren. Anderenfalls wird der Block überschrieben und ebenfalls als belegt markiert. Beide Markierungen müssen in einer Liste gesichert werden. Nachdem der gesamte freie Speicherbereich behandelt wurde, werden alle gemerkten Markierungen wieder entfernt, um den Speicherbereich freizugeben. Das Ergebnis ist mit dem der Methode 1 identisch, jedoch wird der Datenträger geschont. Der Grad der Schonung hängt dabei von der Größe des unbeschriebenen Speicherbereiches ab. Je mehr Speicher unberührt ist, desto weniger Abnutzung findet durch das Füllen statt. Im Gegensatz zu Methode 1 ist diese Methode sehr viel aufwendiger.

Fragmente einer sicher zu löschenden Datei werden beim Füllen ebenfalls entfernt, wenn sie auf einem Block mit noch benötigtem Inhalt gesichert sind. Der Garbage Collector speichert benötigte Seiten an anderer Stelle und macht den frei gewordenen Block wieder beschreibbar. Das Problem, Fragmente einer gelöschten Datei im Fileslack einer nicht gelöschten Datei wiederherstellen zu können, besteht hier ebenfalls nicht. Da eine Seite zunächst gelöscht sein muss, bevor sie beschrieben werden kann, entsteht erst gar kein Fileslack. Das Prinzip des vorgestellten Konzeptes ermöglicht es, sicheres Löschen unabhängig vom Typ des Endgerätes durchzuführen. Bei der Umsetzung kann eine Lösung angestrebt werden, die es dem Nutzer ermöglicht Benutzerdaten auszusuchen, die er sicher gelöscht wissen möchte. Danach wird beim Prozess des Füllens sichergestellt, dass es unmöglich ist, diese gewählten Daten wiederherzustellen.

7 Implementierung und Evaluierung der Applikation

7.1 Inhalt und Struktur der Applikation

Die Applikation umfasst das Löschen von Benutzerdaten, welche durch die Nutzung einer Applikation erzeugt werden. Des Weiteren ist es möglich die standardmäßigen Daten, wie Datenbanken, Cachedateien und Einstellungen für jede beliebige installierte Applikation auf dem verwendeten Endgerät sicher zu löschen. Das Füllen des Datenträgers ist ebenfalls eine Funktion der hier beschriebenen Applikation. Der Anwender kann diese Funktion ausführen, um sicherzugehen, dass sich keine verwaisten Daten auf der Data Partition befinden. Die Applikation besteht aus vier Klassen.

Die Klasse *Home_Activity* liefert die Startoberfläche. Beim Starten wird sichergestellt, dass eine Applikationsbibliothek im App-Verzeichnis angelegt wird. In der Applikationsbibliothek befinden sich Listen mit den Pfaden aller Dateien, die Benutzerdaten enthalten können. Die

Klasse *Apps_Activity* wird mittels Intent von *Home_Activity* aufgerufen und erzeugt eine Liste aller auf dem Endgerät befindlichen Applikationen. Für jede Applikation wird der eindeutige Name ermittelt und beim Anwählen einer App per Intent an die Klasse *Userdata_Activity* gesendet. Der Benutzer ist somit in der Lage alle installierten Apps per Klick anzuwählen. Die Klasse *Userdata_Activity* erhält einen Intent mit dem Namen der angewählten Applikation von der Klasse *Apps_Activity*. Anhand dieser Information werden die User ID, welche eine Applikation eindeutig referenziert und der Pfad des Applikations-Verzeichnisses ermittelt. Anhand der übermittelten Daten zu der ausgewählten App wird die Bibliothek auf Informationen zu Benutzerdaten durchsucht und die dort hinterlegten Pfade für die weitere Verarbeitung in einer Liste gespeichert und zurückgegeben.

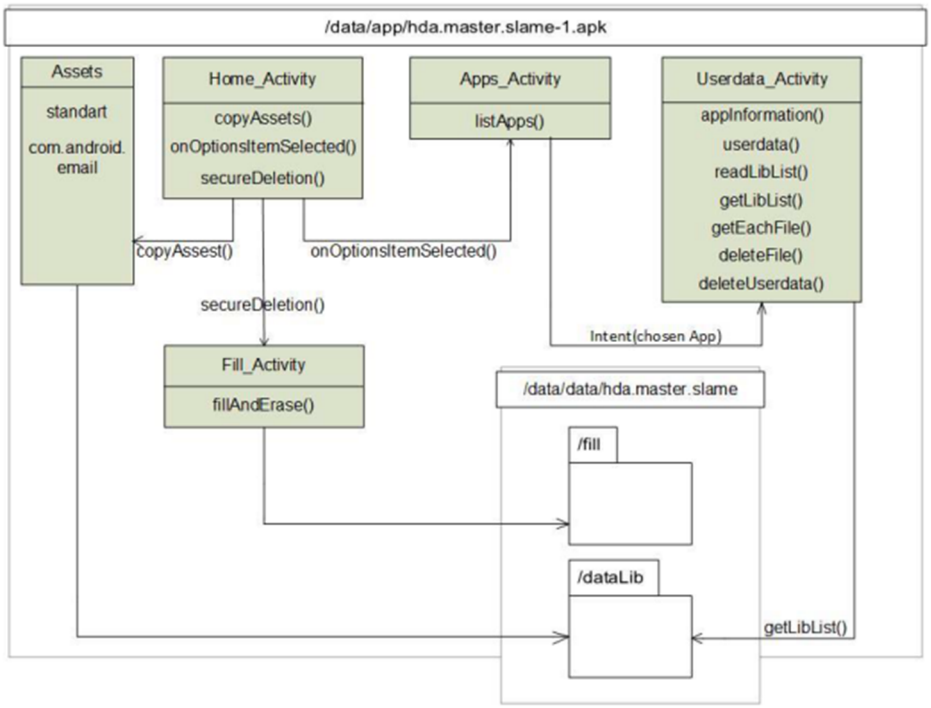

Abb. 1: Klassendiagramm der Anwendung SLamE

Existiert keine spezielle Bibliotheksdatei für die vom Benutzer gewählte App, wird die Standarddatei ausgewählt. Dadurch werden die Standardverzeichnisse Cache, Database und *Shared_prefs* zurückgegeben. Die ermittelten Pfade werden im Anschluss nach ihrem Inhalt durchsucht. Die gefundenen Dateien werden in einer Liste gesichert und zurückgegeben. Das Ergebnis ist eine Liste von Dateien, die auf der Grundlage der in der Bibliothek angegebenen Pfade, alle Benutzerdaten dieser Applikation enthält. Der Löschvorgang wird mittels eines Buttons in Gang gesetzt. An die Methode *deleteFile()* werden die zuvor ermittelten Pfade und Dateien übergeben. Diese wiederum löst, über einen Prozess in der Kommandozeile, das Löschen der entsprechenden Dateien aus. Nachdem alle Dateien gelöscht wurden, muss ein Neustart des Endgerätes durchgeführt werden, um einen Absturz des Systems beim Ausführen einer betroffenen App zu verhindern. Die Klasse *Fill_Activity* wird durch einen Intent aus der *Home_Activity* Klasse ausgeführt. Sie veranlasst, dass der freie interne Speicher des Endgerätes vollkommen beschrieben wird. Mittels der Methode *fillAndErase()* wird der zur Verfügung stehende Speicherplatz in der Data Partition vollkommen befüllt. Dafür werden Dateien, deren Inhalte nur aus Nullen bestehen im eigens dafür erstellten Ordner *fill* erzeugt. Danach wird der Inhalt dieses Ordners gelöscht, so dass der überschriebene Speicherbereich wieder freigegeben

wird. Dieser Vorgang kann abhängig vom freien Speicher ein wenig Zeit in Anspruch nehmen. Nachdem der komplette Vorgang beendet ist, wird auf dem Display erneut das Layout der Klasse *Home_Activity* angezeigt.

7.2 Evaluierung der Software

Die Evaluierung der Software für Szenario eins wird am Beispiel der Applikation *com.android.email* durchgeführt. Die Applikation wurde durch die Eingabe von Accountdaten eines E-Mail-Providers mit Benutzerdaten versehen. Die Applikation sichert dadurch automatisch alle in diesem Account vorhandenen E-Mails im entsprechenden Applikationsverzeichnis. Anschließend wurde eine exakte Kopie der Benutzerdaten-Partition erzeugt, um den Zustand vor dem Löschvorgang festzuhalten. Daraufhin wurde mit der Applikation SLamE die entsprechende Anwendung ausgewählt und ein sicheres Löschen durchgeführt. Zuletzt wurde erneut eine exakte Kopie der Benutzerdaten-Partition erzeugt. Dadurch kann mittels Vergleich von bestimmten Speicherbereichen aus beiden Partitionsabbildern gezeigt werden, ob Dateiinhalte nach dem Löschen wiederhergestellt werden können oder nicht.

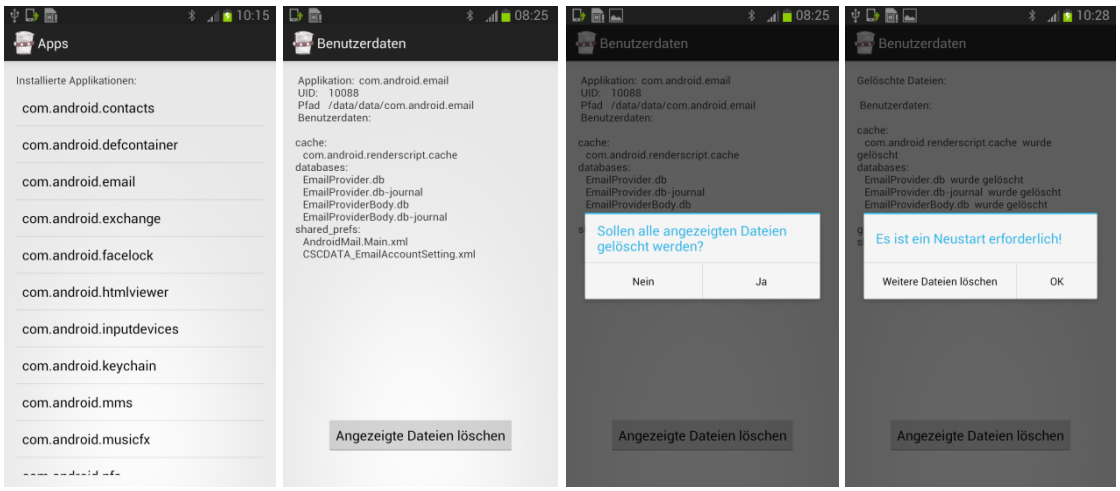

Abb. 2: Löschen aller Benutzerdaten der Applikation *com.android.email*

Das Szenario zwei ist im Vergleich zum ersten, welches ein sicheres Löschen für eine App umsetzt, eine Lösung für das sichere Löschen von allen installierten Applikationen. Das Anwenden eines Werksresets, führt dazu, dass alle installierten Applikationen, beziehungsweise erstellte Daten von installierten Applikationen, gelöscht werden. Ein anschließendes Durchführen der Funktion sicheres Löschen der App SLamE wird das gleiche Maß an Sicherheit für alle beim Zurücksetzen gelöschten Dateien liefern, wie beim sicheren Löschen innerhalb einer einzigen Applikation. Dies ist unabhängig davon ob das Zurücksetzen selbst zu einem sicheren Löschen führt oder nicht.

7.3 Ergebnis

Der Test für Szenario eins hat gezeigt, dass mittels Anwendung der SLamE Applikation auf die E-Mail-Applikation *com.android.email* alle Benutzerdaten, die durch das Erstellen des E-Mail-Accounts auf dem Datenträger geladen wurden, sicher und damit unwiderruflich gelöscht wurden. Durch die spezielle Liste aus dem *dataLib* Verzeichnis werden für diese App alle Dateien, die Benutzerdaten enthalten, berücksichtigt. Die Speicherbereiche der dort aufgeführten Fragmente einer jeden Datei wurden mit denen nach dem Ausführen der Anwendung verglichen.

Das Ergebnis ist, dass der zuvor gesicherte Dateiinhalt im Anschluss mit Nullen überschrieben ist. Im Falle von Szenario zwei ist klar, dass die Anwendung der Applikation SLamE nach einem Werksreset alle Daten sicher vom Gerät entfernt, da der komplette nicht belegte interne benutzbare Speicher mit Nullen überschrieben wird.

8 Zusammenfassung und Ausblick

Es wurde eine Lösung erarbeitet, die es ermöglicht Benutzerdaten im benutzbaren Bereich eines Flash-Speichers auf einem Android Smartphone sicher zu löschen. Die gezeigte Lösung kann unabhängig vom Datenträgertyp angewendet werden. Es ist aber zu beachten, dass noch im sogenannten Spare Area oder in als defekt markierten Blöcken Benutzerdaten gespeichert sein können. Wäre ein Zugriff auf diese Bereiche möglich, auf dem normalerweise nur der Controller des NAND-Speichers Zugriff hat, so könnten auch diese Bereiche sicher gelöscht werden, so dass vom gesamten physikalischen Speicher, keine Daten mehr wieder hergestellt werden könnten.

Ein sicheres Löschen im benutzbaren Speicher konnte nur auf dem Gerät Samsung Galaxy S3 festgestellt werden. Durch das Ausführen des Werksresets werden hier alle auf der Benutzerdaten-Partition gesicherten Inhalte unwiederbringlich gelöscht. Im laufenden System konnte nur für die Option des manuellen Löschens der vom Nutzer gesicherten Dateien ein sicheres Löschen festgestellt werden. Da der Nutzer jedoch nicht die Rechte besitzt, jede Datei auf der Benutzerdaten-Partition zu löschen, ist dies nicht ausreichend. Das Samsung Galaxy S2 hingegen bringt keine Option mit, ein sicheres Löschen durchzuführen. Alle durchgeführten Tests führten zu dem Ergebnis, dass gelöschte Dateien wiederhergestellt werden können. Das vorgestellte Konzept bietet die Möglichkeit, auf beiden Referenzgeräten ein sicheres Löschen im laufenden Betrieb durchzuführen. Die Umsetzung für die Android Version 4 bietet das sichere Löschen von Benutzerdaten innerhalb von Applikationen und nach einem Werksreset an. Ein gezieltes Entfernen von Dateien, die Benutzerdaten enthalten, wurde speziell für die Applikation *com.android.email* umgesetzt. Dies kann für jede beliebige Applikation erweitert werden. Änderungen, die durch jede neue Version von Android einhergehen, liefern stets neue Ansätze. Auch die Weiterentwicklung von NAND-Datenträgern könnte eine performantere Lösung für mobile Endgeräte in Bezug auf sicheres Löschen bieten. Wenn die Kapazität von Datenträgern zunimmt, wird das Konzept des Füllens eine langwierige Option zum sicheren Löschen sein. Eine Umsetzung des vorgestellten Konzeptes ohne die Nutzung von Superuser Rechten würde eine Lösung für jedes beliebige Endgerät liefern.

Für die Umsetzung auf Geräte anderer Hersteller wird im Rahmen einer laufenden Masterarbeit, im ersten Schritt die Sicherheit der vom System bereitgestellten Löschmöglichkeiten auf Apple Geräten geprüft. Im zweiten Schritt wird das unser Konzept zum sicheren Löschen auf iOS übertragen.

Danksagung

Wir danken CASED und dem OPEN C3S Projekt für die Unterstützung bei diesem Beitrag.

Literatur

[BMK+07] M. Breeuwsma; M. de Jongh; C. Klaver; R van der Knijff; M. Roeloffs: Forensic Data Recovery from Flash Memory. Small Scale Digital Device Forensics Journal, Vol. 1, No. 1, June 2007

[Carr05] B. Carrier: File System Forensic Analysis. 9th ed., Addision-Wesley, 2005.

[DFFr13] DFF: Digital Forensics Framework - Open Source Digital Investigation Software, URL: http://www.digital-forensic.org/, zuletzt aufgerufen am 11.04.2014.

[Fox09] D. Fox: Sicheres Löschen von Daten auf Festplatten. In: Datenschutz und Datensicherheit (2009) 2, S. 110 – 113.

[Gutm96] P. Gutmann: Secure Deletion of Data from Magnetic and Solid-State Memory. Auckland: 1996 Sixth USENIX Security Symposium Proceedings, S. 77 - 90.

[Hoog12] A. Hoog: Android Forensik Datenrecherche, Analyse und mobile Sicherheit bei Android. München: Franzis Verlag GmbH, 2012.

[Ilho12] S. Ilhoon: Implementing Secure File Deletion in NAND-based Block Devices with Internal Buffers. IEEE 2012, Transactions on Consumer Electronics, Vol. 58, No. 4, S. 1219 - 1224.

[King14] Kingston: Over-Provisioning leicht verständlich gemacht. URL: http://www.kingston.com/de/ssd/overprovisioning, zuletzt aufgerufen am: 12.04.2014.

[KJDN08] S. Kyoungmoon; C. Jongmoo; L. Donghee; S. Noh: Models and Design of an Adaptive Hybrid Scheme for Secure Deletion of Data in Consumer Electronics. 2008 IEEE, Transactions on Consumer Electronics, Vol. 54, No. 1 S. 100 - 104.

[MaCB07] A. Mathur; M. Cao; S. Bhattachary: The new ext4 Filesystem: current status and future plans. Ottawa, Ontario Canada 2007: IBM, Linux Symposium, Volume Two, S. 21 - 34.

[MiCM10] R. Micheloni; L. Crippa; A. Marelli: Inside NAND Flash Memories. Dordrecht Heidelberg London New York: Springer, 2010.

[Oste13] S. Ostermaier: Smartphone-Verbreitung in Deutschland bei 39,8%, URL: http://stadt-bremerhaven.de/smartphone-verbreitung-in-deutschland-bei-398/, zuletzt aufgerufen am: 23.11.2013.

[Press12] Pressebox: Trendstudie reCommerce: Pro Haushalt schlummern bis zu 3.000 Bücher in den Regalen. URL: http://www.pressebox.de/inaktiv/rebuy-recommerce-gmbh/Trendstudie-reCommerce-Pro-Haushalt-schlummern-bis-zu-3000-Buecher-in-den-Regalen/boxid/556745, zuletzt aufgerufen am: 12.04.2014.

[ReBC13] J. Reardon; D. Basin; S. Capkun: SoK: Secure Data Deletion. Zürich: 2013 IEEE, Symposium on Security and Privacy, S. 301 - 315.

[Sams13] Samsung: NAND-Grundlagen, URL: http://www.samsung.com/at/business-images/resource/white-paper/2013/11/samsung_nand_basics_whitepaper-0.pdf, zuletzt aufgerufen am 17.06.2014

[Subh09] S. Subha: An Algorithm for Secure Deletion in Flash Memories. CA, USA: 2009 IEEE, Sixth International Conference on Information Technology: New Generations, S. 1705 – 1706.

[Wilt13] Knoll Ontrack Blog: How to securely erase data from SSDs: 4 questions answered. URL: http://blog.krollontrack.co.uk/top-tips/securely-erase-ssds-4-questions-answered/, zuletzt aufgerufen am: 12.04.2014.