Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Jan Wienholz

Entwicklung einer gestenbasierten Authentifikation für Android

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Jan Wienholz

Entwicklung einer gestenbasierten Authentifikation für Android

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung

im Studiengang Informations- und Elektrotechnik

am Department Informations- und Elektrotechnik

der Fakultät Technik und Informatik

der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Robert Fitz

Zweitgutachter : Prof. Dr.-Ing. Karl-Ragmar Riemschneider

Abgegeben am 21. Februar 2014

Jan Wienholz

Thema der Bachelorthesis

Entwicklung einer gestenbasierten Authentifikation für Android

Stichworte

Authentifikation, Gesten, Android, Smartphone, Touchscreen

Kurzzusammenfassung

Mobile Geräte wie Smartphones und Tablets enthalten immer mehr persönliche Daten. Der Zugriff auf diese Daten durch Unbefugte, sowie der Missbrauch von Telefonfunktionen soll verhindert werden. Dies geschieht durch eine Authentifikation des Geräteinhabers durch eine persönliche Geste. Eine Anwendung für das Betriebssystem Android, die eine gestenbasierte Authentifikation ermöglicht, wurde mit dieser Arbeit entwickelt.

Jan Wienholz

Title of the paper

Development of gesture based authentification for Android.

Keywords

Authentification, Gesture, Android, Smartphone, Touchscreen

Abstract

On mobile devices such as Smartphones and Tablets are stored more and more personal data. To prevent the access to this data and the phone functions is the aim of the thesis. This will happen with an authentification by the owner of the device with a personal gesture. This thesis describes the development of a gesture-based authentification for Android.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Anz. – Anzahl

AWT – Abstract Window Toolkit

GPS – Global Positioning System

HSDPA – High Speed Downlink Packet Access

SD-Speicherkarte – Secure Digitial-Speicherkarte

SDK – Software Development Kit

SMS – Short Message Service

WLAN – Wireless Local Area Network

1 Einleitung

In den letzten Jahren gibt es einen stetig wachsenden Markt von mobilen Geräten mit Touchscreen. Die größte Gruppe davon sind tragbare Geräte, wie Smartphones und Tablets, die von den Nutzern mittlerweile nahezu überall mit hingenommen werden. Begünstigt wird dieses rasante Marktwachstum durch die fortschreitende technologische Entwicklung auf diesem Gebiet. Die Geräte werden laufend verbessert, erhalten schnellere Prozessoren und größere Speicher. Für jeden erdenklichen Zweck gibt es Anwendungen, sodass nahezu jeder Nutzer eine entsprechende, für ihn nützliche Verwendung finden kann. Die wachsenden Möglichkeiten und Datenspeicher, die mobile Geräte bieten, veranlassen die Benutzer auch immer mehr persönliche Daten wie z.B. Bilder, Adressen oder Zugangsberechtigungen für Onlineportale auf dem Endgerät zu speichern. Daraus resultiert ein steigender Anspruch an Datensicherheit.

Diese Bachelorthesis beschäftigt sich mit der Entwicklung einer gestenbasierten Authentifikation für das Betriebssystem Android. Durch die Anwendung für mobile Geräte mit Touchscreen wird dem Nutzer die Möglichkeit zur Einrichtung eines Sperrbildschirmes gegeben. Dieser Sperrbildschirm lässt sich nur mit der Eingabe einer eigenen Geste des Benutzers umgehen. Dadurch werden persönliche Daten auf dem internen Telefonspeicher durch unbefugten Zugriff Dritter geschützt. Außerdem können Telefonfunktionen nicht genutzt werden. Ein weiteres Ziel bei der Entwicklung der Anwendung ist das Schaffen von mehr Komfort für den Nutzer. So wird das Merken von lästigen Passwörtern oder Zahlenkombinationen überflüssig. Der Benutzer kann sich seine individuelle Geste für die Authentifikation des mobilen Gerätes frei wählen.

In den folgenden Abschnitten wird, nach einer Grundlageneinführung, ein Einblick auf die zur Verfügung stehende Hardware sowie Software und Entwicklungsumgebung gegeben. Anschließend werden die, in der Anwendung implementierten Authentifikationsverfahren erläutert und im Speziellen wird auf die entwickelten Algorithmen mit ihren entsprechenden Parametern eingegangen. Nach der Erläuterung des Testverfahrens und dessen Ergebnisse wird ein

Ausblick auf die Fortführung des Projektes gewährt und ein Fazit nach Beendigung der Arbeit gezogen.

2 Grundlagen

Zur grundlegenden Verständlichkeit der Bachelorthesis werden zunächst wichtige Begriffe definiert und ein kleiner Einblick in die Android-Programmierung mit einigen wichtigen Bestandteile gegeben.

2.1 Begriffsdefinitionen

In diesem Abschnitt werden die Begriffe, die das Thema der Bachelorthesis prägen, genauer beleuchtet.

2.1.1 Geste

Laut dem deutschen Duden versteht man unter dem Begriff Geste eine „[…] spontane oder bewusst eingesetzte Bewegung des Körpers, besonders der Hände und des Kopfes […]“.[1] Die Geste, die in dieser Arbeit gemeint ist, ist entsprechend der Definition eine bewusst eingesetzte Bewegung des Menschen, speziell der Hand bzw. eines Fingers.

Eine weitere Definition hinsichtlich einer Interaktion zwischen Mensch und Computer besagt: „Eine Geste ist eine Bewegung eines Körpers, die Informationen enthält.“[2] Ein reproduzierbarer Informationsgehalt einer Geste bildet die Grundlage des Authentifikationsverfahrens.

Der Touchscreen eines mobilen Gerätes (Smartphone, Tablet, etc.) dient zur Aufnahme der bewussten Bewegung des menschlichen Fingers. Die

[1] [4] Duden. Berlin: Bibliographisches Institut GmbH, 2013, URL: www.duden.de/rechtschreibung/Geste (Stand: 09.12.2013, 16:30 Uhr)
[2] [11] Kurterbach, G.; Hulteen, E.A.; The Art of Human-Computer Interface Design; S. 309-317, 1990

gespeicherten Informationen dieser Bewegung werden in dieser Arbeit als Geste bezeichnet.

2.1.2 Authentifikation

Unter Authentifikation im Sinne der Informationstechnologie „[…] versteht man die Aufgaben- und Benutzerabhängige Zugangs- und/oder Zugriffsberechtigung.“[3] Eine Authentifikation schützt Systeme, Systemfunktionen und Geräte vor Missbrauch durch Außenstehende. Man unterscheidet zwischen einseitiger und zweiseitiger Authentifikation. Das Ergebnis dieser Arbeit, die gestenbasierte Authentifikation, stellt eine einseitige Authentifikation dar. Lediglich der Benutzer muss sich anhand seiner individuellen Gesten beim mobilen Gerät authentifizieren, jedoch nicht das Gerät bei seinem Benutzer.

2.1.3 Android

Android ist ein Betriebssystem, das von Google entwickelt und auf den Markt gebracht wurde. Es ist speziell für mobile Geräte, die durch einen Touchscreen bedient werden, geeignet. Android basiert auf Linux und ist vollständig in Java programmiert, was Entwicklern viele Möglichkeiten der Anwendungsentwicklung bietet.[4]

Google selbst bezeichnet Android als „Die beliebteste mobile Plattform der Welt“.[5] Mittlerweile gibt es Android-Geräte in jeder Größe und Preiskategorie, sodass für jeden Geschmack etwas dabei ist. Geschmack spielt für die Android-Entwickler sowieso eine große Rolle, da jede Android-Version nach einem amerikanischen

[3] [9] ITWissen. Das große Online-Lexikon für Informationstechnologie. Peterskirchen: DATACOM Buchverlag GmbH, URL: www.itwissen.info/definition/lexikon/Authentifizierung-authentication.html (Stand: 09.12.2013, 16:10 Uhr)

[4] Vgl. [14] Ratgeber. Tipps und Tricks: Was genau ist Android? URL: http://www.ratgeber.org/207-was-genau-ist-android.html, (Stand: 06.01.2014, 15:00 Uhr)

[5] [7] Einführung in Android. URL: http://www.android.com/intl/de/about/ (Stand: 06.01.2014, 15:00 Uhr)

Dessert benannt ist (1.5 Cupcake, 1.6 Donut, 2.0 Eclair, 2.2 Froyo, 2.3 Gingerbread, 3.x.x Honeycomb, 4.0.x Ice Cream Sandwich, 4.1.x Jelly Bean).[6]

Tatsächlich setzt sich Android mehr und mehr in der Welt der mobilen Geräte durch. In den USA hat Android den großen Konkurrenten von Apple und seinem Betriebssystem IOS bereits überholt. Ein Grund dafür ist die freie Verwendung des Codes. Mobiltelefonhersteller wie Samsung, Motorola und LG zahlen keine Lizenzgebühren an Google und können das Betriebssystem an ihre Geräte individuell anpassen.[7]

2.2 Androidprogrammierung

Um die Realisierung der gestenbasierten Authentifikation für das Betriebssystem Android zu verstehen, ist eine kurze Einführung in die wichtigsten Begrifflichkeiten der Androidprogrammierung erforderlich. In den folgenden Abschnitten wird eine kurze und prägnante Grundlageneinführung gegeben.

2.2.1 Activity

Im Zusammenhang mit Android spricht man bei einer Activity von einer Benutzeroberfläche, welche, die für den Nutzer sichtbaren Bestandsteile (z.B. Buttons, Eingabefelder, Texte) implementiert. Dementsprechend ist eine Activity in den meisten Fällen mit einer Layoutdatei verbunden. Eine Activity ist verantwortlich für die Kommunikation zwischen Anwender und Programm. Bei

[6] Vgl. [10] Kuhn, Daniel: Android: Von Cupcake bis Ice Cream Sandwich — alle Versionen in der Übersicht. 2011, URL: http://www.androidnext.de/schwerpunkt/android-von-cupcake-bis-ice-cream-sandwich-alle-versionen-in-der-ubersicht/ (Stand: 06.01.2014, 15:20 Uhr)

[7] Vgl. [2] Awuku, Yaw: Was ist eigentlich Android? URL: http://www.t-online.de/handy/smartphone/id_49385388/was-ist-android-google-android-klipp-und-klar-.html (Stand: 06.01.2014, 15:50 Uhr)

komplexen Anwendungen können mehrere Activitys miteinander verbunden werden.[8]

2.2.2 Service

Ein Service wird verwendet, um Hintergrundprozesse zu erledigen. So können aufwendige Rechenoperationen unbemerkt im Hintergrund ablaufen, auch dann noch, wenn die Bedienoberfläche geschlossen wurde. Ein Anwendungsbeispiel hierfür ist das Abspielen von Musik, ohne dass sich der dazugehörige Player im Vordergrund befindet.[9] Außerdem ist es, neben vielen weiteren Anwendungen, möglich, in einem Service dynamische Broadcastreceiver zu registrieren.

2.2.3 Intent

Mit Intents lassen sich die verschiedenen Komponenten (Activity, Service, Broadcast Receiver) einer Anwendung, aber auch Anwendungen unter sich miteinander verbinden. Die häufigste Verwendung von Intents ist das Starten einer Activity aus einer anderen Activity heraus. Dabei können Nachrichten und Daten zwischen den Komponenten ausgetauscht werden. Sogenannte Broadcast Intents (o.a. Systemnachrichten) schaffen eine Verbindung von der Anwendung zum Android-Betriebssystem.[10]

2.2.4 Broadcastreceiver

Broadcastreceiver dienen der Kommunikation zwischen Anwendung und Betriebssystem. Sie sind für den Empfang von Broadcast Intents zuständig. Das Betriebssystem sendet verschiedene Broadcast Intents aus, zum Beispiel bei einem schwachen Akkustand, Störungen der Netzwerkverbindungen oder nach

---

[8] Vgl. [3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung: Heidelberg: dpunkt Verlag GmbH, 2009, S. 5

[9] Vgl. [3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung: Heidelberg: dpunkt Verlag GmbH, 2009, S. 20-21

[10] Vgl. [3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung: Heidelberg: dpunkt Verlag GmbH, 2009, S. 93

vollständigem Bootvorgang des Betriebssystems. Eine Anwendung kann eigene Broadcast Intents verschicken, somit ist auch eine Verbindung zwischen verschiedenen Anwendungen möglich.[11]

2.2.5 Shared Preferences

Mit Hilfe von Shared Preferences lassen sich primitive Datentypen abspeichern und wieder abrufen. Dabei werden sie in Schlüssel-Wert-Paaren hinterlegt. Um die Daten also wieder abrufen zu können, muss der Schlüssel bekannt sein. Die Daten bleiben erhalten, wenn die Anwendung geschlossen wird.[12] Aus der Erfahrung lässt sich feststellen, dass die Daten mit der Deinstallation nicht mehr zur Verfügung stehen.

3 Hardware

Zum Testen der Anwendung zur gestenbasierten Authentifikation wurde das Smartphone Samsung Galaxy 3 I5800 zur Verfügung gestellt. Das Samsung Galaxy 3 kam im Jahr 2010 mit dem Betriebssystem Android 2.1 auf den Markt. Durch ein Update der Firmware ließ sich jedoch das Betriebssystem auf Android 2.2 aktualisieren. Die Entwicklung der gestenbasierten Authentifikation ist dementsprechend spezialisiert für Android 2.2. Das Smartphone besitzt ein TFT-Farbdisplay Touchscreen mit einer Bildschirmdiagonale von 8,1 cm bzw. 3,2 Zoll und einer Auflösung von 240 x 400 Pixel. Der interne Telefonspeicher hat eine Größe von 170 MB. Die Speicherkapazität lässt sich durch eine microSD Karte mit bis zu 32 GB erweitern. Das Samsung Galaxy 3 ist mit einem Gyrosensor ausgestattet. Dieser ermöglicht die Auswertung der aktuellen Lage des Smartphones in X-, Y- sowie Z-Richtung. Darüber hinaus besitzt das Samsung Galaxy 3 viele weitere Funktionalitäten wie WLAN, Bluetooth, HSDPA, GPS, mp3-

[11] Vgl. [3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung: Heidelberg: dpunkt Verlag GmbH, 2009, S. 21

[12] Vgl. [16] Storage Options. URL: http://developer.android.com/guide/topics/data/data-storage.html, (Stand: 04.12.2013, 15:55 Uhr)

Player, die für die Entwicklung der gestenbasierten Authentifikation aber keine große Rolle spielen.

4 Software und Programmierungsumgebung

Für die Erstellung der Anwendung zur gestenbasierten Authentifikation für Android wurden einige Tools und Hilfsmittel verwendet. Auf diese und die verwendete Programmiersprache Java wird im Folgenden eingegangen.

4.1 Java als Programmiersprache

Die Programmierung der Anwendung für Android geschieht mit der plattformunabhängigen, objektorientierten Programmiersprache Java.

Java bildet die Grundlage für jegliche Art von Netzwerkanwendungen und ist ein globaler Standard für die Entwicklung und Bereitstellung von mobilen Anwendungen, Spielen und Web-basierten Inhalten. Mit mehr als neun Millionen Entwicklern auf der ganzen Welt bietet die effiziente Java-Entwicklung viele Anwendungen und Dienste.[13] Die US-Amerikanische Firma Oracle stellt sämtliche Java-Plattformen für die Entwicklung zur Verfügung.

4.2 Eclipse

Eclipse ist eine Open-Source Java-Entwicklungsumgebung. Es ermöglicht das Erzeugen, Starten und Debuggen von Anwendungen. Eclipse ist keine abgeschlossene Anwendung, sondern wurde so entworfen, dass sie sich ständig mit weiteren Tools erweitern und verbessern lässt.[14] Das Eclipse-Projekt wurde

[13] Vgl. [13] Oracle: Why Java? URL: http://www.oracle.com/us/technologies/java/overview/index.html (Stand: 16.12.2013, 11:30 Uhr)

[14] Vgl. [6] Eclipse: FAQ What is Eclipse? URL: http://wiki.eclipse.org/FAQ_What_is_Eclipse%3F (Stand: 20.12.2013, 11:45 Uhr)

ursprünglich im Jahr 2001 von IBM ins Leben gerufen und fand in einer Reihe von Software-Anbietern Unterstützung.[15]

Für die Entwicklung dieses Projektes wurde die Eclipse-Plattform Version 4.2.1 genutzt.

4.3 Android SDK

Das Android Software Development Kit (Android SDK) stellt alle notwendigen Bibliotheken und Entwicklungswerkzeuge für das Erstellen, Testen und Debuggen von Anwendungen für Android zur Verfügung.[16] Dabei ist es auf die geringeren Ressourcen von Smartphones und Tablets hinsichtlich Speicher und Prozessorleistung optimiert. Dementsprechend bietet das Android SDK nicht den kompletten Funktionsumfang der Java SDKs. Unter anderem fehlen die Java-Pakete java.awt und java.swing, die für Oberflächengestaltungen erforderlich sind. Das Android SDK bringt mit der Verwendung von Layouts eine eigene Variante der Oberflächengestaltung mit, die im Gegensatz zu AWT (Abstract Window Toolkit) und Swing ressourcenschonender ist.[17]

4.4 Emulator

Das Android SDK beinhaltet einen Emulator für mobile Geräte. Damit lassen sich Smartphones verschiedenster Displaygrößen bis hin zu Tablets auf dem PC simulieren.[18] Der aktuelle Emulator stellt alle Android Versionen bis zu Android 4.3 (Jelly Bean) zur Verfügung. Für die Simulation des Samsung Galaxy 3 wird ein

---

[15] Vgl. [5] Eclipse: About the Eclipse Foundation. URL: http://www.eclipse.org/org/ (Stand: 20.12.2013, 11:45 Uhr)

[16] Vgl. [8] Get the Android SDK. URL: http://developer.android.com/sdk/index.html, (Stand: 05.12.2013, 15:45 Uhr)

[17] Vgl. [3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung: Heidelberg: dpunkt Verlag GmbH, 2009, S. 328

[18] Vgl. [1] Android Emulator. URL: http://developer.android.com/tools/help/emulator.html, (Stand: 05.12.2013, 17:30 Uhr)

3,3“, Android 2.2 Emulator gewählt. Die nachfolgende Abbildung (Abbildung 1) zeigt die Oberfläche des Emulators.

Abbildung 1: Emulator
Quelle: Screenshot

Die linke Hälfte bildet den Touchscreen des Smartphones und die rechte Hälfte sämtliche Hardwaretasten nach. Mit dem Mauszeiger wird die Berührung mit einem Finger simuliert.

Bei der Entwicklung der gestenbasierten Authentifikation stellte sich eine Einschränkung heraus. Im Gegensatz zu einem richtigen mobilen Gerät, lässt sich der Emulator nicht kippen. Somit können Sensoren, die die Orientation des Gerätes feststellen, nicht sinnvoll ausgewertet werden. Die Arbeit mit dem Emulator erleichtert jedoch die Entwicklung von Android Anwendungen ungemein, da für Testzwecke kein mobiles Gerät vorhanden sein muss.

5 Anforderungen an die Software

In diesem Kapitel wird auf die speziellen Anforderungen der Software eingegangen, die im Vorfeld festgelegt und im Späteren umgesetzt wurden.

Die Anwendung im Allgemeinen soll zum einen bei Verlust des mobilen Gerätes, einen unerlaubten Zugriff auf Daten und Telefonfunktionen durch nicht befugte Personen verhindern und zum anderen dem Nutzer selbst eine komfortable Authentifizierungsmethode bieten. Die Möglichkeit zur Eingabe einer individuellen Geste erlaubt dem Nutzer die freie Entfaltung seiner Kreativität. Außerdem wird durch die Anwendung das Auswendiglernen von Pins oder Passwörtern überflüssig.

Durch einen Hauptbildschirm soll dem Nutzer die Auswahl des Authentifikationsverfahren, das Anlegen seiner personalisierten Geste, sowie eines Super-Pins und die Aktivierung bzw. Deaktivierung der Authentifikation ermöglicht werden.

Über einen Bildschirm zur Speicherung der Gesten bzw. des Pins wird dem Nutzer zum einen das Anlegen und Löschen seiner Geste erlaubt, anderseits kann er sich eine eventuell vergessene Geste nochmals anschauen.

Ein Authentifikationsbildschirm, der die Eingabe der angelegten Geste fordert, darf nur für den Fall der richtigen Geste einen Zugriff erlauben und ansonsten unumgänglich für jeglichen Nutzer sein.

Um den unerwünschten Zugriff auf Daten und Funktionen des Gerätes zu verhindern, muss der Authentifikationsbildschirm in bestimmten Situationen selbstständig starten. Den häufigsten Anwendungsfall hierfür bildet die Rückkehr aus dem Standby-Mode, wenn die Tastensperre des Gerätes deaktiviert wird. Des Weiteren soll sich der Authentifikationsbildschirm im aktivierten Zustand der Anwendung starten, wenn ein Neustart des Gerätes stattgefunden hat.

Bei ankommenden Anrufen und SMS während der Standby-Phase muss sich der Nutzer zunächst mit seiner Geste erfolgreich authentifizieren, bevor die jeweilige Funktion genutzt werden kann.

Sollte es dem Nutzer passieren, dass er seine Geste längerfristig vergessen hat, muss eine Möglichkeit bestehen, das mobile Gerät über einen Super-Pin freizuschalten. Dieser Super-Pin sollte dann nach Möglichkeit an einem sicheren Ort aufbewahrt werden.

Da in extremen Stresssituationen, wie Unfällen mit Personenschaden, die Geste kurzzeitig vergessen werden kann, bzw. eine zittrige Hand zu Fehleingaben führt, muss eine Notruffunktion existieren. Der Nutzer muss in jedem Fall Notrufe absetzen können.

6 Allgemeine Beschreibung der Authentifikationsverfahren

Die Anwendung der gestenbasierten Authentifikation unterscheidet drei verschiedene Authentifikationsverfahren. Darunter befindet sich das bereits bekannte Bounding Box Verfahren. Es ist Teil des Programmes, hat aber nur eine nebensächliche Bedeutung. Deswegen wird auf die Implementierung dieses Verfahrens nicht genauer eingegangen. Hauptaugenmerk gilt den selbst entwickelten Verfahren des Points- sowie Rises-Matching.

6.1 Bounding-Box

Eine einfache Variante der Gestenauthentifikation lässt sich mit dem sogenannten Bounding-Box-Verfahren realisieren. Dabei wird die Eingabefläche in mehrere gleichgroße Quadrate aufgeteilt (ähnlich wie bei einem Tic Tac Toe Spiel). Über diese Quadrate lassen sich verschiedene Muster erzeugen. Zur Authentifikation ist die Anzahl und Reihenfolge der durchlaufenden Quadrate wichtig. Ein Nachteil dieser Variante ist, dass nur sehr schlecht Diagonalen nachgebildet werden

können, da die Quadrate sehr eng zusammen liegen und dadurch benachbarte Felder unbeabsichtigt markiert werden können. Besser ist dagegen die Verwendung von Kreisen (Bounding-Circle). Abbildung 2 zeigt die Realisierung der Oberfläche auf dem Android Smartphone.

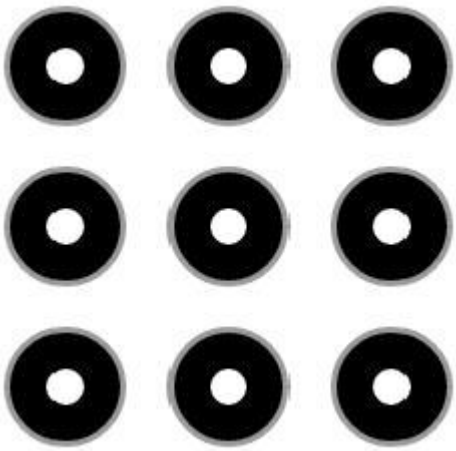

Abbildung 2: Bounding Circle Oberfläche
Quelle: Screenshot Emulator

Wie sich gut erkennen lässt, ist ein Zeichnen von Diagonalen problemlos möglich. Ein Beispiel in Abbildung 3 verdeutlicht dies.

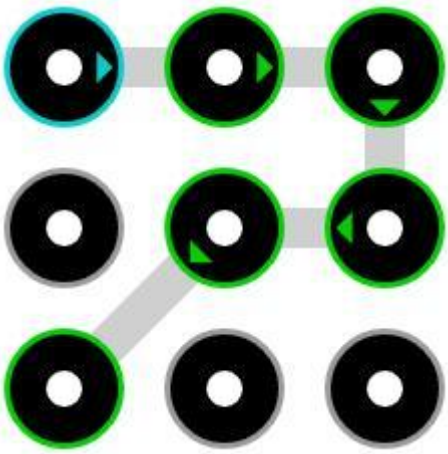

Abbildung 3: Bsp. für Bounding Circle
Quelle: Screenshot Emulator

Ein grundlegender Nachteil der Bounding-Box- bzw. Bounding-Circle – Verfahren ist die geringe Anzahl von Möglichkeiten der Mustereingabe. So bleibt dem Nutzer eine Individualisierung der Geste vorenthalten. Hingegen ist die Komplexität und der Rechenaufwand dieser Methode gering, was für einen sparsameren Stromverbrauch spricht.
Bei der Implementierung dieses Verfahrens wurde auf das Open-Source Projekt „Lock Pattern Generator“ von dem App-Entwickler Michael Shick zurückgegriffen.[19] Die Klasse NodeDrawable.java sowie Point.java wurden ohne Veränderungen übernommen. Die Klasse LockPatternView.java wurde ebenfalls übernommen und an die Anforderungen an eine Authentifikation angepasst.

6.2 Points- Matching

Beim Points-Matching–Verfahren werden die Gesten anhand ihrer Punkte miteinander verglichen. Dem Nutzer steht ein großer Bereich des Displays für die Eingabe eines Musters zur Verfügung. Hierbei kann jegliche denkbare Form verwendet werden. Nachstehende Abbildung (Abbildung 4) zeigt ein Beispiel für eine Geste auf dem Android Smartphone.

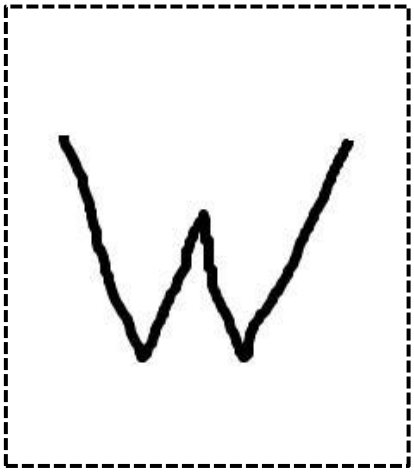

Abbildung 4: Beispiel Geste Points-Matching
Quelle: eigene Darstellung - Screenshot

[19] [15] Shick, Michael: Lock Pattern Generator. URL: http://git.shick.in/lockpatterngenerator/, (Stand: 04.11.2013, 11:10 Uhr)

Der Nutzer kann seinen individuellen Wünschen freien Lauf lassen. Ein Nachteil des Points-Matching-Verfahrens ist allerdings der sehr hohe Rechenaufwand. Einerseits durch die notwendige Vorverarbeitung der Punkte einer Geste, als auch der Vergleich an sich. Der Vergleich findet über die Berechnung von Punkteabständen statt. Nur zwei Punkte deren Abstände im vorgegebenen Toleranzbereich liegen, gelten als übereinstimmend. Bei der Verwendung von sehr aufwendigen Gesten (z.B. das Haus vom Nikolaus) mit sehr vielen Punkten, kann es bei diesem Verfahren zu Komplikationen bei der Authentifikation kommen. Der normale Nutzer ist nämlich meist nicht in der Lage eine zu lange Geste mehrfach in einer ähnlichen Form auf dem Smartphone zu zeichnen.

6.3 Rises-Matching

Beim Rises-Matching-Verfahren werden die Gesten anhand ihres Monotonieverhaltens in X- sowie Y-Richtung verglichen. Unterschieden wird dabei zwischen steigend, fallend und konstant. Im Vergleich zum vorher beschriebenen Points-Matching mit seinem großen Informationsgehalt in Form der Punkte, werden hier nur noch die Anstiege betrachtet. Daher ist es nicht sinnvoll sehr einfache Gesten, wie senkrechte bzw. waagerechte Striche als Geste zu verwenden. Auch hier steht dem Nutzer wieder ein großer Bereich des Smartphone Touchscreens zur Verfügung. Es kann jede beliebige Geste angelegt werden. Der Rechenaufwand ist im Vergleich zum Points-Matching wesentlich geringer. Sehr aufwendige und lange Gesten lassen sich hiermit auch besser zur Authentifikation nutzen.

7 Algorithmen zur Gestenauthentifikation

Im nun folgenden Kapitel werden die Algorithmen der Authentifikationsverfahren aus 7.2 Points- Matching und 7.3 Rises-Matching beschrieben.
Die Algorithmen bestehen im Wesentlichen aus zwei Teilen. Zunächst aus der Vorverarbeitung der Rohdaten nach der Eingabe einer Geste sowie dem Vergleich

zweier Datensätze zur Realisierung der Authentifikation. Die Besonderheit der Algorithmen liegt bei der Messung und Auswertung von Zeitwerten, die es in dieser Form bei Gesten-Authentifikationen noch nicht gegeben hat. Die Zeit bildet, neben der persönlichen Gestenform und der Geräteorientierung, eine weitere Dimension in der Gesten-Authentifikation sowie der Personalisierung einer Geste.

7.1 Preprocessing/ Vorverarbeitung der Koordinatenpaare

Um die Gesten zur Authentifikation vergleichbar zu machen, ist eine Vorverarbeitung der X- sowie der Y-Koordinaten nötig. Im Folgenden wird auf die einzelnen Berechnungen eingegangen.
Angelehnt an die Vorgehensweise aus der Diplomarbeit von Jan Lemke mit dem Thema „Entwicklung einer gestenbasierten Authentifikation für Symbian 60“ wurden die Ideen eines Größennormierungsfilters, Glättungsfilters, Punktedistanzfilters sowie der Steigungsberechnung für die Vorverarbeitung der Koordinaten übernommen.[20]

7.1.1 Verschiebung an Rand der Koordinatenachsen

Als erstes wird die Geste an den Rand der Koordinatenachsen gelegt. Hierfür ist es zunächst einmal notwendig die kleinsten X- und Y-Koordinaten zu ermitteln.

$$minX = \min(x_i)$$
$$minY = \min(y_i)$$

Die Minima werden von allen Koordinaten subtrahiert.

$$xTemp_i = x_i - minX$$
$$yTemp_i = y_i - minY$$

[20] Vgl. [12] Lemke, Jan: Entwicklung einer gestenbasierten Authentifikation für Symbian 60: Diplomarbeit HAW Hamburg, 2012, S. 54-58

Die Auswirkung auf die Lage der Geste veranschaulicht folgende Abbildung (Abbildung 5).

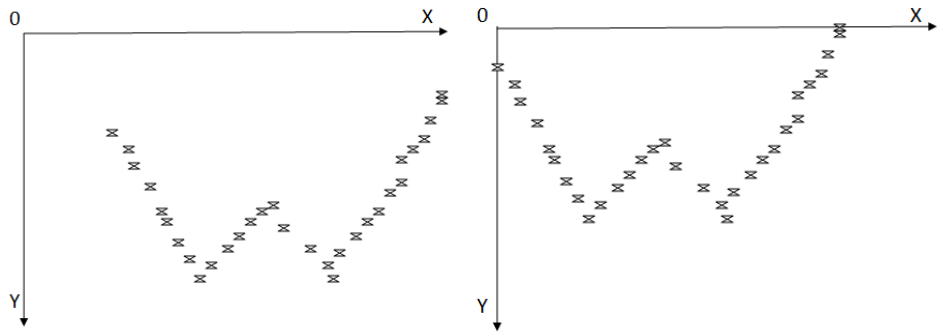

Abbildung 5: Preprocessing Verschiebung
Quelle: eigene Darstellung

7.1.2 Normierung auf einheitlichen Größenbereich

Nun wird die Geste auf eine einheitliche Skalierung gebracht. Dafür wird der maximale X-Wert und Y-Wert der Punkte benötigt.

$$maxX = \max(x_i)$$
$$maxY = \max(y_i)$$

Ein Skalierungsfaktor (scaleFactor) legt den gewünschten Bereich fest. Er ist maßgeblich für die Auflösung der Punkte verantwortlich. Die neuen skalierten Werte berechnen sich nach folgender mathematischer Vorschrift.

$$xNew_i = \frac{x_i * scaleFactor}{maxX}$$
$$yNew_i = \frac{y_i * scaleFactor}{maxY}$$

Je nach Größe der Geste und des Skalierungsfaktors wird das Zeichen auf den festgelegten Bereich gestaucht oder gestreckt. Nachstehende Abbildung (Abbildung 6) verdeutlicht dies an einem Beispiel.

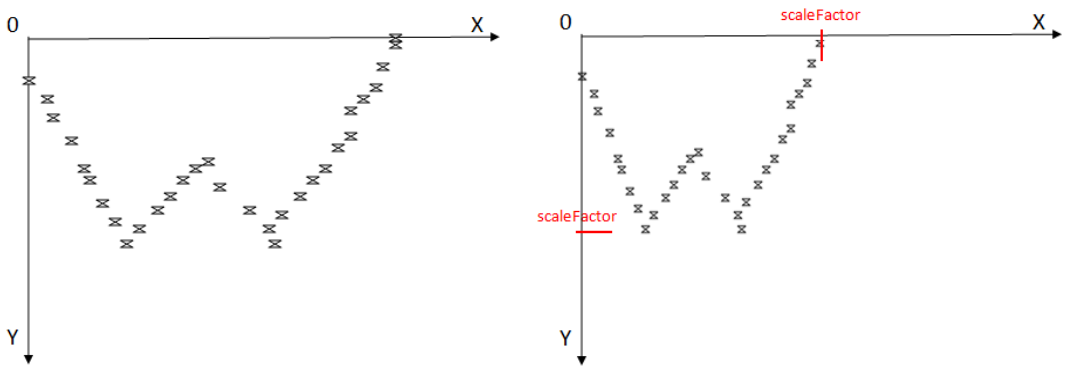

Abbildung 6: Preprocessing Größennormierung
Quelle: eigene Darstellung

7.1.3 Glättung der Daten

Um auftretende Ausreißer zu beseitigen und scharfe Kanten zu glätten, wird ein Glättungsfilter (Formel siehe unten) benutzt. Der Glättungsfaktor (straightFact) gibt die Anzahl der Koordinaten, über die jeweils eine Glättungsoperation ausgeführt wird, an.

$$xNew_i = \frac{1}{straightFact} \sum_{i=0}^{straightFact-1} x_i$$

$$yNew_i = \frac{1}{straightFact} \sum_{i=0}^{straightFact-1} y_i$$

Diese Berechnungsvorschrift gilt allerdings nicht für den ersten und letzten Punkt der Geste. Die beiden Koordinatenpaare werden unverändert übernommen.

7.1.4 Reduktion der Daten

Abschließend werden die Koordinaten der Geste reduziert, um bei den Vergleichsalgorithmen Rechenzeit zu sparen. Dabei bleiben grundlegende Merkmale der Geste erhalten. Für die Punktereduktion wird eine minimale Entfernung (s) zwischen zwei Punkten festgelegt. Der Abstand berechnet sich zunächst wie folgt.

$$distance = (x_i - x_{i+1})^2 + (y_i - y_{i+1})^2$$

Ist der Abstand zwischen zwei Punkten kleiner als die vorgegebene Distanz, so wird der Punkt entfernt bzw. nicht weiterverarbeitet. Bei einem größeren Abstand werden die Koordinaten übernommen.

$$P_{new}(x_i, y_i) = P(x_i, y_i)\ wenn\ distance > s$$

7.1.5 Winkelberechnung

Als ein zusätzliches Überprüfungsmerkmal wird der Steigungswinkel von zwei aufeinanderfolgenden Koordinaten ermittelt.

$$\alpha = \arctan(\frac{y_i - y_{i+1}}{x_i - x_{i++1}})$$

Die Berechnung des Arkustangens wurde mit der in Java implementierten atan2()-Funktion vorgenommen, die wie folgt definiert ist.

$$atan2(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right)\ für\ x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi\ für\ y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi\ für\ y < 0, x < 0 \\ \frac{\pi}{2}\ für\ y > 0, x = 0 \\ -\frac{\pi}{2}\ für\ y < 0, x = 0 \\ undefined\ für\ y = 0, x = 0 \end{cases}$$

7.1.6 Ermittlung der Monotoniewechsel

Die für das Rises-Matching benötigten Monotoniewechsel in X- sowie in Y-Richtung werden aus den reduzierten Koordinatenpaaren ermittelt. Dazu werden jeweils die beiden benachbarten X- bzw. Y-Werte verglichen. Unterschieden wird dabei zwischen steigend, fallend und konstant. Damit nicht jeder kleine Ausreißer in einer Geraden zu einer Anstiegs-Änderung führt, müssen die Werte mindestens um einen festgelegten Wert (Mindestabweichung = 3) voneinander abweichen. Das Ergebnis ist jeweils eine Liste für die Monotoniewechsel in X- sowie in Y-Richtung.

7.2 Gestenvergleich Points-Matching

Das Points-Matching-Verfahren basiert grundlegend auf der Berechnung von Punkteabständen der einzelnen Koordinaten der Gesten zur Authentifikation. Zusätzlich spielt neben der Auswertung von Lagesensoren auch eine Zeitmessung bei der Eingabe der Geste eine Rolle. Im Folgenden wird ausführlich auf den Algorithmus des Points–Matching-Verfahrens eingegangen.

7.2.1 Orientierung und Gestengröße

Nach der Vorverarbeitung liegen nun Gesten vor, die über ihre Koordinatenpaare miteinander vergleichbar sind. Vorab wird überprüft, ob bei der Eingabe der Geste das mobile Gerät in derselben Position gehalten wurde wie beim Abspeichern. Dazu werden die im Smartphone integrierten Lagesensoren ausgewertet. Bei übereinstimmenden Sensorwerten wird anschließend getestet, ob beide Gesten ähnlich viele Punkte aufweisen (siehe auch Struktogramm 1, Abbildung 7). Damit lässt sich verhindern, dass die Eingabe eines kleinen Teils der Geste bereits zur erfolgreichen Authentifikation führt. Außerdem kann so erreicht werden, dass zwischen ähnliche Gesten, die sich nur minimal durch zusätzliche bzw. weniger Punkte voneinander unterscheiden (z.B. wie bei den Großbuchstaben „D“ und „B“), sondiert werden kann.

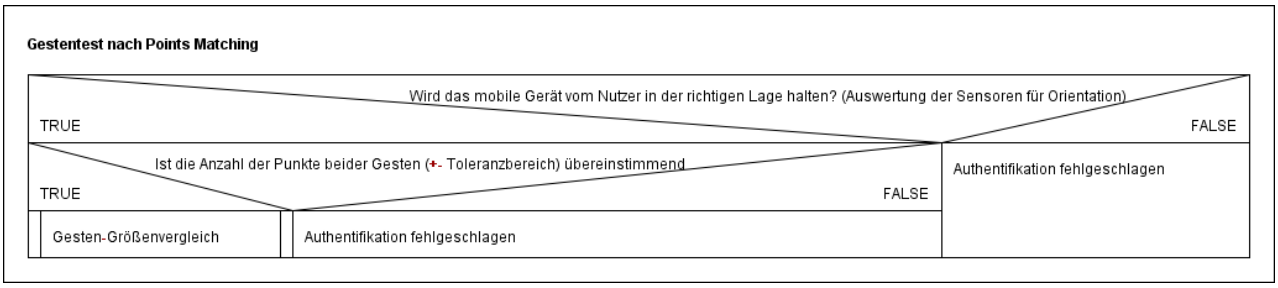

Abbildung 7: Struktogramm 1 Gestentest nach Points-Matching
Quelle: eigene Darstellung

7.2.2 Gestengrößenvergleich

Da die zu vergleichenden Gesten unterschiedlich viele Punkte haben können und dies für den Algorithmus zur Punktedifferenzermittlung von Bedeutung ist, wird zunächst einmal festgestellt, bei welcher Geste es sich um diejenige mit weniger Punkten handelt (siehe auch Struktogramm 2, Abbildung 8). Mit der „SaveGeste“ ist die im Vorhinein im Speicher hinterlegte Geste gemeint. Während die „AuthGeste“ die zur Authentifizierung geforderte Eingabe darstellt.

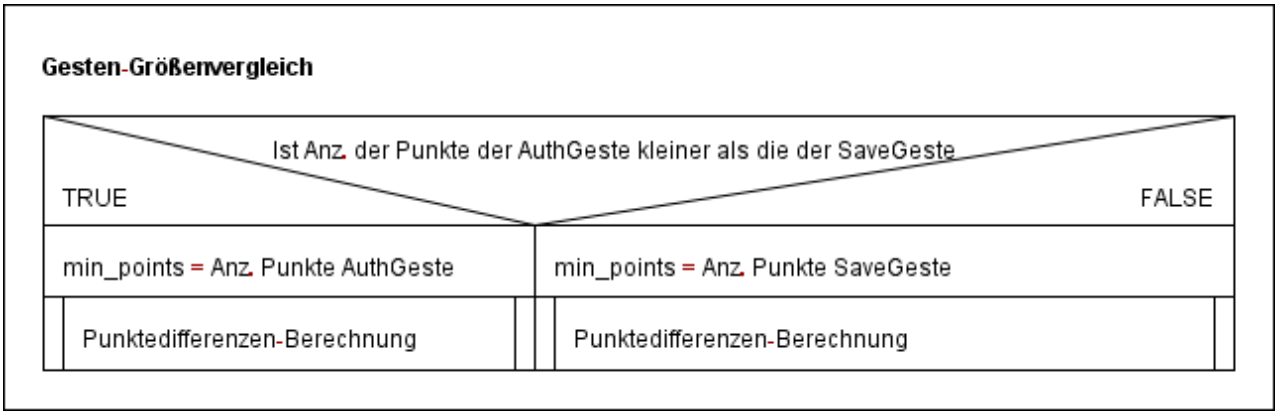

Abbildung 8: Struktogramm 2 Gesten-Größenvergleich
Quelle: eigene Darstellung

7.2.3 Punktedifferenzberechnung

Die Punktedifferenzberechnung bildet den Kern des Algorithmus. Die Schwierigkeit besteht darin, einen bestmöglichen Vergleich der einzelnen Punkte der zwei Gesten zu realisieren. Da ein identisches wiederholtes Zeichnen von Gesten nahezu unmöglich ist, bestehen die zu vergleichenden Gesten unter Umständen aus einer unterschiedlichen Anzahl von Koordinatenpaaren. Auf Grund dieser Tatsache, führt eine Differenzberechnung von Punkten des gleichen Index der zwei Gesten-Arrays zu unbrauchbaren Ergebnissen, da diese mitunter sehr weit auseinander liegen können, und das, obwohl die Gesten übereinstimmend eingegeben wurden.

Weiterhin muss berücksichtigt werden, dass nicht für jeden Punkt der einen Geste ein passender „Partner“ der anderen Geste gefunden werden kann. Wiederum besteht durchaus die Möglichkeit, dass mehrere Punkte auf einen Punkt passen.

Der Algorithmus muss dementsprechend flexibel mit unterschiedlichen Gestengrößen umgehen können und einen variablen Vergleich mehrerer Punkte gewährleisten.

7.2.3.1 Euklidischer Punkteabstand

Die Berechnung des euklidischen Punkteabstandes erfolgt durch nachstehende Formel. Hinzu wird ein gewichteter Betrag der Differenz der Winkel addiert.

$$Punktabstand = \sqrt{(x_{Gest1} - x_{Gest2})^2 + (y_{Gest1} - y_{Gest2})^2} + angleFact * |w_{Gest1} - w_{Gest2}|$$

7.2.3.2 Innere und äußere for-Schleife

Um die einzelnen Punkte der beiden Gesten auf Übereinstimmung bzw. auf minimale Punktabstände zu testen, wird die Geste mit der geringeren Anzahl von

Punkten in der äußeren for-Schleife durchlaufen (Indexvariable i). Jeder dieser Punkte wird mit einer festen Anzahl von Punkten (points_to_test = 5) der jeweils anderen Geste verglichen (innere for-Schleife, Indexvariable j). Der kleinste Punkteabstand der getesteten fünf Punktedifferenzen wird in eine Punkteabstands-Liste übernommen. Zur Veranschaulichung ist in Abbildung 13 der Algorithmus in einem Struktogramm abgebildet.

7.2.3.3 Indexoffset

Um den Punktevergleich flexibler zu gestalten wird zusätzlich ein Indexoffset (k bzw. k_temp) eingeführt. Der temporäre Indexoffset k_temp „lebt“ innerhalb der inneren for-Schleife. Er wird jedes Mal um Eins erhöht, wenn die Punktedifferenz zwischen dem aktuellen Punkt der kleineren Geste und einem Punkt der größeren Geste mit höherem Index kleiner ist als die vorherige. Das heißt, haben die Punkte mit gleichem Index nicht den geringsten Abstand, wird der Indexoffset erhöht. Am besten lässt sich diese Problematik an einem Bild erläutern.
Nachstehende Abbildung (Abbildung 9) zeigt ein Matlab-Plot, in dem zwei Datensätze zusammen dargestellt sind. In diesem Fall hat die blau gekennzeichnete Kurve mehr Koordinatenpaare (43) als die rote (27). Auch wenn diese Konstellation auf Grund des zu großen Punktemengenunterschieds nicht zu einer erfolgreichen Authentifikation führt, verdeutlicht sie gut, dass ein Indexoffset von Nöten ist.

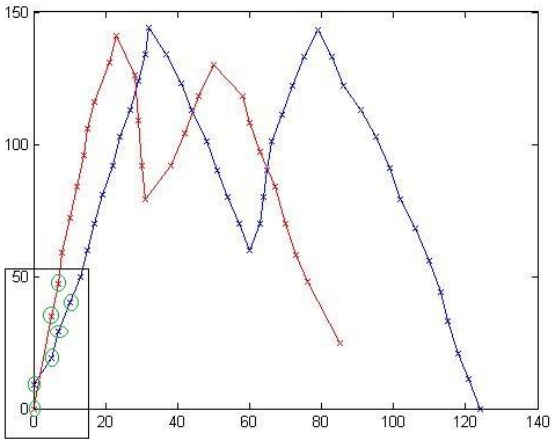

Abbildung 9: Matlab-Plot zweier Gesten Bsp. 1

Die nun folgende Abbildung (Abbildung 10) betrachtet den gekennzeichneten Bereich und die dort enthaltenen Punkte genauer.

Abbildung 10: Ausschnitt aus Matlab Plot Bsp. 1

Wie man sieht liegen zwischen P1_r und P2_r der roten Kurve drei Punkte der blauen Kurve. Während für P1_r offensichtlich P1_b den kleinsten Punkteabstand bedeutet, ist für P2_r hingegen erst P3_b der passende Partner. Der temporäre Indexoffset k_temp hätte in diesem Beispiel den Wert Eins.

Nach jedem Durchlauf der inneren Schleife wird der temporäre Indexoffset k_temp auf den aktuellen Indexoffset k aufaddiert und anschließend auf den Wert Null zurückgesetzt. Dies bedeutet für das Beispiel oben, dass für P3_r zunächst nur die Punkte ab P4_b zur Punkteabstandsberechnung in Frage kommen. Daraus entsteht aber ein Problem, welches in den folgenden Abbildungen (Abbildung 11 und Abbildung 12) verdeutlicht wird.

In nachstehendem Bild sind wiederum zwei Datensätze in einem Matlab-Plot zusammengeführt. Auch hier hat die blaue Kurve mehr Koordinatenpaare (43) als die rote (38).

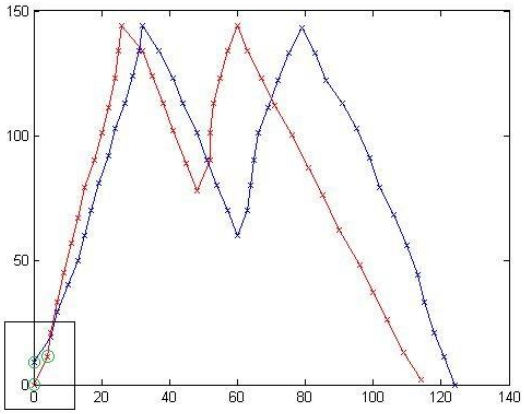

Abbildung 11: Matlab Plot zweier Gesten Bsp. 2

Der im Bild oben gekennzeichnete Bereich ist in folgender Abbildung (Abbildung 12) nochmal vergrößert dargestellt.

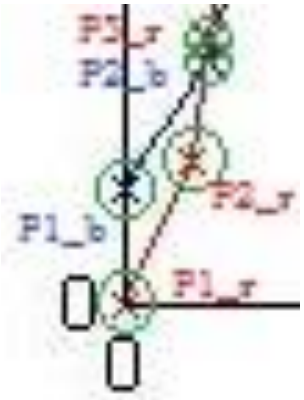

Abbildung 12: Ausschnitt aus Matlab Plot Bsp. 2

Wie man hier gut erkennen kann, besteht die Möglichkeit, dass auch zwischen zwei Punkten (P1_r und P2_r) einer Geste und lediglich einem Punkt der jeweils anderen Geste (P1_b) der kleinste Abstand bestehen kann. Es ist also notwendig, trotz passendem Punktepaar, die Abstandsberechnung des nächsten Punktes der kleineren Geste mit dem vorherig passenden Punkt durchzuführen. Darum muss der temporäre Indexoffset für den Fall, dass er erhöht wurde (egal wie oft), immer um den Wert Eins gesenkt werden. Würde man dies nicht tun, entsteht unter Umständen ein „Aufschaukeln“ der Punkteabstände, die eine Authentifikation unmöglich machen.

7.2.3.4 Struktogramm

Zur Veranschaulichung des gesamten Algorithmus zur Punktedifferenzberechnung zeigt folgende Abbildung (Abbildung 13) das Struktogramm zu diesem Programmteil.

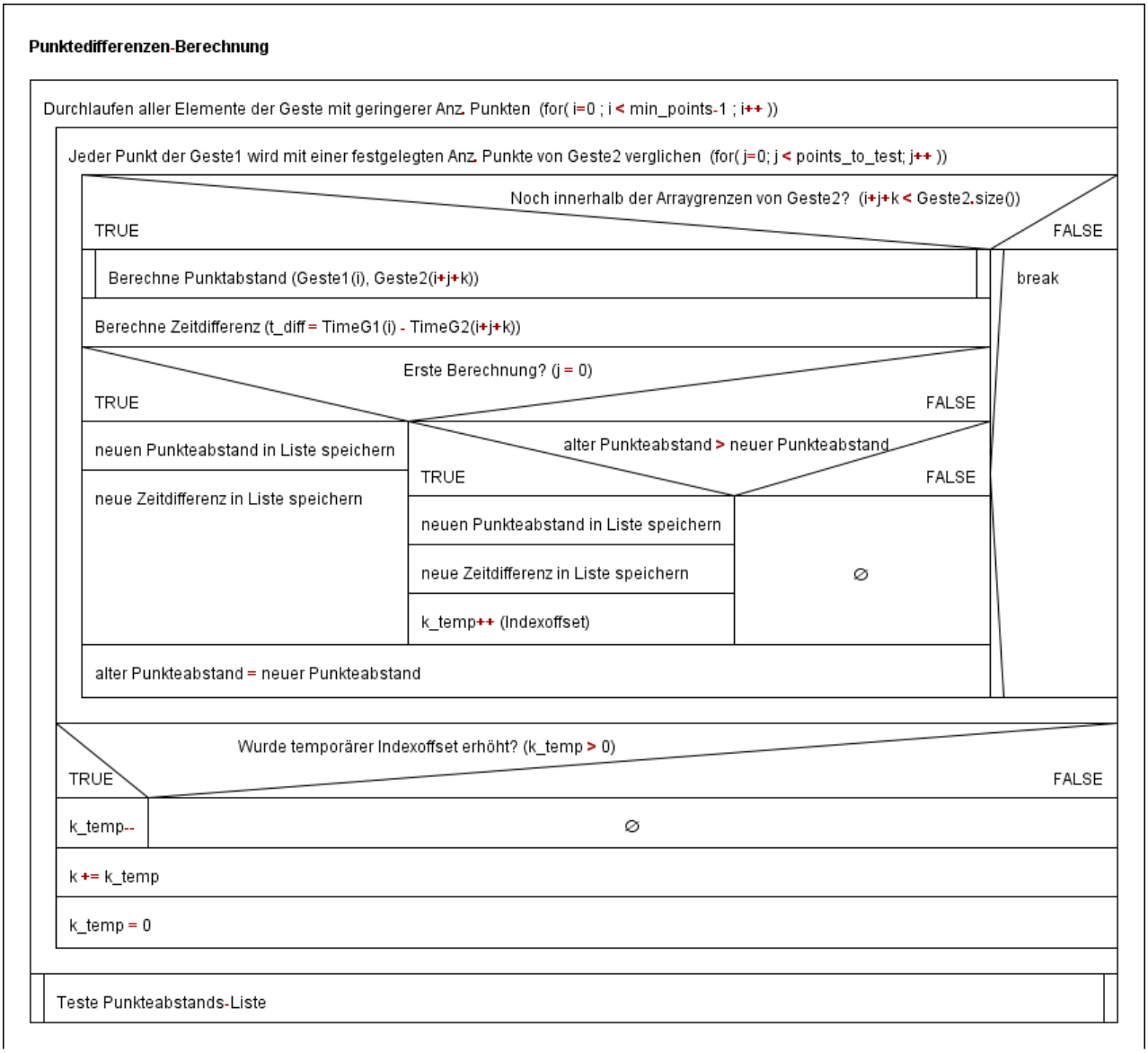

Abbildung 13: Struktogramm 3 Punktedifferenz-Berechnung
Quelle: eigene Darstellung

7.2.4 Test der Punkteabstandsliste

Abschließend wird die Punkteabstandsliste auf Übereinstimmungen geprüft (siehe auch Struktogramm 4, Abbildung 14). Dazu wird ein maximal erlaubter Punkteabstand definiert. Jedes Element der Liste wird mit dem maximalen Abstand verglichen. Liegt der Listenwert darunter, wird ein Übereinstimmungs-Zähler hochgezählt. In Abhängigkeit von der Anzahl der Punkte der Geste wird überprüft, ob eine ausreichende Anzahl von Übereinstimmungen vorliegt (mindestens 90%). Dementsprechend wird die Authentifikation als nicht erfolgreich eingestuft oder der letzte Testschritt, die Überprüfung der Zeitdifferenzen, eingeleitet.

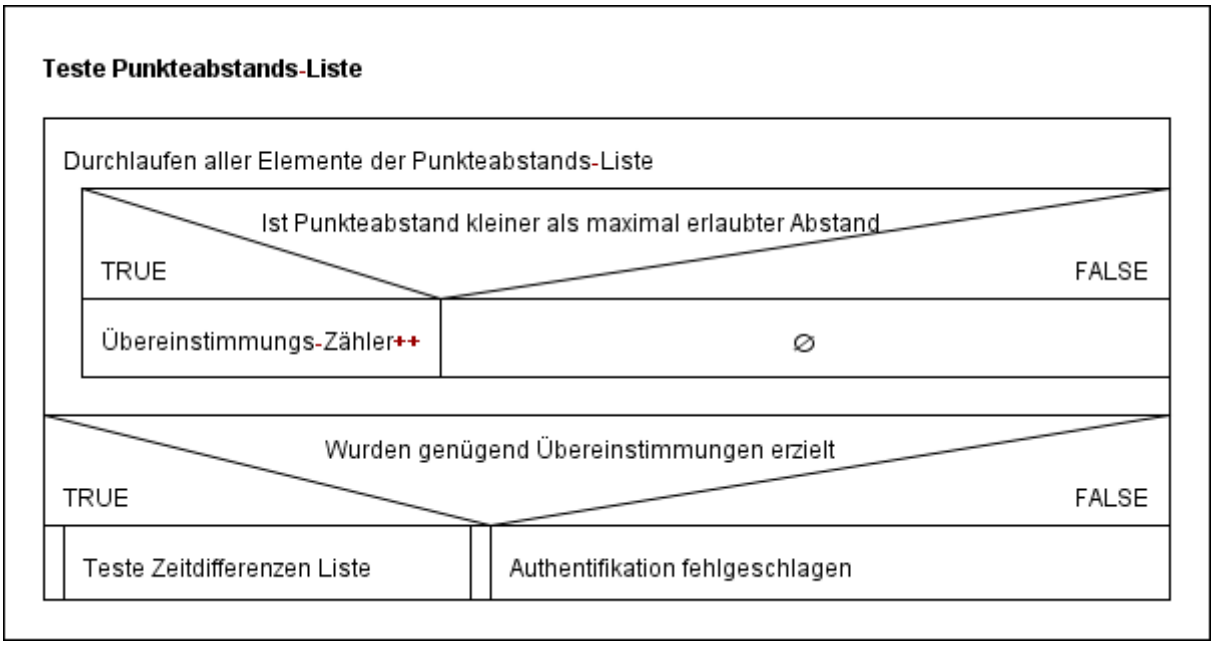

Abbildung 14: Struktogramm 4 Teste Punkteabstands-Liste
Quelle: eigene Darstellung

7.2.5 Test der Zeitdifferenzen

Die Überprüfung der Zeitdifferenzen erfolgt nach dem gleichen Prinzip wie bei der Punkteabstandsüberprüfung (siehe auch Struktogramm 5, Abbildung 15). Jede Zeitdifferenz wird mit einem festgelegten maximalen Zeitwert verglichen. Liegt die Zeitdifferenz unterhalb dieses Wertes wird ein Zähler hochgezählt. Auch hier wird in Abhängigkeit der Anzahl der Zeitdifferenzen ermittelt, ob genug Zeitdifferenzen unterhalb des erlaubten Maximums liegen (80%).

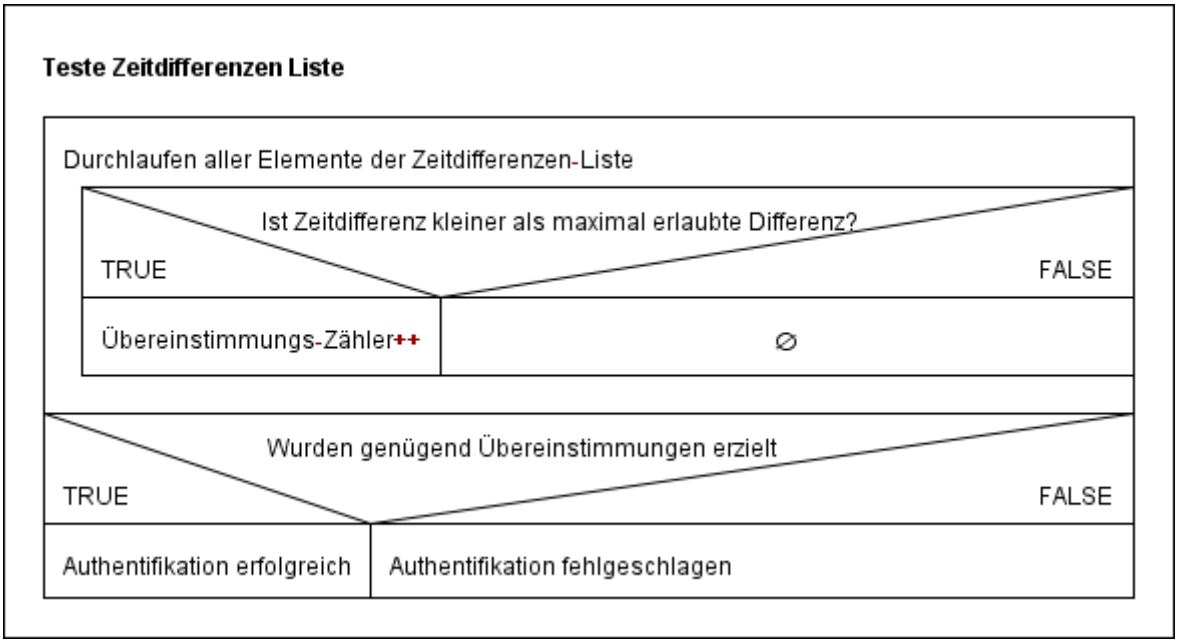

Abbildung 15: Struktogramm 5 Teste Zeitdifferenzen-Liste
Quelle: eigene Darstellung

7.3 Gestenvergleich Rises-Matching

Neben der Gegenüberstellung der Koordinaten lässt sich ein Vergleich zweier Gesten auch über deren Anstiege bzw. Anstiegsänderungen in X- sowie in Y-Richtung realisieren. Unterschieden wird dabei zwischen „steigend“, „fallend“, „konstant“.

Der gesamte Algorithmus wird jeweils für die Anstiegsänderungen in X- sowie in Y-Richtung durchlaufen. Die Authentifikation ist also nur dann erfolgreich, wenn beide Dimensionen mit abschließendem Erfolg verglichen werden konnten.

7.3.1 Orientation und Gestengröße

Ähnlich wie beim Algorithmus des Points-Matching erfolgt zunächst die Auswertung der Sensoren für die Orientierung. Das mobile Gerät muss in derselben Lage in der Hand des Nutzers gehalten werden, wie zuvor beim Abspeichern der persönlichen Geste. Ist dieser Test erfolgreich wird auch hier geprüft, ob beide Gesten in etwa die gleiche Anzahl von Anstiegsänderungen aufweisen (siehe auch Struktogramm 6, Abbildung 16). Dies verhindert eine erfolgreiche Authentifikation bei Eingabe eines kleinen Teils der Geste.

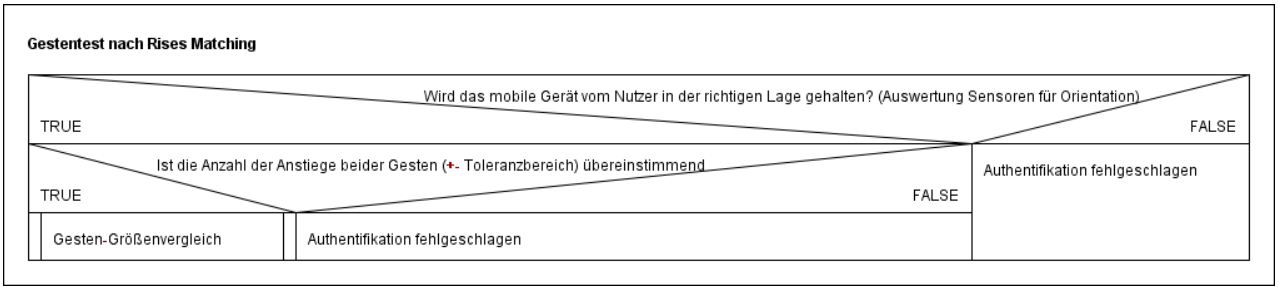

Abbildung 16: Struktogramm 6 Gestentest nach Rises-Matching
Quelle: eigene Darstellung

7.3.2 Gestengrößenvergleich

Als nächstes wird auch hier die Geste mit der geringeren Anzahl an Anstiegen ermittelt (siehe auch Struktogramm 7, Abbildung 17). Eine ungleiche Anzahl von Monotoniewechseln kann durch unterschiedlich schnelles und ungenaues Zeichnen der Gesten, besonders bei runden Formen, auftreten.

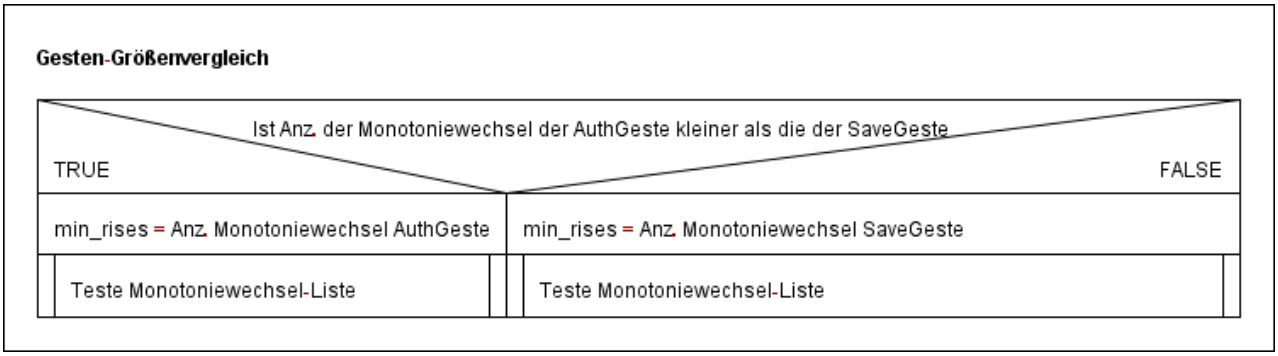

Abbildung 17: Struktogramm 7 Gesten-Größenvergleich
Quelle: eigene Darstellung

7.3.3 Vergleich der Monotoniewechsel

Jeder Anstieg der kleineren Geste (hier Geste1) wird durchlaufen und mit einer festgelegten Menge an Anstiegen der jeweils anderen (hier Geste2) verglichen. Bei Gleichheit wird der Übereinstimmungszähler um den Wert Eins hochgezählt und die entsprechende Zeitdifferenz der jeweiligen Anstiege berechnet. Am Ende wird schließlich geprüft, ob alle Monotoniewechsel Übereinstimmung gefunden haben. Nachstehendes Struktogramm (Struktogramm 8, Abbildung 18) zeigt den beschriebenen Algorithmus.

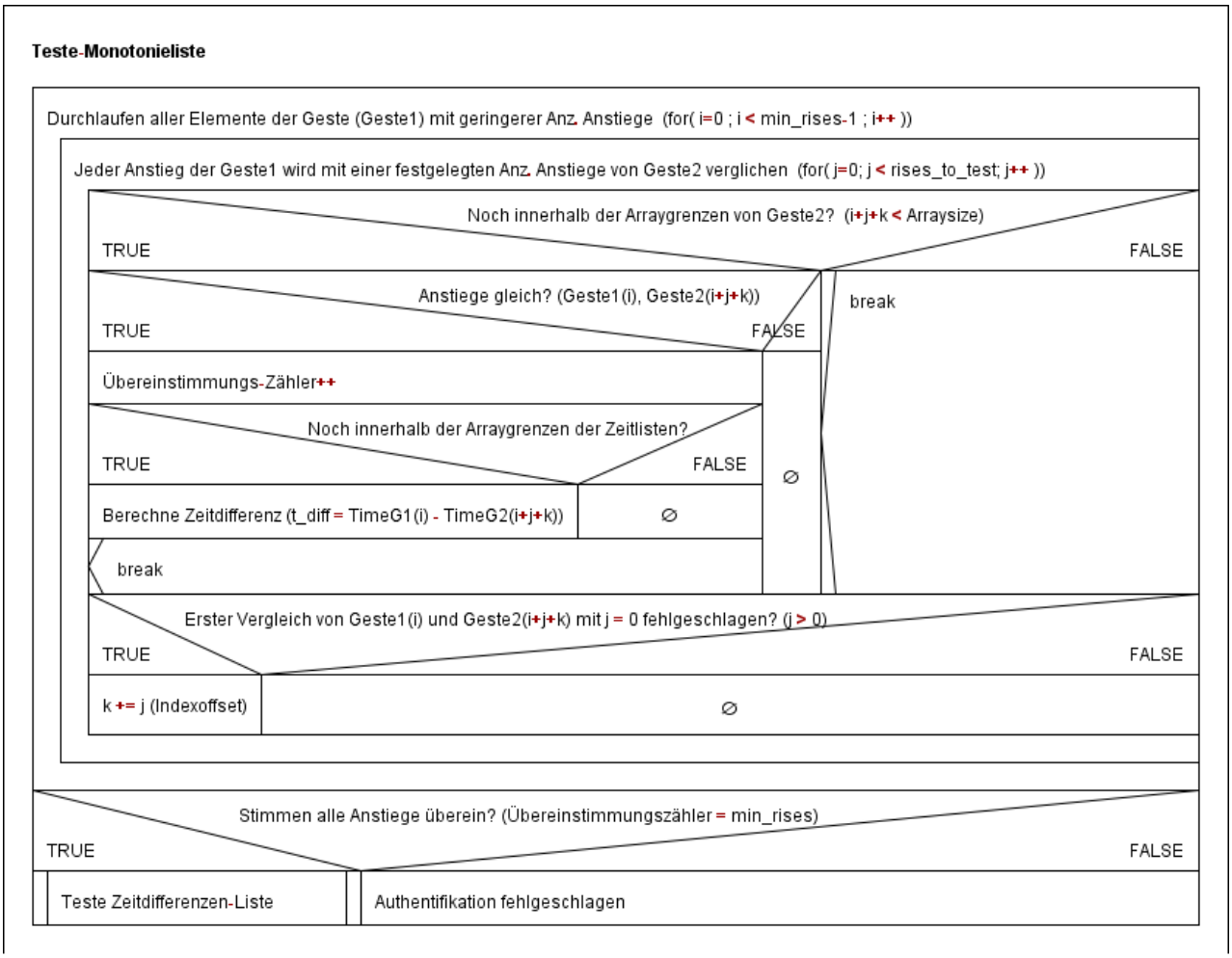

Abbildung 18: Struktogramm 8 Teste Monotonieliste
Quelle: eigene Darstellung

7.3.4 Test der Zeitdifferenzen

Nach erfolgreichem Durchlauf des Anstiegs-Vergleiches werden noch die ermittelten Zeitdifferenzen überprüft. Ein festgelegter maximaler Zeitwert darf nicht überschritten werden. Der Test führt zur erfolgreichen Authentifikation, wenn ein bestimmter Prozentsatz von Zeitdifferenzen unterhalb des Maximalwertes liegt (siehe auch Struktogramm 9, Abbildung 19).

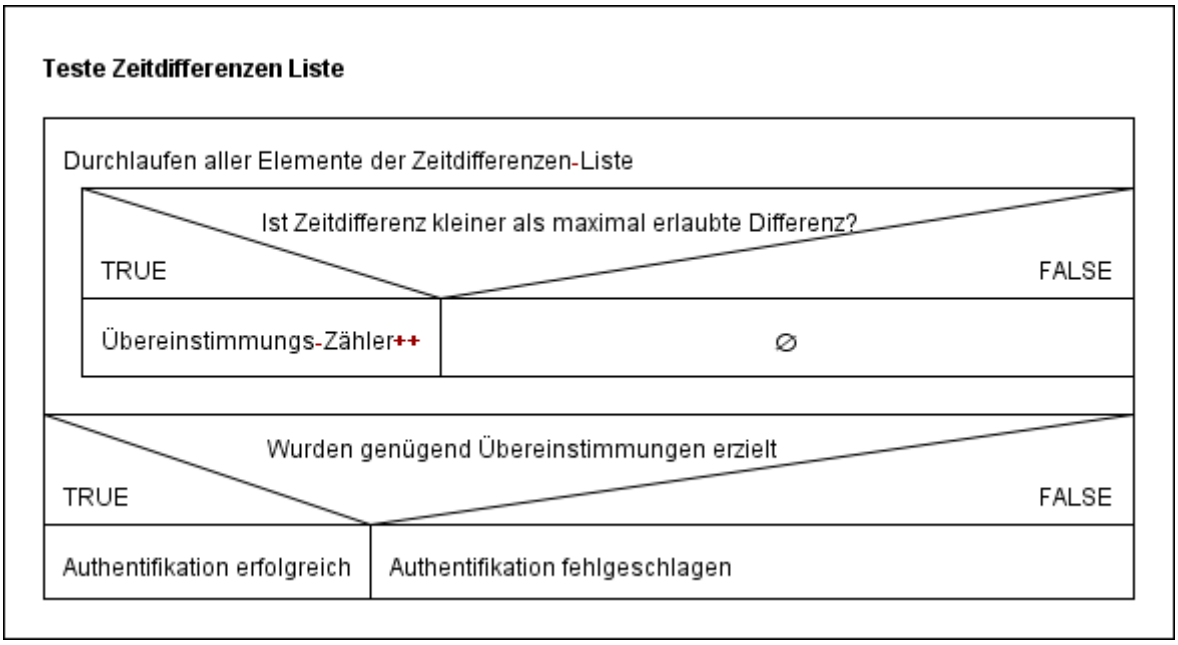

Abbildung 19: Struktogramm 9 Teste Zeitdifferenzen Liste Rises Matching
Quelle: eigene Darstellung

8 Funktionsweise des Programms

Im nun folgenden Abschnitt werden die verschiedenen Programmteile und deren Zusammenspiel erläutert. Es wird die Umsetzung der Anforderungen beschrieben und auf Probleme bei der Entwicklung eingegangen.

Das vollständige Eclipse-Projekt, inklusive dem dokumentierten Quellcode befindet sich auf der beigefügten CD und kann bei Herrn Prof. Dr.-Ing. Robert Fitz eingesehen werden.

8.1 Hauptbildschirm

Der Hauptbildschirm ist die erste Activity und bildet damit die Grundlage des Programmes. Wird die Anwendung zum ersten Mal installiert, gelangt man zunächst direkt zu dieser Benutzeroberfläche (siehe Abbildung 20: Hauptbildschirm). Von hier aus lassen sich alle benötigten Einstellungen treffen.

Abbildung 20: Hauptbildschirm
Quelle: Screenshot

Zunächst lässt sich über einen Button die Authentifikation aktivieren bzw. deaktivieren. Beim ersten Programmstart ist die Einstellung standardgemäß deaktiviert. Nach Betätigung dieses Buttons und der damit verbundenen Aktivierung, lässt sich das Authentifikationsverfahren auswählen. Zur Verfügung stehen hierfür die zuvor bereits erläuterten Bounding-Box- und Matching-Verfahren.

Beim Matching-Verfahren wird unterschieden zwischen dem Points-Matching- und dem Rises-Matching-Verfahren. Welches Verfahren von den beiden letztendlich zur Authentifikation genutzt wird, hängt von der Geste ab. Stellt der Preprocessing

Algorithmus weniger als 16 Monotoniewechsel fest, wird das Points-Matching-Verfahren verwendet, andernfalls das Rises-Matching-Verfahren.

Im unteren Bereich des Hauptbildschirmes weist ein roter Schriftzug darauf hin, dass noch keine Geste vom Nutzer angelegt wurde. Zum Anlegen und Speichern der Gesten ist der Button „Geste speichern“ verantwortlich. Wird dieser betätigt, wird über einen Intent, je nach ausgewähltem Verfahren, eine neue Oberfläche (Activity) gestartet. Der Button „PinCode festlegen“ führt ebenfalls durch einen Intent zu einer neuen Oberfläche, die es ermöglicht einen eigenen Pin festzulegen.

8.2 Activitys zum Speichern der Gesten und des Pin-Codes

Nach der Betätigung des Buttons „Geste speichern“ vom Hauptbildschirm öffnet sich eine neue Activity. Die nachfolgende Abbildung (Abbildung 21) zeigt die entsprechenden Oberflächen.

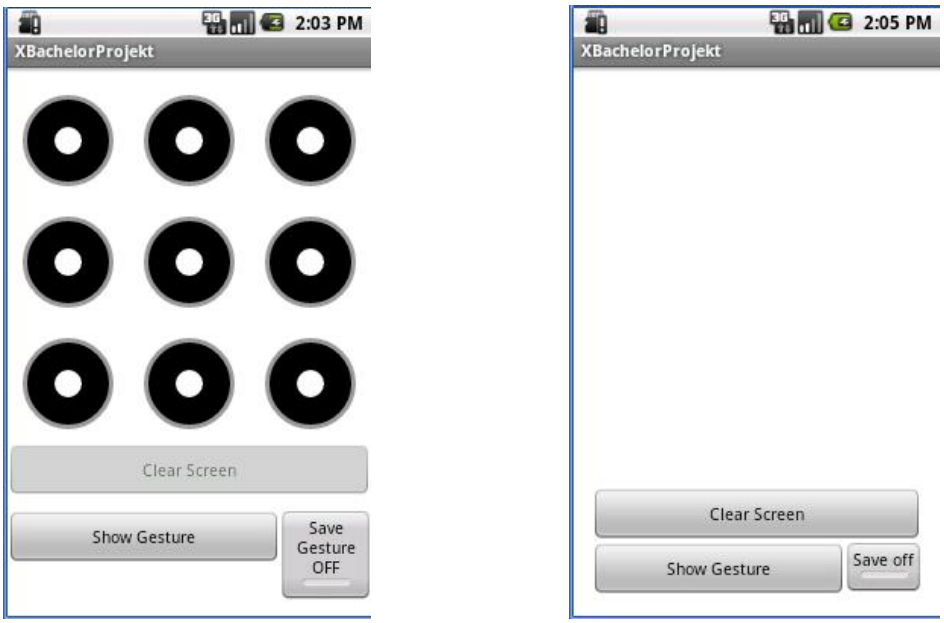

Abbildung 21: Activity zum Speichern der Gesten
Quelle: Screenshot

Beide Oberflächen sind ähnlich aufgebaut und funktionieren nach dem gleichen Prinzip. Um eine Geste anzulegen, muss zunächst der Button „Save Gesture“ bzw. „Save“ aktiviert werden. Nun kann der Nutzer seine individuelle Geste mit Hilfe seines Fingers auf dem Touchscreen zeichnen. Ein Beispiel für die jeweiligen Gesten ist in der folgenden Abbildung (Abbildung 22) gezeigt.

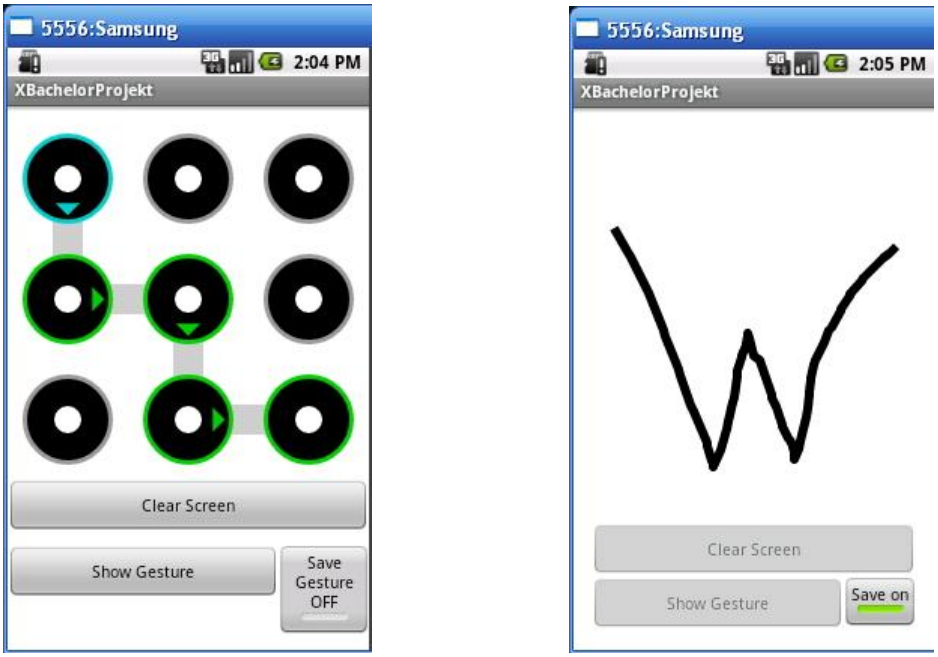

Abbildung 22: Beispiele für gespeicherte Gesten
Quelle: Screenshots

Während im linken Teil der Abbildung 22 der Speichermodus bereits wieder deaktiviert wurde, ist er im rechten Bild noch aktiv (erkennbar am grünen Balken im Button bzw. an der Buttonbeschriftung). Erst nach Deaktivierung durch erneutes Betätigen des „Save“- Buttons, werden die Gesten gespeichert. Im Falle des Matching-Verfahrens durchlaufen die Koordinaten zunächst die Vorverarbeitung, bevor sie abgespeichert werden. Die Daten werden als primitive Datentypen in Shared Preferences abgelegt. Die Verwendung von Shared Preferences ist notwendig, da die Daten auch nach Beendigung des Programmes noch zur Verfügung stehen müssen. Der „Clear-Screen“ Button löscht die Geste vom Bildschirm. Wird die Geste vom Nutzer vergessen, so könnte der „Show

Gesture“ Button an Wichtigkeit gewinnen. Er veranlasst es, die im Speicher hinterlegte Geste auf den Bildschirm zu zeichnen.

Eine weitere Benutzeroberfläche zum Speichern eines Pin Codes zeigt die nun folgende Abbildung (Abbildung 23).

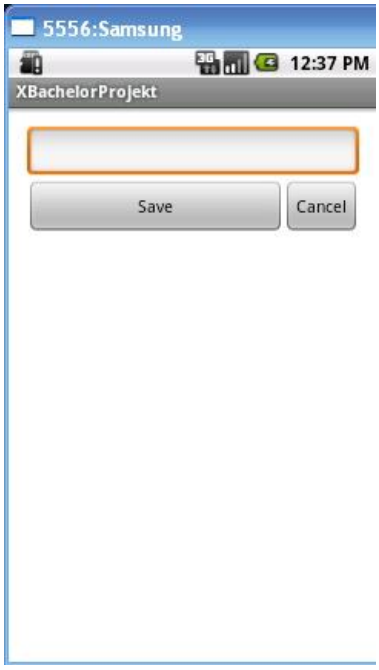

Abbildung 23: Pin Code speichern
Quelle: Screenshot

Wird das Eingabefeld berührt, öffnet sich eine virtuelle Tastatur, mit der ein Pin Code eingegeben werden kann. Mit dem Button „Cancel“ wird die aktuelle Eingabe komplett gelöscht. Der Button „Save“ speichert die Eingabe wiederum in Shared Preferences ab. Der Pin Code muss nicht zwingend eine Reihe von Zahlen sein. Theoretisch lassen sich auch Passwörter und Kombinationen aus Buchstaben, Zahlen und Sonderzeichen anlegen.

8.3 Service und Broadcastreceiver

Mit der Aktivierung der Authentifikation beim Hauptbildschirm wird im Hintergrund durch einen Intent ein Service gestartet. Mit dem Start dieses Services wird ein

dynamischer Broadcastreceiver registriert. Der Broadcastreceiver reagiert auf den vom System gesendeten „ACTION_SCREEN_ON“ Broadcast. Dieser wird versendet, wenn der Standby-Modus des Smartphones beendet, die Tastenspeere deaktiviert und der Touchscreen angeschaltet wird. Beim Empfang des Broadcast wird ein Intent verschickt, der die Authentifikations-Activity startet.

Zusätzlich wurde ein statischer Broadcastreceiver implementiert. Dieser Receiver wartet auf den „BOOT_COMPLETED“ Broadcast. Wenn das mobile Gerät neugestartet wurde, wird dieser Broadcast gesendet. Der Empfang startet zum einen wieder den Service und die damit verbundenen dynamischen Receiver und zusätzlich die jeweilige Authentifikations-Activity.

8.4 Authentifikations-Activitys

Die Authentifikations-Activitys werden jeweils nach dem Empfang der oben beschriebenen Broadcasts gestartet und gelangen damit in den Vordergrund. Die folgende Abbildung (Abbildung 24) zeigt die Oberfläche der Activitys.

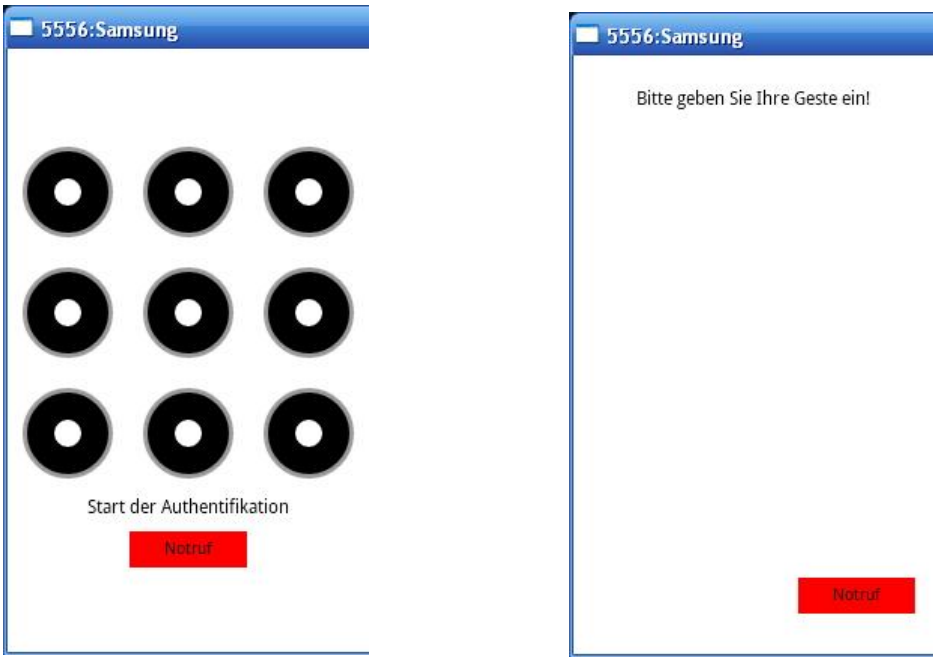

Abbildung 24: Authentifikations-Activitys
Quelle: Screenshots

Der Nutzer wird entsprechend der Art des Verfahrens aufgefordert, seine Geste einzugeben. Wird die Geste als richtig erkannt, so wird die Activity geschlossen und der Zugriff aufs Handy gestattet. Bei falscher Eingabe wird ein Zähler für Fehlversuche mit Eins addiert. Die Anzahl der Fehlversuche werden nach jeder missglückten Eingabe im unteren Bildrand für ca. zwei Sekunden eingeblendet. Hat der User seine Geste zehnmal falsch eingegeben, schließt sich die Activity zur Gesten-Eingabe und es wird nach dem Pin Code verlangt. Nachstehende Abbildung (Abbildung 25) zeigt die Activity zur Pin Code Eingabe.

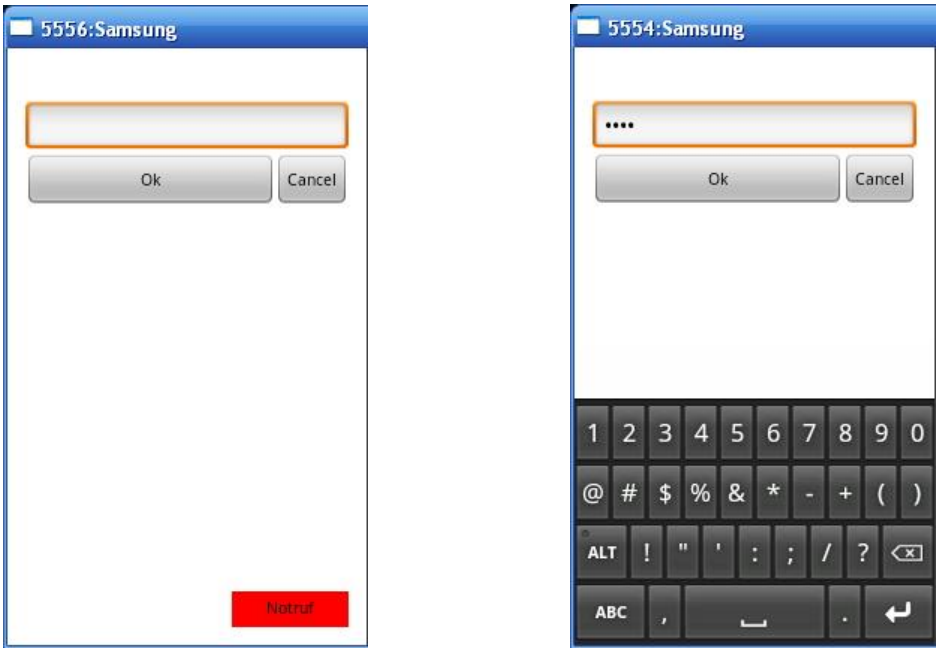

Abbildung 25: Activity zur Pin Code-Eingabe
Quelle: Screenshots

Im linken Teil des Bildes ist die Activity nach dem direkten Start durch einen Intent nach fehlerhafter Eingabe der Geste zu sehen. Im rechten Bild ist bereits ein Pin ins Eingabefeld eingegeben worden. Nach der ersten Installation der Anwendung ist der Standardwert für den Pin auf „0000“ festgelegt. Durch Betätigung des Buttons „Ok“ wird bei dem richtigen Code die Activity geschlossen und ein Zugriff

aufs Smartphone gewährt. Bei falscher Eingabe passiert nichts und das mobile Gerät bleibt weiterhin gesperrt.

8.5 Notruffunktion

Um die Sicherheit des menschlichen Wohles zu gewährleisten, muss die Möglichkeit bestehen, auch ohne korrekte Eingabe von Gesten oder dem Pin Code Notrufe abzusetzen. Hierfür gibt es in den beiden Activitys zur Gesten- und der Pin Code-Eingabe rote Notruf-Buttons. Betätigt man diesen Button öffnet sich eine neue Activity, die wie folgt aussieht (siehe Abbildung 26).

Abbildung 26: Activity Notrufnummer
Quelle: Screenshot

Hier sind zusätzlich zum Eingabefeld noch Nummer-Buttons eingefügt. Über diese Buttons lässt sich entsprechend eine Nummer wählen. Mit dem „Ok“ Button muss die Eingabe nochmal zusätzlich bestätigt werden. Es ist hierbei nur erlaubt, Notrufnummern zu wählen („110“ und „112“ für Deutschland, „911“ für die USA). Wird eine andere Nummer gewählt, wird der Nutzer darauf hingewiesen keine

Notrufnummer gewählt zu haben. Bei der Eingabe einer Notrufnummer und nach dem Betätigen des Buttons „Ok“, wird die Activity geschlossen und es öffnet sich die Standardoberfläche von Android zum Telefonieren. Die Nummer wurde übernommen, sodass lediglich der Anruf gestartet werden muss.

8.6 Unumgänglichkeit der Authentifikations-Activitys

Eine Hauptaufgabe der Anwendung ist es, den Zugriff von Unbefugten auf die Daten und Funktionen des mobilen Gerätes zu verhindern. Dazu müssen die Authentifikations-Activitys unumgänglich sein. Nur bei der richtigen Eingabe der Geste bzw. des Pins darf eine Freischaltung erfolgen. Ebenso muss nach einem erfolgten Notruf die Rückkehr zur Authentifikations-Activity gewährleistet sein.

Hierfür wurden zunächst für die entsprechenden Activitys die Hardwaretasten des Smartphones deaktiviert. Unter normalen Umständen kann jede Anwendung durch den Home-Button des Smartphones beendet werden und der Android Homescreen wird angezeigt. Diese Funktion wird unterdrückt. Ebenso führt die Betätigung der Zurück- und Menü-Taste zu keiner Reaktion des Smartphones. Lediglich die Ein/Aus-Taste behält ihre Funktionalität. Dies stellt aber kein weiteres Problem dar, da nach einem Neustart die Authentifikations-Activity wieder gestartet wird. Somit scheinen die entsprechenden Activitys auf den ersten Blick zunächst unabwendbar zu sein. Nach der Durchführung von mehreren Tests stellte sich jedoch ein schwerwiegendes Problem in den Weg. Die sogenannte Status Bar des Android Betriebssystems (siehe rote Markierung in Abbildung 27) lässt sich komplett herunter scrollen und verdeckt die gesamte Authentifikations-Activity. Dies hat wiederum zur Folge, dass sämtliche Hardwaretasten wieder aktiviert sind und der Nutzer mit Hilfe des Home-Buttons auf den Homescreen gelangt.

Abbildung 27: Status Bar
Quelle: Screenshot

Die erste Idee zur Lösung des Problems war es, die Authentifikations-Activitys als sogenannten Fullscreen-Activitys zu implementieren. Eine Fullscreen-Activity nimmt den kompletten Bildschirm in Anspruch und überdeckt die Status Bar. Bei Tests stellte sich ein unbefriedigendes Ergebnis heraus, da die Status Bar nur gelegentlich und nicht vorhersehbar durch die Fullscreen-Activity verdeckt wurde. Die endgültige Problemlösung stellt ein im Hintergrund laufender Programmteil dar. Dieser veranlasst ein regelmäßiges „Zusammenfallen“ der Status Bar. Damit wird das komplette Herunterscrollen vermieden und die Authentifikations-Activity mit ihren Eigenschaften bleibt stets im Vordergrund. Das Verhindern des Expandierens der Status Bar wird durch Polling in einer Endlosschleife realisiert. Dies hat den großen Nachteil, dass viel Rechenzeit verschwendet und dementsprechend der Akku des Gerätes stark belastet wird.

Ein weiteres Problem ergibt sich durch die Notruffunktion. Um den Anruf zu starten, muss die eigene Activity verlassen werden. Damit verliert man zunächst sämtliche Kontrolle über die Aktivitäten des Nutzers. Ausgehend von einem standardmäßigen Ablauf eines Telefonates, wird der Anruf gestartet und nach

einer unbestimmten Zeit wieder beendet. Durch einen vom Android SDK bereitgestellten PhoneStateListener kann das Beenden des Gespräches festgestellt und die Authentifikations-Activity wieder gestartet werden. Das Problem entsteht dann, wenn das Telefonat erst gar nicht gestartet oder es nicht wieder beendet wird. Um einen Zugriff auf Telefonfunktionen und Daten zu vermeiden, wird eine transparente Oberfläche über den gesamten Bildschirm gelegt. Die für den Nutzer nicht sichtbare Oberfläche lässt sämtliche Touchevents wie gewohnt durch, sodass ein Telefonat entsprechend gestartet und beendet werden kann. Die Oberfläche ermöglicht es die Touchevents zu registrieren und mitzuzählen. Nach einer festgelegten Anzahl von Fingerberührungen mit dem Display wird die Authentifikations-Activity wieder gestartet. Die Anzahl ist groß genug gewählt, um den Ablauf des Telefongespräches zu ermöglichen aber klein genug, um das Deaktivieren der Authentifikations-Anwendung sowie den Missbrauch von Telefonfunktionen zu verhindern.

9 Parametereinstellung mit Matlab

Der Algorithmus zur Gestenauthentifikation beinhaltet einige Parameter die einen maßgeblichen Einfluss auf Erfolg oder Misserfolg beim Gestenvergleich haben. Eine zu weiche Einstellung ermöglicht die Authentifikation bereits durch ähnliche Gesten. Während eine zu harte Abstimmung wohlmöglich nie zu einer erfolgreichen Authentifikation führen kann. Eine sorgfältige Abstimmung aller Parameter ist dementsprechend unerlässlich.

9.1 Anwendung zur Speicherung mehrerer Gesten

Um mit Hilfe von Matlab die Parameter gezielt anzupassen, werden zunächst Datensätze von Gesten benötigt. Hierfür wurde eine zusätzliche Anwendung für das Smartphone entwickelt. Die Zeichenoberfläche sowie der Vorverarbeitungsalgorithmus wurden von dem Gesten-Authentifikations-

Programm übernommen. Hinzugefügt wurden jeweils ein Button zum Speichern und zum Löschen. Nach dem Start der Anwendung kann auf der Zeichenfläche eine Geste eingegeben werden. Betätigt man anschließend den Speichern-Button werden die Koordinatenpaare in einer Textdatei auf dem Telefonspeicher hinterlegt und die Geste vom Display gelöscht. Der Einfachheit halber werden in einer zweiten Textdatei die Zeitwerte und Winkel gespeichert. Bei fehlerhafter Eingabe kann durch den Lösch-Button die Zeichenfläche ebenfalls wieder geleert werden. Die Anwendung ermöglicht beliebig viele Datensätze abzuspeichern. Nach jeder Eingabe einer Geste und der folgenden Speicher-Button-Betätigung werden neue Textdateien angelegt. Mit dem zum Smartphone gehörendem USB-Kabel lassen sich die Textdateien einfach auf einen Rechner übertragen.

Das vollständige Eclipse-Projekt, inklusive dem dokumentierten Quellcode der Anwendung zur Speicherung der Gesten, befindet sich auf der beigefügten CD und kann bei Herrn Prof. Dr.-Ing. Robert Fitz eingesehen werden.

9.2 Auswertung mit Matlab

Die Textdateien mit den Datensätzen der Gesten lassen sich mit Matlab einlesen. Das Ergebnis sind mehrere Matrizen die jeweils eine Geste und die dazugehörigen Zeit- und Winkelwerte wiederspiegeln. Für die Parametereinstellung wurde der Vergleichsalgorithmus des Points-Matching-Verfahrens zunächst eins-zu-eins in Matlab übernommen und implementiert. Ausgenommen davon ist die Auswertung der Informationen der Lagesensoren. Die erste Eingabe und entsprechend die Daten der ersten Textdatei bilden die Geste, mit der alle anderen Eingaben verglichen werden. Um einen ersten optischen Eindruck zu gewinnen, werden die zu vergleichenden Gesten paarweise in einem Matlab-Plot dargestellt. Hier lässt sich bereits über Erfolg oder Misserfolg des Punkteabstandsvergleiches diskutieren. Die folgende Abbildung (Abbildung 28) zeigt Beispiele aus einem Datensatz.

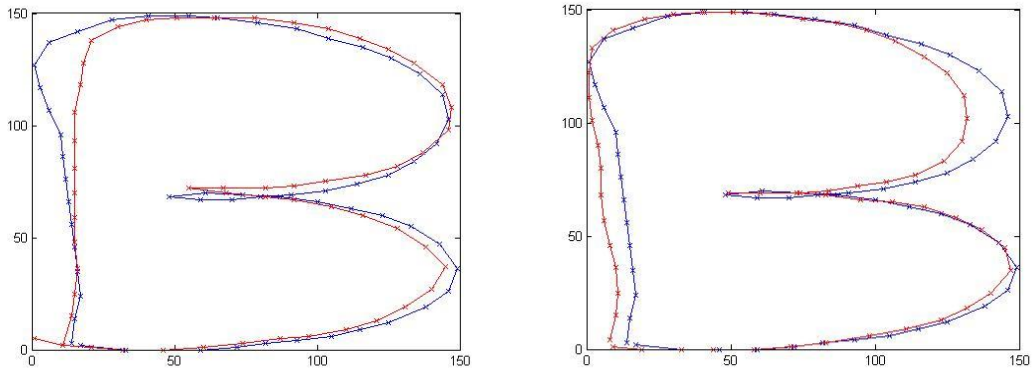

Abbildung 28: Matlab-Plot Beispielgeste 1

Augenscheinlich zu erkennen, sollte der Vergleich dieser Gesten miteinander zu einem positiven Ergebnis führen. Zwei Gegenbeispiele sind in der nachstehenden Abbildung (Abbildung 29) gezeigt.

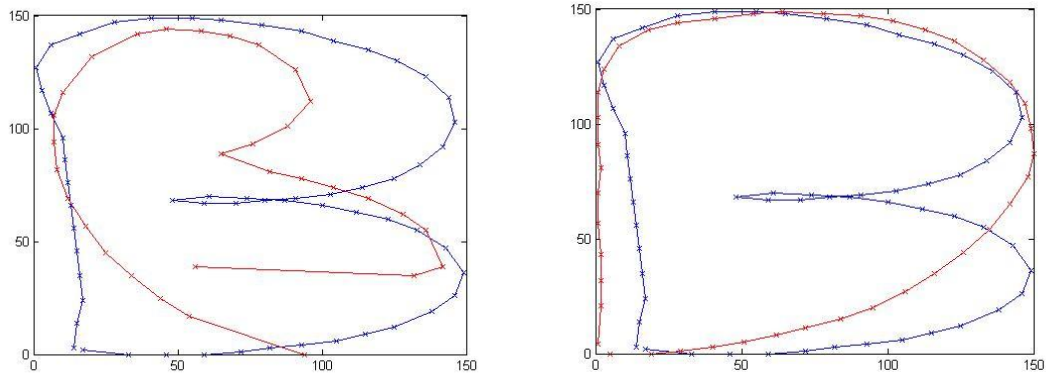

Abbildung 29: Matlab-Plot Beispielgeste 2

Im linken Teil der Abbildung 29 wurde die rote Geste zu ungenau eingegeben. Die unterschiedliche Anzahl von Koordinaten sowie die Punkteabstandsberechnung würde hier eine fehlgeschlagene Authentifikation zur Folge haben. Im rechten Abbildungsteil sind zwei unterschiedliche Gesten (in diesem Fall Großbuchstaben des lateinischen Alphabetes) gezeichnet. Sie sind sich sehr ähnlich und unterscheiden sich nur durch die Einbuchtung des „B“. Die Parametereinstellung muss eine Unterscheidung der beiden Gesten ermöglichen.

Diese Matlab-Plots geben allerdings noch keine Informationen über die Zeit, die für die Eingabe benötigt worden ist. Um auch hier einen ersten Eindruck zu bekommen, werden die aufaddierten Zeitwerte ebenfalls in einem Plot begutachtet (siehe Abbildung 30).

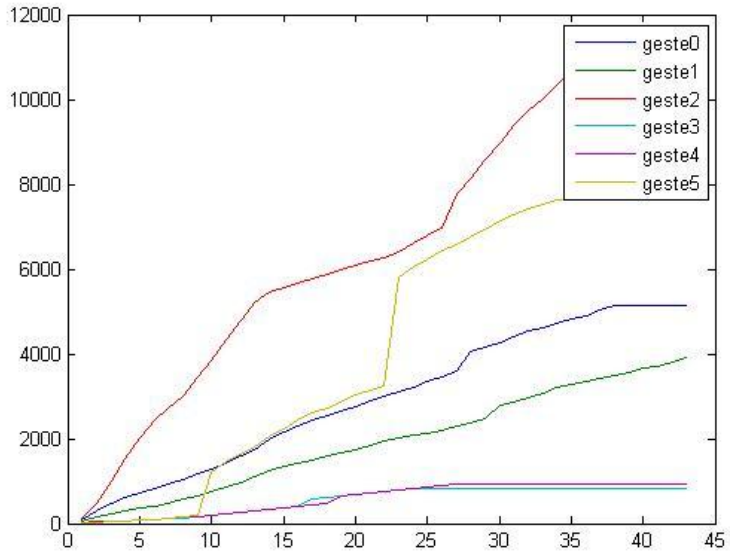

Abbildung 30: Matlab Plot Beispiel Zeit

Auf der Y-Achse ist die Zeit in Millisekunden aufgetragen. Die X-Achse stellt die einzelnen Zeitabschnitte dar. Die Zeitwerte der mittleren blauen Linie (geste0) werden mit den jeweils anderen Werten verglichen. Hieraus lässt sich bereits erkennen, dass lediglich die grüne Kurve ungefähr in der gleichen Zeit und im gleichen Rhythmus eingegeben wurde. Die anderen Kurven sind entweder zu schnell oder entsprechend zu langsam, im Bezug zur Vergleichszeit, gezeichnet worden. Wie man anhand der Linie von geste5 erkennen kann, wurde im mittleren Teil eine kleine Zeichenpause eingelegt. Auch dies kann ein wichtiges Merkmal einer persönlichen Geste sein.

Abschließend durchlaufen alle Daten den Vergleichsalgorithmus des Points-Matching-Verfahrens in Matlab. Der Rückgabewert ist ein Vektor mit den Authentifikations-Ergebnissen. Wird eine Null zurückgegeben so wäre die Authentifikation erfolgreich gewesen. Bei einer Eins, ist der Unterschied der

Anzahl der Punkte beider Geste zu groß. Beträgt der Rückgabewert den Wert Zwei ist die Punkteabstandsüberprüfung fehlgeschlagen. Bei zu großen Zeitdifferenzen wird der Wert Drei zurückgegeben.

Anhand von vielen aufgenommen Testdaten unterschiedlicher Gesten, den Ergebnisvektoren des Vergleichsalgorithmus und den Plots lässt sich nun eine Feinabstimmung der Parameter realisieren.

Das Matlab-Script, die Matlab-Funktionen sowie einige Testdaten befinden sich auf der beigefügten CD und kann bei Herrn Prof. Dr.-Ing. Robert Fitz eingesehen werden.

9.3 Parameter

Der erste wichtige Parameter ist der, bereits in der Vorverarbeitung benötigte, Skalierungsfaktor. Er legt fest auf welchen Bereich die Geste skaliert wird. Bei einem zu kleinen Skalierungsfaktor wird die Geste zunehmend winziger und wichtige Informationen gehen verloren. Bei einem überdimensionierten Skalierungsfaktor können die Punkteabstände zu groß werden. Durch Tests stellte sich ein Skalierungsfaktor von scaleFactor = 150 als geeignet heraus.

Im Weiteren wird zwischen den Parametern des Points-Matching- und des Rises-Matching-Verfahrens unterschieden. Im Grunde sind für beide Verfahren gleiche Einflussgrößen erforderlich. Die Werte unterscheiden sich aber voneinander. In folgender Tabelle sind alle Parameter mit ihren, durch die Tests ermittelten, Werte aufgelistet.

Parameter	Points-Matching	Rises-Matching
Quotient aus Anzahl der Punkte/Anstiegswechsel	0,85 < Quotient < 1,2	0,75 < Quotient < 1,3
Maximaler Punkteabstand	45	-
Prozentuale Übereinstimmung der Punkteabstände/Anstiegswechsel	90%	100%
Maximale Zeitdifferenz	90 ms	240 ms
Prozentuale Übereinstimmung der Zeiten	80%	68%

Tabelle 1: Übersicht über die Parameter der Authentifikationsverfahren
Quelle: eigene Darstellung

Der Quotient aus der Anzahl der Punkte bzw. aus der Anzahl der Anstiegswechsel der zu vergleichenden Gesten ermöglicht eine Unterscheidung zwischen ähnlichen Gesten und verhindert zugleich eine erfolgreiche Authentifikation bei der Eingabe eines kleinen Teiles der Geste. Der maximale Punkteabstand spiegelt den Abstand zweier Koordinaten wieder, die noch als übereinstimmend erkannt werden. Genau gleich verhält es sich mit der maximalen erlaubten Zeitdifferenz zwischen zwei Punkten bzw. zwei Anstiegswechsel. Abschließend wurde noch eine geeignete prozentuale Übereinstimmung der Punkteabstände bzw. Anstiegswechsel und der Zeiten, die für eine erfolgreiche Authentifikation notwendig sind, ermittelt.

10 Test der Anwendung

Nach der Abstimmung aller Parameter wurde die Anwendung zum Test einem Publikum vorgestellt. Die Testgruppe von insgesamt 11 Personen im Alter zwischen 15 und 36 Jahren, spiegelt eine wichtige Zielgruppe von Smartphones und deren vielfältige Anwendungen wieder. Im Vorhinein wurde das Programm in einer kurzen Präsentation vorgestellt und auf wichtige Bestandteile des Algorithmus (Auswertung der Eingabezeit und der Lagesensoren) hingewiesen.

Aufgeteilt ist der Test in vier Phasen, die jeder Teilnehmer mit seiner individuellen Geste durchläuft. Nachdem die Testergebnisse ausgewertet und analysiert wurden, erfolgte ein zweiter Testverlauf nach dem selben Prinzip, an dem fünf Versuchsteilnehmern teilnahmen.

10.1 Test mit den ermittelten Parametern

Zunächst werden die Testphasen mit den ermittelten Parametern aus Punkt 9.3 durchgeführt.

10.1.1 Phase 1: Mehrfache Authentifikation

In der ersten Phase werden die Testpersonen gebeten sich eine Geste auszudenken. Diese soll aber keinem weiteren Versuchsteilnehmer mitgeteilt werden. Nach der Speicherung der Geste, werden die Teilnehmer aufgefordert sich jeweils zehnmal bei dem Mobiltelefon zu authentifizieren. Der Erfolg bzw. Misserfolg bei der Authentifikation sowie der Grund für den Misserfolg werden dabei festgehalten. In den folgenden Tabellen sind die Testergebnisse geordnet nach den Testpersonen aufgeführt.

Testperson 1, Geste: Herz, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Anz. Punkte	Anz. Punkte	Erfolg-reich	Erfolg-reich

Testperson 2, Geste: Schnecke, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Erfolg-reich	Punkte-vergleich	Erfolg-reich

Testperson 3, Geste: 9, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Anz. Punkte	Anz. Punkte	Anz. Punkte	Punkte-vergleich	Erfolg-reich

Testperson 4, Geste: Bohne, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich

Testperson 5: Geste: 4, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Punkte-vergleich	Zeit	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Anz. Punkte	Erfolg-reich	Punkte-vergleich	Erfolg-reich

Testperson 6: Geste: Kreis, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Zeit	Zeit	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Punkte-vergleich	Erfolg-reich

Testperson 7, Geste: Ω, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Zeit	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 8, Geste: S, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Erfolg-reich	Erfolg-reich	Anz. Punkte	Punkte-vergleich	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 9, Geste: Karo, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Anz. Punkte	Erfolg-reich	Anz. Punkte	Anz. Punkte

Testperson 10, Geste: Tannenbaum, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anstiegs-Vergl.	Anstiegs-Vergl.	Erfolg-reich	Anstiegs-Vergl.	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich	Anz. Anst.	Anstiegs-Vergl.	Anstiegs-Vergl.

Testperson 11, Geste: Haus vom Nikolaus, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich.	Anstiegs-Vergl.	Erfolg-reich

Tabelle 2: Übersicht Testergebnisse aus Test 1 in Tabellen zur Phase 1

Durch die freie Wahl der Teilnehmer, sich ihre persönliche Geste zu suchen, kam lediglich zwei Mal das Rises-Matching-Verfahren zum Einsatz. Der Test dieses Verfahrens fällt daher etwas schmaler aus. Die Versuchsteilnehmer wählten im Allgemeinen eher einfachere und leichter reproduzierbare Gesten.

Die neun Testpersonen, bei denen das Points-Matching-Verfahren herangezogen wurde, erzielten eine erfolgreiche Authentifikation mit 58,89%. Im Schnitt ist also mehr als jede zweite Eingabe erfolgreich. Leichte Unterschiede zeigen sich bei

Testpersonen mit unterschiedlich viel Erfahrung beim Umgang von Smartphones und Touchscreens. Die Versuchsteilnehmer mit wenig bis gar keiner Erfahrung scheitern öfter an der Authentifikation.

Die häufigste Ursache für eine fehlgeschlagene Authentifikation ist, in 23,33% aller Eingaben, eine zu große Differenz der Anzahl der Punkte. Gefolgt von einem fehlerhaften Punktevergleich mit 13,33% und zu großen zeitlichen Abweichungen mit 4,44%.

Beim Rises-Matching-Verfahren gelang eine erfolgreiche Authentifikation mit 55%. Dabei gab es aber enorme Unterschiede bei den Gesten. Während der Tannenbaum mit über 30 Monotoniewechseln eine sehr schwer reproduzierbare Geste darstellt und nur dreimal erfolgreich nachgezeichnet werden konnte, ist das Haus vom Nikolaus mit unter 20 Monotoniewechseln eher einfach und führte daher zu acht gelungenen Authentifikationen.

Mit 40% scheiterten die Versuchsteilnehmer bei diesem Verfahren am häufigsten an dem Anstiegs-Vergleich sowie mit 5% an der zu großen Differenz der Anstiegs-Anzahlen der zu vergleichenden Gesten.

10.1.2 Phase 2: Nicht bekannte Geste

In Phase Zwei sind die Testpersonen aufgefordert, die Ihnen unbekannte Geste eines anderen Teilnehmers zu erraten und somit Zugriff auf das mobile Gerät zu erhalten. Dabei sind zehn Versuche erlaubt, da nach dem zehnten Fehlversuch die Pin-Code-Eingabe aktiviert wird. Hier stellte sich, wie zu erwarten war, ein klares Ergebnis heraus. Keine Geste konnte innerhalb der zehn Versuche erfolgreich erraten bzw. eine erfolgreiche Authentifikation erzielt werden.

10.1.3 Phase 3: Bekannte Geste

Im Gegensatz zur Phase Zwei sind den Testpersonen die Gesten nun bekannt. Die Versuchsteilnehmer versuchen erneut mit der Geste einer anderen Person die Authentifikation erfolgreich zu gestalten. Die Testergebnisse sind hierbei stark abhängig von der Geste selbst. Formen, die im alltäglichen Leben eine größere

Rolle spielen (Herz, Ziffern, Buchstaben), werden im Durchschnitt mit drei Versuchen von den Testpersonen erfolgreich umgesetzt. Die eher außergewöhnlichen Gesten (Bohne, Schnecke, Karo) können erst dann zur erfolgreichen Authentifikation führen, wenn die Gesteneingabe von der entsprechenden Testperson mehrfach vor den Augen der anderen Versuchsteilnehmer vorgeführt wird. Eine wesentliche Rolle spielt auch die Zeit mit der die Geste eingegeben wurde. Ist die Dynamik nicht bekannt, hilft auch das Wissen über die Gestenform nicht aus.

10.1.4 Phase 4: Wiederholung der mehrfachen Authentifikation

In der ersten Testphase wurden die Testpersonen direkt nach dem Abspeichern ihrer persönlichen Geste gebeten sich zehnmal beim Mobiltelefon zu authentifizieren. Dabei ist der Fingerverlauf über den Touchscreen noch gut im Kurzzeitgedächtnis verankert und führt daher zu guten bis sehr guten Authentifikations-Ergebnissen. Nach Abschluss von Phase Zwei und Phase Drei sind einige Minuten vergangen. Die Versuchsteilnehmer werden nun gebeten den Test aus Phase Eins zu wiederholen. Die nachfolgenden Tabellen zeigen das Testergebnis.

Testperson 1, Geste: Herz, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Erfolg-reich	Anz. Punkte	Punkte-vergleich	Anz. Punkte	Anz. Punkte	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich

Testperson 2, Geste: Schnecke, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Punkte-vergleich	Punkte-vergleich	Zeit	Erfolg-reich

Testperson 3, Geste: 9, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Anz. Punkte	Anz. Punkte	Punkte-vergleich	Punkte-vergleich	Anz. Punkte	Anz. Punkte	Zeit	Erfolg-reich	Punkte-vergleich

Testperson 4, Geste: Bohne, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Anz. Punkte	Anz. Punkte	Punkte-vergleich	Punkte-vergleich	Erfolg-reich	Erfolg-reich

Testperson 5: Geste: 4, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Anz. Punkte	Punkte-vergleich	Erfolg-reich	Zeit	Erfolg-reich	Anz. Punkte	Punkte-vergleich	Punkte-vergleich	Erfolg-reich

Testperson 6: Geste: Kreis, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Zeit	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 7, Geste: Ω, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Punkte-vergleich	Punkte-vergleich	Zeit	Zeit	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Anz. Punkte

Testperson 8, Geste: S, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Anz. Punkte	Anz. Punkte	Anz. Punkte	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich

Testperson 9, Geste: Karo, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Anz. Punkte	Anz. Punkte	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Anz. Punkte	Erfolg-reich	Anz. Punkte	Erfolg-reich

Testperson 10, Geste: Tannenbaum, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Anst	Anstiegs-Vergl.	Anstiegs-Vergl.	Anstiegs-Vergl.	Anstiegs-Vergl.	Erfolg-reich	Anz. Anst.	Anstiegs-Vergl.	Anstiegs-Vergl.	Anz. Anst.

Testperson 11, Geste: Haus vom Nikolaus, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Anst	Anstiegs-Vergl.	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich	Erfolg-reich	Zeit	Erfolg-reich	Erfolg-reich

Tabelle 3: Übersicht Testergebnisse aus Test 1 in Tabellen zur Phase 4

Auch bei diesem Test stellten sich die Erwartungen ein. Die Quote für eine erfolgreiche Authentifikation beim Points-Matching-Verfahren fiel auf 40%. Die häufigste Ursache für eine fehlerhafte Eingabe bleibt die zu große Differenz der Punkte mit 33,33%, weiterhin gefolgt vom Punktevergleich mit 20% und der Zeitdifferenzen mit 6,66%.

Beim Rises-Matching-Verfahren fiel die erfolgreiche Authentifikation von 55% auf 35% der Eingaben. Der Anstiegs-Vergleich bleibt mit 45% häufigste

Fehlerursache. In 15% aller Eingaben führte die Differenz der Anstiegs-Anzahlen und in 5% eine zeitliche Abweichung zu einer fehlgeschlagenen Authentifikation.

10.1.5 Testfazit

Die Testergebnisse hinsichtlich der Authentifikation sind zufriedenstellend ausgefallen. Zwar ist eine 40%ige (Points-Matching) bzw. 35%ige (Rises-Matching) Wahrscheinlichkeit einer erfolgreichen Gesteneingabe etwas gering, jedoch würde eine Lockerung der Parameter zu einer einfacheren, aber ungewollten Authentifikation durch Dritte führen. Beim Gebrauch der Anwendung über einen längeren Zeitraum führen darüber hinaus gesammelte Erfahrungen und Verankerungen im Langzeitgedächtnis zu besseren Authentifikations-Ergebnissen.

Das Testergebnis von der Geste des Tannenbaumes zeigt deutlich, dass zu komplizierte Gesten nur sehr selten zur erfolgreichen Authentifikation führen. In solchen Fällen weist der Algorithmus noch einige Schwächen auf. Jedoch tendierte die große Anzahl der Teilnehmer zu einfacheren Gesten, die mit dem Points-Matching-Verfahren gut abgedeckt werden konnten.

Äußerst beachtlich sind die Testergebnisse aus Phase Zwei. Ist die Geste unbekannt, so ist eine Authentifikation nicht möglich.

Während der Testphasen stellte sich ein kleines Problem der Implementierung heraus. Es gab Schwierigkeiten mit der Auswertung der Lagesensoren, sodass eine Authentifikation nicht mehr möglich war. So wurde für den Test kurzfristig die Abfrage der richtigen Orientierung entfernt. Im Nachhinein wurde die Ursache des Problems gesucht, gefunden und anschließend behoben.

10.2 Test mit gelockerten Parametern

Auf Grund der etwas zu gering ausgefallenen Wahrscheinlichkeit einer erfolgreichen Authentifikation und zur Überprüfung der aufgestellten Hypothese, dass eine Lockerung der Parameter zu einer einfacheren, aber ungewollten Authentifikation durch Dritte führen kann, wird der Test nochmals mit einer kleineren Testgruppe durchgeführt. Hierfür wurden die entsprechenden Parameter gelockert und in folgender Tabelle 4 aufgelistet.

Parameter	Points-Matching	Rises-Matching
Quotient aus Anzahl der Punkte/Anstiegswechsel	0,75 < Quotient < 1,3	0,7 < Quotient < 1,4
Maximaler Punkteabstand	60	-
Prozentuale Übereinstimmung der Punkteabstände/Anstiegswechsel	85%	Zwei ermittelte Anstiegswechsel dürfen sich unterscheiden
Maximale Zeitdifferenz	120 ms	240 ms
Prozentuale Übereinstimmung der Zeiten	80%	68%

Tabelle 4: Übersicht über die gelockerten Parameter der Authentifikationsverfahren
Quelle: eigene Darstellung

10.2.1 Phase 1: Mehrfache Authentifikation

In den folgenden Tabellen sind die Testergebnisse geordnet nach den Testpersonen aufgeführt.

Testperson 1, Geste: 2, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich

Testperson 2, Geste: Welle, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Zeit	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 3, Geste: Möwe, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Erfolg-reich

Testperson 4, Geste: Dreieck, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 5, Geste: Haus vom Nikolaus, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich

Tabelle 5: Übersicht Testergebnisse aus Test 2 in Tabellen zur Phase 1

Beim Points-Matching-Verfahren mit gelockerten Parametern wurde eine erfolgreiche Authentifikation mit einer Wahrscheinlichkeit von 85% erzielt. Ein zu großer Unterschied in der Anzahl der Punkte ist mit 7,5% die häufigste Ursache für das Scheitern am Algorithmus. Der Punktevergleich mit 5% und die zu großen Zeitunterschiede mit 2,5% bilden die weiteren Ursachen der fehlerhaften Authentifikation.

Mit einer Wahrscheinlichkeit von 80% gelingt beim Rises-Matching-Verfahren eine erfolgreiche Authentifikation. Der Vergleich der Monotoniewechsel führt in 20% der Fälle zur fehlgeschlagenen Zugriffsberechtigung. Dass die Zeit sowie die Anzahl der Anstiege bei diesem Testergebnis keine Rolle spielen, ist zum einen auf die gelockerten Parameter, allerdings hauptsächlich auf die geringe Anzahl von Testpersonen zurückzuführen.

10.2.2 Phase 2: Nicht bekannte Geste

Die Testpersonen wurden nun wieder angehalten die Geste eines anderen Teilnehmers zu erraten, um Zugriff auf das Mobiltelefon zu erhalten. Dies gelang bei allen Gesten des Points-Matching-Verfahrens erstaunlich einfach. Nach durchschnittlich fünf bis sechs Fehleingaben wurden die Gesten erfolgreich umgesetzt und ein unbefugter Zugriff damit erlaubt. Die Geste von Testperson 5 (Haus vom Nikolaus) konnte hingegen nicht erraten werden.

10.2.3 Phase 3: Bekannte Geste

Die Ergebnisse aus Phase 2 machen die Durchführung der Phase 3 nahezu überflüssig, da bereits die unbekannte Geste zur unbefugten Authentifikation führte. Einzig die Geste des Rises-Matching-Verfahrens wurde getestet. Hier konnte durchschnittlich jeder dritte Versuch umgesetzt werden. Begünstigt durch die Lockerung war es möglich, sogar durch verschiedene Varianten des Hauses vom Nikolaus die Authentifikation erfolgreich zu gestalten.

10.2.4 Phase 4: Wiederholung der mehrfachen Authentifikation

In den folgenden Tabellen sind die Testergebnisse geordnet nach den Testpersonen aufgeführt.

Testperson 1, Geste: 2, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Anz. Punkte	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 2, Geste: Welle, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anz. Punkte	Zeit	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 3, Geste: Möwe, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Erfolg-reich	Erfolg-reich	Punkte-vergleich	Punkte-vergleich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 4, Geste: Dreieck, Authentifikations-Verfahren: Points-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Zeit	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich

Testperson 5, Geste: Haus vom Nikolaus, Authentifikations-Verfahren: Rises-Matching

Versuch	1	2	3	4	5	6	7	8	9	10
Ergebnis	Anstiegs-Vergl.	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich	Erfolg-reich	Erfolg-reich	Anstiegs-Vergl.	Erfolg-reich	Erfolg-reich

Tabelle 6: Übersicht Testergebnisse aus Test 2 in Tabellen zur Phase 4

Die Wahrscheinlichkeit der erfolgreichen Authentifikation beim Points-Matching-Verfahren bleibt, wie schon in Phase 1, bei 85%. Die Fehlerursachen verteilen sich gleichmäßig mit jeweils 5% auf die Anzahl der Punkte, auf die Zeit sowie den Punktevergleich.

Beim Rises-Matching-Verfahren fällt die Wahrscheinlichkeit für eine erfolgreiche Zugriffsberechtigung von 80% auf 70%. Gleichzeitig steigt der fehlerhafte Anstiegs-Vergleich von 20% auf 30%.

10.2.5 Testfazit

Die mäßigen Testergebnisse des ersten Testes wurden erwartungsgemäß wesentlich verbessert. Die erfolgreichen Authentifikationen beim Points-Matching-Verfahren steigen im Vergleich zum ersten Test deutlich von 40% auf 85%. Ähnlich verhält sich die höhere Erfolgswahrscheinlichkeit beim Rises-Matching-Verfahren von 70% im Gegensatz zu den 35% des ersten Versuches.

Äußerst unbefriedigend sind hingegen die Ergebnisse aus Testphase zwei. Der Zugriff durch Unbefugte wird viel zu einfach ermöglicht. Durch die Lockerung genügt es, bereits nur Teile der Geste zu erraten, um trotzdem eine erfolgreiche Authentifikation zu erzielen.

Im Endeffekt muss herausgestellt werden, dass die Parameter deutlich zu stark gelockert worden sind. Um eine optimale Abstimmung zwischen einer möglichst hohen Erfolgsquote und einer sicheren Authentifikation zu gewährleisten sind noch einige Test mit unterschiedlichen Parametern nötig.

11Ausblick und Fortführung

Die Anwendung zur gestenbasierten Authentifikation lässt noch sehr viel Spielraum zur Fortführung des Projektes. Das bis hierher entwickelte Programm bietet zunächst nur einen Schutz vor unerlaubtem Zugriff auf Telefonfunktionen und den internen Telefonspeicher. Um einen kompletten Schutz der Daten vor Dritten zu gewährleisten, ist eine Verschlüsselung der SD-Speicherkarte unumgänglich. Die neueren Android-Versionen bringen bereits eine passwortgeschützte Verschlüsselung mit, die sich mit der Anwendung koppeln lässt. Zur besseren Kontrolle und für höhere Sicherheitsstandards ist die Implementierung eines eigenen Verschlüsselungsalgorithmus allerdings die bessere Wahl.

Die Implementierung der Notruffunktion stellt noch keine zufriedenstellende Lösung dar. Der transparente Hintergrund, der die Touchevents auf dem Display

registriert und somit bei einem untypischen Telefonverhalten zur Authentifikations-Activity zurückführt, ist lediglich eine Notlösung. Im Rahmen der Bachelorthesis konnte leider keine bessere Variante gefunden werden.

Um die Anwendung marktreif zu gestalten sind eine programminterne Anleitung bzw. Hilfetexte zusätzlich zu implementieren. Außerdem sollte eine Internationalisierung erfolgen, sodass sämtliche Texte nicht nur in der deutschen Sprache sondern auch im englischsprachigen Raum verständlich sind.

12 Fazit

Mit der Entwicklung einer Anwendung zur gestenbasierten Authentifikation ist ein komfortables Sicherheitskonzept für mobile Endgeräte mit Android Betriebssystem entstanden. Auch wenn einige Funktionen nicht bis zur endgültigen Zufriedenheit fertiggestellt werden konnten, so ist mit dieser Arbeit ein Schritt in zukünftige Authentifikations-Mechanismen getan. Besonders die Auswertung von Zeitwerten als zusätzliches Gesten-Merkmal ist eine Neuerung auf diesem Gebiet. Die freie Entfaltungsmöglichkeit der Nutzer in der Wahl ihrer individuellen Geste und die daraus resultierenden, nahezu unendlichen Möglichkeiten beim Points-Matching bzw. Rises-Matching-Verfahren erhöhen die Sicherheit vor Zugriff von Unbefugten im Vergleich zu bereits etablierten Musterauthentifikationen wie das Bounding-Circel-Verfahren. Daraus resultierend besteht die Möglichkeit, dass sich die, in dieser Bachelorarbeit entwickelten Authentifikations-Verfahren in Zukunft auf dem Markt durchsetzten können.

13 Literaturverzeichnis

[1] Android Emulator. URL:
http://developer.android.com/tools/help/emulator.html, (Stand: 05.12.2013, 17:30 Uhr)

[2] Awuku, Yaw: Was ist eigentlich Android? URL: http://www.t-online.de/handy/smartphone/id_49385388/was-ist-android-google-android-klipp-und-klar-.html (Stand: 06.01.2014, 15:50 Uhr)

[3] Becker, Arno; Pant, Marcus: Android Grundlagen und Programmierung. Heidelberg: dpunkt Verlag GmbH, 2009

[4] Duden. Berlin: Bibliographisches Institut GmbH, 2013, URL:
www.duden.de/rechtschreibung/Geste (Stand: 09.12.2013, 16:30 Uhr)

[5] Eclipse: About the Eclipse Foundation. URL: http://www.eclipse.org/org/ (Stand: 20.12.2013, 11:45 Uhr)

[6] Eclipse: FAQ What is Eclipse? URL: http://wiki.eclipse.org/FAQ_What_is_Eclipse%3F (Stand: 20.12.2013, 11:45 Uhr)

[7] Einführung in Android. URL: http://www.android.com/intl/de/about/ (Stand: 06.01.2014, 15:00 Uhr)

[8] Get the Android SDK. URL:
http://developer.android.com/sdk/index.html, (Stand: 05.12.2013, 15:45 Uhr)

[9] ITWissen. Das große Online-Lexikon für Informationstechnologie. Peterskirchen: DATACOM Buchverlag GmbH,
URL: www.itwissen.info/definition/lexikon/Authentifizierung-authentication.html (Stand: 09.12.2013, 16:10 Uhr)

[10] Kuhn, Daniel: Android: Von Cupcake bis Ice Cream Sandwich — alle Versionen in der Übersicht. 2011, URL: http://www.androidnext.de/schwerpunkt/android-von-cupcake-bis-ice-cream-sandwich-alle-versionen-in-der-ubersicht/ (Stand: 06.01.2014, 15:20 Uhr)

[11] Kurterbach, G.; Hulteen, E.A.: The Art of Human-Computer Interface Design. 1990

[12] Lemke, Jan: Entwicklung einer gestenbasierten Authentifikation für Symbian 60: Diplomarbeit HAW Hamburg, 2012

[13] Oracle: Why Java? URL: http://www.oracle.com/us/technologies/java/overview/index.html (Stand: 16.12.2013, 11:30 Uhr)

[14] Ratgeber. Tipps und Tricks: Was genau ist Android? URL: http://www.ratgeber.org/207-was-genau-ist-android.html, (Stand: 06.01.2014, 15:00 Uhr)

[15] Shick, Michael: Lock Pattern Generator. URL: http://git.shick.in/lockpatterngenerator/, (Stand: 04.11.2013, 11:10 Uhr)

[16] Storage Options. URL: http://developer.android.com/guide/topics/data/data-storage.html, (Stand: 04.12.2013, 15:55 Uhr)

14 Anhang

Anhang 1: Projektdaten_____ vorgelegt auf CD-ROM

- Gestenauthentifikation:
  - Eclipse-Projekt zur Anwendung zur Gestenauthentifikation
  - Ausführbare Datei: Gestenauthentifikation.apk
- Gestenspeicherung:
  - Eclipse-Projekt zur Anwendung zur Speicherung von Gesten
  - Ausführbare Datei: Gestenspeicherung.apk
- Matlab:
  - Matlab-Script und Funktionen zur Parametereinstellung
  - Testdaten

15 Eidesstaatliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Wolgast, 21. Februar 2014

_____

Jan Wienholz