Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2423

Evaluierung der Effizienz von schmiersicheren Passwortsystemen

Frank Steimle

Studiengang:	Informatik
Prüfer:	Prof. Dr. Albrecht Schmidt
Betreuer:	M.Sc. Stefan Schneegaß Dr. Florian Alt
Beginn am:	9. April 2013
Beendet am:	9. Oktober 2013
CR-Nummer:	H.5.2, K.6.5

Kurzfassung

Authentifizierungssysteme für touchfähige mobile Geräte haben eine große Schwäche: Jedes Mal wenn sich der Benutzer authentifiziert, entsteht eine Fettspur auf dem Display. Diese Fettspur kann von Dritten genutzt werden um das Passwort herauszufinden und damit das Authentifizierungssystem zu umgehen. In dieser Arbeit wurde ein Authentifizierungssystem für mobile Geräte entwickelt, das grafische Passwörter verwendet und diese Gefahr minimieren soll. Dazu bedient sich das System geometrischer Transformationen wie Verschiebung, Rotation, Scherung, Spiegelung und Skalierung. Dadurch wird erreicht, dass sich die Position an der das Passwort eingegeben werden muss bei jedem Loginvorgang ändert. Abschließend wurde in einer Studie mit 20 Teilnehmern untersucht, ob das System wirklich die Sicherheit erhöht. In der Studie sollten die Probanden mit Hilfe hochauflösender Fotos von Fettspuren vergangener Loginversuche versuchen das Passwort herauszubekommen. Die Studie hat gezeigt, dass das grafische Passwörter in Verbindung mit geometrischen Transformationen schwerer machen das Passwort zu stehlen als aktuelle Authentifizierungssystem wie PIN und Lockpattern.

Abstract

Authentication systems for touch-enabled mobile devices suffer from one big disadvantage: every time the user authenticates, a smudge trace is generated on the display. This traces can be used by others to identify the passwort and to bypass the authentication system. In this work an authentification system for mobile devices is developed, which uses graphical passwords and hence should minimize this threat. For that purpose, the system uses geometrical transformations like translation, rotation, shearing, flipping, and scaling. Thereby is achieved, that the location where the password has to be entered changes its location on the screen every time the user tries to login. In order to evaluate the security of the approach a study with 20 participants was carried out to test whether the system really increases the security. During the study the participants tried to identify the password using high quality pictures from smudge traces from former logins. The study showed that graphical passwords in combination with geometrical transformations make it harder to reveal the password from the smudge traces than state-of-the-art systems like PIN and lock pattern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

1. Einleitung

Die Anzahl der Benutzer, die Dienste wie Online-Banking, Einkaufen oder soziale Netze mit ihrem Smartphone verwenden, steigt immer weiter an. All diese Dienste sind zugangsbeschränkt, jedoch speichern viele Benutzer ihre Passwörter direkt auf ihrem Smartphone. Muslukhov et al. haben in [MBK+13] erst vor kurzem gezeigt, dass 64% aller Smartphonenutzer ihr Gerät durch Passwörter schützen. Außerdem stellten sie fest, dass die meisten dabei auf PIN und Lockpattern setzen, obwohl immer mehr alternative Systeme wie Gesichtserkennung [Kel13] oder grafische Passwörter, bei denen man beispielsweise Passwortpunkte auf einem Bild definiert, verfügbar sind.
Einer der größten Nachteile ist in diesem Zusammenhang, dass beim Entsperren touchfähiger Geräte mit den Fingern immer Fettspuren auf dem Display entstehen. Diese Fettspuren sind eine große Schwäche von grafischen Passwörtern, insbesondere von Lockpattern wie sie im Smartphonebetriebssystem Android verwendet werden ([ZKDLH13]), da die Spuren es erlauben das Passwort mit Hilfe des letzten erfolgreichen Loginvorgangs zu erraten. Text-basierte Passwörter und PINs sind mit Hilfe von Fettspuren nicht so leicht zu umgehen, da man zwar leicht die einzelnen Elemente des Passworts (Ziffern, Buchstaben) erkennt, aber die wichtigere Reihenfolge nicht aus der Fettspur ablesen kann.
Es gibt nur zwei Möglichkeit solchen Angriffen zu entgehen: das Display nach dem Benutzen gründlich zu reinigen oder beim Benutzen des Geräts so viele zusätzliche Fettspuren zu erzeugen, dass die Fettspur des Passworts nicht mehr erkennbar ist. Leider ist es im Alltag nicht praktikabel, nach jeder Interaktion mit dem Gerät das Display zu putzen und wenn das Handy nach dem Benutzen wieder in die Hosentasche geschoben wird, wird die Fettspur nicht genügend verwischt. Andererseits besteht die alltägliche Interaktion nur aus Aktionen wie Mails/SMS lesen, wobei nur vertikale Spuren entstehen, oder telefonieren, wobei auch vertikale Spuren entstehen oder nur ein Paar Ziffern gedrückt werden. Die Fälle in denen das Gerät nur entsperrt wird, um auf die Uhr zu schauen oder um nachzusehen, ob man neue E-Mails erhalten hat, müssen auch zur alltäglichen Interaktion zählen. Dadurch entstehen nicht genügend andere Spuren, die die Fettspur des Passworts (das mehrmals am Tag eingegeben wird) auf eine Art und Weise überlagern würden, sodass man sie nicht mehr erkennt.
Um der Problematik dieser Angriffe in Zukunft zu entgehen, soll in dieser Studienarbeit ein Android-Prototyp entwickelt werden, der es schwieriger macht, die oben beschriebenen Fettspuren dazu zu nutzen das Passwort zu erraten. Dazu soll der Prototyp grafische Passwörter verwenden und die Passworteingabe bei jedem Loginvorgang in einer anderen Lage oder an einer anderen Position anzeigen. Um die Passworteingabe bei jedem Versuch zu verändern, sollen geometrische Transformationen, wie beispielsweise Rotation verwendet werden. Anschließend soll im Rahmen einer Studie getestet werden, ob der Prototyp die Sicherheit des Passworts wirklich erhöht hat.

Überblick über die weiteren Kapitel

Kapitel 2 - Related Work und Android Background gibt einen Überblick über die relevanten Themen dieser Studienarbeit und stellt Möglichkeiten vor ein Lockscreen-Replacement für Android zu implementieren.

Kapitel 3 - Konzept stellt die grundlegende Idee dieser Arbeit vor.

Kapitel 4 - Implementierung stellt den im Rahmen dieser Arbeit entwickelten Prototypen vor.

Kapitel 5 - Benutzerstudie: Evaluierung der Sicherheit berichtet von der Studie, die durchgeführt wurde um zu zeigen, dass der in dieser Arbeit beschriebene Ansatz die Sicherheit erhöht.

Kapitel 6 - Zusammenfassung und Ausblick fasst die Ergebnisse dieser Arbeit zusammen.

Eingereichter Artikel auf Grundlage dieser Arbeit

Auf Grundlage der im Rahmen dieser Studienarbeit entstandenen Ergebnisse wurde ein Artikel erarbeitet und für die *31. SIGCHI International Conference on Human Factors in Computing Systems (2014)* eingereicht. Aus diesem Grund basiert diese Studienarbeit auf dem folgenden Artikel:
Schneegaß, S., Alt, F., Bulling, A., Steimle, F., Schmidt, A., SmudgeSafe: Geometric Image Transformations for Smudge-resistant User Authentication. *Submitted to the 32nd SIGCHI International Conference on Human Factors in Computer Systems (2014)*

2. Related Work und Android Background

Zuerst soll in den ersten Abschnitten die Literatur zu den für diese Studienarbeit relevanten Themen zusammengefasst werden. Im letzten Abschnitt dieses Kapitels werden die Möglichkeiten ein Lockscreen-Replacement für Android zu implementieren vorgestellt.

2.1. Grafische Passwortsysteme

Grafische Passwörter erfreuen sich immer größerer Beliebtheit, da sie tendenziell höhere Sicherheit bieten und dabei nicht die Usability in Mitleidenschaft ziehen [ML07] [SWKW13]. Es gibt drei Arten grafischer Passwörter, die im folgenden vorgestellt werden.

Recognition-based Systems oder *cognoemtric schemes* setzen darauf, dass der Benutzer aus einer Menge an Bildern, das korrekte auswählen muss. Dabei spielt es keine Rolle welcher Art die Bilder sind. Bei *Déjà Vu* sucht sich der Benutzer fünf Bilder als Passwort aus [DP00]. Während der Anmeldung werden dann 25 Bilder gezeigt und es müssen die richtigen fünf Bilder in der richtigen Reihenfolge ausgewählt werden. In *Déjà Vu* werden nur synthetische Bilder verwendet, da die Autoren verhindern wollten, dass Benutzer schwache Passwörter wählen. Außerdem sollte verhindert werden, dass die Passwörter verbal beschrieben werden können. Hätte man persönliche Bilder zugelassen, wäre die Sicherheit in Mitleidenschaft gezogen worden, da die Benutzer wahrscheinlich Bilder ausgewählt hätten, die ihnen etwas bedeuten. Davis et al. entwickelten Story Scheme [DMR04]. In diesem Authentifizierungssystem sollten sich die Benutzer eine Geschichte ausdenken um, sich ihr Passwort, das aus Bildern von Gesichtern, Tieren oder Gegenständen bestand, zu merken. Allerdings stellten sie fest, dass sich die Probanden meistens einfach nur die Reihenfolge der Bilder gemerkt haben statt einer Geschichte. *Passfaces*[1] ist ein weiteres recognition-based system, das allerdings mit einem höheren Trainingsaufwand verknüpft ist. Anfangs muss der Benutzer aus einer Menge von Gesichtern vier auswählen und diese Gesichter dann in mehreren Trainingsrunden wieder erkennen. Der eigentliche Loginvorgang besteht aus vier Schritten und in jedem Schritt wird ein Grid mit Gesichtern angezeigt. Pro Schritt muss eines der am Anfang gewählten Gesichter erkannt werden.

[1] http://www.passfaces.com/

Recall-based Systems oder *drawmetric schemes* verlangen vom Benutzer das Passwort aus dem Gedächtnis ohne Hinweise zu zeichnen. Jermyn et al. stellten 1999 ein solches System namens *Draw-A-Secret* vor, in dem der Benutzer ein Passwort auf einem $2D$-Gitter mit einem oder mehreren Strichen festlegen musste. Nachdem das Passwort gezeichnet wurde, wurde es vom System als Sequenz von benutzten Zellen kodiert und abgespeichert. Um sich zu authentifizieren musste der Benutzer wieder ein Bild zeichnen, dass zu derselben Sequenz führt. Von Dunphy et. al wurde festgestellt, dass Benutzer von Draw-A-Secret dazu tendierten zu einfache Passwörter erstellten, indem sie häufig die Mitte des Grids und symmetrische Passwörter verwendeten. Deswegen stellten sie in [DY07] *Background Draw-A-Secret* vor. Bei diesem System wurde das Passwort auf einem Bild definiert. In einer Studie stellten die Autoren fest, dass das Bild für die Sicherheit und die Merkfähigkeit von großer Bedeutung ist.

Cued-recall Systems oder *Locimetric schemes* lassen den Benutzer sein Passwort auf einem Bild definieren. Ein Beispiel dafür ist *Passpoints*, dass von Wiedenbeck et al. in [WWB+05] präsentiert wurde. In diesem System muss der Benutzer auf dem Bild Punkte als Passwort definieren und diese wieder in richtiger Reihenfolge anwählen um sich einzuloggen. Microsoft hat für Windows 8 eine Kombination aus recall-based und cued-recall implementiert[2]. In diesem System sollen die Benutzer auf einem Bild verschiedene Formen (Kreise, Linien) zeichnen. Ein großer Nachteil des Systems ist, dass die Benutzer nicht nur das Bild auswählen können, sondern auch die Lage der Passwortelemente, was dazu führt das meistens die offensichtlichsten genommen werden (so genannte Hotspots [TO07]). Um dieses Problem zu lösen, haben Bulling et al. Saliency Masks in cued-recall Systemen verwendet, um die markantesten Teile des Bildes von der Passwortdefinition auszuschließen [BAS12]. Dadurch wurde eine wesentliche Verbesserung der Sicherheit erreicht.

2.2. Sicherheit mobiler Geräte

Mit der Fähigkeit mobiler Geräte große Datenmengen zu speichern und zu verarbeiten, wird die Notwendigkeit größer diese Daten mittels Authentifizierungsverfahren unberechtigten Zugriffen schützen. Chin et al. untersuchten ob Smartphonebenutzer Bedenken hinsichtlich ihrer Privatsspäre haben, wenn Apps auf ihre privaten Daten zugreifen [CFSW12]. Dabei fanden sie raus, dass die Benutzer sich im Umgang mit Handys mehr um die Sicherheit ihrer Daten sorgen, als im Umgang mit Laptops. Da sie Angst haben, dass ihr Telefon gestohlen wird, verwenden sie vertrauliche Daten sehr selten auf dem Handy. Muslukhov et al. fanden zudem heraus, dass die Benutzer von Smartphones eher Angst davor haben, dass ihnen bekannte Personen sich Zugriff zu ihren Daten verschaffen, als Fremde [MBK+13]. In einer anderen Arbeit fanden dieselben Autoren heraus, dass sich die Daten, die ein Benutzer auf Mobiltelefonen speichert, in verschiedene Kategorien einteilen lassen und untersuchten warum die Benutzer die Daten als vertraulich oder wertvoll einstufen [MBK+12]. Dorflinger

[2] http://windows.microsoft.com/en-us/windows-8/picture-passwords

et al. untersuchten wie Benutzer verschiedene neuartige Authentifizierungsverfahren, wie beispielsweise biometrischen Verfahren oder Gestenerkennung im zwei- und dreidimensionalen Raum, wahrnehmen [DVKF10]. Insbesondere interessierten sie sich dafür, wie sicher die Benutzer die einzelnen Verfahren einschätzten. Sie fanden dabei heraus, dass die meisten Benutzer die PIN für nicht sicher halten.

2.3. Smudge Attacks

Versuche mit Hilfe einer Fettspur das Passwort herauszufinden, nennt man Smudge Attacks. Andriotis et al. untersuchten die Sicherheit von Androids Lockpattern [ATOY13]. Sie zeigten verschiedene Ansätze, die verwendet werden können um Passwörter mit Hilfe unvollständiger Fettspuren zu erraten. Aviv et al. untersuchten ebenfalls Androids Lockpattern, legten jedoch ihren Fokus darauf unter welchen Bedingungen Smudge Attacks einfach durchgeführt werden können [AGM+10]. Sie fanden heraus, dass durch die richtige Beleuchtung und die richtige Kamera die Mehrheit von Fettspuren dazu benutzt werden konnte das ursprüngliche Pattern zu erkennen. Auf eine ähnliche Art zeigten Zhang et al., dass Fingerabdrücke die von Displays genommen werden, dazu benutzt werden können eine PIN herauszufinden.
De Luca et al. schlugen das Konzept der impliziten Authentifizierung vor [DLHB+12]. Dabei wird der Benutzer nicht nur durch das Passwort authentifiziert, sondern auch über die Art und Weise wie das Passwort eingegeben wird. Das bedeutet, dass neben dem Passwort beispielsweise auch die Geschwindigkeit, in der das Passwort eingegeben wird oder der Druck, mit dem das Passwort eingegeben wird, überprüft werden. Zezschwitz et al. präsentierten drei Alternativen zum Lockpattern, die dazu führen, dass die Benutzer beim Eingeben des Passworts zusätzliche Fettspuren hinterlassen, sodass die Spuren nicht mehr dazu genutzt werden können, das Passwort zu rekonstruieren [ZKDLH13]. Der erste vorgestellte Prototyp, Pattern-90, funktioniert wie das Android-Lockpattern, nur dass das Grid seine Position und seine Ausrichtung ändern kann. Der zweite, Marbles, besteht aus einem Ring von zufällig angeordneten Farbpunkten, wobei das richtige Passwort in korrekter Reihenfolge in die Kreismitte gezogen werden muss. Marble Gap, der dritte Prototyp, ist ein dreigeteilter Bildschirm, bei dem im oberen und unteren Teil wieder bunte Farbpunkte sind, die in der farblich richtigen Reihenfolge in die Mitte gezogen werden müssen. AlRowaily und AlRubaian präsentierten ein System namens *WhisperCore*. Das System verlangt am Ende des Loginprozesses, das der Benutzer noch einmal große Teile des Bildschirms abwischt, um die Fettspur des Passworts durch eine neue zu überdecken [AA11]. Oakley und Bianchi stellten ein System vor, das Multitouchpasswörter verwendet um das Beobachten des Passworts bei der Eingabe (sogenanntes shoulder-surfing) und das Verwenden der Fettspur um das Passwort herauszufinden zu erschweren [OB12].

2.4. Lockscreen-Replacement in Android

In diesem Abschnitt werden zwei Möglichkeiten vorgestellt, wie ein Lockscreen-Replacement in Android implementiert werden kann. Für beide Varianten ist es notwendig einen

Listing 2.1 Java-Code: Mit dem KeyguardManager die Tastatur sperren

```
KeyguardManager km = (KeyguardManager) getSystemService(KEYGUARD_SERVICE);
KeyguardLock kl = km.newKeyguardLock("lock");
kl.reenableKeyguard();
```

Broadcast Receiver zu implementieren, der auf die die Intents *ACTION_SCREEN_ON* und*ACTION_SCREEN_OFF* reagiert.

KeyguardManager

Mit dem KeyguardManager kann man mit drei Zeilen Code die Tastatur des Geräts sperren, wie in Listing 2.1 dargestellt. Das heißt insbesondere, dass der Homebutton nicht mehr reagiert. Um die Tastatur wieder freizugeben, muss der die Methode *kl.reenableKeyguard()* aufgerufen werden. In Verbindung mit einem Homescreen-Replacement kann damit ein Lockscreen-Replacement implementiert werden. Ein Homescreen-Replacement ist eine Activity, bei der in der AndroidManifest.xml im Intent-Filter die Action *android.intent.action.MAIN* und die Categorys *android.intent.category.HOME* und *android.intent.category.DEFAULT* angegeben sind. Das Homescreen-Replacement muss nun jedes Mal, wenn es aufgerufen wird mit der Methode *isKeyguardLocked()* der Keyguard-Klasse prüfen, ob die Tastatur gesperrt wurde. Wenn die Tastatur gesperrt wurde, wird die eigentliche Lockscreen-Activity aufgerufen um es dem Benutzer zu ermöglichen das Gerät zu entsperren, andernfalls wird der zuvor festgelegte eigentliche Homescreen aufgerufen. Im Broadcast Receiver muss nun nur noch implementiert werden, dass die Tastatur gesperrt werden soll wenn der Bildschirm ausgeht und wenn er wieder angeht muss die das Homescreen-Replacement aufgerufen werden.
Dieser Ansatz hat leider folgende Nachteile:

- Die verwendeten Funktionen wurden mit dem API Level 13 (Android 3.2) als deprecated markiert.
- Die verwendeten Funktionen funktionieren nicht wenn ein eine App als aktiver Geräteadministrator registriert ist.
- Es wird nur die Tastatur gesperrt. Man hätte, im Gegensatz zu einem gesperrten Gerät, beispielsweise Zugriff auf den kompletten Inhalt der SD-Karte, wenn man das Gerät per USB mit einem Computer verbindet.

Device Policy Manager

Die Device Policy Manager API ist ursprünglich dafür vorgesehen gewesen, im Firmenumfeld gewisse Sicherheitsrichtlinien durchsetzen zu können. Mit ihrer Hilfe lassen sich auf Dienstgeräten bestimmte Passwortrichtlinien durchsetzen, die Kamera lässt sich in bestimmten Situationen deaktivieren oder es kann sogar der gesamte Speicher des Geräts gelöscht werden. Um mit dieser API ein Lockscreen-Replacement zu implementieren, werden allerdings nur zwei der Richtlinien benötigt. Und zwar sind das diejenigen, die es ermöglichen

das Passwort zurückzusetzen und das Gerät zu sperren.
Über den BroadcastReceiver muss nun jedes Mal wenn der Bildschirm ausgeht, ein Passwort gesetzt werden und das Gerät gesperrt werden. Das Gerät ist dann mit diesem Passwort geschützt. Wenn der Bildschirm wieder angeht, muss der Broadcast Receiver den Lockscreen starten, der im Fall eines erfolgreichen Loginversuchs das Passwort auf den leeren String setzt.
Im Gegensatz zur obigen Methode muss kein Homescreen-Replacement implementiert werden. Allerdings muss der Broadcast Receiver an einen Service gekoppelt werden, der gestartet werden kann, wenn das Gerät bootet. Bei der Methode mit dem Homescreen-Replacement ist dies nicht nötig, da der Broadcast Receiver mit dem Homescreen-Replacement als eines der ersten Module gestartet wird, wenn das Gerät bootet.

Entscheidungsfindung

Da die KeyguardManager API als deprecated markiert ist, kann es passieren, dass die API in einer der zukünftigen Androidversionen gar nicht mehr vorhanden ist. Das würde bedeuten, dass ein auf diese Art implementierter Lockscreen nicht mehr funktioniert.
Google stellt mittlerweile auch ein Programm mit dem man das Handy orten lassen kann und aus der Ferne sämtliche Daten löschen kann. Dieses Programm ist auf allen Geräten, die von Google hergestellt werden (Nexus-Serie), installiert. Will man dieses Programm nutzen, muss man es als Geräteadministrator registrieren und dadurch funktionieren Lockscreens, die mit der KeyguardManager API implementiert wurden, nicht mehr.
Außerdem kann durch die Benutzung der KeyguardManager API nicht sichergestellt werden, dass auf neueren Geräten wirklich alle Tasten gesperrt werden. Wenn nicht alle Tasten gesperrt werden, kann es sein, dass man aus dem Lockscreen über die Statusleiste am oberen Bildschirmrand in die Geräteeinstellungen kommt.
Da diese Nachteile doch sehr gravierend sind, wurde für diese Arbeit die Device Policy Manager API verwendet um einen Lockscreen-Replacement zu implementieren.

3. Konzept

In diesem Kapitel wird die Grundidee, grafische Passwörter mit Hilfe von geometrischen Transformationen sicherer zu machen, vorgestellt und auf geometrische Transformationen eingegangen, die verwendet werden könnten. Abschließend werden im Abschnitt über den Design Space einzelne Implementierungsaspekte vorgestellt, die beachtet werden müssen, wenn man ein solches System implementiert.

3.1. Grundidee

Bei der Benutzung von Geräten mit Touchscreens, wie Handys oder Tablets, entstehen unweigerlich Fettspuren auf dem Display (siehe Abbildung 3.1), die dazu benutzt werden können, das Passwort zu erraten, wie in [AGM+10] gezeigt wurde. Von Zezschwitz et al. haben in [ZKDLH13] gezeigt, dass dies insbesondere für Passwortsysteme wie Androids Lockpattern gilt, bei denen das Passwort direkt aus der Fettspur abgeleitet werden kann. Passwortsysteme, die auf textbasierte Passwörter oder PINs setzen, sind mit Hilfe von Fettspuren nicht so leicht zu umgehen. Man kann mit Hilfe der Fettspuren zwar die einzelnen Elemente (Buchstaben, Zahlen) des Passworts erkennen, aber es ist nicht möglich die richtige Reihenfolge der Elemente zu erkennen, wenn man nur die Fettspur betrachtet.

Die Grundidee um diese Fettspuren nutzlos zu machen, ist es grafische Passwörter sicherer zumachen, indem geometrische Transformationen auf das Bild angewendet werden, auf dem man das Passwort definiert. Dabei können Transformationen wie beispielsweise Verschiebung (das Bild wird an einer anderen Stelle angezeigt), Rotation (das Bild wird um einen zufälligen Winkel α gedreht), Skalierung (das Bild wird um einen zufälligen Faktor S skaliert), Scherung (das Bild wird um einen Faktor D geschert) oder Spiegelung (das Bild wird horizontal oder veritkal gespiegelt) verwendet werden. Wenn bei jedem Loginversuch eine zufällige Transformation benutzt wird, kann die Fettspur des vorherigen Loginversuchs nicht mehr direkt auf das Bild angewandt werden, was es schwieriger macht das Passwort mit Hilfe der Fettspur zu erraten. Da sich schon nach wenigen Logins die Fettspuren überlagern (siehe Abbildung 3.1 (c)), ist es fast unmöglich ist das ursprüngliche Passwort herauszubekommen.
Die Transformationen müssen gleichermaßen auf das Passwortbild wie auf die Passwortpunkte angewendet werden. Insbesondere muss darauf geachtet werden, dass dabei kein Passwortpunkt nach der Transformation außerhalb des Bildschirms liegt. SmearSafe trägt diesem Umstand Rechnung, indem es immer wenn ein neues Password definiert wird zu jeder Transformation Grenzwerte für die Parameter berechnet.
Die Lage der Passwortpunkte im Bild hat auch Einfluss auf die Sicherheit des Passworts und

Abbildung 3.1.: Fettspuren von drei aufeinanderfolgenden Loginversuchen: (a) Lockpattern, (b) PIN und (c) SmearSafe, das hier vorgeschlagene System.

die Anzahl der möglichen Passwörter. Werden die Passwortpunkte nah am Rand definiert, wird beispielsweise bei Rotation die Anzahl der möglichen Winkel verkleinert und dadurch wird es einfacher aus der resultierenden Fettspur das Passwort zu erraten. Aus diesem Grund darf es das System nicht zulassen, dass am Rand des Bildes Passwortpunkte definiert werden (siehe Abbildung 3.2). Dadurch wird sichergestellt, dass Transformationen, unabhängig von den Passwortpunkten, auf jeden Fall Einfluss nehmen auf die Lage des Passworts, und gleichzeitig die Anzahl der möglichen Passwörter nicht zu sehr beschränkt wird.

3.2. Geometrische Transformationen

Im folgenden werden geometrische Transformationen vorgestellt, die verwendet werden können um die Sicherheit grafischer Passwörter zu erhöhen. Es wird zusätzlich für jede Transformation eine Matrix angegeben, mit der aus jedem Pixel $\vec{v} = \binom{x}{y}$ des Originalbildes das Pixel $\vec{v}' = \binom{x'}{y'}$ des transformierten Bilds berechnet werden kann.

Verschiebung Die Verschiebung im $2D$-Raum beschreibt, wie man in Abbildung 3.3 (a) sehen kann, eine Bewegung des Bildes in X- und/oder Y-Richtung um einen Parameter D. Er kann sowohl positiv als auch negativ sein. Das bedeutet, dass das Bild sowohl nach links und rechts verschoben werden kann als auch nach oben und unten. Wenn man die Verschiebung entlang der X- und Y-Achse kombiniert, kann man den Bildmittelpunkt auch beispielsweise in Richtung der linken unteren Bildschirmecke verschieben. Man

Abbildung 3.2.: Um zu verhindern, dass Passwortpunkte am Rand den Nutzen des Systems schmälern, verhindert SmearSafe die Platzierung von Passwortpunkten am Rand.

Abbildung 3.3.: Geometrische Transformationen, die in diesem Abschnitt vorgestellt werden: (a) Verschiebung, (b) Skalierung, (c) Rotation, (d) Scherung und (e) Spiegelung.

muss bei der Verschiebung auch beachten, dass nach ihrer Anwendung Teile des Bildes nicht mehr sichtbar sind.

$$\vec{v}' = \begin{pmatrix} 1 & 0 & dX \\ 0 & 1 & dY \\ 0 & 0 & 1 \end{pmatrix} \vec{v}$$

Skalierung Als Skalierung (Abbildung 3.3 (b)) wird die Anwendung eines Parameters S auf die X- und Y-Dimension bezeichnet. Wenn S zwischen null und eins liegt, wird das Bild verkleinert. Ist S größer eins, wird das Bild vergrößert. Im Fall einer Vergrößerung muss beachtet werden, dass die Passwortpunkte nach der Transformation noch im sichtbaren Bereich des Bildes liegen. Beim Verkleinern des Bildes muss zusätzlich beachtet werden, dass das Bild nach Anwendung der Transformation noch erkennbar ist und dass die Passwortpunkte noch gut mit dem Finger getroffen werden können.

$$\vec{v}' = \begin{pmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{v}$$

Rotation Die Rotation beschreibt eine Drehung des Bildes um einen bestimmten Pivotpunkt (meistens der Bildmittelpunkt) um einen Winkel α, wie in Abbildung 3.3 dargestellt. Dabei muss wieder beachtet werden, dass die Transformation Teile des Bildes aus dem sichtbaren Bereich herausdreht. Außerdem bleiben Teile des Bildschirms leer, wenn das Bild nicht groß genug ist oder es werden neue Teile des Bildes sichtbar, wenn das Bild größer war als der Bildschirm.

$$\vec{v}' = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{v}$$

Scherung Bei der Scherung (Abbildung 3.3 (d)) oder Transvektion im $2D$-Raum wird jeder Punkt der Achse, entlang der die Transformation durchgeführt wird (X- oder Y-Achse), auf sich selbst abgebildet. Jeder andere Punkt wird parallel zu dieser Achse verschoben. Dabei ist die Länge des Vektor der Verschiebung proportional zum Abstand des Punktes zur Achse. Die Scherung kann auch Bildpunkte aus dem sichtbaren Bereich des Bildes verschieben.

$$\vec{v}' = \begin{pmatrix} 1 & D_X & 0 \\ D_Y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{v}$$

Spiegelung Die Spieglung (Abbildung 3.3 (e)) kann entweder entlang der X- oder der Y-Achse durchgeführt werden. Sie kommt ganz ohne Parameter aus und nach der Transformation befinden sich auch noch alle Bildpunkte im sichtbaren Bereich, die sich vorher dort befunden haben.

$$\vec{v}' = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{v}$$

3.3. Design Space

In diesem Abschnitt werden anhand des Design Spaces Implementierungsaspekte vorgestellt, die beachtet werden müssen, wenn die Transformationen aus dem vorherigen Abschnitt umgesetzt werden sollen, da sie Einfluss auf die Sicherheit und die Usability haben können.

Räumliche Dimensionen Transformationen können in verschiedenen räumlichen Dimensionen angewendet werden, und zwar $2D$ und $3D$. Gerade dadurch kann es aber zu ungewollten Effekten kommen. Beispielsweise hat eine Verschiebung im dreidimensionalen Raum entlang der der Z-Achse denselben Effekt wie eine Skalierung im zweidimensionalen. $3D$-Transformationen können zwar die Anzahl der möglichen Passwörter, den Theoretical Password Space (TPS), erhöhen, aber auch gleichzeitig die Usability beeinflussen, da beispielsweise Passwortpunkte verdeckt werden können. $3D$ Transformationen könnten in Zukunft eine größere Rolle spielen, wenn Geräte mit autostereoskopischen Displays auf den Markt kommen.

Bewahrung der Form Ob eine Transformation die Form des Bildes und des Passworts bewahrt oder nicht, ist auch ein wichtiger Aspekt, da er Einfluss haben kann auf die Einprägsamkeit des Passworts. Wenn ein Passwort beispielsweise eine runde Form enthält und eine Transformation wie beispielsweise die Scherung angewandt wird, die nicht die Form bewahrt, kann es dadurch für den Benutzer schwieriger werden sein die zuvor definierten Passwortpunkte zu finden.

Kombinierte Transformationen Es ist auch möglich mehrere Transformationen hintereinander anzuwenden, zum Beispiel kann das Bild zuerst verschoben und dann rotiert werden. Dadurch wird zwar die Sicherheit erhöht, aber auch die Usability gesenkt, da es dem Benutzer schwerer fallen kann seine Passwortpunkte zu finden. Mathematisch betrachtet ist eine Kombination von Transformationen eine Multiplikation der im vorherigen Abschnitt angegebenen Matrizen. Man muss zusätzlich bedenken, dass verschiedene Reihenfolgen verschiedene Ergebnisse verursachen.

Bildkontext und Bildausschnitt In [BAS12] wurde gezeigt, dass sich Benutzer ihr Passwort manchmal mittels einer Geschichte merken. Zum Beispiel zeigt ein Bild eine Straße, mehrere Personen, eine Bushaltestelle, fahrende Autos und eine Ampel. Eine Merkhilfe für ein Passwort könnte beispielsweise sein "Der Mann der neben dem Bushaltestellenschild wartet, springt vor das rote Auto und über die Ampel". Die Passwortpunkte sind folglich der Mann,

das Auto und die Ampel. In diesem Fall ist das Haltestellenschild kein Teil des Passworts aber ein wichtiges Detail, da man darüber den richtigen Mann findet. Wird nun durch durch eine Transformation beispielsweise das Haltestellenschild aus dem sichtbaren Bildauschnitt heraus bewegt, wird es schwerer für den Benutzer sich an sein Passwort zu erinnern. Manche Transformationen wie beispielsweise die Spiegelung bewahren im Allgemeinen den Kontext, allerdings wird es durch sie schwerer den Kontext zu erfassen, vorallem im Fall textlastiger Bilder.

Komplexität des Bildes In [PS90] wurde gezeigt, dass unter anderem die Bildmerkmale einen Einfluss haben, auf die Sicherheit des Passworts. Bilder mit niedriger Komplexität, sind anfälliger dafür, dass das Passwort erraten wird, da es nur wenige markante Punkte gibt. Transformationen wie beispielsweise die Skalierung ändern durch ihre Anwendung eventuell die Komplexität. Insbesondere wird die Komplexität durch Transformationen geändert, die Teile des Bildes sichtbar machen, die bei der Definition des Passworts nicht sichtbar waren.

Hintergrund Die Idee der geometrischen Transformationen ist auch auf andere Passwortsysteme anwendbar. Deswegen könnte man auch über Systeme nachdenken, die Transformationen nicht auf statische Bilder anwenden, sondern auf $3D - Bilder$, wie beispielsweise einen sich drehenden Würfel, bei dem man die richtige Seite auswählen muss. Man könnte auch Systeme wie das in [PSFW12] vorgestellte VidPass um diese Transformationen erweitern.

3.4. Hypothese

Die Implementierung des hier vorgestellten Konzepts soll später genutzt werden um mit Benutzerstudien die folgende Hypothese zu zeigen:

H1 : Der Schutz vor Passwortklau durch Fettspuren wird durch das Anwenden von geometrischen Transformationen erhöht.

4. Implementierung

Im Folgenden werden der Prototyp namens SmearSafe und einige Details der Implementierung vorgestellt. Zuerst wird der Wizard vorgestellt, mit dem die PIN, das Hintergrundbild des Lockscreens und das Unlockpattern festgelegt wird. Danach wird auf die Komponenten eingegangen, die das Gerät sperren und entsperren, nämlich der Lockservice und der Lockscreen. Abschließend werden noch kurz die Logging Komponente und die Manifest-Datei des Prototyps vorgestellt.

4.1. Wizard

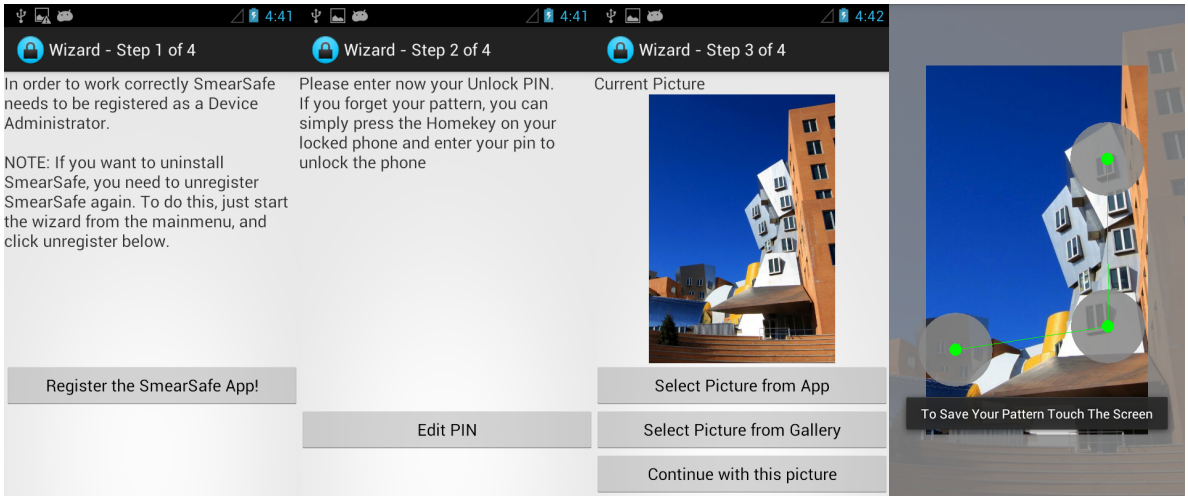

Abbildung 4.1.: Alle vier Schritte des Wizards: (a) Registrierung als Geräteadministrator, (b) PIN anlegen, (c) Bild auswählen und (d) Pattern festlegen

Schritt 1 - Registrierung des Prototyps als Geräteadministrator

Nach dem ersten Start der App wird automatisch der Wizard gestartet um den Lockscreen zu konfigurieren. Im ersten Schritt wird dazu die App mit Hilfe des Device Policy Managers als Geräteadministrator registriert. SmearSafe verwendet von den verfügbaren Policies des Device Policy Managers *reset-password* und *force-lock*. Für eine spätere Deinstallation muss

der Wizard nochmals gestartet werden und die App als Geräteadministrator deregistriert werden, da eine registrierte App nicht deinstalliert werden kann.
Wird der Wizard über das Menü gestartet, wird außerdem der Service, der den Lockscreen erzeugt, beendet. Würde dies nicht passieren und der Wizard vor der Komplettierung abgebrochen, könnte es vorkommen, dass der Lockscreen mit einer falschen Bild-Pattern-Kombination gestartet wird. Es genügt nicht, dass überprüft wird, ob der Wizard vollständig durchlaufen wird und dann gegebenenfalls. der Service beendet wird, da der Aufruf von *stopService()* den betreffenden Service nicht sofort beendet.

Schritt 2 - PIN anlegen

Der Prototyp ist als Overlay für die PIN-Eingabe implementiert. Über die Device Policy Manager API wird, sobald der Bildschirm ausgeht, das Gerät gesperrt und als Passwort die PIN festgelegt, die der Benutzer hier eingibt. Wenn der Bildschirm wieder angeht, ist die PIN-Eingabe unter dem Lockscreen versteckt. Falls der Benutzer seine grafisches Passwort nicht eingeben kann oder vergessen hat, kann er die Home-Taste drücken und damit die PIN-Eingabe wieder zum Vorschein bringen.

Schritt 3 - Bild auswählen

Der Benutzer kann nun ein Hintergrundbild für den Lockscreen festlegen. Dafür kann er ein Bild aus seiner eigenen Gallery wählen, oder eines der Bilder, die die App bereitstellt. Die Bilder, die der Prototyp bereitstellt, werden über einen BaseAdapter in ein GridView geladen, der über den Menüpunkt *Select Picture from App* erreichbar ist.
Um ein Bild aus der Gallery des Telefons auszuwählen, muss der Code aus Listing 4.1 verwendet werden.

Listing 4.1 Java-Code: Auswahl eines Bildes aus der Gallery

```
Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
photoPickerIntent.setType("image/*");
startActivityForResult(photoPickerIntent,GALLERY_PICK);
```

Durch die hohe Vielfalt an Displaygrößen muss das Bild nach dem Auswählen noch an das Telefon angepasst werden. Das geschieht in einem AsyncTask und nicht in der Activity selbst um zu verhindern, dass das Betriebssystem meldet, dass die Anwendung nicht mehr reagiert.
Der AsynchTask speichert eine an die Displaygröße angepasste Version des Bildes im internen Speicher des Telefons, indem er das Bild in ein Bitmapobjekt lädt, die passenden Dimensionen ausrechnet und per *Bitmap.createScaledBitmap()* eine neue Version dieses Bildes erzeugt. Dabei muss man allerdings beachten, dass das Bild zu groß sein kann für das Bitmapobjekt. Um zu verhindern, dass das Programm in diesem Fall abstürzt, sollte man ein Objekt des Typs *BitmapFactory.Options* und die zugehörigen Werte *inJustDecodeBounds* und *inSampleSize* verwenden, so wie in Listing 4.2 dargestellt. Mit Hilfe von *inJustDecodeBounds*

Listing 4.2 Java-Code: Erzeugung eines ans Display angepassten Bilds (vereinfacht)

```
BitmapFactory.Options options = new BitmapFactory.Options();
options.inJustDecodeBounds=true;
Bitmap b = BitmapFactory.decodeStream(input,null,options);
int picture_height = options.outHeight;
int picture_width = options.outWidth;
int sampleSize = 1;
if (picture_height > 2*display_height){
       sampleSize = Math.floor(picture_height / (display_height));
}
options.inJustDecodeBounds = false;
options.inSampleSize = sampleSize;
b = BitmapFactory.decodeStream(input,null,options);
int width = picture_width;
int height = picture_height;
if (picture_height > display_height){
       height = display_height;
       width = Math.rint(picture_width*(height/picture_height));
}
Bitmap c = Bitmap.createScaledBitmap(b, width, height, true);
```

kann nur die Dimensionen des Bildes geladen werden und mit Hilfe von *inSampleSize* kann eine kleinere Version des Bildes in ein Bitmap geladen werden und anschließend die zur Displaygröße passende Version des Bildes berechnet werden. Die Methode *createScaledBitmap()* arbeitet schneller, wenn für *inSampleSize* eine Zweierpotenz angegeben wurde.
Bei allen Geräten des Herstellers *Samsung* muss man zusätzlich beachten, dass die Orientierung des Bildes (Hoch- oder Querformat) aus den EXIF-Informationen des Bildes gelesen wird. Wenn man also auf einem Samsung-Gerät ein Bild in ein Bitmap lädt und das Bitmap dann wieder im Dateisystem speichern will, muss man davor eventuell das Bitmap drehen, um das neue Bild in der gleichen Orientierung zu erhalten. Um die EXIF-Informationen auszulesen, kann die Klasse *ExifInterface* verwendet werden und um die Orientierung auszulesen den Aufruf *getAttribute(ExifInterface.TAG_ORIENTATION)*. Wenn dieser Aufruf den Wert 3, 6 oder 8 zurückliefert, muss das Bitmap noch um 90, 180 oder 270 Grad gedreht werden.

Schritt 4 - Pattern festlegen

Nachdem der Benutzer das Bild ausgewählt hat, kann er nun sein Unlockpattern festlegen. Dazu muss er auf dem sichtbaren Teil des Bildes einen Punkt als Anfang wählen und auf jedem weiteren Punkt den er zum Pattern hinzufügen will kurz verharren. Dort erscheint dann ein kleiner grüner Punkt, der den eigentlichen Passwortpunkt darstellt. Dieser grüne Punkt ist umgeben von einem grauen Schatten, der einen Toleranzbereich darstellt. Wenn der Benutzer in diesem Bereich einen Punkt auswählt, wird dies so gewertet, als ob der Benutzer den Passwortpunkt ausgewählt hätte. Die Größe dieses Toleranzbereich lässt sich im Hauptmenü der App unter *Touchtolerance* verändern. Des Weiteren wird sicher gestellt, dass zwei aufeinander folgende Punkte einen gewissen Abstand zueinander haben und das

Pattern mindestens zwei Punkte beinhaltet.
Für einige Transformationen müssen nach der Eingabe des Patterns noch Grenzwerte berechnet werden, damit die Punkte des Patterns nicht aus dem sichtbaren Bereich des Bildes gedreht oder geschoben werden. Da Patternpunkte am Rand des Bildes einige Transformationen, wie beispielsweise die Verschiebung, nutzlos machen würden, ist es nicht möglich Patternpunkte am Rand des Bildes zu definieren. Der Rand des Bildes ist daher mit einem grauen Schatten gekennzeichnet (siehe Abbildung 3.2). Am oberen und unteren Rand ist der Rand $\frac{1}{8}$ der Bildschirmhöhe und am linken und rechten Rand $\frac{1}{8}$ der Bildschirmbreite.
Um die Grenzwerte für die Verschiebung zu berechnen, werden die minimalen und maximalen X- beziehungsweise Y-Werte ermittelt und mit Hilfe der Bildschirmauflösung die maximal mögliche Verschiebung in alle vier Richtungen ermittelt. Dabei wird zusätzlich darauf geachtet, dass der Punkt nach der Verschiebung nicht auf dem Rand des sichtbaren Bereichs liegt, sondern etwas weiter Richtung Bildmitte, damit man der Punkt in jedem Fall noch ein einem Bereich ist, wo er mit dem Finger erreicht werden kann.
Um die Grenzwerte für die Skalierung zu berechnen, kann man auf die Grenzwerte für die Verschiebung zurückgreifen. Mit diesen Grenzwerten kann man feststellen, welcher dieser Punkte die größte Entfernung zum Mittelpunkt des Displays hat. Wenn man nun diese Entfernung ins Verhältnis setzt mit der Strecke von Mittelpunkt bis zum Rand des Displays hat man den maximalen Skalierungsfaktor. Um den minimalen Faktor auszurechnen werden alle Punkte des Patterns paarweise betrachtet und die minimale Entfernung ins Verhältnis gesetzt mit der doppelten maximalen *Touchtolerance*. Damit wird sichergestellt, dass die Punkte im verkleinerte Pattern noch weit genug auseinander liegen um sie einzeln zu treffen. Sollte die minimale Entfernung kleiner sein als die doppelte maximalen *Touchtolerace*, wird der minimale Skalierungsfaktor auf 1.0 gesetzt.
Die Grenzwerte für die Verschiebung werden ermittelt, indem für jeden Punkt des Patterns der Kreis betrachtet wird, auf dem der Punkt liegt, wenn der Displaymittelpunkt der Kreismittelpunkt ist. Mit Hilfe dieses Kreises wird der Winkel ausgerechnet, um den der Punkt (im Uhrzeigersinn) auf dem Kreis gedreht werden muss, damit er am Bildschirmrand liegt. Hierbei wird auch eine gewisse Toleranz berücksichtigt, damit der Punkt nach der Drehung nicht genau auf dem Displayrand liegt und damit eventuell. nicht mehr richtig erreicht werden kann. Der minimale Wert aller Winkel wird als Grenzwert für das Minimum abgespeichert. Der Grenzwert für das Maximum wird berechnet, indem der Punkt auf dem Kreis gegen den Uhrzeigersinn bewegt wird und das Ergebnis anschließend von 360 abgezogen wird.

4.2. Lockservice

Nachdem der Wizard komplett durchlaufen wurde, kann man im den Lockscreen im Menü aktivieren. Dadurch wird der Lockservice gestartet. Während der Initialisierung des Service werden zwei Broadcast Receiver gestartet. Der eine reagiert auf das Ein- und Ausschalten des Bildschirms und der andere auf Anrufe.
Der Broadcast Receiver für den Bildschirm reagiert auf die Intents *ACTION_SCREEN_ON*, *ACTION_SCREEN_OFF* und *ACTION_USER_PRESENT*. Im Fall dass der Bildschirm ausgeht,

wird nun *ACTION_SCREEN_OFF* empfangen und der Lockservice sperrt das Telefon und startet den Lockscreen, wie in Listing 4.3 dargestellt. Der Lockscreen wird hier schon gestartet, um zu vermeiden, dass wenn der Bildschirm wieder angeht die Android-PIN-Eingabe sichtbar wird. Auf Geräten des Herstellers HTC bewirkt der Code allerdings, dass das Handy ausgeht und dann ein paar Sekunden später wieder angeht. Aus diesem Grund wird auf HTC-Geräten nur das Gerät gesperrt, wenn der Bildschirm ausgeht.

Listing 4.3 Java-Code: Sperren des Telefons

```
DevicePolicyManager deviceManger = getSystemService(Context.DEVICE_POLICY_SERVICE);
deviceManger.resetPassword(mPIN, 0);
deviceManger.lockNow();
if (!android.os.Build.BRAND.equals("htc_europe")){
        Intent start = new Intent(context, ActivityLockScreen.class);
        start.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(start);
}
```

Wenn der Bildschirm wieder angeht, wird der Code, der im Listing 4.3 in der if-Abfrage steht, ausgeführt, um den Lockscreen wieder sichtbar zu machen oder um einen neuen zu erzeugen.
Beim Empfangen des Intents *ACTION_USER_PRESENT* wird versucht das Log an den Server zu übertragen. Außerdem wird das Intent mit Timestamp ins Log eingetragen, um später feststellen zu können, ob die PIN zum entsperren verwendet wurde oder nicht.
Um zu verhindern, dass der Lockscreen während eines Telefonats gestartet wird, reagiert der andere Broadcast Receiver auf den Intent *ACTION_PHONE_STATE_CHANGED* aus der Klasse TelephonyManager und setzt im Falle eines Anrufs ein Flag, damit der Lockscreen nicht gestartet wird.
Wie man sieht bräuchte man nur den Broadcast Receiver um auf die Intents *ACTION_SCREEN_ON und ACTION_SCREEN_OFF* zu reagieren und damit hätte man schon die grundlegende Lockscreenfunktionalität. Allerdings würde man dann jedes Mal, nach einem Neustart des Geräts, zuerst die App öffnen müssen, damit der Receiver gestartet wird und dann der Lockscreen gestartet werden kann. Um dieses Problem zu umgehen, kann man den Broadcast Receiver als Teil eines Services schreiben und diesen Service jedes Mal starten wenn das Gerät hochgefahren wird.
Um einen Service (oder eine Activity) beim hochfahren des Geräts zu starten, muss man einen Broadcast Receiver implementieren, der auf den Intent *BOOT_COMPLETED* (auf HTC-Geräten wird anstatt dessen *QUICKBOOT_POWERON* benötigt!) reagiert, indem er den betreffenden Serivce startet.
Abschließende Bemerkung zu den Intents *ACTION_SCREEN_ON, ACTION_SCREEN_OFF* und *ACTION_USER_PRESENT*: Im Gegensatz zu allen anderen Intents kann man sich für die genannten nur über Programmcode registrieren(wie Listing 4.4 demonstriert) und nicht der AndroidManifest.xml.

Listing 4.4 Java-Code: Receiver für die Bildschirm-Intents korrekt initalisieren

```
IntentFilter filter = new IntentFilter(Intent.ACTION_SCREEN_ON);
filter.addAction(Intent.ACTION_SCREEN_OFF);
filter.addAction(Intent.ACTION_USER_PRESENT);
registerReceiver(mScreenReceiver, filter);
```

4.3. Lockscreen

Damit der Lockscreen überhaupt angezeigt wird, wenn das Handy gesperrt ist, müssen in der Methode *onAttachedToWindow()* die beiden Flags *FLAG_SHOW_WHEN_LOCKED* und *FLAG_FULLSCREEN* gesetzt werden (nur *FLAG_SHOW_WHEN_LOCKED* reicht nicht!). Es ist auch wichtig, dass man nicht aus versehen *addFlags()* verwendet, da sonst eventuell Flags gesetzt bleiben, die verhindern, dass die genannten Flags, das beabsichtigt bewirken. Auf langsameren Geräten muss zusätzlich beachtet werden, dass das Laden des Lockscreens etwas länger dauern kann. Dann kann es passieren, dass der Code, der ausgeführt wird, wenn der Bildschirm angeht, für das System schon abgearbeitet wurde, der Lockscreen aber noch nicht angezeigt wird. In diesem Fall geht der Bildschirm einfach wieder aus. Um das zu verhindern, wird beim Laden des Lockscreens ein *WakeLock* geholt und für zehn Sekunden gehalten. Der *WakeLock* verhindert in diesen zehn Sekunden, dass der Bildschirm ausgeht. Diese Lösung hat auch den Vorteil, dass der Bildschirm ohne Interaktion nach zehn Sekunden wieder ausgeht.
Des Weiteren wird die Android-Statuszeile ausgeblendet, da man über diese (auch bei gesperrtem Gerät) beispielsweise ungelesene SMS lesen kann oder sogar Zugriff auf die kompletten Geräteeinstellungen hat.
Sobald der Lockscreen geladen wird, lädt der Lockscreen alle nötigen Werte aus den Shared Settings, u.a. das Pattern und das Bild, und übergibt sie dem View. Dann wählt der Lockscreen per Zufallszahl aus welche Transformation verwendet wird. Da es mit dem Android-Zufallszahlen-Generator vorkommen könnte, dass zwei Mal hintereinander dieselbe Zahl "gezogen"wird, d.h. zwei Mal hintereinander dieselbe Transformation ausgewählt wird, wurden Gewichte eingesetzt um diese Wahrscheinlichkeit zu minimieren. Die grundlegende Idee ist, das Gewicht der ausgewählten Transformation auf eins zu setzen und alle anderen um eins hochzuzählen. Die nächste gezogene Zufallszahl wird dann aus dem Bereich zwischen null und der Summe der Gewichte sein.
Für einen Teil der Modifikationen müssen zusätzliche Parameter ausgewählt werden, was wiederum über Zufallswerte implementiert ist:

Zoom: Per booleschem Zufallswert wird der minimale Zoomwert oder der maximale ausgewählt. Diese Vereinfachung liegt darin begründet, dass viele Werte dazwischen aussehen, als wäre gar keine Transformation angewendet worden.

Rotation: Zuerst wird die Differenz zwischen minimalem und maximalem Winkel berechnet ($Differenz = 360 - maxWinkel + minWinkel$), dann wird eine durch zehn teilbare Zahl aus dem Intervall $[0; Differenz]$ ausgewählt. Abschließend wird noch entschieden, ob das Bild vor dem addieren des Winkelmodifikators um 180 Grad gedreht wird oder nicht.

Scherung: Bei der Scherung wird ebenfalls per booleschem Zufallswert entschieden, ob die Scherung in X- oder Y-Richtung geschieht.

Verschiebung: Durch eine Zufallszahl aus dem Intervall $[1;4]$ wird entschieden in welche Richtung verschoben wird.

Spiegelung : Für diese Transformation muss kein zusätzlicher Parameter bestimmt werden.

Nach diesem Schritt wird die ausgewählte Transformation an den View übergeben. Der View wendet die ausgewählte Transformation auf das Bild und auf das Pattern an. Danach ist der Lockscreen bereit benutzt zu werden. Bei einem beginnenden Loginversuch, wird zuerst geprüft, ob beim ersten Berühren des Bildschirms, die Toleranzzone des ersten Punkts getroffen wurde. Bei jeder weiteren Bewegung wird geprüft, ob der im Pattern als nächstes kommende Punkt getroffen wird. Wenn der Punkt getroffen wird, wird bei einer neuen Bewegung mit dem nächsten Punkt verglichen. Sobald der Benutzer den Finger im letzten Patternpunkt vom Display nimmt und alle Passwortpunkte davor nacheinander besucht wurden, gilt der Loginversuch als erfolgreich, das Handy wird entsperrt und der Lockscreen beendet sich selbst.
Das Handy wird entsperrt indem das Passwort mit Hilfe des Device Policy Managers auf den leeren String gesetzt wird. Auf den meisten Geräten erscheint, nachdem das Passwort auf den leeren String gesetzt und der Lockscreen beendet wurde, nochmal Androids PIN-Abfrage, die allerdings dann jede Eingabe akzeptiert. Um das zu vermeiden, wird eine Activity aufgerufen, die nur eine Zeile ausführt und sich dann beendet. Diese eine Zeile setzt für das diese Activity das Flag *FLAG_DISMISS_KEYGUARD* und blendet damit die PIN-Eingabe aus.
Der Lockscreen besitzt auch einen Broadcast Receiver, der auf *ACTION_PHONE_STATE_CHANGED* reagiert. Wenn dieser Intent empfangen wird, wird der Lockscreen beendet, damit der eingehende Anruf angenommen werden kann.

4.4. Logging

Pro Loginversuch wird ein Datensatz erzeugt, der die folgenden Daten beinhaltet:

- der aktuelle Timestamp in Millisekunden
- der Erfolg des Loginversuchs
- die aktuelle Transformation mit eventuellen Parametern
- das verwendete Bild als String, wenn es ein Bild aus der App ist. Andernfalls der die Nummer des Custom-Bilds gefolgt von ".jpeg"
- das aktuelle Pattern, kodiert durch eine Nummer
- die aktuelle Touchtolerance
- die aktuelle Anzahl der fehlgeschlagenen Loginversuche
- die aktuelle Anzahl der erfolgreichen Loginversuche

Diese Datensätze werden, bis sie per http an den Server geschickt werden, auf der SD-Karte des Geräts zwischengespeichert. Da das Passwort die Passwortpunkte beinhaltet, wird das Passwort kodiert im Log gespeichert. Jedes Mal wenn ein neues Passwort gesetzt wird, wird ein Passwortzähler erhöht und jedes Mal wenn beim Senden des Logs an den Server der Passwortzähler einen größeren Wert beinhaltet, als die Variable die, angibt wie viele Passwörter übertragen wurden, wird das Passwort/die neuen Passwörter mit den Datensätzen übertragen.
Die App unterstützt es auch, dass Kopien der Bilder, die aus der Gallery des Geräts, an den Server übertragen werden. Jedes Mal wenn ein neues Bild aus der Gallery ausgewählt und an den Bildschirm angepasst wurde, wird eine Kopie dieses angepassten Bilds auf der SD-Karte gespeichert. Bei Senden der Datensätze wird, wieder mittels Zählern, geprüft, ob es ein neues Bild auf der SD-Karte gibt. Wenn es ein neues gibt, wird es mit BASE64 kodiert und an den Server übertragen. Es wird pro Verbindung mit dem Server nur ein Bild übertragen.

4.5. AndroidManifest.xml

In diesem Kapitels wird noch auf einige Details der AndroidManifest.xml eingegangen. Die folgenden Attribute sind zu beachten:

android:installLocation muss auf *internalOnly* gesetzt werden, da der Service sonst nicht gestartet werden kann, wenn das Gerät hochfährt.

android:excludeFromRecents ist bei allen Komponenten auf *true* gesetzt, da man sonst im Taskmanager den Service beenden kann.

android:screenOrientation ist beim Lockscreen und im vierten Schritt des Wizard auf *portrait* gesetzt.

Abschließend wird noch darauf eingegangen wofür welche Permission gebraucht wird:

RECEIVE_BOOT_COMPLETED wird benötigt, um den Service nach dem Booten des Geräts zu starten.

DISABLE_KEYGUARD wird benötigt, um das Anzeigen des Keyguard nach dem entsperren mit dem Lockscreen zu verhindern.

READ_PHONE_STATE wird benötigt, um die Intents empfangen zu können, die ausgelöst werden, wenn das Gerät angerufen wird.

WRITE_EXTERNAL_STORAGE wird benötigt um auf der SD-Karte das temporäre Log und die angepassten Bilder aus der Gallery abzuspeichern.

WAKE_LOCK wird benötigt, um die WakeLocks benutzen zu können, die verhindern das der Bildschirm nach dem anschalten sofort wieder ausgeht.

INTERNET wird benötigt, um die Logs an den Server zu übertragen.

ACCESS_NETWORK_STATE wird benötigt, um zwischen WLAN und 3G-Verbindung unterscheiden zu können.

VIBRATE wird benötigt, um bei einem fehlgeschlagenen Loginversuch Feedback zu geben.

5. Benutzerstudie: Evaluierung der Sicherheit

In diesem Kapitel wird die Studie behandelt, die durchgeführt wurde, um zu überprüfen, ob der entwickelte Prototyp wirklich die Sicherheit erhöht. Zuerst wird die das Vorgehen bei der Studie beschrieben und dann die Ergebnisse vorgestellt.

5.1. Vorgehen

Um zu überprüfen, ob grafische Passwörter kombiniert mit geometrischen Bildtransformationen, wirklich mehr Sicherheit bieten, wurde eine Studie entworfen, in der solche grafischen Passwörter mit den am meisten verwendeten Sicherheitsmechanismen von Android, PIN und Lockpattern, verglichen wurde. In einer Vorstudie wurden die Probanden gebeten, Passwörter anzugeben, die sie selbst auch verwendet würden. In der Studie selbst wurden andere Probanden gebeten, mit Hilfe von hochauflösenden Fotos der Fettspuren das Passwort herauszufinden.

Bedrohungsszenario

In dieser Studie sollte untersucht werden, ob es möglich ist anhand der Fettspur auf dem Gerät das Passwort herauszufinden. Dabei sollte der Worst-Case betrachtet werden. Das bedeutet, dass davon ausgegangen wurde, dass der Angreifer in Besitz des Gerätes ist und das Display vor der Passwort-Eingabe gut geputzt wurde, d.h. nur die Passwort-Fettspur auf dem Display ist. Des Weiteren ist der Angreifer in der Lage, das Display unter optimaler Ausleuchtung zu fotografieren. Da durch die Implementierung der App, die Wahrscheinlichkeit sehr klein ist, dass zweimal hintereinander dieselbe Transformation verwendet wird, wurde davon ausgegangen, dass der Angreifer eine andere Transformation zur Eingabe bekommt, als er aufgenommen hat.

Vorstudie: Generierung der Passwörter

In der Vorstudie wurden vier Probanden (ein männlicher und drei weibliche) im Alter von 22 bis 42 Jahren ($M = 29.25$, $SD = 9.22$) gebeten, sich Passwörter zu überlegen, wie sie sie auch für ihr Telefon verwenden würden. Jeder Proband musste eine vierstellige PIN und ein Android Lockpattern angeben. Anschließend wurde ihnen der Android-Prototyp gezeigt und wie man mit Hilfe des Wizards ein grafisches Passwort erstellt. Dann hat jeder Proband auf sechs vorher ausgewählten Bildern jeweils ein grafisches Passwort erstellt. Am Ende von

Abbildung 5.1.: Aufbau, der genutzt wurde um die Fotos der Fettspuren zu machen

dieser Phase standen dann vier PINs, vier Android Lockpattern und 24 grafische Passwörter zur Verfügung.

Fotos der Fettspuren

Um den die Studie vorzubereiten mussten Fotos, wie in Abbildung 5.2 dargestellt, von den Fettspuren gemacht werden, mit deren Hilfe die Probanden die Passwörter herausfinden sollten. Für die Aufnahmen wurden ein Nexus S (Bildschirmauflösung 800x480), eine Canon EOS 7D und ein 800-Watt-Scheinwerfer für die optimale Ausleuchtung der Fettspuren verwendet. Zuerst wurde mit verschiedenen Kamera- und Scheinwerferwinkeln experimentiert, bis die Winkel gefunden wurden mit denen man die besten Bilder erstellen konnte (siehe Abbildung 5.1). Jedem Passwort wurde dann eine Transformation zugeordnet unter der das Passwort fotografiert wird, dabei wurde darauf geachtet, dass alle Transformationen gleich oft verwendet wurden. Für jedes Bild wurden die folgenden drei Schritte durchgeführt:

1 Das Display wurde gründlich gereinigt.
2 Das Passwort wurde (immer von derselben Person) eingegeben
3 Nachdem es im Versuchsaufbau (an immer derselben Stelle) platziert wurde, wurde ein Foto gemacht.

Abbildung 5.2.: Hochauflösende Bilder der Fettspuren wie sie für die Studie benutzt wurden: Lockpattern, PIN und grafisches Passwort mit Transformation (v.l.n.r.).

Theoretical Password Space

Der Theoretical Password Space (TPS) ist ein Maß für die theoretische Anzahl der Passwörter eines Systems. Im folgenden sollen die TPS-Werte für die Systeme angegeben werden, die in dieser Studie gegenüber gestellt werden. Für die PIN *PIN* (vier Ziffern, zwischen 0 und 9), ist der Wert $TPS_{PIN} = log_2(10^4) \approx 13.29$. Das *Lockpattern* besteht aus mindestens vier Punkten aus einem 3x3 Grid, wobei jeder Punkt nur einmal besucht werden kann. Der TPS-Wert für das Lockpattern ist also mindestens $TPS_{Pattern} = log_2(9 * 8 * 7 * 6) \approx 11.56$ und maximal $TPS_{Pattern} = log_2(9!) \approx 18.47$
Der TPS-Wert für *grafische Passwörter* hängt von der Auflösung des Bildschirms ab und von der Größe des Toleranzbereiches eines Passwortpunktes. In dieser Studie wurde ein Gerät verwendet, dass eine Auslösung von 800*480 hat und ein Toleranzbereich mit 100 Pixel Durchmesser. Das führt zu einem Wert von $TPS_{grafPW} = log_2(32^4) \approx 20.00$. Da es der Prototyp nicht erlaubt an Rand Passwortpunkte zu definieren (siehe Abbildung 3.2) sinkt der Wert auf $TPS_{grafPW} = log_2(18^4) \approx 16.67$.

Durchführung der Studie

Für die Studie wurden alle Passwörter nach der Transformation sortiert mit der das jeweilige Foto gemacht wurde, da auch die empfundene Schwierigkeit der Transformationen untereinander verglichen werden sollte. Um dem Bedrohungsszenario zu entsprechen, wurde für

jedes Foto eine andere Transformation ausgewählt, als die mit der das Foto aufgenommen wurde. Dabei wurde darauf geachtet, dass jede Kombination aus Foto und Transformation gleich oft getestet wurde.
Am zweiten Teil der Studie haben 20 Probanden (15 männliche und 5 weibliche) im Alter zwischen 19 und 54 Jahren ($M = 26.15$, $SD = 7.04$) teilgenommen. Als erstes wurde jedem Proband der Zweck der Studie und das Konzept der grafischen Passwörter erklärt. Dann hat der Probanden einen deomgrafischen Fragebogen ausgefüllt. Danach wurde ihm an einem Beispiel erklärt, wie er anhand der Fotos der Fettspuren die Passwörter knacken sollte. Es wurde auch darum gebeten, sich zu überlegen an welchen Bildfeatures der Passwortersteller sich eventuell orientiert haben könnte.
Anschließend wurde dem Proband erst alle Bilder der Gruppe A gezeigt, dann alle der Gruppe B usw. Es gab keine zeitliche Vorgabe in der ein Passwort mit Hilfe des Fotos zu knacken gab. Es gab nur die Einschränkung, dass man pro Foto nur drei Versuche hat, in das Handy einzubrechen. Nach den vier Fotos einer Gruppe wurde den Probanden gesagt, welche Transformation dieser Gruppe zugrunde lag und sie wurden gebeten einen kleinen Fragebogen zu den letzten vier Fotos auszufüllen. In diesem Fragebogen abgefragt, ob die Probanden der Meinung waren, dass sie mit mehr Versuchen, die Passwörter herausbekommen hätten und wie schwer die letzte Gruppe empfunden wurde. Nach der letzten Gruppe wurden die Probanden gebeten einen letzten Fragebogen ausfüllen, indem sie den Einfluss des Bildes beurteilen sollten und die Transformationen nach Schwierigkeit ordnen sollten. Jeder Teilnehmer wurde zum Schluss mit fünf Euro entschädigt.

5.2. Ergebnis

Zuerst wurde die Sicherheit des in dieser Arbeit untersuchten Ansatzes mit PIN und Lockpattern verglichen. In diesem Vergleich schnitt das Lockpattern am schlechtesten ab (Mittlere Anzahl der erratenen Passwörter pro Proband $M = 3.50$, $SD = 0.51$), gefolgt von der PIN ($M = 1.10$, $SD = 0.91$). Die grafischen Passwörter mit geometrischen Tranformationen schnitten am besten ab ($M = 0.51$, $SD = 0.33$). Diese Werte werden in Abbildung 5.3 dargestellt. Eine Friedman Analysis of Variance (ANOVA) zeigt, dass es statistisch signifikante Unterschiede zwischen den drei Systemen gibt, $\chi^2(2) = 33.50$, $p = .00$. Um dieses Ergebnis weiter zu untersuchen wurden drei Wilcoxon Tests durchgeführt, auf die jeweils eine Bonferroni Korrektur angewandt wurde. Diese Wilcoxon Tests zeigten, dass der in dieser Arbeit vorgestellte Ansatz statistisch gesehen besser ist als das Lockpattern ($Z = -4.03, p = .00, r = -.64$) und besser als die PIN ($Z = -2.62, p = .01, r = -.41$). Außerdem zeigten die Wilcoxon Tests, dass die PIN statistisch gesehen besser ist als das Lockpattern ($Z = -3.87, p = .00, r = -.61$).
Außerdem wurden noch die Wirkungen der Transformationen aufeinander untersucht. Das heißt es wurde untersucht, ob es bestimmte Ausgangstransformationen (Transformationen mit denen die Bilder gemacht wurden) gibt, durch die es schwerer wird die das Passwort in der Zieltransformation (die Transformation die auf dem Gerät angezeigt wird) zu erraten. Das Ziel hierbei war unsichere Kombinationen zu finden, die nach Möglichkeit nicht aufeinander folgen sollten. Dazu wurden zwei Friedman ANOVAs durchgeführt, einmal mit den Ausgangstransformationen und einmal mit den Zieltransformationen. Die Ergebnisse sind in

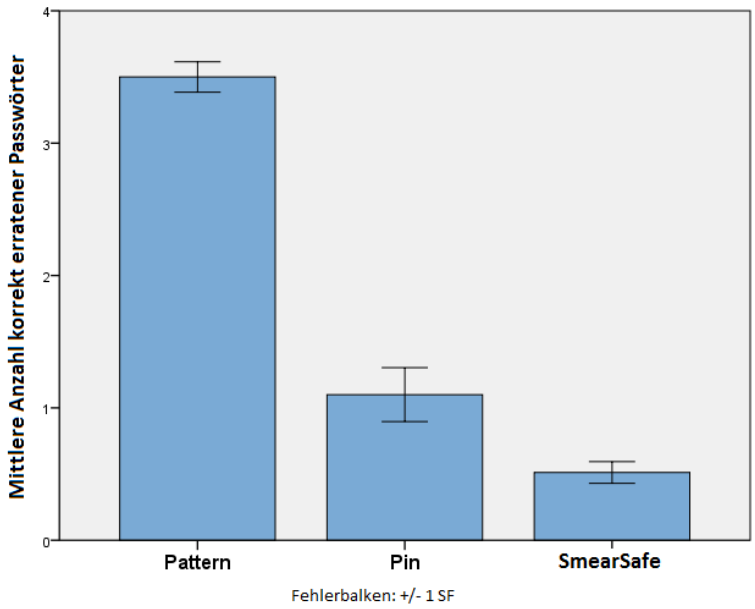

Abbildung 5.3.: Vergleich Pattern vs. PIN vs. SmearSafe: Mittlere Anzahl der erratenen Passwörter

Abbildung 5.4 dargestellt. Abbildung 5.5 zeigt die Erfolgsraten jeder Transformationskombination in Prozent. Man sieht, dass in 87,5% der Fälle das Lockpattern mit maximal drei Versuchen erraten wurde. Die PIN wurde in 27,5% aller Fälle erraten. Wenn man den hier vorgestellen Ansatz betrachtet, sieht man, dass die schlechteste Kombination, diejenige ist die als Ausgangstransformation die Verschiebung und als Zieltransformation die Skalierung hat. Diese Kombination wurde in 30% der Fälle erraten. Die beste Kombination des Ansatzes ist Rotation-Skalierung. Diese Kombination wurde nie erraten. Eine Friedman ANOVA mit paarweisen Vergleichen der Transformationen durch Bonferroni-korrigierte Wilcoxon-Tests ergab keinen statistisch signifikanten Unterschied.
Abbildung 5.6 zeigt Einschätzung der Probanden über die Sicherheit der einzelnen Transformationen beziehungsweise Verfahren. Die Probanden empfanden das Lockpattern am unsichersten, gefolgt von der PIN. Die als am sichersten empfundenen Transformationen sind Spiegelung und Scherung und die Transformation, die am schlechtesten bewertet wurde ist die Verschiebung. Jeder Proband wurde am Ende der Studie gebeten ein solches Ranking aufzustellen.
Die Studie hat somit gezeigt, dass die Hypothese H1, die besagt, dass der Schutz vor Passwortdiebstahl durch Fettspuren durch das verwenden von geometrischen Transformationen erhöht wird, stimmt.

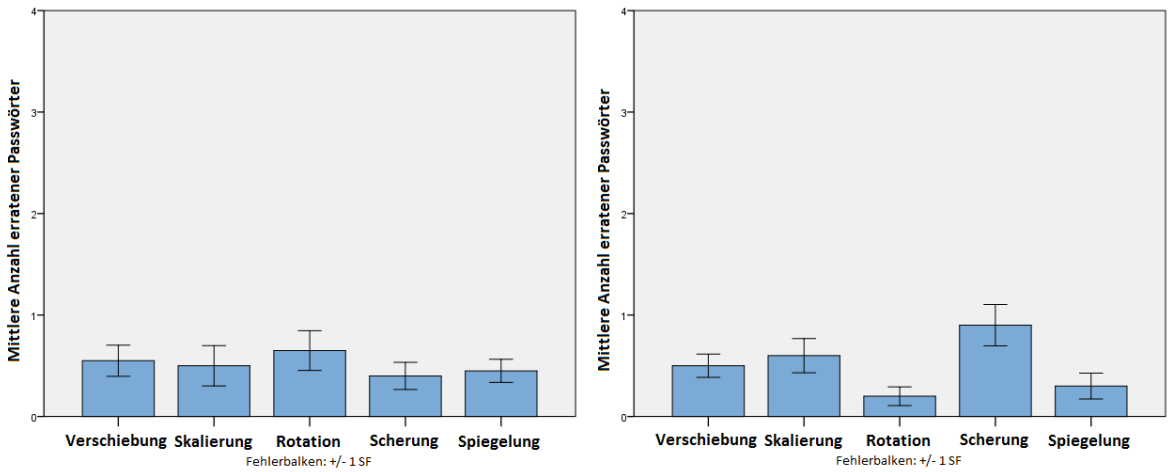

Abbildung 5.4.: Mittlere Anzahl korrekt erratener Passwörter. Linkes Diagramm: Ausgangstransformation, rechtes Diagramm: Zieltransformation

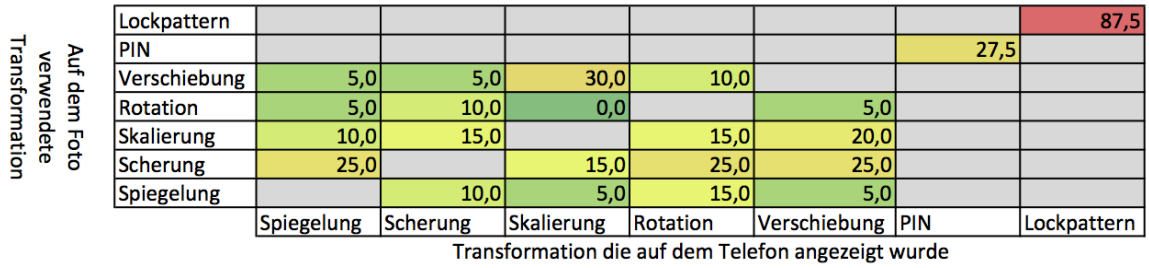

Abbildung 5.5.: Übersicht über alle erratenen Passwörter in Prozent

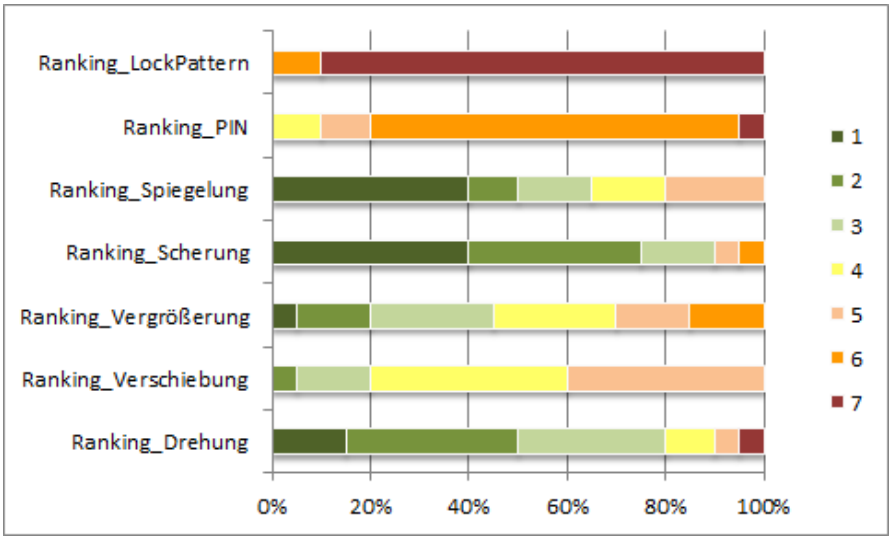

Abbildung 5.6.: Übersicht über das Sicherheitsranking der Probanden

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

In dieser Arbeit wurde ein Prototyp eines Authentifizierungssystems entwickelt, der es schwieriger machen soll, mit Hilfe von Fettspuren Passwörter herauszufinden. Der Prototyp implementiert ein grafische Passwort der Kategorie cued-recall System. Der Benutzer kann dabei auf einem frei wählbaren Bild ein Passwort definieren, dass aus mindestens zwei Punkten bestehen muss. Es werden geometrische Transformationen wie Verschiebung, Rotation, Spiegelung, Skalierung und Scherung verwendet um das dem Passwort zugrunde liegende Bild zu transformieren und damit die Eingabe bei jedem Loginversuch in einer anderen Lage anzeigen zu können. In einer mit 20 Probanden durchgeführten Studie, mussten die Probanden mit Hilfe von hochauflösenden Fotos von Fettspuren das Passwort herausfinden und eingeben. In der Studie wurde der Prototyp, verglichen mit den weitverbreiteten Verfahren Lockpattern und PIN. Das Ergebnis der Studie bestätigt, dass grafische Passwörter in Verbindung mit geometrischen Transformationen sicherer sind als PIN und Lockpattern. Bei der Studie wurde auch untersucht, ob es Transformationen gibt, die nicht aufeinander folgen dürfen, da sie eventuell das Passwort erraten begünstigen. Zwar gab es in der Studie eine Kombination, die am schlechtesten abgeschnitten hat, aber statistische Tests haben keinen signifikanten Unterschied zwischen den Kombinationen gefunden.

6.2. Ausblick

In diesem Prototyp wurden einfache geometrische Transformationen verwendet, um das grundlegende Konzept zu testen und vorzustellen. In Zukunft könnte man auch komplexere Transformationen wie beispielsweise einen Fisheye-Effekt zurückgreifen. Damit der Fisheye-Effekt aber jedes Mal dynamisch berechnet werden kann, wird man eine Grafikbibliothek zu Hilfe nehmen müssen, da die Berechnung eines Fisheye-Effekts mit der Android API lange dauert. Es könnten aber auch Transformationen untersucht werden die nicht geometrisch sind. Beispielsweise könnte ein System untersucht werden, dass nach der Passworteingabe das Bild wie ein Grid aufteilt und für einen Loginversuch die Teile dieses Grids vermischt. Die Kombination von Transformationen, die hier nicht gemacht wurde, könnte auch Thema einer künftigen Arbeit sein. Interessant wären hierbei die Auswirkungen auf Sicherheit und Usability. Sobald es Geräte mit autoskereoskopischen Displays gibt könnte auch untersucht werden, wie der hier vorgestellte Ansatz im dreidimensionalen Raum anwenden lässt.
Zum andern könnte der vorgestellte Ansatz sehr einfach auf andere schon Systeme angewendet werden, da geometrische Transformationen sehr leicht zu implementieren sind und

wenig Rechenzeit brauchen. Interessant sind hierbei auch Systeme, die im Gegensatz zum hier vorgestellten Prototyp auf dynamische Hintergründe setzen. VidPass ist beispielsweise ein System beim dem auf kurzen Videos ebenfalls ein Passwort durch auswählen einzelner Passwortpunkte definiert wird [PSFW12]. Generell kann man sagen, dass der hier vorgestellte Ansatz auf alle Systeme portierbar ist, die es erlauben Passwortpunkte auf Bildern zu definieren.
Man könnte auch andere schmiersichere Passwortsysteme implementieren und untersuchen. Beispielsweise könnte man ein System implementieren, bei dem sich Objekte auf dem Bildschirm bewegen. Man muss dann die passenden Objekt auswählen oder verbinden. Es ist auch vorstellbar ein System zu entwerfen, dass sich beim Loginverfahren bestimmte Sensoren des Geräts ausliest. Beispielsweise ein System, bei dem man einen Finger auf eine bestimmte Stelle legen muss und dann das Gerät in verschiedene Richtungen bewegen muss. Allerdings sollte man darauf achten, dass in zukünftigen Systemen nicht der ganze Bildschirm verwendet wird. Auf diesem Teil des Bildschirms könnte man dann die Uhrzeit oder die Zahl der verpassten Anrufe anzeigen und Smartphonebenutzer haben sich daran gewöhnt schnell einen Blick darauf werfen zu können.

6.3. Fazit

Es wurde gezeigt, dass hier vorgestellte Ansatz, in dem auf grafische Passwörter zufällig ausgewählte geometrische Transformationen angewendet werden, sicherer ist als weitverbreitete Loginmechanismen wie PIN und Lockpattern. Da die verwendeten Transformationen einfach zu berechnen, kann man ihn auch leicht auf andere Loginverfahren anwenden, solang sie auch auf einer Menge von Passwortpunkte basieren.

A. Anhang

A.1. Fragebogen zur Benutzerstudie über die Sicherheit

Information

Vielen Dank für die Teilnahme an unserer Benutzerstudie! In dieser Studie wollen wir untersuchen, wie sicher verschiedene Passwortsysteme sind. Hierbei betrachten wir insbesondere die Bedrohung durch Fettspuren auf dem Handydisplay.

Fragebogen

A. Demographie
Geschlecht: [ ] männlich [ ] weiblich
Alter:
Beruf / Studiengang:

B. Mobiltelefonssicherheit
Welches Authentifizierungsverfahren nutzen Sie?
o Pin
o Android Lock-Pattern
o Keins
o Passwort
o Ein anderes:

Wie lang ist Ihr Passwort?

Für wie wichtig halten Sie Passwörter auf Mobiltelefonen
Gar nicht wichtig - Sehr wichtig
o o o o o

Und warum?

Nach jeder Bildergruppe bekamen die Probanden den folgenden Zwischenfragebogen (insgesamt acht Mal):
C. Bedrohungs-Scenario
Es war einfach für mich das Passwort anhand der Fettspur herauszufinden.

Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

Mit mehr als drei Versuchen hätte ich das Passwort herausgefunden.
Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

Der Erfolg hing vom Bild ab.
Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

D. Abschlussfragebogen
Die Sicherheit der grafischen Passwörter hing stark vom Bild ab.
Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

Ich konnte durch Bildfeatures (markante Punkte in den Bildern) auf das Pass-wort schließen.
Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

Ich würde ein System, das grafische Transformationen benutzt nutzen.
Ich stimme voll zu - Ich stimme gar nicht zu
o o o o o

Bitte stellen Sie eine Rangliste mit Hinblick auf die Sicherheit der Passwörter auf. Fangen Sie bei dem sichersten an (1) bis zum unsichersten (8).
Keine Transformation:
Drehung:
Verschiebung:
Vergrößerung:
Scherung:
Spieglung:
PIN:
Lock Pattern:

Literaturverzeichnis

[AA11] K. AlRowaily, M. AlRubaian. Oily Residuals Security Threat on Smart Phones. In *Proc. of the 1st International Conference on Robot, Vision and Signal Processing*, S. 300–302. 2011. doi:10.1109/RVSP.2011.92. (Zitiert auf Seite 11)

[AGM+10] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, J. M. Smith. Smudge attacks on smartphone touch screens. In *Proc. of the 4th USENIX Conference on Offensive Technologies*, S. 1–7. 2010. URL http://dl.acm.org/citation.cfm?id=1925004.1925009. (Zitiert auf den Seiten 11 und 15)

[ATOY13] P. Andriotis, T. Tryfonas, G. Oikonomou, C. Yildiz. A pilot study on the security of pattern screen-lock methods and soft side channel attacks. In *Proc. of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, S. 1–6. 2013. doi:10.1145/2462096.2462098. URL http://doi.acm.org/10.1145/2462096.2462098. (Zitiert auf Seite 11)

[BAS12] A. Bulling, F. Alt, A. Schmidt. Increasing the security of gaze-based cued-recall graphical passwords using saliency masks. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, S. 3011–3020. 2012. doi:10.1145/2207676.2208712. URL http://doi.acm.org/10.1145/2207676.2208712. (Zitiert auf den Seiten 10 und 19)

[CFSW12] E. Chin, A. P. Felt, V. Sekar, D. Wagner. Measuring user confidence in smartphone security and privacy. In *Proc. of the 8th Symposium on Usable Privacy and Security*, S. 1:1–1:16. 2012. doi:10.1145/2335356.2335358. URL http://doi.acm.org/10.1145/2335356.2335358. (Zitiert auf Seite 10)

[DLHB+12] A. De Luca, A. Hang, F. Brudy, C. Lindner, H. Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, S. 987–996. 2012. doi:10.1145/2207676.2208544. URL http://doi.acm.org/10.1145/2207676.2208544. (Zitiert auf Seite 11)

[DMR04] D. Davis, F. Monrose, M. K. Reiter. On user choice in graphical password schemes. In *Proc. of the 13th USENIX Security Symposium*, S. 11–11. 2004. URL http://dl.acm.org/citation.cfm?id=1251375.1251386. (Zitiert auf Seite 9)

[DP00] R. Dhamija, A. Perrig. Déjà Vu: a user study using images for authentication. In *Proc. of the 9th USENIX Security Symposium*, S. 4–4. 2000. URL http://dl.acm.org/citation.cfm?id=1251306.1251310. (Zitiert auf Seite 9)

[DVKF10] T. Dörflinger, A. Voth, J. Krämer, R. Fromm. "My Smartphone is a Safe! The User's Point of View Regarding Novel Authentication Methods and Gradual Security Levels on Smartphones. S. 155–164. 2010. URL http://dblp.uni-trier.de/db/conf/secrypt/secrypt2010.html#DorflingerVKF10. (Zitiert auf Seite 11)

[DY07] P. Dunphy, J. Yan. Do background images improve "draw a secret"graphical passwords? In *Proc. of the 14th ACM Conference on Computer and Communications Security*, S. 36–47. 2007. doi:10.1145/1315245.1315252. URL http://doi.acm.org/10.1145/1315245.1315252. (Zitiert auf Seite 10)

[Kel13] L. Kelion. Google facial password patent aims to boost Android security. BBC News – Technology, 2013. http://www.bbc.co.uk/news/technology-22790221. (Zitiert auf Seite 7)

[MBK+12] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, K. Beznosov. Understanding Users' Requirements for Data Protection in Smartphones. In *Proc. of the 28th IEEE International Conference on Data Engineering*, S. 228–235. IEEE, 2012. (Zitiert auf Seite 10)

[MBK+13] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, K. Beznosov. Know your enemy: the risk of unauthorized access in smartphones by insiders. In *Proc. of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, S. 271–280. 2013. doi:10.1145/2493190.2493223. URL http://doi.acm.org/10.1145/2493190.2493223. (Zitiert auf den Seiten 7 und 10)

[ML07] W. Moncur, G. Leplâtre. Pictures at the ATM: exploring the usability of multiple graphical passwords. In *Proc. of the SIGCHI International Conference on Human Factors in Computing Systems*, S. 887–894. 2007. doi:10.1145/1240624.1240758. (Zitiert auf Seite 9)

[OB12] I. Oakley, A. Bianchi. Multi-touch passwords for mobile device access. In *Proc. of the ACM Conference on Ubiquitous Computing*, S. 611–612. 2012. doi:10.1145/2370216.2370329. URL http://doi.acm.org/10.1145/2370216.2370329. (Zitiert auf Seite 11)

[PS90] R. A. Peters, R. N. Strickland. Image complexity metrics for automatic target recognizers. In *Automatic Target Recognizer System and Technology Conference*, S. 1–17. 1990. (Zitiert auf Seite 20)

[PSFW12] S. Penninger, T. Schneidermeier, H. Federrath, C. Wolff. VidPass Passwörter in Raum und Zeit. Zur Usability von Videopasswörtern. In H. Reiterer, O. Deussen, Herausgeber, *Mensch & Computer 2012: interaktiv informiert allgegenwärtig und allumfassend!?*, S. 273–282. Oldenbourg Verlag, München, 2012. (Zitiert auf den Seiten 20 und 40)

[SWKW13] F. Schaub, M. Walch, B. Könings, M. Weber. Exploring the design space of graphical passwords on smartphones. In *Proc. of the 9th Symposium on Usable*

Privacy and Security, S. 11:1–11:14. 2013. doi:10.1145/2501604.2501615. URL http://doi.acm.org/10.1145/2501604.2501615. (Zitiert auf Seite 9)

[TO07] J. Thorpe, P. C. van Oorschot. Human-seeded attacks and exploiting hot-spots in graphical passwords. In *Proc. of 16th USENIX Security Symposium*, S. 8:1–8:16. 2007. URL http://dl.acm.org/citation.cfm?id=1362903.1362911. (Zitiert auf Seite 10)

[WWB+05] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, N. Memon. PassPoints: design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies*, 63(1-2):102–127, 2005. doi:10.1016/j.ijhcs.2005.04.010. URL http://dx.doi.org/10.1016/j.ijhcs.2005.04.010. (Zitiert auf Seite 10)

[ZKDLH13] E. von Zezschwitz, A. Koslow, A. De Luca, H. Hussmann. Making graphic-based authentication secure against smudge attacks. In *Proc. of the International Conference on Intelligent User Interfaces*, S. 277–286. 2013. doi:10.1145/2449396.2449432. URL http://doi.acm.org/10.1145/2449396.2449432. (Zitiert auf den Seiten 7, 11 und 15)

Alle URLs wurden zuletzt am 06. 10. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift