Technische Berichte in Digitaler Forensik

Herausgegeben vom Lehrstuhl für Informatik 1 der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Kooperation mit dem Masterstudiengang Digitale Forensik (Hochschule Albstadt-Sigmaringen, FAU, Goethe-Universität Frankfurt am Main)

Google Drive Android App

Version 5.0.36

com.google.android.apps.docs

Christian Strebe

02.03.2016

Technischer Bericht Nr. 5

Zusammenfassung

Dieses Dokument beschreibt die Analyse der Android App Google Drive. Diese App speichert Metadaten in einer Datenbank auf dem Smartphone ab. Mit diesen Informationen ist es möglich die zugehörigen, teilweise verschlüsselten Dateien sicherzustellen. Daher wird die Struktur der enthaltenen Datenbank analysiert.

Entstanden im Rahmen des Moduls Browser- und Anwendungsforensik des Studiengangs Digitale Forensik im Wintersemester 2015/2016 unter der Anleitung von Felix Freiling, Holger Morgenstern und Michael Gruhn.

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit sollen Spuren der Android App Google Drive[1] untersucht werden. Die App besitzt den Paketnamen: “com.google.android.apps.docs”. Im Rahmen der Untersuchung sollen die zur App gehörenden Dateien bzw. Datenbanken gesucht und analysiert werden. Dafür sind verschiedene Szenarien wie die Erstellung, Editierung und Betrachtung von Dokumenten vorgesehen. Damit liegt der Fokus auf den gespeicherten Dateien und der Möglichkeit Metadaten zu erhalten bzw. die Dateien wiederherzustellen.
Im Abschnitt 2 ab Seite 2 wird zunächst die verwendete Hard- und Software beschrieben. Als Grundlage wurde eine virtuelle Umgebung auf Basis von Google “Android SDK Tools” genutzt. Somit ist die Möglichkeit gegeben in einem Emulator Snapshots anzulegen, die dann mit Tools des qemu Projektes und The Sleuth Kit analysiert werden können.
Die Vorgehensweise wird im Abschnitt 3 ab Seite 4 beschrieben. Zunächst wird ein neues Nutzerkonto als Ausgangspunkt angelegt. Danach werden verschiedenen Szenarien umgesetzt.
Die so entstandenen Snapshots, werden anschließend im Abschnitt 4 untersucht. Die relevanten Spuren werden identifiziert und analysiert. Weiterhin wird auf Seite 20 gezeigt, wie eine verschlüsselte Datei mit den gewonnenen Informationen entschlüsselt werden kann.
Die wesentlichen Ergebnisse werden im Abschnitt 5 ab Seite 21 zusammengefasst.

2 Untersuchungsumgebung

Nachfolgend wird die Untersuchungsumgebung dargestellt. Als Host wurde eine Workstation mit Fedora 19 x86_64 gewählt. Die aktuelle Versionen der android-sdk-tools[2] sowie tsk installiert. Als virtuelle Maschine wurde Android 5.0 gewählt, da es zum Zeitpunkt der Hausarbeit mit 16.9%[3] recht verbreitet ist und zu erwarten ist, dass sich die Erkenntnisse auch auf die Andoid Version 5.1 übertragen lassen, welche einen ähnlichen Verbreitungsgrad hat. Vermutlich lassen sich die Erkenntnisse auch auf die Android Version 6.0 übertragen. Weiterhin wurde eine x86 Architektur gewählt, damit die Möglichkeit der CPU zur Beschleunigung der Virtualisierung genutzt werden kann. Wesentlich war die Erweiterung des internen Speichers und das Hinzufügen einer virtuellen SD-Karte, da sonst nicht genug Speicherplatz zur Verfügung steht, um alle Schritte der Untersuchung durchführen zu können.

1 https://de.m.wikipedia.org/wiki/Google_Drive abgerufen am 9.1.2016
https://en.m.wikipedia.org/wiki/Google_mobile_services abgerufen am 9.1.2016
2 http://developer.android.com/tools/devices/emulator.html abgerufen am 10.12.2015
http://www.flinkd.org/2015/02/installing-google-play-on-the-android-emulator-api-21-lollipop/ abgerufen am 10.12.2015
3 http://developer.android.com/about/dashboards/index.html abgerufen am 9.1.2016

2.1 Host

CPU: Intel(R) Xeon(R) CPU E31230 @ 3.20GHz
OS: Fedora release 19 (Schrödinger's Cat)
android-sdk-tools: 24.4.1 (Update am 2.1.2016)
tsk: 4.2.0
Remotdesktop:XSpice

2.2 VM

AVD-Name:	df-test
Device:	Nexus 7 (2012)
Target:	Google APIs (Google Inc.) - API Level 21
CPU/ABI:	Google APIs Intel Atom (x86)
Keyboard:	Hardware Keyboard present
Skin:	Skin with dynamic hardware controls
Front Camera:	Emulated
Back Camera:	None
Memory Options:	Ram: 1024 VM Heap: 32
Internal Storage:	2000 MiB
SD Card:	1000 MiB
Emulation Options:	Snapshot
Software:	Google Play: 5.0.36 Android Version: 5.0.2 API 21 Kernel Version: 3.4.67 Build number: sdk_google_phone_x86_eng 5.0.2 LSY6602473544 test-keys

3 Vorgehensweise

Um die Untersuchung auf einem Emulator entsprechend durchführen zu können muss dieser mit der Google Play App ausgestattet werden. Anschließend kann die eigentliche Untersuchung beginnen. Diese beiden Schritte sind nun nachfolgend näher beschrieben. Allgemein werden nachfolgend mehrere Snapshots erstellt. Dies kann mittels der Emulator Console durchgeführt werden.
Dazu wird mittels Telnet auf den Consolen Port des Emulators verbunden. In diesem Fall ist das Port 5554. Anschließen kann mit avd snaphost save das Speichern der Snapshots gestartet werden.

3.1 Vorgehensweise zur Vorbereitung des Emulators

Für die Installation der Google Play App auf dem ausgewählten Emulator kann die Anleitung “Installing Google Play on the Android Emulator (API 21 - Lollipop)” genutzt werden. Diese wurde von “pyoor” auf “Flinkd!” am 11.02.2015 bereitgestellt. Zwar ist die Anleitung für die Android Version 5.0.1 geschrieben worden, kann jedoch für die Version 5.0.2 genutzt werden. Da bereits eine Virtuelle Maschine erstellt worden ist, kann der Abschnitt “Create the AVD” übersprungen werden und dann wie nachfolgend nochmals beschrieben, verfahren werden.

3.1.1 google-drive-init0

Zunächst wird ein Snapshot erstellt, um bei Bedarf rückspringen zu können. Für die eigentliche Untersuchung spielt dieser Snapshot keine Rolle. Dieser wird “google-drive-init0” genannt.

3.1.2 google-drive-init1

Anschließend wird die heruntergeladen Version des Google Apps Paketes verifiziert und die entsprechenden Dateien entpackt.

```
md5sum gapps-lp-20141109-signed.zip
367ce76d6b7772c92810720b8b8b0c931e  gapps-lp-20141109-signed.zip
unzip -j gapps-lp-20141109-signed.zip \
  system/priv-app/GoogleServicesFramework/GoogleServicesFramework.apk
unzip -j gapps-lp-20141109-signed.zip \
  system/priv-app/GoogleLoginService/GoogleLoginService.apk
unzip -j gapps-lp-20141109-signed.zip \
  system/priv-app/Phonesky/Phonesky.apk
unzip -j gapps-lp-20141109-signed.zip \
  system/priv-app/GmsCore/GmsCore.apk
```

Nun werden die entsprechenden Dateien mittels adb auf die virtuelle Maschine hochgeladen. Und der Emulator neu gestartet. Das “Optimieren” der App kann dann ein wenig Zeit in Anspruch nehmen

```
./adb remount
./adb push GmsCore.apk /system/priv-app/
./adb push GoogleServicesFramework.apk /system/priv-app/
./adb push GoogleLoginService.apk /system/priv-app/
./adb push Phonesky.apk /system/priv-app/
./adb shell stop
```

```
./adb shell start
```

Abschließend wird ein Snapshot “google-drive-init1” erstellt, um bei Bedarf zurück springen zu können. Für die eigentliche Untersuchung spielt dieser Snapshot keine Rolle

Die oben dargestellten Schritte habe ich in dem foglenden Shell-Skript “./prepare-android-emulator” zusammengefasst:

```
$ cat prepare-android-emulator
#!/bin/bash

echo "avd snapshot save google-drive-init0" | nc 127.0.0.1 5554

echo "checking md5sum of gapps"
md5sum gapps-lp-20141109-signed.zip
#367ce76d6b7772c92810720b8b0c931e  gapps-lp-20141109-signed.zip

echo "unzipping files"
GAPPS=download/gapps-lp-20141109-signed.zip
ADB=/home/uipo/android-sdk-linux/platform-tools/adb

mkdir tmp
unzip -u -j $GAPPS \
  system/priv-app/GoogleServicesFramework/GoogleServicesFramework.apk -d ./tmp
unzip -u -j $GAPPS \
  system/priv-app/GoogleLoginService/GoogleLoginService.apk -d ./tmp
unzip -u -j $GAPPS \
  system/priv-app/Phonesky/Phonesky.apk -d ./tmp
unzip -u -j $GAPPS \
  system/priv-app/GmsCore/GmsCore.apk -d ./tmp

$ADB remount
$ADB push ./tmp/GmsCore.apk /system/priv-app/
$ADB push ./tmp/GoogleServicesFramework.apk /system/priv-app/
$ADB push ./tmp/GoogleLoginService.apk /system/priv-app/
$ADB push ./tmp/Phonesky.apk /system/priv-app/
$ADB shell stop
$ADB shell start

echo "avd snapshot save google-drive-init1" | nc 127.0.0.1 5554
```

3.2 Vorgehensweise der Untersuchung

Die Untersuchung selbst wird in mehrere Abschnitte unterteilt, um die relevanten Spuren identifizieren und im Idealfall gleich analysieren zu können. Zunächst wird ein initialer Snapshot von der Installation der eigentlichen App erstellt. Danach werden für die jeweiligen Testabschnitte die jeweiligen Snapshots erstellt.

3.2.1 google-drive-install

Nachdem die Google Play App auf dem Emulator installiert ist, kann der Google Account angemeldet und die eigentliche App Google Drive installiert werden. Danach wird ein Snapshot “google-drive-install” erstellt. Dieser dient als Referenz und initialer Ausgangszustand.

3.2.2 google-drive-test1

Als erstes werden über die Weboberfläche “drive.google.com” ein Textdokument “test1-text1” mit dem Inhalt “The quick brown fox jumps over the lazy dog. 1234567890 üäö !”§$%&/()=?+*#’-_^” erstellt.
Zusätzlich wird ein Tabellendokument “test1-tabelle1” erstellt. Der Inhalt wird in der Zelle A1 “Test1” und in Zelle B1 “Tabelle1” sein.
Außerdem soll eine Präsentation “test1-präsentation1” mit einer einzigen Folie, auf der “Test1” steht, erstellt werden.
Weiterhin wird ein Ordner “test1-ordner1” erstellt. In diesem wird ein weiteres Textdokument “test1-text2” mit dem Inhalt “Test text2” erstellt.
Nun sollen über die Weboberfläche die nachfolgend aufgelisteten Dateien zusätzlich hochgeladen werden. Damit soll untersucht werden, welche Spuren auftreten, wenn Google Drive als reines Storagesystem genutzt wird.

	Size	**md5sum**
Patern_test.jpg	36623	77d2a6bf840622331df62963174df72d
test1.bin	10240	ea3fbf3b9ad017558612428882bc0178
test1.txt	18	30d26580320f588c4c15e711569f8420
kapselhalter.stl	17884	6ee33af22cb0896cf136c5c807e0fb62
test1.zip	10567	f23485097405cee56a392ee451874a6b
test2.bin (Inhalt von test1.zip)	10240	539677773eac4958713c74aa90e40fcf
test2.txt (Inhalt von test1.zip)	13	9bcedaf428322c6c282b0806f890b2bd
test1.pdf	73012	b47ff84b7ed0d240076cba3f02ca6cad

Jetzt wird die App das erste mal geöffnet, die Einführung angesehen, die Mitteilungen der App bestätigt und der Bereich “Meine Ablage” angesehen. Anschließend wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test1” erstellt.

3.2.3 google-drive-test2

Nun soll wieder die App geöffnet werden und der Ordner “test1-ordner1” wird angesehen.
Anschließend wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test2” erstellt.

3.2.4 google-drive-test3

Diesmal sollen alle Dateien Offline verfügbar gemacht werden. Zunächst muss dafür die Einstellung “Dateien nur über WLAN übertragen” deaktiviert werden, da der Emulator keine WLAN-Verbindung

besitzt. Der Hinweis auf die Datennutzung in diesem Zusammenhang wird bestätigt. Nachdem die Dateien offline verfügbar sind, wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test3” erstellt.

Um diesen Test durchführen zu können, wurde bei der Erstellung der VM der interne Speicher des Android Emulators auf 2000 MiB vergrößert. Die Standardeinstellung reicht nicht aus, da sonst Downloadfehler auftreten.

3.2.5 google-drive-test4

Nun sollen alle Dateien betrachtet bzw. heruntergeladen werden. Anschließend wird die App wieder geschlossen. Dabei können die Dateien test1.bin und kapselhalter.stl nur im “WebMode” heruntergeladen werden. Die beiden Google-Drive eigenen Text Dateien test1-text1 und test1-text2 werden nicht richtig dargestellt. Die Google-Drive Präsenation test1-präsentation1 wird im WebBrowser dargestellt. Anschließend wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test4” erstellt.
Um diesen Test durchführen zu können, wurde bei der Erstellung der VM eine virtuelle SD-Karte mit 1000 MiB dem Emulator hinzugefügt, da sonst die Dateien test1.bin und kapselhalter.stl nicht heruntergeladen werden können.

3.2.6 google-drive-test5

Nun soll in die Textdatei, Präsentation und Tabelle der Text “Test5” eingefügt werden. Dazu werden die Apps Google Docs, Google Sheets und Google Slides installiert. Anschließend werden alle Apps wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test5” erstellt.

3.2.7 google-drive-test6

Nun soll geprüft werden, welche Spuren durch Markieren, Verschieben, Link freigeben, Entfernen, Umbenennen und Person hinzufügen entstehen. Die Änderungen werden wie in der nachfolgenden Tabelle dargestellt durchgeführt.
Nachdem die Änderungen vorgenommen worden sind, wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test6” erstellt.

Datei	**Änderung**
Patern_test.jpg	Markieren
test1.bin	Verschieben in test1-ordner1
test1.txt	Link freigeben
kapselhalter.stl	Umbennen in test6.stl
test1.zip	Entfernen
test1.pdf	Person hinzufügen strebech@albsig.de

3.2.8 google-drive-test7

In diesem Test sollen Dateioperationen wie Datei senden, Drucken und Datei bzw. Ordner erstellen vorgenommen werden. Die Operationen sind in der nachfolgenden Tabelle aufgeführt.
Nachdem die Änderungen vorgenommen worden sind, wird die App wieder geschlossen.
Danach wird ein Snapshot mit dem Namen “google-drive-test6” erstellt.

Datei	Änderung
Patern_test1.jpg	Datei senden (simuliert mittels standard MMS Dienst)
test1.pdf	Drucken (in PDF “test1-print.pdf” speichern in Downloads)
StudienführerDigitaleForensik.pdf	Hochladen (heruntergeladen von alb-sig)
test7-text1	Erstellen Textdokument mit Inhalt “Test7”

4 Analyse

Nachfolgend werden die vorbereiteten Snapshots genutzt um die eigentliche Analyse durchzuführen.

4.1 Aufbereitung der Snapshots

Um die Snapshots mit dem Sleuthkit analysieren zu können, müssen die Snapshots aus dem qcow2 Format in das raw Format jeweils extrahiert werden. Leider lässt sich das Tool qemu-img dafür nicht direkt nutzen. Zwar kann das tool die Snapshots des Emulators anzeigen, jedoch werden nur leere raw Images extrahiert. Dies liegt daran, dass der Android Emulator qemu benutzt und den qemu eigenen “savevm” Mechanismus nutzt um Snapshots zu erstellen. Dabei werden zusätzliche Daten, die über die eigentlichen Festplatteninformationen hinausgehen, mit abgespeichert. Dies kann dann von qemu-img nicht mehr extrahiert werden. Daher musste ein anderer Weg gefunden werden[4].
Die Snapshots können unter Nutzung des Emulators selbst extrahiert werden. Dazu habe ich das unten aufgeführte Skript “analyse” erstellt. Zunächst wird der Emulator angehalten. Da bereits alle gewünschten Änderungen vorhanden sind, soll nichts mehr verändert werden. Anschließend werden die Snapshots der Reihe nach geladen und die Datei “userdate-qemu.img” kopiert. Diese Datei enthält das jeweilige Festplattenabbild im raw Format und kann somit mit Tools von sleuthkit untersucht werden. Danach werden die Festplattenabbilder mittels idifference2.py mit ihrem jeweiligen Vorgänger verglichen und die idiff Dateien für eine spätere Analyse abgespeichert.

4https://juliofaracco.wordpress.com/2015/02/19/an-introduction-to-qcow2-image-format/ abgerufen am 10.12.2015
https://people.gnome.org/~markmc/qcow-image-format.html abgerufen am 10.12.2015

```
#!/bin/bash
copyimages() {
  echo "stoping avd"
  echo "avd stop" | nc 127.0.0.1 5554
  for i in test1 test2 test3 test4 test5 test6 test7 install init0 init1; do
    echo "########################################"
    echo "Loading image google-drive-$i"
    echo "avd snapshot load google-drive-$i" | nc 127.0.0.1 5554
    echo "copy /home/uipo/.android/avd/df-test.avd/userdata-qemu.img ./userdata-qemu-google-drive-$i.img"
    cp /home/uipo/.android/avd/df-test.avd/userdata-qemu.img ./userdata-qemu-google-drive-$i.img
    ls -l ./userdata-qemu*img
  done
}

# Diese Funktion erstellt die diff Files aus den jeweiligen Festplattenimages
# Das erste Argument ist der Name des ersten Tests
# Das zweite Arguement ist der Name des zweiten Tests
creatediff() {
  FILEPREFIX="snapshots/userdata-qemu-"
  echo "erstelle diff $2.diff fuer $1 zu $2"
  python3 ~/dfxml/python/idifference2.py $FILEPREFIX$1.img $FILEPREFIX$2.img > idiffs/$2.idiff
}

declare -a snapshots=("init0" "init1" "install" "test1" "test2" "test3" "test4" "test5" "test6" "test7")

createAllDiffs() {
  SNAPPREF="google-drive-"
  arraylength=${#snapshots[@]}
  for (( i=2; i<${arraylength}+1; i++ ));
  do
    creatediff $SNAPPREF${snapshots[$i-2]} $SNAPPREF${snapshots[$i-1]}
  done
}

copyimages
createAllDiffs
```

4.2 Dateien von Interesse

Um die Dateien von Interesse besser identifizieren zu können wird das Skript “spuren.tcl” aus Modul 10 angepasst[5]. Die Änderung am Skript betrifft lediglich die Namen der Aktionen, für die die me und ce Files erstellt werden. Unverändert bleibt der gesamte Teil, der aus den idiff Files die Daten extrahiert und in einer Sqlite Datenbank preparedevidences.db speichert. Das Skript selbst ist im Anhang dargestellt.

Durch den typischen Aufbau der Android Apps mit einem eigenen Verzeichnis in /data war zu erwarten, das wesentliche Dateien unter /data/com.google.android.apps.docs zu finden sind. Dies hat sich auch direkt nach der Installtion von Google Drive bestätigt. Dies ist unten aufgeführt.

```
$ fls -r -D snapshots/userdata-qemu-google-drive-install.img | grep com.google.android.apps.docs
+ d/d 14187:    com.google.android.apps.docs-1
++++ r/d * 99203(realloc):      com.google.android.apps.docs1760057093.apk
+ d/d 99205:    com.google.android.apps.docs
$ fls -r snapshots/userdata-qemu-google-drive-install.img 99205
l/l 99206:      lib
d/d 99203:      cache
```

5https://www.sqlite.org/tempfiles.html abgerufen am 9.1.2016
https://www.sqlite.org/cli.html abgerufen am 9.1.2016

```
+ d/d 99218:    diskCache
+ d/d 99219:    docs_glide
++ d/d 99220:   temp
++ d/d 99221:   data
d/d 98911:      shared_prefs
+ r/r 99222:    com.google.android.gms.analytics.prefs.xml
+ r/r 99207:    google_apps_features_per_account_prefs.xml
+ r/r 99227:    accountFlagsdigitaleforensik01@gmail.com.xml
+ r/r 99229:    com.google.android.apps.docs_preferences.xml
+ r/r 99216:    flags-application.xml
+ r/r 99228:    flags-account-digitaleforensik01@gmail.com.xml
+ r/r * 99224(realloc): accountFlagsdigitaleforensik01@gmail.com.xml.bak
d/d 99213:      databases
+ r/r 99214:    DocList.db
+ r/r 99215:    DocList.db-wal
+ r/r 99217:    DocList.db-shm
+ r/r 99212:    google_analytics_v4.db
+ r/r 99223:    google_analytics_v4.db-journal
d/d 99225:      files
+ r/r 99226:    gaClientId
```

Bei Betrachtung der Ordner und Dateien direkt nach der Installation der App fällt auf, dass ein Ordner “databases” existiert, in dem eine Datei mit dem Namen DocList.db angelegt wurde. Weiterhin existiert ein Ordner “files”. Der manuelle Vergleich mit dem Zustand der Verzeichnisstruktur nach Test 7 legt direkt nahe, dass hier weiter untersucht werden kann.

```
$ fls -r snapshots/userdata-qemu-google-drive-test7.img 99205
l/l 99206:      lib
d/d 99203:      cache
+ d/d 99218:    diskCache
++ d/d 99230:   fetching
+++ d/d 99248:  accountCache_1
++++ r/r 99255: previewImage-1600-1600-
d_downloaded_image_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjoxMA==_1452050509950
++++ r/r 99254: previewImage-1600-1600-
d_downloaded_image_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjo4_1452207867373
++++ r/r 99256: documentContent_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjoxMQ==_1452050349010
++++ r/r 99257: documentContent_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjoxMg==_1452050256616
++++ r/r 99366: previewImage-1600-1600-
d_downloaded_image_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjo2_1452207891013
++++ r/r 99352: previewImage-1600-1600-
d_downloaded_image_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjoz_1452208026646
++++ r/r 99367: documentContent_ZGlnaXRhbGVmb3JlbnNpazAxQGdtYWlsLmNvbS1kYjo5_1452050614209
+ d/d 99219:    docs_glide
++ d/d 99220:   temp
+++ r/r * 100296(realloc):     tmp-1147471019
++ d/d 99221:   data
+++ r/r 99241:  2db77f4102a49476b91aa93cac59534e1a433f0665fa122524df415a6b8d8fd7
+++ r/r 99242:  cc2e2695d99d3da32ae1b8e64bc3595fd8abdbfdcf70c11fa4c2b2ddbf5a4445
+++ r/r 99243:  2afb38c11268ae1d08291cedee2d89145f6c0932e76b0eaa8c5edfc6b1b99951
+++ r/r 99244:  4eec12635c52282709a64b5276e655a34ae6afeec2f8f0417132f2a211b64211
+++ r/r 99245:  e6e62db5b6f4f35d8b7c41bf115a1e611e46b822bf77903ba8f58e78268ef286
+++ r/r 99246:  ce3894acddcb3b807deb5e809655abf077eaea3f36049ed541b6825a914e75b0
+++ r/r 99234:  dd090183e13c56685fcf6141546473420574cdd9af29506574cebca7474a8e13
+++ r/r 99227:  9420ac971ae4540c1f2a31f4f8ee11961058c891fc47d2c22fc3512d8d309dac
+++ r/r 99229:  5d575e816d4979f01772a44045ec13038953ee9b4573cc1f9388cac555d30912
+++ r/r 99251:  92634ba83e04aeee4a20bfc9f923e86987449d9b8352531b12238c92defab8a0
+++ r/r 99252:  7cd5039302647376094855 5da33a92e2fb9566839bbe6d049092504fad9f8d85
+++ r/r 100045: 2b01e35e3c4e72e5e9f944ae6c7ef7393020e909467996150eea760447f44ae4
+++ r/r 99207:  f5778c0978ac9cfb1b0f0ae7797638c568f7520d1dd25212f32d07376929a424
+++ r/r 100296: f9c963cfe0671563bf4b978458389dabbce195c31b3743817b462f21337af9b3
+ d/d 100287:   filecache2
+ d/d * 99361(realloc): projector
```

+ d/r * 99362(realloc): projector-tmp
+ r/r * 99377(realloc): temp1690470044temp
+ r/r * 99366(realloc): temp1558684014temp
+ r/r * 99367(realloc): temp-1848650905temp
+ r/r * 99368(realloc): temp-1742547174temp
d/d 98911: shared_prefs
+ r/r 99088: com.google.android.gms.analytics.prefs.xml
+ r/r 99222: google_apps_features_per_account_prefs.xml
+ r/r 99232: accountFlagsdigitaleforensik01@gmail.com.xml
+ r/r 99363: com.google.android.apps.docs_preferences.xml
+ r/r 100054: flags-application.xml
+ r/r 99216: flags-account-digitaleforensik01@gmail.com.xml
+ r/r 99201: GoogleDriveSharedPreferences.xml
+ r/r 100289: HelpCard.xml
+ r/r 99239: CrossAppPromoPreferences.xml
+ r/r 99247: WarmWelcomePersister.xml
+ r/r 99253: DocumentContentSpecCrypter.GuidKeyStore.V2.xml
+ r/r 99259: WebViewChromiumPrefs.xml
+ r/r 99267: webview.xml
+ r/r 100288: upload-history.xml
+ r/r * 99207(realloc): HelpCard.xml.bak
d/d 99213: databases
+ r/r 99214: DocList.db
+ r/r 99215: DocList.db-wal
+ r/r 99217: DocList.db-shm
+ r/r 99212: google_analytics_v4.db
+ r/r 99223: google_analytics_v4.db-journal
d/d 99225: files
+ r/r 99226: gaClientId
+ d/d 99233: fileinternal
++ d/d 99324: 5804c462cf91e9b2070236548fd6eced
+++ r/r 99335: test1.txt
++ d/d 99300: 71af733d8bda375e663a064752981227
+++ r/r 99360: test1.pdf
++ d/d 99228: aa1396239ac2f628c829f6b23c5ff6bb
+++ r/r 100264: Patern_test.jpg
++ d/d 100278: 63624ae569e764ebad5e32bfd56e3699
+++ r/r 100290: StudienführerDigitaleForensik.pdf
d/d 99231: app_apps_1452255030692769769084
+ r/r 99235: app2092076871.cache
d/d 99237: app_apps_14522550309141242425506
+ r/r 99238: app1760057093.cache
d/d 99236: app_apps_145225503111181360591988
+ r/r 99240: app991025956.cache
d/d 99260: app_webview
+ d/d 99261: paks
+ r/r 99262: Web Data
+ r/r 99263: Web Data-journal
+ r/r 99265: Cookies
+ r/r 99266: Cookies-journal
+ d/d 99268: Cache
++ r/r 99269: index
++ d/d 99270: index-dir
+++ r/r * 99249(realloc): temp-index
+++ r/r 99249: the-real-index
++ r/r 99272: 6824eac908b6f7bb_0
++ r/r 99273: f13d26ff9186d8c7_0
++ r/r 99274: 1f353e91b59b50d8_0
++ r/r 99275: 6ea78edf880dee9d_0
++ r/r 99276: f394e3f4b03a1c95_0
++ r/r 99277: 56ff954fd337e90b_0
++ r/r 99278: f29dbc1312a44bc8_0
++ r/r 99279: 97d9eb15e68c200c_0
++ r/r 99280: 62223d06dab02811_0
++ r/r 99281: d8f60bb08f1a517e_0
++ r/r 99282: 0f88c08ea611aef3_0

```
++ r/r 99283:   170fcbe78cda12c6_0
++ r/r 99284:   daaa961a916c353f_0
++ r/r 99285:   490c4ebc564819ea_0
++ r/r 99286:   e1d283537aa20118_0
++ r/r 99287:   a62ffb291a2e5fba_0
++ r/r 99288:   90a28d2c89650458_0
++ r/r 99289:   a9bed136b22c8623_0
++ r/r 99290:   6493e4e1f1d61cb5_0
++ r/r 99291:   d0f156c41f45309d_0
++ r/r 99250:   eae87d66710dfa8d_0
++ r/r 99293:   6c36660ecfa2606a_0
++ r/r 99294:   d677e96425f9b0ec_0
++ r/r 99349:   8f7301f4122ccf09_0
++ r/r 99350:   70d49d00d97a7e19_0
++ r/r 99351:   b856926f3c5ee33d_0
++ r/r 99302:   697dd58e186af34b_0
++ r/r 99353:   b4f11aff39f4e01c_0
++ r/r 99354:   2dd000ef4bbc48c0_0
++ r/r 99355:   966da3876ebd8a43_0
++ r/r 99356:   9de81e9c7f60082b_0
++ r/r 99357:   e071b60469310f06_0
++ r/r 99358:   304389c35dda0ea7_0
++ r/r * 99359(realloc):        6f797497e63f9100_0
d/d 99264:      app_appcache
d/d * 100287(realloc):  app_filecache2
```

Weiterhin kann man der Datenbank preparedevidences.db die Dateien mit häufigem Zugriff entnehmen. Diese sind nachfolgend dargestellt. Auch hier ist die Datei DocList.db enthalten. Diese Datei selbst wurde fünf mal verändert, aber die Files shm, wal wurden bei jedem Snapshot verändert. Dabei handelt es sich um zwei Dateien die im Write Ahead Log Modus erstellt werden, wenn die Datenbank geöffnet wird und wieder gelöscht werden, wenn die Datenbank geschlossen wird. Damit ist bestätigt, dass die Datei DocList.db bei jedem öffnen der App selbst auch geöffnet wird und somit eine Datei von Interesse ist. Weiterhin sind Systemdateien und Dateien der App “com.google.android.gms”. Bei der App handelt es sich um die Google Play Dienste. Diese stellt auch für Google Drive wesentliche Funktionalitäten bereit. Ich möchte mich im Rahmen dieser Arbeit jedoch auf die App google Drive selbst beschränken.

```
sqlite> select pfad, stempeltyp, COUNT() from pe group by pfad, stempeltyp HAVING count() > 3 order by COUNT()
DESC;
data/com.google.android.apps.docs/databases/DocList.db-shm|c|8
data/com.google.android.apps.docs/databases/DocList.db-shm|m|8
data/com.google.android.apps.docs/databases/DocList.db-wal|c|8
data/com.google.android.apps.docs/databases/DocList.db-wal|m|8
system/recent_tasks|c|8
system/recent_tasks|m|8
system/sync|c|8
system/sync|m|8
system/sync/status.bin.bak|a|8
system/sync/status.bin.bak|c|8
system/sync/status.bin.bak|cr|8
system/sync/status.bin.bak|m|8
data/com.google.android.apps.docs/shared_prefs|c|7
data/com.google.android.apps.docs/shared_prefs|m|7
system/sync/pending.xml|c|7
system/sync/pending.xml|m|7
system/users/0/package-restrictions-backup.xml|a|7
system/users/0/package-restrictions-backup.xml|c|7
system/users/0/package-restrictions-backup.xml|cr|7
system/users/0/package-restrictions-backup.xml|m|7
data/com.google.android.apps.docs/cache/docs_glide/data|c|6
```

data/com.google.android.apps.docs/cache/docs_glide/data|m|6
data/com.google.android.apps.docs/cache/docs_glide/temp|c|6
data/com.google.android.apps.docs/cache/docs_glide/temp|m|6
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml.bak|a|6
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml.bak|c|6
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml.bak|cr|6
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml.bak|m|6
data/com.google.android.apps.docs/shared_prefs/accountFlagsdigitaleforensik01@gmail.com.xml|a|6
data/com.google.android.apps.docs/shared_prefs/accountFlagsdigitaleforensik01@gmail.com.xml|c|6
data/com.google.android.apps.docs/shared_prefs/accountFlagsdigitaleforensik01@gmail.com.xml|cr|6
data/com.google.android.apps.docs/shared_prefs/accountFlagsdigitaleforensik01@gmail.com.xml|m|6
data/com.google.android.gms/app_dg_cache/5FFB35BE606DEFA7F87F0AB2B1A2559DA140EC50/t|c|6
data/com.google.android.gms/app_dg_cache/5FFB35BE606DEFA7F87F0AB2B1A2559DA140EC50/t|m|6
data/com.google.android.gms/databases/dg.db|c|6
data/com.google.android.gms/databases/dg.db|m|6
data/com.google.android.gms/databases/dg.db-journal|c|6
data/com.google.android.gms/databases/dg.db-journal|m|6
data/com.google.android.gms/databases/ns.db|c|6
data/com.google.android.gms/databases/ns.db|m|6
data/com.google.android.gms/databases/ns.db-journal|c|6
data/com.google.android.gms/databases/ns.db-journal|m|6
data/com.google.android.gms/shared_prefs|c|6
data/com.google.android.gms/shared_prefs|m|6
system|c|6
system|m|6
system/sync/status.bin|a|6
system/sync/status.bin|c|6
system/sync/status.bin|cr|6
system/sync/status.bin|m|6
system/users/0/package-restrictions-backup.xml|d|6
data/com.google.android.apps.docs/databases/DocList.db|c|5
data/com.google.android.apps.docs/databases/DocList.db|m|5
data/com.google.android.apps.docs/databases/google_analytics_v4.db|c|5
data/com.google.android.apps.docs/databases/google_analytics_v4.db|m|5
data/com.google.android.apps.docs/databases/google_analytics_v4.db-journal|c|5
data/com.google.android.apps.docs/databases/google_analytics_v4.db-journal|m|5
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml|a|5
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml|c|5
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml|cr|5
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml|m|5
data/com.google.android.gms/app_sslcache/www.googleapis.com.443|c|5
data/com.google.android.gms/app_sslcache/www.googleapis.com.443|m|5
data/com.google.android.gms/databases/google_analytics_v4.db|c|5
data/com.google.android.gms/databases/google_analytics_v4.db|m|5
data/com.google.android.gms/databases/google_analytics_v4.db-journal|c|5
data/com.google.android.gms/databases/google_analytics_v4.db-journal|m|5
system/dropbox|c|5
system/dropbox|m|5
system/sync/accounts.xml|a|5
system/sync/accounts.xml|c|5
system/sync/accounts.xml|cr|5
system/sync/accounts.xml|m|5
system/usagestats/0/daily|c|5
system/usagestats/0/daily|m|5
system/usagestats/0/monthly|c|5
system/usagestats/0/monthly|m|5
system/usagestats/0/monthly/1451520410557|a|5
system/usagestats/0/monthly/1451520410557|c|5
system/usagestats/0/monthly/1451520410557|cr|5
system/usagestats/0/monthly/1451520410557|m|5
system/usagestats/0/weekly|c|5
system/usagestats/0/weekly|m|5
system/usagestats/0/weekly/1452125210557|a|5
system/usagestats/0/weekly/1452125210557|c|5
system/usagestats/0/weekly/1452125210557|cr|5
system/usagestats/0/weekly/1452125210557|m|5

system/usagestats/0/yearly|c|5
system/usagestats/0/yearly|m|5
system/usagestats/0/yearly/1450656410557|a|5
system/usagestats/0/yearly/1450656410557|c|5
system/usagestats/0/yearly/1450656410557|cr|5
system/usagestats/0/yearly/1450656410557|m|5
system/users/0/accounts.db|c|5
system/users/0/accounts.db|m|5
system/users/0/accounts.db-journal|c|5
system/users/0/accounts.db-journal|m|5
data/com.android.providers.downloads/databases/downloads.db|c|4
data/com.android.providers.downloads/databases/downloads.db|m|4
data/com.android.providers.downloads/databases/downloads.db-journal|c|4
data/com.android.providers.downloads/databases/downloads.db-journal|m|4
data/com.google.android.apps.docs|c|4
data/com.google.android.apps.docs|m|4
data/com.google.android.apps.docs/cache|c|4
data/com.google.android.apps.docs/cache|m|4
data/com.google.android.apps.docs/cache/docs_glide/data/e6e62db5b6f4f35d8b7c41bf115a1e611e46b822bf77903ba8f58e78268ef286|c|4
data/com.google.android.apps.docs/cache/docs_glide/data/e6e62db5b6f4f35d8b7c41bf115a1e611e46b822bf77903ba8f58e78268ef286|m|4
data/com.google.android.apps.docs/shared_prefs/HelpCard.xml.bak|d|4
data/com.google.android.apps.docs/shared_prefs/com.google.android.apps.docs_preferences.xml|a|4
data/com.google.android.apps.docs/shared_prefs/com.google.android.apps.docs_preferences.xml|c|4
data/com.google.android.apps.docs/shared_prefs/com.google.android.apps.docs_preferences.xml|cr|4
data/com.google.android.apps.docs/shared_prefs/com.google.android.apps.docs_preferences.xml|m|4
data/com.google.android.gms/app_chimera/chimera-module-root/module-a3e4fba11e705727c59ff3116ef21fa4834b9f56/.optimization_in_progress|c|4
data/com.google.android.gms/app_chimera/chimera-module-root/module-a3e4fba11e705727c59ff3116ef21fa4834b9f56/.optimization_in_progress|m|4
data/com.google.android.gms/app_sslcache/android.clients.google.com.443|c|4
data/com.google.android.gms/app_sslcache/android.clients.google.com.443|m|4
data/com.google.android.gms/databases|c|4
data/com.google.android.gms/databases|m|4
data/com.google.android.gms/databases/context_feature_default.db|c|4
data/com.google.android.gms/databases/context_feature_default.db|m|4
data/com.google.android.gms/databases/context_feature_default.db-journal|c|4
data/com.google.android.gms/databases/context_feature_default.db-journal|m|4
data/com.google.android.gms/databases/fitness.db.digitaleforensik01_gmail.com-shm|a|4
data/com.google.android.gms/databases/fitness.db.digitaleforensik01_gmail.com-shm|c|4
data/com.google.android.gms/databases/fitness.db.digitaleforensik01_gmail.com-shm|cr|4
data/com.google.android.gms/databases/fitness.db.digitaleforensik01_gmail.com-shm|m|4
data/com.google.android.gms/databases/playlog.db|c|4
data/com.google.android.gms/databases/playlog.db|m|4
data/com.google.android.gms/databases/rmq.db|c|4
data/com.google.android.gms/databases/rmq.db|m|4
data/com.google.android.gms/databases/rmq.db-journal|c|4
data/com.google.android.gms/databases/rmq.db-journal|m|4
data/com.google.android.gms/files/service.connections|c|4
data/com.google.android.gms/files/service.connections|m|4
system/recent_images|c|4
system/recent_images|m|4
system/sync/status.bin.bak|d|4
system/usagestats/0/daily/1452211610557.bak|a|4
system/usagestats/0/daily/1452211610557.bak|c|4
system/usagestats/0/daily/1452211610557.bak|cr|4
system/usagestats/0/daily/1452211610557.bak|m|4
system/usagestats/0/monthly/1451520410557.bak|a|4
system/usagestats/0/monthly/1451520410557.bak|c|4
system/usagestats/0/monthly/1451520410557.bak|cr|4
system/usagestats/0/monthly/1451520410557.bak|m|4
system/usagestats/0/weekly/1452125210557.bak|a|4
system/usagestats/0/weekly/1452125210557.bak|c|4
system/usagestats/0/weekly/1452125210557.bak|cr|4
system/usagestats/0/weekly/1452125210557.bak|m|4

```
system/usagestats/0/yearly/1450656410557.bak|a|4
system/usagestats/0/yearly/1450656410557.bak|c|4
system/usagestats/0/yearly/1450656410557.bak|cr|4
system/usagestats/0/yearly/1450656410557.bak|m|4
sqlite>
```

Allerdings ist die Datei DocList.db, wie unten zu sehen, nicht unter den charakteristischen Spuren der App zu finden. Dies liegt aber daran, dass die Datei bei der Installation bereits angelegt wurde und somit auch in “install.me” zu finden ist. Weiterhin wird sie nicht immer verändert. Trotzdem werde ich mich in der folgenden Analyse auf diese Datei konzentrieren.

```
$ cat test.ce
data/com.google.android.apps.docs/shared_prefs  c
data/com.google.android.apps.docs/shared_prefs  m
system/sync/pending.xml c
system/sync/pending.xml m
$
```

4.3 Analyse der identifizierten Dateien

4.3.1 Analyse von DocList.db

Bei der Datei DocList.db handelt es sich um eine Sqlite3 Datenbank. Diese kann mit den üblichen zur Verfügung stehenden Tools geöffnet werden.
Die Datei beinhaltet 27 Tabellen.

```
$ sqlite3 spuren/google-drive-test7-DocList.db
SQLite version 3.8.3 2014-02-03 14:04:11
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
Account150                  Entry150
AccountMetadata150          EntryView
App150                      FileList150
AppMetadata150              GokartPartialFeed150
CachedSearch150             Manifest150
CachedSearchResult150       NotificationList150
Collection150               PartialFeed150
CollectionView              PendingOperation150
ContainsId150               SyncRequest150
Document150                 SyncRequestJournalEntry150
DocumentContent150          UniqueId150
DocumentRequestedToSyncView  UriToContent150
DocumentView                android_metadata
DocumentWithFontFamiliesView
```

Von den 22 Tabellen sind acht auch nach Test 7 komplett leer. Das sind CachedSearch150, CachedSearchResult150, AppMetadata150, NotificationList150, PendingOperation150, SyncRequest150, SyncRequestJournalEntry150, UniqueId150 und UriToContent150. Diese werden nicht weiter betrachtet.
Nach einer ersten Sichtung erscheinen Account150, Document150, DocumentContent150 und Entry150 besonders interessant. Auch von Bedeutung sind Collection150 und Containsid150 mit diesen läßt sich die aktuelle Ordnerstruktur ermitteln. Alle genannten Tabellen werden nachfolgend weiter betrachtet. Die zusätzlichen Views werden nicht weiter betrachtet, da sie nach erster Sichtung keine neuen Informationen enthalten.

4.3.1.1 Account150

Diese Tabelle enthält die Account Informationen. Im Rahmen dieses Test wurde lediglich der Account digitaleforensik01@gmail.com benutzt. Daher befindet sich hier nur ein Eintrag. Nachfolgend ist das Schema der Tabelle dargestellt.

```
sqlite> .schema account150
CREATE TABLE "Account150" ("Account_id" INTEGER PRIMARY KEY AUTOINCREMENT, "accountHolderName" TEXT NOT
NULL, "forceFullSync" INTEGER NOT NULL DEFAULT 1, "lastSyncTime" INTEGER NOT NULL DEFAULT 0,
"folderSyncClipTime" INTEGER, "documentSyncClipTime" INTEGER, "lastSyncChangeStamp" INTEGER NOT NULL
DEFAULT 0, "syncInProgress" INTEGER NOT NULL DEFAULT 0, "lastSyncSequenceNumber" INTEGER NOT NULL DEFAULT
0, "offlinePolicyEnabled" INTEGER, "lastOfflineEnabledState" INTEGER, "lastForceSyncLevel" INTEGER,
"gokartLastSyncSequenceNumber" TEXT, "features" TEXT);
CREATE UNIQUE INDEX "Account150_accountHolderName_ui" ON "Account150" (accountHolderName);
```

Die interessanten Spalten sind in der nachfolgenden Tabelle zusammengefasst.

Spalte	Beschreibung
Account_id	Primärer Schlüssel
accountHolderName	Name des Accounts - Unique Index
lastSyncTime	Zeit des letzten Syncs (Unix-Time in Millisekunden)
lastSyncChangeStamp	noch keine gesicherten Erkenntnisse
lastSyncSequenceNumber	noch keine gesicherten Erkenntniss

4.3.1.2 Entry150

In der Tabelle Entry150 befindet sich eine Vielzahl von wesentlichen Einträgen. An dieser Stelle sind alle wesentlichen Metadaten zu allen Dateien und Ordnern in Google Drive abgelegt. Das Schema der Tabelle ist nachfolgen dargestellt.

```
sqlite> .schema Entry150
CREATE TABLE "Entry150" ("Entry_id" INTEGER PRIMARY KEY AUTOINCREMENT, "title" TEXT NOT NULL, "owner" TEXT
NOT NULL, "creationTime" INTEGER NOT NULL, "lastModifiedTime" INTEGER NOT NULL, "lastModifierAccountAlias" TEXT
NOT NULL, "lastModifierAccountName" TEXT, "lastOpenedTime" INTEGER, "sharedWithMeTime" INTEGER,
"sharedWithMeAccountName" TEXT, "recencyReason" INTEGER DEFAULT 100, "recencyTime" INTEGER, "shared"
INTEGER NOT NULL DEFAULT 0, "modifiedByMeTime" INTEGER, "metadataEtag" TEXT, "resourceId" TEXT, "isLocalOnly"
INTEGER NOT NULL DEFAULT 0, "mimeType" TEXT, "externalAppEntryMimeType" TEXT, "kind" TEXT NOT NULL, "canEdit"
INTEGER NOT NULL, "starred" INTEGER NOT NULL, "trashed" INTEGER NOT NULL DEFAULT 0, "pinned" INTEGER NOT
NULL DEFAULT 0, "lastPinnedStateChangeTime" INTEGER NOT NULL DEFAULT 0, "lastOfflineContentUpdateTime"
INTEGER NOT NULL DEFAULT 0, "changeFeed" INTEGER NOT NULL, "placeHolder" INTEGER NOT NULL, "accountId"
INTEGER NOT NULL, "sequenceNumber" INTEGER NOT NULL DEFAULT 0, "thumbnailStatus" INTEGER NOT NULL
DEFAULT 0, "plusMediaAttribute" INTEGER DEFAULT 0, "downloadRestricted" INTEGER DEFAULT 0, "contentUriAvailable"
INTEGER DEFAULT 1, "serializedDriveId" TEXT, "invariantDriveId" TEXT, "gokartContentAvailability" INTEGER DEFAULT 0,
"folderColorRgb" TEXT, "gokartContentTransfers" INTEGER NOT NULL DEFAULT 0, "numericNormalizedTitle" TEXT,
"gokartIsSubscribed" INTEGER NOT NULL DEFAULT 1, "pinnedInGokart" INTEGER NOT NULL DEFAULT 0,
"gokartContentFresh" INTEGER NOT NULL DEFAULT 0, FOREIGN KEY("accountId") REFERENCES
"Account150"("Account_id") ON DELETE CASCADE);
CREATE INDEX "Entry150_accountId_i" ON "Entry150" ("accountId");
CREATE UNIQUE INDEX "Entry150_resourceId_accountId_ui" ON "Entry150" (resourceId,accountId);
CREATE INDEX "Entry150_sequenceNumber_i" ON "Entry150" ("sequenceNumber");
CREATE INDEX "Entry150_invariantDriveId_i" ON "Entry150" ("invariantDriveId");
sqlite>
```

Die Bedeutung in der einiger Einträge ist in der nachfolgenden Tabelle aufgelistet:

Spalte	**Beschreibung**
Entry_id	Primärer Schlüssel
title	Klartextname der Datei/des Ordners
owner	Besitzer der Datei
creationTime lastModifiedTime lastOpenedTime modifiedByMeTime	Zeitstempel für Erstellung, letzten Modifikationen und letztes Öffnen (Unix Time in Millisekunden)
lastModifierAccountAlias lastModifierAccountName	Account/Alias des letzten Bearbeiters
sharedWithMeTime sharedWithMeAccount	Noch keine gesicherten Erkenntnisse
recencyReason recencyTime	Noch keine gesicherten Erkenntnisse (Ist Reason gleich 2 stimmt der Zeitstempel mit der lastModifiedTime überein. Bei Reason gleich 3 sind creationTime, lastModifiedTime und recencyTime identisch. Für Reason gleich 1 ist noch kein Zusammenhang erkannt)
shared	1 für Dateien deren Link freigegeben wurde
metadataEtag	noch keine gesicherten Erkenntnisse
resourceId accountId	Unique Index
isLocalOnly	noch keine gesicherten Erkenntnisse (1 - für die hochgeladene Datei)
mimeType kind	Art der Datei
canEdit	noch keine gesicherten Erkenntnisse (für alle Dateien bisher 1)
starred	1 für Markierte Dateien
trashed	3 für gelöschte Dateien
pinned lastPinnedStateChangeTime	1 für alle offline verfügbaren Dateien mit Zeitstempel (Unix-Time in Millisekunden)
changeFeed placeHolder	noch keine gesicherten Erkenntnisse
sequenceNumber	Index - noch keine gesicherten Erkenntnisse
thumbnailStatus	2 für Dateien für die es nur ein typbedingtes Thumbnail gibt; 1 für Dateien für die es ein individuelles Thumbnail

	gibt 0 bei Meine Ablage und der neu hochgeladenen Datei
invariantDriveId	Index - keine gesicherten Erkenntnisse (bisher für alle Einträge leer)
numericNormalizedTitle	keine gesicherten Erkenntnisse (beinhaltet häufig den Titel mit zusätzlichen Nullen und Einsen aufgefüllt)

Ein Sondereintrag stellt “Meine Ablage” dar. Dieser weicht in vielen Punkten von allen anderen Einträgen ab. Dies ist aber zu erwarten, da dieser Eintrag immer existiert und vom Nutzer nicht verändert werden kann.

4.3.1.3 Document150

Die Tabelle Document150 ermöglicht es für Dateien eine Korrelation zwischen einem Eintrag in der Tabelle Entry150 und den Metadaten zum Inhalt in der Tabelle DocumentContent150 herzustellen. Das Schema ist nun nachfolgend dargestellt.

```
sqlite> .schema Document150
CREATE TABLE "Document150" ("Document_id" INTEGER PRIMARY KEY AUTOINCREMENT, "doSync" INTEGER NOT NULL,
"entryId" INTEGER NOT NULL, "contentId" INTEGER, "pdfContentId" INTEGER, "htmlUri" TEXT, "md5Checksum" TEXT,
"size" INTEGER, "quotaBytesUsed" INTEGER, "relevanceSynced" INTEGER NOT NULL DEFAULT 0, "syncReason" INTEGER
NOT NULL DEFAULT 0, FOREIGN KEY("entryId") REFERENCES "Entry150"("Entry_id") ON DELETE CASCADE, FOREIGN
KEY("contentId") REFERENCES "DocumentContent150"("DocumentContent_id") ON DELETE SET NULL, FOREIGN
KEY("pdfContentId") REFERENCES "DocumentContent150"("DocumentContent_id") ON DELETE SET NULL);
CREATE UNIQUE INDEX "Document150_entryId_ui" ON "Document150" (entryId);
CREATE INDEX "Document150_contentId_i" ON "Document150" ("contentId");
CREATE INDEX "Document150_pdfContentId_i" ON "Document150" ("pdfContentId");
sqlite>
```

Nachfolgend sind nun die wesentlichen Einträge in der Tabelle erklärt.

Spalte	**Beschreibung**
Document_id	Primärer Schlüssel (Unique Index)
entryId	Referenz zum Eintrag in Entry150 - Index
contentId	Referenz zum Eintrag in DocumentContent150 - Index
htmlUri	Uri des Dokumentes online
md5Checksum	MD5 Checksumme des Dokumentes in Originalform
size	Größe des Dokumentes in Bytes
quotaBytesUsed	Größe die auf das Nutzerquota angrechnet wird
doSync pdfContentId relevanceSynced syncReason	noch keine Erkenntnisse

4.3.1.4 DocumentContent150

Durch diese Tabelle kann der Pfad zu einem Dokument auf dem Gerät ermittelt werden. Zu beachten ist, dass bei der Pfadangabe keine Moint-Points berücksichtigt werden. Weiterhin können die wesentliche Parameter zur Verschlüsselung ermittelt werden, um ein Dokument entschlüsseln zu können. Das Schema der Tabelle ist nachfolgend dargestellt.

```
sqlite> .schema DocumentContent150
CREATE TABLE "DocumentContent150" ("DocumentContent_id" INTEGER PRIMARY KEY AUTOINCREMENT, "contentETag"
TEXT, "contentType" TEXT NOT NULL, "encryptionKey" BLOB, "encryptionAlgorithm" TEXT,
"encryptionAlgorithmParameters" TEXT NOT NULL DEFAULT '', "encryptionAlgorithmIv" BLOB,
"allPendingCommandsPersisted" INTEGER NOT NULL DEFAULT 0, "hasPendingChanges" INTEGER NOT NULL DEFAULT 0,
"hasPendingComments" INTEGER NOT NULL DEFAULT 0, "filePath" TEXT, "notOwnedFilePath" TEXT, "lastOpenedTime"
INTEGER, "lastModifiedTime" INTEGER, "serverSideLastModifiedTime" INTEGER, "md5Checksum" TEXT,
"isDocumentSnapshotted" INTEGER NOT NULL DEFAULT 1, "isTemporary" INTEGER, "referencedContentId" INTEGER,
"isDirty" INTEGER NOT NULL DEFAULT 0, "gcLockExpiryTime" INTEGER NOT NULL DEFAULT 0, "manifestId" INTEGER,
"documentId" TEXT, "referencedFontFamilies" TEXT, FOREIGN KEY("referencedContentId") REFERENCES
"DocumentContent150"("DocumentContent_id") ON DELETE SET NULL, FOREIGN KEY("manifestId") REFERENCES
"Manifest150"("Manifest_id") ON DELETE CASCADE);
CREATE INDEX "DocumentContent150_referencedContentId_i" ON "DocumentContent150" ("referencedContentId");
sqlite>
```

Die wesentlichen Einträge in der Tabelle sind nun nachfolgend erklärt.

Spalte	**Beschreibung**
DocumentContent_id	Primärer Schlüssel
contentType	MimeType des Dokumentes
encryptionKey encryptionAlgorithm encryptionAlgorithmParameters encryptionAlgorithmIv	Parameter für das Entschlüsseln des Dokuments; Keys sind als Blob gespeichert und können als Hexwerte exportier werden
filePath	Pfad zum Dokument auf dem Gerät
notOwnedFilePath	noch keine gesicherten Erkenntnisse
lastOpenedTime lastModifiedTime serverSideLastModifiedTime	Zeitstempel (Unix-Time in Millisekunden)
referencedContentId	noch keine gesicherten Erkenntnisse - Index
allPendingCommandsPersisted hasPendingChanges hasPendingComments	noch keine gesicherten Erkenntnisse
contentETag isDocumentSnapshotted isTemporary isDirty gcLockExpireTime manisfestId documentId referencedFontFamilies	noch keine gesicherten Erkenntnisse

Mit Hilfe der Tabelle Entry150 kann für die Datei test1.bin die Entry_id 7 ermittelt werden. In der Tabelle Document150 kann die contentId 5 ermittelt werden. Danach können in dieser Tabelle DocumentContent150 sowohl der Pfad als auch die Angaben zur Verschlüsselung ermittelt werden. Im Anschluss kann die Datei extrahiert und entschlüsselt werden. Teile diese Vorganges sind nachfolgend dargestellt.

```
sqlite> select filePath, hex(encryptionKey), hex(encryptionAlgorithmIv), md5Checksum from DocumentContent150
where DocumentContentId == 5;
Error: no such column: DocumentContentId
sqlite> select filePath, hex(encryptionKey), hex(encryptionAlgorithmIv), md5Checksum from DocumentContent150
where DocumentContent_Id == 5;
/storage/sdcard/Android/data/com.google.android.apps.docs/files/pinned_docs_files_do_not_edit/37a8ea3073e53ac091
4f7ac1b52c0fbb/test1.bin|671C91C85C905DF0849758AA1B8C9441|9E6785031BFD4C3734A83D388095B0C7|
ea3fbf3b9ad017558612428882bc0178
sqlite> .exit
.... Extrahieren Ausgelassen ....
$ openssl aes-128-cbc -in test1.bin-enc -out test1.bin -d -K 671C91C85C905DF0849758AA1B8C9441 -iv
9E6785031BFD4C3734A83D388095B0C7
$ md5sum test1.bin*
ea3fbf3b9ad017558612428882bc0178  test1.bin
c1f35bc7530c0cf43f3c42a2e6902536  test1.bin-enc
```

4.3.1.5 Collection150

Diese Tabelle listet alle Ordner auf und Correliert die Entry_id aus Entry150 mit der entsprechenden Collection_id. Das Schema ist nachfolgend dargestellt.

```
sqlite> .schema collection150
CREATE TABLE "Collection150" ("Collection_id" INTEGER PRIMARY KEY AUTOINCREMENT, "daysToSync" INTEGER,
"syncNewDocsByDefault" INTEGER, "entryId" INTEGER NOT NULL, FOREIGN KEY("entryId") REFERENCES
"Entry150"("Entry_id") ON DELETE CASCADE);
CREATE UNIQUE INDEX "Collection150_entryId_ui" ON "Collection150" (entryId);
```

In der nachfolgenden Tabelle sind die wesentlichen Einträge näher erläutert.

Spalte	**Beschreibung**
Collection_id	Primary Key
entryId	ID des Eintrages in Entry150
daysToSync syncNewDocsByDefault	keine gesicherten Erkenntnisse

4.3.1.6 Containsid150

Mit Hilfe der Tabelle Containsid150 lässt sich zuordnen welcher Eintrag in welchem Ordner/welcher Collection enthalten ist. Das Schema der Tabelle ist nachfolgend dargestellt.

```
sqlite> .schema containsid150
CREATE TABLE "ContainsId150" ("ContainsId_id" INTEGER PRIMARY KEY AUTOINCREMENT, "entryId" INTEGER NOT
```

```
NULL, "collectionId" INTEGER NOT NULL, FOREIGN KEY("entryId") REFERENCES "Entry150"("Entry_id") ON DELETE
CASCADE, FOREIGN KEY("collectionId") REFERENCES "Collection150"("Collection_id") ON DELETE CASCADE);
CREATE INDEX "ContainsId150_entryId_i" ON "ContainsId150" ("entryId");
CREATE INDEX "ContainsId150_collectionId_i" ON "ContainsId150" ("collectionId");
```

In der nachfolgenden Tabelle sind die wesentlichen Einträge näher erläutert.

Spalte	Beschreibung
ContainsId_id	Primärer Schlüssel
entryId	ID des Eintrages in Entry150
collectionId	ID des Eintrages in Collection150

5 Zusammenfassung

Google Drive ist ein Filehosting-Dienst. Betrieben wird er durch Google Inc. Zusätzlich zum eigentlichen Speichern von Dateien können auch Office Dokument erstellt und bearbeitet werden. Für Android existiert eine eigene App. Die Struktur der gespeicherten Daten unterscheidet sich wesentlich von der Desktopverison[6].

5.1 DocList.db

Alle wesentlichen Metadaten zu den Dateien werden in der SQLite Datenbank DocList.db abgespeichert. Die Datenbank befindet sich im Verzeichnis /data/data/com.google.android.apps.docs/databases/. Sie enthält fast alle wesentlichen Metadaten zu den verwalteten Dateien und Ordnern.

5.2 Files

Die Dateien werden entweder im Verzeichnis /data/data/com.google.android.apps.docs/files/pinned_docs_files_do_not_edit/ oder /storage/sdcard/Android/data/com.google.android.apps.docs/files/pinned_docs_files_do_not_edit/ gespeichert. Die Dateien auf der SD Karte werden dabei verschlüsselt. Der genaue Ort und die Daten zur Verschlüsselung können den Tabellen DocumentContent150, Document150 und Entry150 entnommen werden.

6http://bitforensics.blogspot.de/2012/12/google-drive-artifacts-explained.html abgerufen am 9.1.2016
http://forensicswiki.org/wiki/Google_Drive abgerufen am 9.1.2016
http://forensicswiki.org/wiki/Mozilla_Firefox abgerufen am 9.1.2016
http://kb.mozillazine.org/Profile_folder_-_Firefox abgerufen am 9.1.2016

5.3 Cache und Thumbnails

Im Verzeichnis /data/data/com.google.android.apps.docs/cache/fetching/ werden zum einen inhaltliche Daten der Google Drive eigenen Officedokumente und Vorschaubilder von anderen Dokumenten gespeichert. Die Google eigenen Officedokumente sind als Zip-Archiv abgelegt.

Im Verzeichnis /data/data/com.google.android.apps.docs/cache/docs_glide/data werden Thumbnail-Bilder in verschiedenen Größen für die jeweiligen Dokument gespeichert.
Eine konkrete eindeutige Zuordnung der Dateien zu ihren jeweiligen Originalen auf Basis der Daten der DocList.db war bisher noch nicht möglich.

Neben der Datenbank DocList.db befindet sich eine weitere Datenbank google_analytics_v4.db im Ordner /data/data/com.google.android.apps.docs/databases.

Im Verzeichnis /data/data/com.google.android.apps.docs/shared_prefs befinden sich verschiedene XML-Dokumente mit diversen Einstellungen.

6 Anhang

6.1 spuren.tcl

```
#!/usr/bin/tclsh

set idiffsfolder "./idiffs"

if { $::argc > 0 } {
   set i 1
   foreach arg $::argv {
      puts "argument $i is $arg"
      incr i
   }
   set idiffsfolder [lindex $arg 0]
   puts "nutze $idiffsfolder"
} else {
   puts "nutze $idiffsfolder"
}


# opening the database
package require sqlite3
sqlite3 pedb ./preparedevidence.db

# drop old table if it exists
catch {pedb eval {DROP TABLE pe}}

# create the new table
pedb eval {CREATE TABLE pe (pfad text, stempeltyp text, nr int, aktion text, unique (pfad, stempeltyp, nr, aktion))}


proc insertvalue {pfad stempeltyp nr aktion} {
      if { [ catch {pedb eval {INSERT INTO pe VALUES($pfad, $stempeltyp, $nr, $aktion)}}] } {
            puts "$pfad $stempeltyp $nr $aktion bereits enthalten"
      }
}
```

```
proc handleline { line mode aktion nr } {
	set linelist [split $line "\t"]
	#puts "mode: $mode linelist: $linelist"
	switch $mode {
		"new"	{
				# der zweite eintrag der zeile ist der Dateiname und wird in die Datenbank geschrieben
				set fileentry [lindex $linelist 1]
				insertvalue $fileentry a $nr $aktion
				insertvalue $fileentry m $nr $aktion
				insertvalue $fileentry c $nr $aktion
				insertvalue $fileentry cr $nr  $aktion

			}
		"deleted"	{
				set fileentry [lindex $linelist 1]
				insertvalue $fileentry d $nr $aktion
			}
		"renamed"	{
			# kaennen nach studienbrief ignoriert werden
			}
		"modified"	{
				switch -regexp -matchvar match -- [lindex $linelist 1] {
					"(.*)time changed"	{
						set fileentry [lindex $linelist 0]
						insertvalue $fileentry [lindex $match 1] $nr $aktion
					}
				}
			}
		"properties"	{
				switch -regexp -matchvar match -- [lindex $linelist 1] {
					"(.*)time changed"	{
						set fileentry [lindex $linelist 0]
						insertvalue $fileentry [lindex $match 1] $nr $aktion
					}
				}
			}
		default { puts "mode unknown" }
	}
}

proc parsefile { fd file } {
	puts "parsing file $fd $file"
	set dateiname [file tail $file]
	#puts $dateiname
	regexp {([a-z]*)([0-9]*).idiff} $dateiname matched aktion nr
	puts "name der aktion $aktion"
	while {[gets $fd line] >= 0} {
		# zuerst pruefen ob es sich um eine Sektionszeile handelt
		switch -regexp -matchvar m1 $line {
			"^New files:$"	{
					puts "New files mode - every entry here creates a m c cr timestamp"
					set mode "new"
				}
			"^Deleted files:$"	{
					puts "Delted files mode - every entry here creates d timestamp"
					set mode "deleted"
				}
			"^Renamed files:$"	{
					puts "Renamed files mode - every entry here is ignored as desired"
					set mode "renamed"
				}
			"^Files with modified contents:$"	{
					puts "modified files mode - every entry here creates the appropiate timestamp"
					set mode "modified"
				}
```

```
                "^Files with changed properties:$"      {
                        puts "changed properties files mode - every entry here creates the appropiate timestamp"
                        set mode "properties"
                    }
                "^===*$"        {
                    }
                "^$"    {
                        #  puts "empty line"
                    }
                default {
                        handleline $line $mode $aktion $nr
                    }
            }

    }
}

proc parseallfiles {} {
    global idiffsfolder
    foreach file [glob -directory $idiffsfolder *.idiff] {
        set fd [open $file]
        parsefile $fd $file
        close $fd
    }
}

proc formatierteausgabe { daten spaltenanzahl dateipfad} {
    set file [open $dateipfad w]

    set i 0
    foreach j $daten {
        puts -nonewline $file "$j"
        puts -nonewline "$j"
        incr i
        if {$i==$spaltenanzahl} {
            set i 0
            puts $file ""
            puts ""
        } else {
            puts -nonewline $file "\t"
            puts -nonewline "\t"
        }
    }
    close $file
}

# me ausgabe a
proc generateme aktion {
    puts "ME fuer $aktion ============"

    set x [pedb eval {SELECT pfad, stempeltyp, COUNT(*) FROM pe WHERE aktion=$aktion GROUP BY pfad,
stempeltyp}]
    formatierteausgabe $x 3 "$aktion.me"
}

proc generatece aktion {
    puts "CE fuer $aktion  ============================="

    set x [pedb eval {SELECT A.pfad, A.stempeltyp FROM pe as A WHERE NOT EXISTS
            (SELECT B.pfad, B.stempeltyp FROM pe AS B WHERE A.pfad = B.pfad AND A.stempeltyp = B.stempeltyp
AND NOT B.aktion=$aktion)
            GROUP BY A.pfad, A.stempeltyp HAVING COUNT(*)=(SELECT MAX(nr) FROM pe WHERE
aktion=$aktion)}]

    formatierteausgabe $x 2 "$aktion.ce"
```

```
	set x [pedb eval {SELECT A.pfad, A.stempeltyp FROM pe as A WHERE NOT EXISTS
			(SELECT B.pfad, B.stempeltyp FROM pe AS B WHERE A.pfad = B.pfad AND A.stempeltyp = B.stempeltyp
AND NOT B.aktion=$aktion)
			GROUP BY A.pfad, A.stempeltyp HAVING COUNT(*)>(SELECT MAX(nr) FROM pe WHERE
aktion=$aktion)*2/3}]
	formatierteausgabe $x 2 "ce-sonstige/$aktion-medium.ce"


	set x [pedb eval {SELECT A.pfad, A.stempeltyp FROM pe as A WHERE NOT EXISTS
			(SELECT B.pfad, B.stempeltyp FROM pe AS B WHERE A.pfad = B.pfad AND A.stempeltyp = B.stempeltyp
AND NOT B.aktion=$aktion)
			GROUP BY A.pfad, A.stempeltyp}]
	formatierteausgabe $x 2 "ce-sonstige/$aktion-minimum.ce"

}

parseallfiles

foreach aktion [list init install test] {
	generateme $aktion
	puts "########"
	generatece $aktion
}


pedb close
```