TOWARDS EFFECTIVE BEHAVIOURAL BIOMETRICS FOR MOBILE DEVICES

by
Michael Bingham

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of
the requirements for the degree of
MASTER OF COMPUTER SCIENCE

at

CARLETON UNIVERSITY
Ottawa, Ontario
May 13, 2016

Table of Contents

List of Tables

List of Figures

Abstract

Mobile devices store immense amounts of personal and sensitive data, and so have become targets for attackers. The first line of defence against attacks is authentication — verifying the identity of an agent accessing the device. We examine behavioural biometrics as an effective authentication mechanism on mobile devices. Behavioural biometrics observe and model an agent's behaviours to establish their identity. We construct an authentication system based on profiling the device sensors (touch screen, accelerometer and gyroscope) during a swipe, defined as a quick, simple movement across a touch screen. In addition, we explore the relationship between problem setting and evaluation methodology in authentication systems, producing a list of requirements necessary for accurate evaluation. Finally, we perform a user study to ascertain the effectiveness of our behavioural biometric mechanism. We conclude that this system is resistant to attacks which trivially bypass standard mechanisms such as PINs, while also potentially lightening the usability load imposed by authentication.

Chapter 1

Introduction

1.1 Introduction

Smartphones store immense amounts of personal, financial and other sensitive information. Additionally, in many systems the device itself acts as a token which authenticates the user. In light of this, device authentication — determining whether the person holding the phone should be granted access to the device — is a task of critical importance.

To date, most authentication systems on consumer smartphones are direct ports or simple adaptations of authentication techniques from desktop systems. In particular, passwords and PINs dominate the authentication landscape. However, the implementation of these schemes has left many users unsatisfied with the usability of authentication [57]. In particular, the constraints of frequent access (and by extension frequent authentication), small screen sizes and touch input make an approach based on text entry a difficult process [40]. Though some alternatives exist, such as Android's swipe pattern unlock, many users still choose weak credentials or disable authentication completely to lighten the usability load of the system [28]. In fact, researchers have shown that less than half of users have their devices protected by locks [36].

Mobile devices have an additional factor which heavily effects the design of authentication systems. The devices are often used in public spaces and carried around with users, where it is possible for them to be lost or stolen. This means that the threat of an attacker attempting to gain physical access to a device is significantly increased from a desktop context. Moreover, it is more than possible that the attacker has observed the user entering their authentication credentials, and so is able to imitate some portion of authentication. Researchers have shown that this kind of shoulder surfing attack can greatly reduce the security offered by authentication mechanisms,

particularly those which rely solely on an alphanumeric credential [70, 63].

This leaves us with a situation where strong authentication is a primary concern. Unfortunately, the added usability constraints of mobile devices often lead to comprises in authentication strength. We previously mentioned that many users opt out of authentication completely. Of those who elect to lock their devices, many select weak credentials [40]. There are two main sources of friction which influence users towards simpler, lower entropy credentials. They are the number of times devices require authentication, and the difficulty of entering alphanumeric input and special characters in on-screen keyboards. These usability considerations have a direct effect on the practical security of authentication. The difficult problem the research community faces is how to subvert the usability-security tradeoff to achieve a solution which satisfies both usability and security concerns.

1.2 Biometrics

In parallel to the trend toward user adoption of mobile devices, advances in hardware and software have made consumer grade biometric authentication a possibility. In classic biometric authentication, a physical measurement of a person is used for authentication. A typical example is a fingerprint. By using a specialized hardware sensor, unique physical characteristics of a fingerprint pattern, such as the number and location of ridges, can reliably identify an individual. Biometrics have long been used in high security environments, however their introduction at the consumer level has led some commentators to raise concerns, particularly about privacy. For example, the ubiquity of CCTV cameras in the UK, capable of facial recognition, has led to many arguments in the public media [58, 5].

There are also security considerations raised by the prevalent use of biometrics for authentication. In particular, if the biometric is compromised it is difficult for users to recover. It is much easier to rotate a password than a fingerprint, much less a face or eye. There exist storage techniques by which the exposure of the data used to verify a biometric does not necessarily reveal the biometric itself (similar to storing the hash of a password) [61], however this technology may not always be in use. For example, in the security breach of the United States' Office of Personnel Management

(OPM), an organization which manages government employees. More than 5 million employee fingerprints were exposed as a result of a data breach [31].

1.3 Behavioural Authentication

In response to this problem, a recent area of research focuses on *behavioural authentication*. Rather than explicitly prompting the user for credentials, a behavioural authentication system will attempt to use the behaviour of a user as a means to recognize them. This is done by observing user behaviour over some period of time and constructing a model of that behaviour. When new behaviour is observed, it is compared to the user model to judge whether it is typical of the user. In other words, the algorithms perform *anomaly detection* on the data stream of the user's behaviour. Typical examples of systems may observe the pattern of phone calls the user makes [41], the sequence of touch points as a user interacts with the screen [10], or even the way the user walks [23] as a means of recognition. By observing and characterizing behaviour which the user normally performs during operation of the device, researcher's hope to sidestep the security-usability tradeoff by avoiding the need to explicitly prompt for authentication.

Broadly, two categories of behavioural authentication emerge from the literature and from real world implementations. The first category consists of *continuous authentication* systems. In continuous systems, user behaviour is constantly monitored and characterized. For example, every call which a user makes [41], or every website the user accesses [65] or some combination of multiple behaviours, is incorporated into the model of user behaviour. In this way, the authentication system takes a macro view of the user and their behaviour.

The second category is *task-based* authentication. Whereas continuous schemes take a macro view of user behaviour, task-based authentication takes a micro view. Task-based authentication observes and characterizes how a user performs a single task — for example, how the user slides the unlock bar [21]. This single task constitutes the basis for authentication.

1.4 Requirements for a Solution

At this point, we observe that behavioural authentication is a subset of anomaly detection. In brief, anomaly detection attempts to build a statistical model of patterns in an input stream (representing "normal" operation), and then uses that model to detect new input which is abnormal. This is useful because deviations from normal operation often warrant a manual response: for example, they may represent unanticipated errors or failure conditions. In the context of behavioural biometrics, the system attempts to learn and model user behaviour, then use that knowledge to recognize when anomalous behaviour occurs. If the behaviour is sufficiently anomalous, it is judged to be an attack and whatever appropriate response is taken.

With this background, we assert that there are three properties necessary for a behavioural solution to effectively solve the usability-security problem of authentication. These requirements are:

1. *Model stable observables.* In order to accurately characterize normal and abnormal behaviour, it is important for the behaviour to be something the user performs without excessive variation. Moreover, the algorithm should represent sensor data in such a way that the invariant components of the behaviour are obvious and the variant components can be looked over. This is because, if a highly variant behaviour is modelled, the algorithm will acquire a model which is too broad, opening the path for an attacker to match the behaviour as well as the legitimate user.

2. *Imitation resistance.* If a solution is going to be an improvement over passwords or PINs, it should exhibit some amount of imitation resistance. This means that solely observing the authentication process (or having knowledge of the credential by some other means) should not be enough to reliably authenticate.

3. *Non-intrusive authentication.* Authentication which occurs without the user performing an explicit action is to be preferred. This is because explicit authentication is an action which interrupts the task the user wants to perform, and so user's tend to disable the mechanism or lighten it's effects. If authentication is non-intrusive, it will not disturb the users workflow and so the user

will not compromise its security.

The first requirement ensures that the algorithm is able to accurately identify users and distinguish them from each other. The second requirement serves to elevate the security of the system above that of existing solutions, while the third requirement allows it to improve overall usability.

1.5 Evaluation Requirements and Philosophy

We contend that the true challenge of implicit authentication lies not in the design of an authentication algorithm, but rather in design and evaluation methodologies. In this way, the conceptualization of a threat model and a usability model deserve as much attention as the technical specifications of an authentication algorithm. A well defined threat model is necessary to be able to meaningfully determine the strength of the authentication system, as well as to meaningfully contrast the strength of different schemes under similar conditions. If the threat model does not capture the attacks which the system is meant to protect against, it is difficult to determine if progress is being made. In tandem with a threat model, all systems make assumptions about how users will interact with the system. For example, a typical password-based system might contain the assumptions that users are able to remember high entropy strings of characters and that such character strings are easy to enter into the system. We refer to these assumptions as a usability model. Even if these assumptions are not explicitly stated, every authentication mechanism intended for commodity use has implicit assumptions that the mechanism does not interfere significantly with normal use of the system. This is because if it does, users will opt to disable it, and the mechanism will not achieve its purpose of protecting the device. We are not aware of security and usability literature which explores this concept. We note it here as a consequence of our analysis of evaluation methodologies, however as our focus is on security analysis we leave a full exploration of this area to future research.

To construct a threat model, one must sketch out the different scenarios under which an attacker can interact with the system. Since we are considering an authentication mechanism, we assume that an attacker has gained physical possession of the device while it is in a locked state. In this scenario, we can vary the amount of

knowledge the attacker possesses. For simplicity, we can consider two attackers: one which has no knowledge of the credential, and one who has observed the entire credential input process. Now we have a simple threat model which we can use to make informed measurements of the locking mechanisms strength. It should be noted, however, that there are many other nuances which could be added to this threat model. For example, you could vary the amount of knowledge the attacker has (perhaps they have only observed the credential partially, or obstructed in some way). You could also constrict the amount of time the attacker has, or the number of attempts an attacker can make.

1.6 Swipes as an Authentication Task

In the previous sections we have set the problem of mobile device authentication and gave an overview of evaluation methodology. Here we discuss some considerations for algorithmic design. We previously classified behavioural biometrics into continuous and task-based approaches. An example of a single task is the *swipe*. Swiping is a common UI element of many mobile applications. It is often used to transition between screens or items, for example in the Android or iOS photo gallery applications and home screens. It is also used in both iPhone and Android unlock screens, which unlock the device by sliding a bar or moving a lock icon across the screen.

From the device's perspective, swipes have two broad components. The first component is the shape of the swipe drawn on the screen. The device typically reports this as a series of Cartesian coordinate points separated unequally in time. The second component is the physical movement of the device. While the swipe is happening, the device will move in the user's hand. This movement is reported in the device's accelerometer and gyroscope sensors, which return a series of (x, y, z) points capturing some property of physical movement.

We hypothesize that, because swipes are a ubiquitous part of touch-screen interfaces, users are habituated to the gesture and perform it subconsciously. Furthermore, because the action is subconscious, users perform it in a consistent way. This means that User A's swipes consistently have similar properties, and are distinct and recognizable from the properties of another user's swipes.

We further hypothesize that this task is difficult for an attacker to imitate. This is for two reasons. First, the physical motion of the device happens at a scale which is difficult for an attacker to perceive. Second, when an attacker tries to imitate a swipe, the action moves from being subconscious muscle memory to being conscious imitation. This change will have effects on the form of the swipe. For example, if an attacker is consciously attempting to imitate the shape of the swipe on the screen, it may result in the swipe taking a longer than average amount of time to complete. Similar processes have been observed in handwriting forgery [16, 17]. Researchers have also noted difficulties in imitating gait patterns [34, 30]. In order for an attacker to come near imitating a person's gait, it is typically necessary for the attacker to have a similar physical makeup as the legitimate user, or in some cases be wearing similar clothing.

Considering the hypotheses that users will swipe consistently and that they will be difficult to imitate, swipes seem to be a prime candidate for task-based behavioural authentication. We developed several algorithmic implementations of swipe-based authentication. These schemes take the touchscreen, accelerometer and gyroscope sensors and transform them into a feature representation. The algorithms observe a small number of the user's swipes, and then construct a model of expected values for the features. When a new swipe comes in, the algorithm then compares these values to the expected values to make an authentication decision.

1.7 Contribution

We make several key contributions. First, we give an extensive treatment of the evaluation framework necessary to make meaningful statements about the security of behavioural biometrics, going into detail about the proper design of attacker models and techniques for formulating accurate studies. Then, we give a treatment of *swipes* as a task for authentication. The form of swipes, as well as their suitability to authentication, is given in detail. Next, we construct an authentication scheme with swipes as the basic task. To our knowledge, this is the earliest work which uses swipes as a behavioural biometric in a primary authentication system. We evaluate this system in a laboratory user study and compare it to PINs, the current authentication

standard on mobile devices. Once we have this worked out, we proceed to describe the actual design of the evaluation studies. This framework, taking the form of a set of guidelines (see Chapter 3), is intended to ensure that research moves in a direction which produces ecologically valid and effective authentication for mobile users which provides usability and security without compromise.

1.8 Organization

The rest of this thesis is structured as follows. In Chapter 2, we give background information on authentication and the new concerns introduced by the mobile context. Within this context, we move on to discuss the specific evaluation challenges inherent to this domain in Chapter 3. Chapter 4 begins by outlining several features that a behavioural authentication solution on a mobile platform should exhibit. It then continues by describing the physical characteristics of swipes and how they meet these features. Next, Chapter 5 introduces our candidate authentication algorithms and describes the motivations for their design. In order to validate the performance of our algorithms, we performed a user study which is described in Chapter 6. Finally, Chapter 7 concludes the thesis while presenting limitations and future work.

1.9 Statement of Joint Work

Several people have made contributions to aspects of the work presented in this thesis. Sarah Liske collaborated on a Java implementation of the Bit Vector version of the algorithm as an Android lock screen. Bheesham Persaud worked on technical infrastructure to run the user study. Sarah Dorey ran numerous participants for both user studies and collaborated on the ethics applications. Reid Van Melle designed the user interface for the second user study application. Shiven Sharma designed several key elements of the Mean Distance version of the algorithm. In particular, he contributed the accelerometer and gyroscope features and the design of the authentication decision. His analysis also aided in the selection of touch features for the Dynamic Range and Mean Distance versions.

Chapter 2

Background

The issue of user authentication has been a pernicious problem for security researchers since personal computing first brought users in contact with computer systems. In this chapter, we first discuss the problems researchers face with user authentication to provide the motivation for behavioural biometrics. We then introduce the concept of behavioural biometrics and their application on mobile devices. Finally we present related work which has constructed behavioural authentication systems for mobile devices.

2.1 Desktop Authentication

The most common form of user authentication is the *password*. Passwords are conceptually simple and inexpensive to operate. They are also familiar to users and easy to create and rotate. These factors may explain their resilience as the mechanism of choice for almost all computing environments [37]. Nevertheless, passwords have been criticized by researchers on many fronts, including both their security and their usability. For example, passwords are ideally long strings of random characters. However, the literature has shown that humans perform poorly at remembering these sequences [77, 1]. To compensate, many users choose passwords that are short and less random [26].

The security properties of a password are typically measured using the concept of *entropy*. Entropy is meant to be a measure of the randomness, or information, that is contained in a password. A higher level of entropy implies that an attacker would take more attempts to guess the correct password than with a lower level of entropy. This concept can be quantified in several ways. For example, it could be modelled using Shannon's measure of information [48], or by simulating an attacker guessing against a population of passwords [43].

A number of high profile security breaches have underscored these weaknesses with passwords. It is not uncommon for password databases to be leaked, either in hashed or unhashed form. Low entropy passwords are much more susceptible to cracking than those with high entropy. Additionally, password re-use, where a user reuses the same credentials for multiple online services, often means that once an attacker has gained knowledge of a single account's password they can compromise additional accounts which share the same credentials.

Despite these problems, passwords remain a stable part of desktop authentication [37] and most implemented advances involve securely reducing the frequency of authentication rather than replacing passwords completely. One way this is done is through Single Sign On (SSO), which relies on a central authentication server which may give the user a token (for example, a cookie) which the client can automatically present to services to gain access [60]. Another solution is the use of password managers, software which stores or procedurally generates passwords for arbitrary services, which are put behind a single master password the user knows [56].

2.2 Biometrics

Advances in hardware have opened up the possibility of biometrics being a consumer-grade authentication option. Physical biometrics (as opposed to the behavioural biometrics which we will discuss presently) are distinctive measurements of some physical characteristic of the human body. The most basic form is that of a single static image, a common example being fingerprint recognition. Fingerprint patterns as a systematic means of identification has been in use since at least the 19th century [39]. More recently, recognition uses a specialized scanner to detect unique characteristics in the pattern of ridges on the fingerprint or other physiological phenomena [53]. Iris patterns have similar properties and can also be used for recognition [12, 15]. Rather than a static images, biometrics can also be a dynamic stream of data. A popular example of this is voice recognition [51]. Still others may use both static and dynamic measurements. For example, facial recognition may be based off of a static image of a persons face, or it may consist of a video stream of the persons face (perhaps for the purpose of liveness detection — discussed below) [38]. It is also common to combine

multiple biometric factors (for example, face and voice) into one mechanism, called multimodal biometrics or biometric fusion [64, 20, 75].

Biometrics represent a usability advance in that users do not need to remember a credential, and can also be seen as a security advance in that the credential is closely tied to a person's identity. However, biometrics also introduce their own issues. From a usability perspective, biometrics still constitute an explicit authentication action, which users will still choose to disable in large numbers if given the option. Moreover, because biometrics take unique physical measurements, users may find them to be overly invasive. From a security perspective, biometrics in practice have proven to be unable to resist replay attacks. Since biometric recognition comes down to taking a picture (in the case of fingerprint or facial recognition) or a recording (in the case of voice), it is always possible for an attacker to replay input to the sensor [13]. For example, an attacker may take a picture of a face [18, 49] or a recording of a voice [47]. In general, as long as the replay data stream is of a higher resolution than the sensor is capable of reading, biometric recognition can be defeated.

To mitigate these attacks, biometric mechanisms have implemented various "liveness detection" features. For example, fingerprint liveness detection may work by attempting to measure a pulse in the finger, or by detecting perspiration across multiple image frames [19]. Though these techniques advance the field beyond standard attacks, it is difficult to see how defenders could reliably win an arms race between the resolution of biometric detectors and the various replay vectors for attackers.

An additional dimension to security considerations is that, once a biometric has been compromised, it is unclear how the user is to recover from the breach. In the case of a password, the user can simply change the credential to restore security. A fingerprint, face or voice is not so easily changed. Rigid credentials can benefit security, but may also harm the ability of users to recover from breaches, and consequently impair the overall usability of the system. Though there exist techniques to limit the harm caused by a breach of the information used to validate a biometric [61], disclosure of biometric credentials can happen in other ways. For example, an attacker may lift a fingerprint off of a surface and create a replica, or may take a picture of person's face [9]. Additionally, as biometrics become more common, it is

conceivable that attackers would begin phishing for biometric credentials.

2.3 Mobile Authentication

At first, mobile authentication seems to be a simple case of standard authentication. However, many issues with authentication are exacerbated by the characteristics of mobile platforms. In particular smaller on-screen keyboards make text entry, particularly text entry including non standard characters, challenging for users. Unfortunately, achieving the level of entropy required for secure passwords necessitates a large number of unique characters. However, because of the small screen sizes of mobile devices, these characters are often placed on alternate keyboards where they are even less accessible. Moreover, users authenticate on mobile devices much more frequently than in a desktop environment, which means users encounter authentication friction more frequently as well. The standard response to this issue is the use of PINs (short sequences of numbers) rather than passwords. However, this greatly reduces the potential entropy of the credential. Additionally, the large size of the key pad may increase the visibility of the authentication process for an attacker.

Mobile devices also highlight new security concerns. The devices are often used in public places, where they are lost or stolen. This means an attacker who has physical access to the device is the principle threat model. Additionally, the risk of shoulder surfing — where the attacker observes the credentials as they are being entered — is elevated. This is a problem for standard password / PIN schemes because knowledge of the credential (which can be directly gained from observation) is the only thing needed to authenticate. This leaves traditional what-you-know authentication in a difficult place, a concern shared and further examined by Ben-Asher et al [6].

To date, the standard mechanisms for authentication on mobile devices remains a PIN or password unlock. Besides these mechanisms, another option available to users is the swipe pattern unlock screen, the prototypical example of which was introduced on Android devices. This mechanism substitutes alphanumeric character entry for an ordered sequence of connected points. Research has shown that this method constitutes a usability improvement on touch screen devices, particularly in the way it allows users to recover from an input error [74]. However, because it is not in

principle different from PIN entry, the same security issues (observation, low entropy credentials) remain. Additionally, new avenues of attack, such as reconstruction of the pattern from smudges left on the screen [3], are opened.

2.4 Mobile Biometrics

Mobile devices have also moved to incorporate biometric authentication. Recently, Apple has incorporated fingerprint recognition into many of its consumer products [69]. Samsung has made similar moves [32]. Research has also explored facial recognition [25], voice recognition [73], or multimodal authentication combining different biometrics [35, 72] using hardware available on commodity mobile devices.

Biometric authentication on mobile devices is attractive because it removes the need for the user to remember an authentication credential and many devices are already equipped with the necessary hardware. In many practical applications, however, biometrics are used as a way to avoid entering a PIN or password, rather than completely replacing it. For example, Apple still requires users of its fingerprint recognition system to occasionally reauthenticate with a knowledge-based credential [2]. As discussed previously, these biometric sensors are also vulnerable to spoofing and replay attacks.

2.5 Behavioural Authentication

As discussed previously, mobile devices have more difficult usability and security considerations than a typical desktop environment. Behavioural authentication is one recent approach to overcoming these unique circumstances. This approach takes the techniques of anomaly detection and applies them to an authentication context. Anomaly detection is the process of determining whether a pattern deviates from "normal behaviour". Anomaly detection has a long history in the security literature, particularly for intrusion detection at both the network [50] and host [67] levels.

In broad strokes, anomaly detection (and by extension behavioural authentication) has the following components. First, the incoming data is deconstructed into features in a process termed feature extraction. This often takes the form of a vector

of meaningful measurements from the data. Next, the algorithm constructs a model of how it expects the system to behave. This model takes some amount of previously observed behaviour as input, called learning or training data. The algorithm then contains a component which is able to compare new behaviour to the model. Typically, features are extracted from the new behaviour and their values are compared to the distribution of values from the model. If the new behaviour is close enough to the modelled behaviour, then the behaviour is considered normal. Otherwise, it is considered anomalous.

Though we will exclusively be considering a mobile context, research into behavioural authentication has roots going back much further. One example of this is keystroke timing analysis, which uses the amount of time between different keystrokes as the basic features for anomaly detection [46]. Another classic example of a behavioural biometric is gait recognition, where the movement pattern of a person's gait is used for identification [29].

Mobile devices have a wide range of sensors which can be used to gather data on the behaviour of the user. The user interacts directly with the touch screen of the device. Physical movements in the device can be recorded by accelerometer and gyroscope. Positioning of a device can be captured by a magnetometer and its location is available by GPS. Most devices are also equipped with front facing and rear facing cameras, as well as a microphone. Additionally, software events like network, cellular and application activity can be monitored. A behavioural authentication algorithm can extract features from these sensors to characterize the behaviour of a user.

Given the number of sensors and possible behaviours, the design space of behavioural authentication algorithms is exceedingly large. However, we can group solutions together based on their approach to the problem. Specifically, we define three broad classes of behavioural authentication schemes based on the scope of user behaviour observed. These are *continuous*, *secondary* and *task-based* systems. Continuous schemes observe all user behaviour at all times from some subset of the device's sensors. In this way, the whole behaviour of a user on the device is the target of the model. Secondary schemes overlay a layer of behavioural authentication on top of an existing method of authentication. These schemes model how the user performs

another authentication action and add additional security. An example of such a scheme would be performing keystroke timing analysis on the user as they enter a password. Task-based schemes observe user behaviour as they perform a specific task on the device. It can be seen as an analogous to secondary authentication, except that the observed task is not limited to authentication (such as entering a password), but can be any task the user performs on the device.

2.5.1 Continuous Authentication

Continuous authentication schemes apply a broad lens to user behaviour by continuously monitoring sensors to construct the user model. The work done by Jakobsson et al [41] explicitly frames continuous authentication in terms of anomaly detection. More specifically, the approach is highly reminiscent of network intrusion detection systems (NIDS). NIDS sit at some point on the network and monitor packets travelling through them. Typically, they decompose a packet into its constituent parts (various headers and potentially payload) which is used as the basis for anomaly detection.

Some systems, such as Yazji et al [78] operate in a way directly analogous to NIDS by monitoring the network activity (as well as a similar technique for file access) for intrusion detection, the assumption being that any meaningful activity from an attacker would exhibit an anomalous effect on these data streams. The same principle can be applied to many other sensors on the phone. For example, an attacker who picks up the phone will have a different gait, and so continuous monitoring of gait patterns could detect a change in user [66] [73], or an attacker may use cellular services differently [65]. Additionally, when a user interacts with a mobile device, much of their interaction occurs through the touch screen. In response, systems may incorporate some kind of continuous monitoring of touch screen gestures to detect an intrusion [27]. In this work, touch screen gestures are represented as 31 discrete features, including measurements such as the starting and stopping points, pressure, and speed. This is then fed into a binary classifier to create a discriminator which classifies gestures as belonging to one of several observed users. Since monitoring is continuous, it follows that the space of patterns in the behaviour is large and complex

— because every possible action from the user has an effect on the sensor. Similar work is done by Bo et al [10]. Bo et al consider three distinct touch gestures: tapping, scrolling and flinging. For each gesture, they build a seven feature representation. The first two features are the application currently in use and the type of gesture. The next three features are extracted from the touch screen: the coordinate of the gesture, the duration (in milliseconds), and the pressure. They also consider one feature extracted from the accelerometer and one feature extracted from the gyroscope. This data is fed into a support vector machine (SVM) (one or two-class depending on if data from another user is available). After observing a small number of gestures the system identifies whether it is being used by the owner or a guest.

In general terms, a continuous scheme will monitor data streams for anomalous events and raise a security event when behaviour moves outside of its expected patterns. In a classic intrusion detection architecture, a log or report would be sent to a systems administrator for analysis. This approach does not translate directly to a consumer mobile context. Instead, designers take different approaches as to when a security point should be enforced. For example, if recent behaviour is anomalous, explicit authentication (such as a password) can be required on the next device unlock [65], or it may be done at regular intervals during usage [78]. An increasingly common approach is to only enforce a security point when the user attempts to perform a "secure" operation [73] [62], such as launching an application which has access to personal information.

2.5.2 Secondary Authentication

Secondary authentication acts as an overlay on top of an existing authentication scheme. It characterizes the way that the user performs the authentication action itself. For example, the way that the user performs the Android swipe unlock pattern can be modelled and is distinct between users, even if they use the same pattern, as examined in the work of De Luca et al[21]. From each unlock action, the authors collect a time series of touch coordinates. During a training window, the application learns a reference set containing five timeseries. When the application transitions from training to testing, it uses Dynamic Time Warping (DTW)[7] to calculate a

distance between this reference set and the incoming authentication attempt. If this distance is too large, the authentication fails.

Secondary authentication attaches an additional layer of security onto an already existing scheme, so that it can be made more secure without requiring users or developers to change their behaviour. Whereas continuous authentication applies a broad lens to user behaviour, secondary authentication applies a narrow lens. This reduces the space of patterns in the behaviour, both in terms of size and complexity. Additionally, since the mechanism is predicated upon an existing explicit authentication scheme, a non-authentication result can be enforced immediately without unexpectedly interrupting the user's workflow.

2.5.3 Task-Based Authentication

Like secondary authentication, task-based authentication takes a narrow, well defined view of user behaviour. The algorithm focuses in on a single action that the user typically performs and characterizes it. Given that touch screen gestures are the most common form of interaction between user and device, most task-based systems use them as the observed behaviour. This task can be imposed (either explicitly or implicitly as part of the user interface) at the point where authentication would typically happen, such as when the device is being unlocked or when an application is accessed. One example of this is the work of Lin et al [52]. In this work, the authors focus exclusively on the orientation sensor of the device. As a user "flicks" in either an up-down or left-right direction, the system collects a sequence of orientation readings (consisting of pitch, roll and azimuth). It then extracts features representing the mean, maximum, minimum, range, and standard deviation of these values. This feature representation is fed into a k-nearest neighbour classifier to discriminate flicks between known classes of users.

Besides this narrower focus, the design of a task-based authentication algorithm is not significantly different in principle from that of continuous or secondary schemes. The system will observe tasks for some defined training period, and construct a model of how the user performs that task, as reported by sensors on the device. When a new task is observed, the system will decide whether to authenticate based on whether

that task was anomalous.

2.5.4 Summary

There are a wide range of approaches to solving the authentication problem, particularly on mobile devices. The most familiar remain the PIN and password, with pattern unlocks gaining traction. These typical solutions however have been unsatisfying to many users. In response, researchers have sought to leverage the numerous sensors of mobile devices for authentication. Biometric authentication, particularly fingerprint and facial recognition, is currently available to consumers. Notably researchers have also attempted to recognize users based on how their behaviours are recorded in the various sensors. We have divided these efforts into three categories: continuous, secondary and task-based. For a complete picture of the research, however, it is necessary to explore in depth the evaluation methodologies. We do this in the next chapter.

Chapter 3

Evaluation Principles and Framework

3.1 Evaluation of Authentication Systems

In order to decide between different authentication systems, it is imperative to establish criteria by which the schemes can be compared. Moreover, it is necessary that the criteria accurately represent performance in a real world usage scenario — that is, that the methodology for evaluating the systems maintain ecological validity. Though determining the strength of an authentication system may seem like a straightforward problem, the examination of the field in this section will demonstrate otherwise.

First, consider a typical password. Assume that the password is composed of n characters, and the password is created from a typical keyboard containing 62 characters (as considered in, for example, [26]). Then the total size of the password space — meaning, the number of unique passwords — is 62^n. Given this information, we can construct a simple security analysis using the following assumptions:

1. Each user selects a password randomly from the password space.

2. An attacker, for each attack attempt, selects a password randomly from the password space.

Assumption 1 implies that each user will choose a password with probability $1/62^n$. Similarly, given that a user has chosen password p, the probability that an attacker will guess p is also $1/62^n$. Assuming (for example) an 8 character password, then roughly one out of every 218 trillion attacks will succeed, which is rater extraordinary performance.

Unfortunately, neither Assumption 1 nor Assumption 2 are observed in real usage patterns and attack scenarios. Studies of leaked or anonymized password corpora have shown that the probability of a user selecting a password from the password space is far from a uniform distribution — users tend to cluster toward certain password

patterns (see [26, 11] for discussion). Attackers are able to leverage this discrepancy to make huge gains for themselves. In particular, dictionary based guessing attacks, where an attacker constructs a dictionary of commonly used passwords, are effective tools. As can be expected, these attacks continue to improve over time (the interested reader is directed to [55] and [22] for analyses of password guessing attacks).

Looking closer at the real world usage of passwords, an additional problem emerges. When using a password based authentication system, just knowing the password isn't adequate for authentication — a user has to use whatever input method is available to them to enter the password. A number of errors can be introduced at this stage. For example, the user could incorrectly enter an adjacent letter, or a user may transpose two or more letters. Any of these errors will result in non-authentication, which should be registered in a measurement of the performance of an authentication system.

Given these two observations, new evaluations of password security in light of their practical security have been suggested in the literature. In particular, researchers such as Kelley et al [43] have proposed password strength evaluations based on how resistant a password is to cracking attempts. Using this methodology, one can determine the security of a password against an online attacker (where the number of guesses may be limited) or against an offline attacker (for example, one who has compromised a hashed password database).

It is unclear, then, what the correct empirical measurement of security is for passwords. Looking solely at the theoretical size of the password space cannot be sufficient, because the practical size of the password space is much reduced. However, it cannot be discounted, because a security-conscious user could construct a random (or pseudorandom) password which does approach the theoretical security properties.

3.1.1 Evaluation of Behavioural Biometric Systems

Since it is important to compare behavioural biometric schemes to each other, as well as to more typical authentication schemes like passwords, it is also necessary to discuss their evaluation criteria.

The theoretical space of most behavioural biometrics is very difficult to accurately define. In some cases, however, it is possible to construct a theoretical limit. For

example, assuming that there are n independent features, and each feature can take one of x values, then the theoretical behaviour space is x^n. In practice, however, features may be continuous or unbounded (making determining x difficult), and each feature may not be completely independent, introducing complexities in calculating each features space. Finally, though an individual biometric may be able to take x^n different values, the sensitivity of the machine learning algorithm may treat many different values similarly. Still, it is not unreasonable to conclude that, for most behavioural biometrics, the theoretical limit is as large or larger than an 8 character password.

Considering this difficulty in evaluating the theoretical space of behavioural biometrics, it is more practical to use alternate evaluation techniques. One common approach in the previous work is to report results as the level of discrimination achieved by a binary classifier which is trained and evaluated on input from two users. This approach can give important insight — particularly, it provides an upper bound on the level of performance one can expect from the system. However, since this evaluation methodology assumes that there are only two possible classes of input (namely, the two users), it does not accurately encapsulate real world usage. This is because, in a typical scenario, the system only has knowledge of one user (the device owner), and the second class comprises of every other person who may come in contact with the device.

A second technique involves training a one-class classifier on a user's data and then testing it against data collected from other users. This approach effectively captures the knowledge that a system has in the real world (i.e, that its knowledge is limited to one user). This approach also accurately models a random attacker — one who has no prior knowledge of the legitimate users behaviour. However, this approach rests on the assumption that an attacker is going to behave the same way that a normal user would. Though for some systems this assumption is defensible, for others it is instructive to consider more involved attacker models.

A third technique is to train a one-class classifier on a user's data and then perform predetermined attacks (for example, the researcher could try to imitate the user's behaviour, or the methodology could involve two participants who imitate each other).

This technique most closely matches a real world attack scenario. However, since it involves the engagement and coordination of multiple people, it is the most complex and the most susceptible to various biases and methodological flaws.

As previously mentioned, the literature takes all three of these evaluation approaches. Though multiple evaluation techniques are useful for illustrating various dimensions of system performance, it makes cross comparison of authentication schemes at best difficult and at worst misleading. Useful work has been done by Khan et al [45, 44] et al contemporary to our own to reconcile this difficult problem. We discuss these results in more detail below.

3.2 Threat Modelling and Evaluation Methodology

From the above discussion, one emergent theme is the importance of threat and attacker modelling. It has long been noted that systems are not secure in and of themselves: rather, systems are secure against defined attacks [59]. Measuring security is always comprised of comparing a system against predefined threats and attacks. In the case of behavioural biometrics, much of the limitations with evaluation methodology can be ascribed to ill-defined threat modelling.

For example, measuring security as the result of a binary classifier fails to take into account the presence of an attacker who is unknown to the system. Measuring security as simple user confusion of a one-class classifier fails to consider an attacker who has some knowledge of the credential. To set the correct course for evaluation methodology, we propose the following systematization of threat models for behavioural biometrics. First, we start with the assumption that the attacker has unhindered physical access to the device. This is because physical access is the only way an attacker can attempt authentication against a locking mechanism.

With this assumption, we develop two main attacker models. They are:

- random attacker. This attacker has physical access, but no other knowledge of the authentication process or credential.

- imitation attacker. This attacker has physical access and full knowledge of the authentication process. That is, they have observed the user entering their

credential.

These attacker models are realistic. The random attacker could be someone who has found a lost device and attempts to gain entry. The second could represent an attacker who has stolen a device after watching the user authenticate. More importantly, though, these attackers represent a weak attacker and a strong attacker. Meaning, we would expect the random attacker to have a low chance of success compared to other attackers because they have less knowledge. The imitation attacker we would expect to be stronger, because they have observed the authentication process or otherwise have complete knowledge of the behaviour used to authenticate. These two models are the baseline necessary to make accurate statements about behavioural biometric security.

It is important to note, however, that they are not the only conceivable attacker models. Rather, these two models book-end a spectrum of possible attackers. At one end is the random attacker, and at the other is the imitation attacker. In between are numerous gradations of attackers who have partial knowledge of the authentication process. Perhaps an attacker who has observed the authentication process for a limited amount of time, or one who had their observation obstructed somehow. The reason we describe these two attacker models as the minimum necessary is because the represent the maximum and minimum security performance.

There are also other variables which we could vary to construct additional attacker models. For example, we could relax our initial assumption of unhindered physical access. We could consider an attacker who has a limited amount of time with the device, or intermittent access. For simplicity at the moment we will concern ourselves just with the random and imitation attackers.

Finally, it is important to note that the random attacker does not necessarily select their attack from a uniform distribution of all possible behaviours. Since we are considering that these attacks will be carried out by a human, their attacks will tend towards whatever distribution is typical for that behaviour. For example, under a typist recognition system a random attacker would not have an equal chance of selecting an attack which has several seconds between keystrokes. This is because

human typing behaviours follows a non-uniform distribution. It is possible to construct such an attacker; however we do not believe it is practical for evaluating the real world security implications of behavioural biometrics.

3.2.1 Measurements and Result Reporting

Now that we have established a threat model we can discuss how to report measurements against this threat model. As stated previously, behavioural biometrics are a special case of anomaly detection. In all anomaly detection, the goal of the detector is to distinguish between two classes: the normal class, and the anomalous class. There are two kinds of errors that can be made. First, an instance of the normal class may be mistaken for an instance of the anomalous class. This is typically referred to as a Type I error or a false positive (FP). Second, an instance of the anomalous class may be mistaken for an instance of the normal class. This is a Type II error or a false negative (FN).

Results in anomaly detection are reported as some measure of these FPs and FNs. Additionally, the detector typically has controls which tradeoff between FPs and FNs, such that there are multiple pairs of results (i.e, at configuration x the FP rate is FP_x and the FN rate is FN_x, while at configuration y the FP rate is FP_y and the FN rate is FN_y). To get a full picture of the performance of the system, multiple measurements must be taken at different configurations. It is useful to plot these measurements as a receiver operating characteristic (ROC), which form a curve illustrating the systems performance and the tradeoff between FPs and FNs.

An account must be made, however, of the limitations of ROC reporting when applied to a security domain. The form of ROC reporting tends to implicitly give equal weighting to false positive and false negatives (or Type I and Type II errors). Though in many situations this is a useful perspective to judge success, when applied to security this assumption may no longer be justified. In general, a false negative has a much more adverse security impact than a false positive. Consider: in authentication, a false positive implies that an attacker successfully gained entry into a phone, which has a much higher cost than prompting a user for additional factors of authentication. In a security administration context, a false negative may mean that

an attack is able to persist well beyond when it should have been discovered, which has a much higher cost than alerting an administrator to a false positive. Taking this into account ROC reporting, along with its derivative the Equal Error Rate (ERR), should not be the only measure of success. We propose that, in security domains, along with ROC reporting the researcher should establish a goal for the false negative rate — that is, the rate of attacks succeeding, and compare schemes based on their corresponding false positive rate (the additional cost to the user or administrator). Though it is possible to deduce this from a ROC, stating it explicitly is is clearer, more direct, and has less risk of unknowingly falling into the assumption of equally weighting FP and FN rates.

Additionally, in order to make the terminology clearer for our domain, we propose to replace the terms false positive and false negative with domain specific terms. We will refer to the user acceptance rate (UAR), and the attack reject rate (ARR). The user acceptance rate is defined as $(1 - FP)/N$, where FP is the number of false positives and N is the total number of user authentication attempts. The AAR is defined as FN/N, where FN is the number of false negatives and N is the total number of attack attempts. So, at a glance, a well performing system will have a high UAR and a low AAR. These metrics, and their corresponding terminology in the machine learning and biometric literature, are summarized here:

1. *User Acceptance Rate.* The user acceptance rate (UAR) is the number of times, out of the total number of attempts, the legitimate user was authenticated. It is equivalent to the true negative rate in standard machine learning. A higher UAR is more usable, since it means that the user does not encounter undue difficulty in accessing the service. The UAR is collected for both swipes and PINs.

2. *User Reject Rate.* The user reject rate (URR) is the number of times, out of the total, the legitimate user was not authenticated. It is calculated as 1 — UAR. It is equivalent to the false positive rate in standard machine learning, and is a Type I error in statistics.

3. *Attack Acceptance Rate.* The attack acceptance rate (AAR) is the number of

times, out of the total number of attempts, that an attacker was authenticated. This metric encapsulates the system's imitation resistance. It is equivalent to the false negative rate in standard machine learning, and a type II error in statistics. A low AAR is more secure, because it is more difficult for the attacker to gain access. The AAR is collected for both swipes and PINs.

4. *Attack Reject Rate.* The attack reject rate (ARR) is the number of times, out of the total, that an attacker was not authenticated. It is calculated as 1 — AAR. It is equivalent to a true positive in machine learning, and is a Type II error in statistics.

5. *Input Time.* The input time is the amount of time it takes for the user to input their authentication credentials. For both swipes and PINs this is measured from the first touch event to the last touch event. A lower input time is a sign of usability, because it implies that the user spends less time in the authentication process and more time actively using the service.

3.2.2 Principles of Accurate Study Design

Now that we have discussed the form evaluation measurements should take, we can step down another level and discuss how these measurements should be taken — that is, how a study should be set up so that measurements of UAR and AAR are meaningful.

As a starting point, significant literature exists on evaluating application interfaces for usability and the effectiveness of different evaluation methods (see for example [4, 33, 54]). General principles, such as having a sizable and varied population of users, and not biasing user behaviour in certain directions, will crossover. However, our focus in this paper is not on evaluating the usability of behavioural biometric systems, but rather on evaluating their security properties.

Significantly less literature exists on the evaluation of security systems in general and authentication systems in particular. The most closely related and largest body of research is likely that of graphical passwords. Graphical passwords [8], like

behavioural biometrics, are proposed as a replacement for traditional passwords, particularly on mobile devices. Rather than the input mode being a keyboard, the user inputs credentials on an image. For example, a user may select an ordered sequence of points on an image [76]. In this way, graphical passwords intend to leverage findings from cognitive sciences which imply human memory is more suited to images than text [42].

Evaluation of graphical passwords proceeds in a few directions. It will typically begin with a theoretical analysis of the credential/password space — essentially, the number of possible unique credentials. This establishes a baseline comparison with passwords, whose password space is easily calculable. However, researchers acknowledge two realities: first, the practical size of the password space (that is, the number of unique credentials users actually choose) is significantly smaller than the theoretical space. In the domain of graphical passwords, a standard example is that of "hot spots" — regions of the image which are more often selected than others [71]. Second, the presence of an attacker can substantially reduce the security of the system. For example, if an attacker is watching the user input credentials, they may be able to narrow down the users password to one of several options, rather than one of the many theoretically possible options.

The impact of these issues on security is evaluated by conducting a user study. A number of participants will use a graphical password scheme, which researchers will use to approximate the size of the practical password space. Additionally, the study will be designed to accommodate some sort of attacker. The role of the attacker is well defined and consists of trying to compromise the user's password.

These methods ensure that, as well as knowing about the theoretical security properties of the scheme, researchers also obtain results in an ecologically valid setting — a setting which corresponds in some key ways to the way the system would be used in the real world. Taking cues from this research, and from our previous discussion, we here formulate a list of requirements for the effective evaluation of a behavioural biometric system:

- Result Measurement. Results should be based on empirical measurement of the performance of the system in a user study.

- Result Reporting Format. Among whatever other results researchers wish to present, the UAR and AAR should be included. These measurements should be given for multiple configuration settings, and plotted for comparison. Additionally, researchers should establish a target AAR, and compare their schemes to other ones based on the corresponding UAR.

- User Population. The user population should be at least a comparable size to populations in the related literature (typically 20 to 30 people) and should be demographically varied.

- Usage Scenarios. The participants usage of the device should correspond to how they would use it in the field. In particular, an excessive number of tasks should not be performed in a short period of time and users should not be biased to behave in one way.

- Attack Scenarios. The study should have well defined attack scenarios and results for the attack scenarios should be included. At a minimum, two attackers should be considered: first, an attacker who naively tries to authenticate to the system, and second, an attacker who has knowledge of the user's behavioural pattern and attempts to imitate it.

Khan et al [45, 44] have done considerable work in the problem of accurate comparisons of implicit authentication schemes. They have presented a couple of important considerations. The first is what they term "data availability". This deals with whether it is possible for a scheme to collect enough data over a typical usage period to make an authentication decision. This corresponds to our requirement for usage scenarios. A second they call "detection delay". This is the amount of time an attacker has, once they have gained access, until the scheme detects an attack and takes action. The authors rightly note that a scheme with high accuracy is useless if an attack is able to persist for enough time that the attacker is able to accomplish whatever goals they had. This is a highly important consideration for continuous authentication schemes in particular and a full security analysis would not be possible without it. Indeed, this is one reason why we propose task-based authentication as the best suited approach to device authentication. In a task-based system, the

detection delay is essentially zero, since sensitive tasks should always be behind an authentication barrier.

Finally, though it is not the focus of this analysis, usability considerations in evaluation are important. The usability concerns of authentication systems [14] tend to focus on the failure rate (how many times the user fails to authenticate, which is here called the URR), which is typically tied to the memorability of the credential. Additionally there may be some quantification of the users' difficuly in interacting with the authentication mechanism itself.

At a high-level, a perfectly usable authentication system would impose no friction between the user and whatever task the user is trying to accomplish. The usability cost then can be measured as anything which adds friction. Authentication failures require the user to spend time and effort undergoing the authentication task again, but there are other areas that can be considered. As above, we look at the input time, since the longer the user spends authenticating the more time there is between them and their desired task.

3.3 Review of Previous Methodologies

Now that we have established and organized some important considerations for evaluating authentication systems, we can use this basis to compare previous approaches to evaluation methodology. We begin with a quick summarization of previous work, focusing especially on evaluation methodologies. Then we move on to compare the body of research to our above requirements.

3.3.1 Overview of Previous Work

One of the earliest entries in the field of behavioural biometrics is the work of Jakobsson et al [41], followed by the work of E. Shi et al [65]. These systems monitor some event stream (for example, phone calls made from or to the device) to detect anomalies. When enough anomalous events are observed, the user is locked out of the device. E. Shi performs a more thorough evaluation. User data is collected from an app in the Android Market. In total, 50 users were used in the evaluation of user models, while additional users were included to evaluate attackers. Results are reported as

the number of times a legitimate user uses the device before failed authentication, vs the number of times an attacker uses the device before failed authentication. Two attack models are considered: an uninformed attacker, who does not know anything about the security of the system, and an informed attacker, who does. The uninformed attacker is modelled off-line by splicing data from other users directly into the input stream of a legitimate user and trained user profile. The informed attacker is modelled by splicing known-good events into the attacker event stream, so that the attacker is able to pollute one authentication feature (for example, browser history).

The work of Riva et al [62] continues in a similar vein. Device sensors are monitored to establish a level of confidence that the device is in the possession of a legitimate user (for example, monitoring the microphone for voice recognition, or monitoring bluetooth for proximity to other trusted devices). Nine users, all involved in the research group, were involved in the user study. In the first component of the study, data is collected from each user to train a model of behaviour. In the second part, the system is evaluated. The system is fed data to determine if it can recognize legitimate users. The researchers also conduct a number of attack scenarios to gather attack data. Again, two different attackers are modelled: one who is unaware of the security system, and so behaves as they normally would otherwise, and one who is aware of the security mechanism, and so tries to compensate for it (for example, by not speaking, so as to hopefully not trip the voice recognition system).

A third similar work is that of W. Shi et al [66], which attempts to prompt for explicit authentication only when it is confident that they user has changed, based on observation of the accelerometer, touch screen, voice recognition, and location history. Data was collected from the accelerometer monitoring eight users walking gait, and from seven users performing touch gestures. Results are reported as the success of binary classification (in the case of the accelerometer), or are not reported directly (voice, touchscreen). No results are given for a fusion of these elements, and no participants use a full prototype.

Other research, rather than attempting to fuse the results of multiple sensors together to come to an authentication decision, take a narrower focus and consider only a specific action or sensor. Recent examples of this work include Feng et al [24],

Lin et al [52] and Bo et al [10]. Feng et al analyze touch screen gestures to perform user authentication. Features include x and y coordinates, finger speed, and finger pressure. Forty users provided samples of touch gestures. Classifiers were trained on the resulting data, and the results reported are the results of these classifiers. Lin et al propose a similar system, however they focus on the orientation sensors rather than the touch screen. Twenty participants provided over 1,800 samples each. Again, results are reported as the results of a classifier on the participant data.

Bo et al extract features for user identification from the touch screen, as well as small movements in the accelerometer and gyroscope. In their usage scenario, there is a user who acts as the owner and several users who act as guests. Similar to the previous two systems, results are reported as the classification accuracy between the owner and the guests. Accuracy is reported after observing a single swipe, as well as after observing a sequence of swipes. Frank et al [27], similar to Feng et al [24], extract a large number of features from touch screen input to perform authentication. Subjects participated in two sessions one week apart where they used a mobile device, performing tasks which involve swipes. Results are reported as the classification accuracy between different users.

3.3.2 Previous Work compared to Requirements

- Result Measurement: Most of the systems discussed previously base their results, at least in part, off of empirical measurement from a user study. However, there are several subtle ways this requirement can be avoided. One way is to measure the results from various components of the system, but not from the system as a whole (c.f W. Shi et al).

- Result Reporting: Most authors ([24], [52], [10], [27], [21]) report the FP and FN rates. Some authors, particularly those which perform authentication continually over an extended usage period, tailor their results by, for example, reporting the length of time until an attacker is prompted for explicit authentication ([65]).

- User Population: Many evaluations include 30 or more participants ([65], [24],

System	Result Measurement	Result Reporting	User Population	Usage Scenarios	Attack Scenarios
Jakobsson et al.	yes	N/A	unknown	yes	none
E. Shi et al.	yes	ERR	50	yes	partial
Riva et al.	yes	confusion matrix	9	yes	partial
W. Shi et al.	no	confusion matrix	7	yes	none
Feng et al.	partial	FP/FN of binary classifier	40	yes	none
Lin et al.	yes	FP/FN of kNN	20	no	none
Bo et al.	yes	FP/FN/ERR of one and two class SVM	10	yes	none
Frank et al.	yes	FP/FN/ERR of binary classifier	41	yes	none
De Luca et al.	yes	FP/FN of DTW distance	48	yes	none

Table 3.1: Summary of Related Evaluation Methodologies

[27], [21]), however some include less than 10 ([62], [66]). With less than ten it may be possible to obtain preliminary information on the feasibility of a design, however it is not possible to draw concrete conclusions or accurate comparisons.

- Usage Scenarios: Previous work varies in the amount of detail that is given on how data is collected. Some authors, for example E. Shi et al, collect data from a field study, where some data collection application is given to users who then use their devices as they normally would. Most authors, however, opt for a laboratory study where participants come in explicitly to collect data. In these situations care must be taken that the way data is collected does not impact the effectiveness of the results. In particular, collecting a large number of sample (for example, Feng et al [24]) without compensating for user fatigue or artificially high consistency induced by repetition has a negative impact on effectiveness.

- Attack Scenarios: Previous work varies even more in its approach to attack evaluation. One common approach is to simulate an attacker by constructing a detector for each participant, and then simulating an attacker by feeding the detector data from other participants. Though this may approximate an attacker who has no knowledge of the system or legitimate user's behaviour, it can not judge the success of a knowledgeable attacker. Of the systems which do attempt to approximate a knowledgeable attacker, some ([65]) do so by splicing other user's normal usage data into a stream of usage data from the legitimate user. Again, though this may approximate a knowledgeable attacker, it can not be considered a true substitute for an ecologically valid attack.

This analysis is summarized in Table 3.1. There is no one scheme which meets all the evaluation criteria. There are two main areas where previous work is particularly lacking. First, results are reported in a number of different formats, which makes cross comparison difficult. Moreover, almost all of the results are reported as the result of binary classifiers, which as we discussed previously would not be possible in a real world scenario since only one class of data (that of the legitimate user) is available to the device. Notable exceptions to this are the work of Bo et al [10], who present both one class and binary classification results, and De Luca et al [21], whose DTW classifier is one-class. Second, attack scenarios are largely missing from methodologies. Most evaluations simply imply the binary classification results as being representative of an attacker. This is a weak case of the random attacker, however still suffers from being based on binary classification. Notable attempts at more formal attacker modelling is made by E. Shi et al [65], who splice legitimate user data into data collected from other users to represent an attacker who can imitate some forms of user behaviour. Riva et al [62] are the only scheme which attempts to model legitimate attack scenarios. In their laboratory study, once the user has trained the device, they leave the room and the attacker enters. The attacker either uses the phone normally (similar to a random attacker) or uses the phone while attempting to evade security measures (for example, by not speaking and triggering voice recognition).

We present our own methodology in Chapter 6. Before a discussion of our study can make sense, however, we must first discuss the specific behaviour we have chosen to target in Chapter 4 and the algorithmic approach we take to building the anomaly detector in Chapter 5.

Chapter 4

Swipes as a Behavioural Biometric

In this chapter, we begin by reviewing the requirements for a behavioural authentication solution outlined in Chapter 1 and connecting these requirements to the continuous, secondary and task-based classification in Chapter 2. We then describe how we can fulfill these requirements using the concept of a *swipe*. In this way, we have ensured that the behaviour we are targeting is derived from our problem setting and is able to be judged against an evaluation methodology which will flow from that. In the rest of the chapter we provide an in-depth analysis of the form of swipes from the perspective of the device's sensors.

4.1 Requirements for a Solution

In the introduction, we introduced three requirements for behavioural authentication solutions. The first requirement is to model stable observables. The behaviours which the mechanism is targeting should not have excessive variation. Trying to model a highly variant behaviour has a significant security risk in the model becoming too broad and losing its security properties. The second requirement is imitation resistance. Given the unique constraints of mobile devices (such as frequent public use), an attacker observing and imitating a legitimate user's actions should not be sufficient for reliable authentication. The third and final requirement is non-intrusiveness. Authentication tasks that can be baked into the normal use of an application should be preferred for not imposing additional friction on users.

Requirement	Stable Observables	Imitation Resistance	Non-intrusive
Continuous		x	
Secondary	x	x	
Task-Based	x	x	x

Table 4.1: Requirement fulfillment for continuous, secondary and task-based solutions

With the requirements given in Chapter 1 and outlined above, we can compare the three approaches (continuous, secondary and task-based) discussed in the earlier chapters. This comparison is outlined in Table 4.1. We defined continuous authentication as those schemes which constantly monitor the behaviour of users. Given this broad view of behaviour, it is unlikely that the input from the sensor will qualify as a stable observable. This is because, when a larger scale is considered, the space of possible behaviours is exceedingly large and the amount of dimensions in the behaviours make them complex. This makes it difficult for an algorithm to hone in on a clear pattern in the user's behaviour. Moreover, some approaches to continuous authentication do not meet the requirement for non-intrusiveness, particularly if an explicit authentication event can be triggered any time the authentication score falls below a certain threshold. This is because, from the user's perspective, the occurrence of explicit authentication is unpredictable — it can interrupt them at any point in their workflow. This unpredictability means that the user is not able to form an accurate mental model of the behaviour of the system, which is critical for usability [68]. This effect can be lessened by only triggering explicit authentication at security barriers (such as at application launch), however because of the continuous property these events are still partially unpredictable.

Secondary authentication schemes do not satisfy the third requirement, since they are based on a preexisting explicit authentication mechanism. However, they do satisfy the first two requirements. Since the space of possible behaviour is tightly constrained, it is much more likely for the behaviour to constitute a stable observable. Additionally, since the authentication is layered on top of an explicit mechanism, knowledge of the credential is not the sole means of authentication — which is imitation resistance.

Since task-based authentication is essentially secondary authentication but without being constrained to an explicit mechanism, it inherits the first two requirements. Additionally, by dropping the basis of an explicit authentication action for some other action, it fulfills the third requirement as well.

4.2 Swipes

We decided to select *swipes* as the targeted behavioural biometric which fulfills our requirements. Swipes are a standard navigational gesture on touch-based mobile platforms. They consist of a quick movement sliding the finger along the screen. We do not place restrictions on the direction or length of the swipe. We hypothesize that, since this action is a standard navigational task which users perform frequently, they become habitualized into the user's muscle memory. We further posit that this implies the user will perform the task subconsciously with physical regularity — meaning the motion will be very similar each time the user performs it, making it a stable observable in accordance with the requirement to model stable observables. An additional corollary to this is that the exact action is difficult for an attacker to imitate, giving imitation resistance. This is because, when imitating an action, the action moves from a person's subconscious muscle memory to their explicit attention, which may manifest itself in physical differences in the swipe. As noted in Chapter 1, researchers have observed these effects in similar habitual actions, such as handwriting and walking. Finally, since swipes are a standard part of touchscreen navigation and UI design, most applications requiring authentication can easily instrument a screen to act as the authentication mechanism without disrupting the workflow of the application, which is in accordance with non-intrusivity.

4.2.1 Other Candidate Tasks

Swipes are not the only possible task which could be the subject of behavioural authentication, though we do believe at this point it is the approach which holds the most promise. Other potential behaviours include:

1. *keystroke dynamics.* Keyboards are still a typical input modality for mobile devices. However, the usability of mobile device keyboards has been heavily criticized. In fact, the design of mobile application interfaces will often try to mitigate the need for text input so as to avoid the usability strain of a keyboard. Basing a usable authentication solution on an input modality with significant usability problems will not advance the overall balance of usability in the system.

2. *application navigation.* It could also be possible to profile the way the user progresses through the application (for instance, the time between various button presses or menu selections). However, this tightly couples the authentication algorithm to a specific interface design, reducing its general applicability and usefulness.

3. *other interface widgets.* Swipes are one example of an interface widget which users commonly encounter, however there are numerous other ones which could be suitable for authentication. For example, dragging and pinching are common gestures in mobile platforms. Typically, however, these actions are much more contextualized than swiping because they interact directly with an on screen element, or may have different effects depending on where they occur in the interface.

4.2.2 Form of Swipes

To capture the swipe, three timeseries are read from the devices sensors as the user swipes. The first is from the touchscreen. The touchscreen reports the location of the users finger on the screen as a Cartesian coordinate at a given timepoint. The second sensor is the accelerometer. This reports the accelerational force (essentially, a velocity) acting on the device along the three physical axes. The third sensor is the gyroscope. This measures the rotational velocity around the three physical axes. It is important to note that, though the sampling rate is specified by the developer, the actual series of events returned are not necessarily uniform time intervals.

With these three sensors, we hope to capture the crucial information about the way the user interacts with the device when swiping. The touchscreen timeseries will capture, to the finest granularity available to the device, the shape of the users movement on the screen. The other two sensors are meant to capture the physical movement of the device in the user's hand. This includes things such as how the user turns the device as their wrist moves (captured by the gyroscope), and how the device moves back and forth with small movements in the user's arm (by the accelerometer). In the next section we give examples of these three sensors from data collected during our user study and pull out a number of observations. The purpose

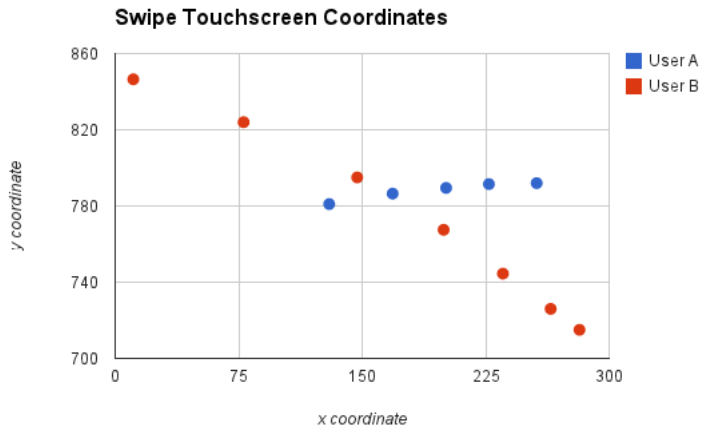

Figure 4.1: Cartesian coordinates of touch points

of this is to give an example of the kind of close-up analysis which forms the basis of feature extraction.

4.2.3 Touchscreen

Figure 4.1 gives a visual example of the touch timeseries to gain a better understanding of the space. Each point represents a single reading from the touchscreen sensor of the swipe. The figure shows data for two different users. Though the series for User A is small, we can note a couple of properties. First, the points have a slight upward trend from left to right. This trend is common when the phone is held with the right hand and the user swipes from right to left with the thumb, primarily due to the physical anatomy of the thumb which moves upward as it is rotated. We can also observe that the points toward the end of the series are spaced closer together than the first two points. This suggests that the velocity of movement may decrease over the course of the swipe (though this would have to be confirmed by analyzing the timestamps, since the readings are not necessarily uniform time intervals).

User B's swipe is quite distinct from User A's. First, the number of coordinates leads us to conclude that the swipe is longer (both in physical distance as well as in time) than that of User A. Second, the swipe has a reversed trend from that of User A — meaning it goes downward when viewed from left to right, rather than upward.

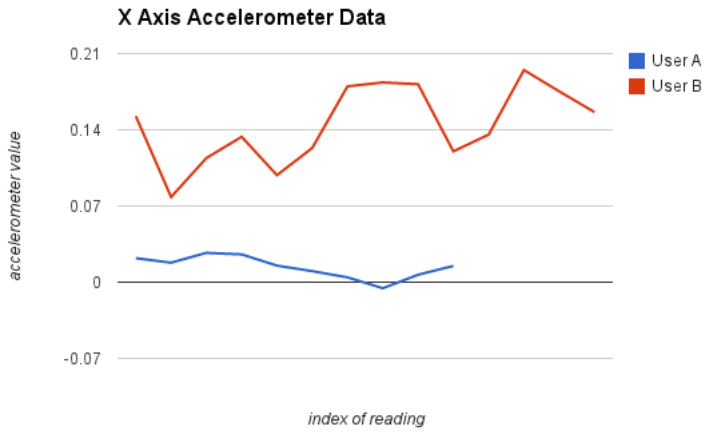

Figure 4.2: Acceleration along the x axis

This occurs because of the same property which caused the upward trend it User A — handedness. User B swiped with the left hand from left to right, causing a downward trend. Finally, similar to User A, the points become closer together towards the right end of the swipe, indicating an increase in velocity.

In summary, the two user's touch timeseries illustrate two important points. First, that user's swipes are distinct. Second, that physical differences between the users (in this case handedness, but equally other differences such as hand size and finger positioning) has noticeable effects on the timeseries.

4.2.4 Accelerometer

Figure 4.2 gives the timeseries for the x-axis accelerometer readings over the same swipes as Figure 4.1. Again, it is visible that User B's swipe is longer (in this case, only that it takes more time) than User A's because of the number of readings in each series. Besides this, there are several important observations. First, User A's values are much less variable than User B's, which have comparatively large and steep changes in acceleration velocity. This indicates that the phone was much more stable in User A's hand, whereas over the course of User B's swipe much more movement occurred. Second, the range of values for each user is distinct (visually, the lines do not cross).

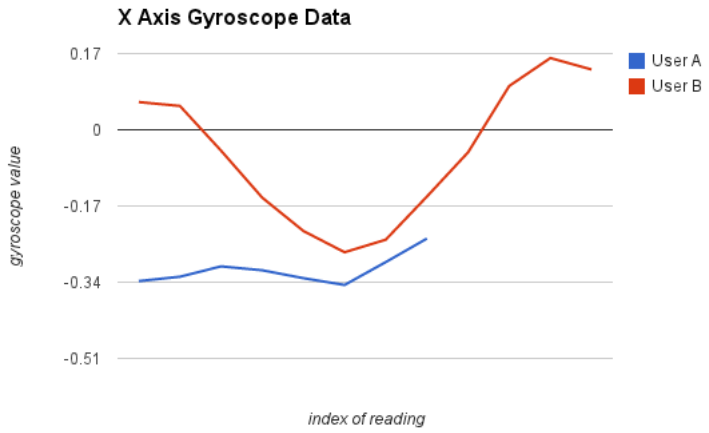

Figure 4.3: Rotational velocity (gyroscopic readings) along the x axis

More observations can be made by taking a closer look at the trends. Both series begin with a downward trend, which bottoms out before rising temporarily and then falling again. After that, however, the trends diverge with User A continuing downward and User B fluctuating. We can also not that User A's values temporarily reaching negative values. This indicates that, for a short time, User A's device began accelerating in the opposite direction (note that, because we are dealing with acceleration, this means that the device's movement was slowing down, not that the device itself was moving in the opposite direction).

Similar to the touch timeseries discussed previously, the key point is that two different user's accelerometer timeseries, and by extension the movement of the device itself, are distinct from each other. This means that physical differences in how the users handle the device have visible effects. The y and z axes are not different in principle from the x axis illustrated here.

4.2.5 Gyroscope

Figure 4.3 gives the timeseries for the x-axis gyroscope readings over the same swipes as the previous figures. The analysis for this is much the same as for the accelerometer readings so it will not be repeated in full. However, we can not that both swipes share a generally parabolic trend — however the trend is much more pronounced in User

B's swipe than it that of User A. Again, the y and z axes do not vary in analysis from the axis illustrated here.

4.2.6 Swipes for Authentication

From the observations above, swipes emerge as a prime candidate behaviour to target for authentication. This is because they have the potential to be the basis for a scheme which fits the requirements derived from the problem setting. In particular, since they are quick, repetitive, near unconscious movements, it is likely that their reflection in the device's sensors will be stable for one user, but variant across a population of users. This same property makes them difficult to imitate, especially when imitation brings the action to a conscious level and so changes its characteristics. Finally, because they are a ubiquitous interface element of touch enabled devices, it is simple to insert them at any point where authentication is desired.

Chapter 5

Algorithms for Swipe-Based Behavioural Authentication

In this chapter, we present several designs for authenticating using swipes. Each successive design is based on refining the previous design with performance feedback from testing. In total there are three versions. The first and simplest version derives features as discretized static ranges represented by bitstrings. In the second version, static ranges are traded for a dynamic range calculated from a retained history of past features. In addition, the second version includes additional features for the accelerometer and gyroscope data streams. In the third and final version, rather than comparing a feature directly to the history of features, the algorithm compares the distance of that feature from a mean to the history of feature's distance's to the mean. For each algorithm, we discuss its approach to feature extraction, its user model and its authentication decision.

5.1 Sensor Readings

We consider three data streams collected from the device's sensors: the accelerometer, the gyroscope, and the touch screen. During a swipe (which begins when the touch screen first reports a touch, and ends when the finger is lifted from the screen) the sensors report readings to a given callback point at uneven intervals. For the accelerometer and gyroscope, these readings are reported in the form {x, y, z, t} where x, y, and z are sensor measurements along the respective axes, and t is a timestamp. The touch screen, these readings are reported as {x, y, t} where x and y are Cartesian coordinates with respect to the top left corner of the screen, and t is a timestamp. When the swipe is complete, these sensor readings have been collected into three arrays, one for each sensor. These structure of these arrays is shown in Table 5.1. All the following algorithmic descriptions assume the availability of data in this form.

accelerometer	$[\{x_0, y_0, z_0, t_0\}, \{x_1, y_1, z_1, t_1\}, ..., \{x_{n_a}, y_{n_a}, z_{n_a}, t_{n_a}\}]$
gyroscope	$[\{x_0, y_0, z_0, t_0\}, \{x_1, y_1, z_1, t_1\}, ..., \{x_{n_g}, y_{n_g}, z_{n_g}, t_{n_g}\}]$
touch screen	$[\{x_0, y_0, t_0\}\{x_1, y_1, t_1\}, ..., \{x_{n_t}, y_{n_t}, t_{n_t}\}]$

Table 5.1: sensor reading structure

In the following sections, we will refer to these arrays as *accel*, *gyro*, and *touch*. An individual reading is then referenced as, for example, accel[n]. The per axis readings are accessed by, for example, accel[n].x.

5.1.1 Approach to Feature Extraction

We chose to leverage domain specific knowledge to manually define and select features for authentication. Since we begin with a small amount of data and data collection is relatively slow and expensive, this approach allows us to prototype and iterate quickly. From previous experimental experience, this approach allows the creation of tighter models that can learn quickly under the constraints of limited data.

5.1.2 Preprocessing

Prior to performing feature extraction there are a few steps of preprocessing. In the first we determine the peak point of the touch array. The peak point is the point of maximum distance (in pixels) from any point in the touch array to a linear interpolation from touch[0] to touch[n_t]. To calculate the distance, we can begin with the point-slope form of a line

$$y - y_1 = m(x - x_1) \tag{5.1}$$

where y_1 and x_1 are any point on the line, and m is the slope of the line. Since we are calculating the linear interpolation, we by definition know two points: (touch[0].x, touch[0].y) and (touch[n_t].x, touch[n_t].y). These are the beginning and the ending points of the line segment. Since we know these two points (call them (x_1, y_1) and (x_2, y_2) respectively) we can determine the slope m as $\Delta y/\Delta x$, or

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{5.2}$$

We can then derive the standard form of the line in the following way:

$$(x_2 - x_1) \cdot (y - y_1) = (x_2 - x_1) \cdot (\frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1))$$
$$x_2 \cdot y - x_1 \cdot y - x_2 \cdot y_1 + x_1 \cdot y_1 = x \cdot y_2 - x_1 \cdot y_2 - x \cdot y_1 + x_1 \cdot y_1$$
$$y \cdot (x_2 - x_1) - x_2 \cdot y_1 + x_1 \cdot y_1 = x \cdot (y_2 - y_1) - x_1 \cdot y_2 + x_1 \cdot y_1$$
$$x_1 \cdot y_2 - x_2 \cdot y_1 = x \cdot (y_2 - y_1) - y \cdot (x_2 - x_1)$$

Given that the standard form of a line is $ax + by + c = 0$, we can re-arrange the above for:

$$(y_2 - y_1) \cdot x - (x_2 - x_1) \cdot y - x_1 \cdot y_2 + x_2 \cdot y_1 = 0$$
$$(-1) \cdot (y_2 - y_1) \cdot x - (x_2 - x_1) \cdot y - x_1 \cdot y_2 + x_2 \cdot y_1 = (-1) \cdot 0$$
$$(y_1 - y_2)x + (x_2 - x_1)y + (x_1 \cdot y_2 - x_2 \cdot y_1) = 0$$

Substituting $(\text{touch}[0].\text{x}, \text{touch}[0].\text{y})$ and $(\text{touch}[n_t].\text{x}, \text{touch}[n_t].\text{y})$ for (x_1, y_1) and (x_2, y_2) results in:

$$\begin{aligned}(\text{touch}[0].\text{y} - \text{touch}[n_t].\text{y})x + (\text{touch}[n_t].\text{x} - \text{touch}[0].\text{x})y \\ +(\text{touch}[0].\text{x} \cdot \text{touch}[n_t].\text{y} - \text{touch}[n_t].\text{x} \cdot \text{touch}[0].\text{y})\end{aligned} \tag{5.3}$$

Once in the standard form, the distance from an arbitrary point (x_p, y_p) to this line is calculated with

$$\frac{|a \cdot x_p + b \cdot y_p + c|}{\sqrt{a^2 + b^2}} \tag{5.4}$$

Where a, b and c are taken from Equation (5.3). The point of maximum distance is the peak point, which we refer to as (x_p, y_p, t_p) which has a distance of d_p.

In the second step we calculate the *band sizes* based on the screens physical dimensions. The band sizes divide the screen (horizontally and vertically) into eight spatial ranges and are used to discretize the first four features. Assume that the horizontal length of the screen (in pixels) is $\text{size}_{\text{horiz}}$ and the vertical length is $\text{size}_{\text{vert}}$. Then the horizontal band size is $\text{band}_{\text{horiz}} = \text{size}_{\text{horiz}}/8$ and similarly the vertical band size is $\text{band}_{\text{vert}} = \text{size}_{\text{vert}}/8$.

5.2 Bit Vector Version

The first version of the algorithm represents a swipe as a bit vector of discretized features. There are 36 total features, comprising measurements from touch screen data (such as the starting and ending points) as well as from the accelerometer. This version did not incorporate gyroscope measurements. The overall design goal was to be conceptually simple and lightweight, which motivated the representation as a bit vector. This representation reduces the authentication decision to a hamming distance measurement, which is easy to understand and compute.

5.2.1 Feature Definitions

Features were selected by inspection of the timeseries data from a small (3-5) initial group of participants. Most features correlate closely to the geomtric shape of a swipe. In particular the starting and ending points locate the swipe on the screen, and the curvature is captured in the peak features. The accelerometer is treated as a simple indication of which way the device moved during the swipe. Each feature was constructed so that it would be discrete. This means that the value of a feature is an integer representing which range a measurement falls into. For example, consider the amount of time (in ms) a swipe takes. We construct four ranges for this feature: 0-100ms, 100ms-200ms, 200-300ms, and 300 or more ms. In this case, the feature is an integer in the range $[1-4]$. The specific ranges for each feature (for example, eight ranges for the screen size, three ranges for the peak size) were determined empirically by observing and balacing such that a user would typically stay within the same range, while mulitple users would fall into different ranges. These features are defined below:

The first group of features concern the starting and ending coordinates. Each x and y coordinate point fits into one of eight ranges defined by $\text{band}_{\text{horiz}}$ and $\text{band}_{\text{vert}}$.

$$\begin{aligned}
\text{starting x coordinate} &\leftarrow \lfloor \frac{\text{touch[0].x}}{\text{band}_{\text{horiz}}} \rfloor \\
\text{starting y coordinate} &\leftarrow \lfloor \frac{\text{touch[0].y}}{\text{band}_{\text{vert}}} \rfloor \\
\text{ending x coordinate} &\leftarrow \lfloor \frac{\text{touch[n}_t\text{].x}}{\text{band}_{\text{horiz}}} \rfloor \\
\text{ending y coordinate} &\leftarrow \lfloor \frac{\text{touch[n}_t\text{].y}}{\text{band}_{\text{vert}}} \rfloor
\end{aligned}$$

The second group concern the peak point (see Equation 5.4). From the peak point (x_p, y_p) which occupies index p_i in the touch array, the following features are extracted.

$$\begin{aligned}
\text{peak size horizontal} &\leftarrow d_p > \text{band}_{\text{horiz}} \\
\text{peak size vertical} &\leftarrow d_p > \text{band}_{\text{vert}} \\
\text{peak location} &\leftarrow \lfloor \frac{p_i}{(n_t/3)} \rfloor \\
\text{peak time} &\leftarrow t_p > \frac{\text{touch[n}_t\text{].t}}{2}
\end{aligned}$$

The next feature is extracted from the angle of the swipe. The angle is from the starting point to the ending point, relative to a horizontal line, and measured in degrees from 0^o to 360^o and discretized into eight ranges.

$$\text{direction} \leftarrow \frac{\tan^{-1}\left(\frac{\text{touch[n}_t\text{].x} - \text{touch[0].x}}{\text{touch[n}_t\text{].y} - \text{touch[0].y}}\right)}{360^o/8}$$

The below features detect whether the swipe has touched one of the four borders of the screen.

$$\begin{aligned}
\text{left bound} &\leftarrow \text{touch[n}_t\text{].x} == 0 \\
\text{top bound} &\leftarrow \text{touch[n}_t\text{].y} == 0 \\
\text{right bound} &\leftarrow \text{touch[n}_t\text{].x} == \text{size}_{\text{horiz}} \\
\text{bottom bound} &\leftarrow \text{touch[n}_t\text{].y} == \text{size}_{\text{vert}}
\end{aligned}$$

The total time of the swipe (in ms) is discretized into four categories of 100ms each. The final range is open ended and includes all swipes longer than 300ms.

$$\text{time} \leftarrow \lfloor \frac{\text{touch}[n_t].t}{100} \rfloor$$

The next three features concern the accelerometer. They represent whether the phone has made a definitive move in a positive or negative direction over the course of the swipe. A definitive move occurs when the magnitude of a vector represented by (x_i, y_i, z_i) is greater than a threshold (empirically set to be 0.4).

$$\begin{aligned}
\text{x-axis acceleration} &\leftarrow \sqrt{\text{touch[i].x}^2 + \text{touch[i].y}^2 + \text{touch[i].z}^2} > 0.4 \text{ AND touch[i].x} > 0 \\
\text{y-axis acceleration} &\leftarrow \sqrt{\text{touch[i].x}^2 + \text{touch[i].y}^2 + \text{touch[i].z}^2} > 0.4 \text{ AND touch[i].y} > 1 \\
\text{z-axis acceleration} &\leftarrow \sqrt{\text{touch[i].x}^2 + \text{touch[i].y}^2 + \text{touch[i].z}^2} > 0.4 \text{ AND touch[i].z} > 1
\end{aligned}$$

5.2.2 Feature Representation

For each of the features above, the feature extraction is either an equation which evaluates to an integer number in some range [0...n] or a boolean conditional. In order to make the comparison of swipes conceptually simple, we represent each swipe as a bitstring. Each feature is represented as a sequence bits equal to the number of discrete states that feature can occupy. The bit at the index corresponding to the state occupied by the feature is set to 1. For example, the screen's length is divided into eight equal ranges. If the starting x coordinate occupies the second range the bit representation will be 01000000, which has the second bit set to 1. The entire swipe is represented as the concatenation of these feature level bitstrings into a single bitstring. An full example of a swipe with its corresponding bitstring representations is given in Table 5.2.

In this example, assume that the screen is 50px x 100px. Coordinates are given relative to the top left corner, which is standard for touch screen measurements (note that this is different from a typical Cartesian plane, where coordinates are relative to the bottom left corner). Then the horizontal band size is $50/8 = 6.25$ pixels. The vertical band size is $100/8 = 12.5$ pixels. This means the starting y coordinate is in the 6th band, giving a bit representation of 00000100 (that is, the sixth bit is set to one). Similarly, the starting x coordinate occupies the 2nd band, giving a representation of 01000000 (the second bit is set to one). Using the same calculations

Feature name	value	bit representation
starting point y coordinate	76	00000100
starting point x coordinate	10	01000000
ending point y coordinate	20	00000100
ending point x coordinate	50	00000001
peak size	12.79	11
peak location (x, y)	(20, 40)	100
peak time	80ms	10
direction	305.54 degrees	00000010
bounding	collision screen right	0010
time	300ms	0010
x-axis acceleration	0.98	10
y-axis acceleration	-0.23	01
z-axis acceleration	0.53	10

Table 5.2: feature decomposition of a swipe

we find the ending x and y coordinate bands.

The peak size is the maximum distance from the true curve (that is, the series of points reported from the touch screen) compared to a linear interpolation from the starting to ending point. Using the equations explained above, this gives a peak size of approximately 12.79 pixels. This size is larger than both the horizontal band and the vertical band, so both bits get set to 1. The peak location coordinate is in the first third of the swipe, giving it a value of 100. Given that the total time is 300ms, the peak time of 80ms is in the first half ($300ms/2 = 150ms$), meaning we set the first bit in the two bit representation.

The angle of the line is calculated using the arctangent as shown above. This calculation gives an angle of approximately $-54.46°$, which is equivalent to $360° - 54.46° = 305.54°$. This falls into the seventh angle range, and so the resulting bit sequence is 00000010.

Note that the ending x coordinate is 50, which is the same as the width of the screen. This means the swipe has collided with the right edge of the screen. For this reason, the third bit is set in the bounding feature. The time, at 300ms, occupies the third range (200-300ms). The overall x-axis and z-axis acceleration are positive, giving a representation of 10, while the y-axis acceleration is negative, giving the opposite representation of 01.

The overall representation is achieved by concatenating all the feature level bitstrings in the last column into one bitstring.

5.2.3 User Model

The user model is also represented by a bitstring. The algorithm keeps track of a rolling window of the past ten swipes in their bitstring representation. The user model is constructed with the following algorithm:

Algorithm 1 Bit Vector Version User Model Construction

```
1:  procedure UserModel
2:      length ← number of swipes in history
3:      numBits ← total number of bits
4:      for i = 0; i < length; i++ do
5:          for j = 0; j < numBits; j++ do
6:              bitCount[j] += history[i][j]
7:      for i = 0; i < numBits; i++ do
8:          if numBits[i] > (length/2) then
9:              userModel[i] ← 1
10:         else
11:             userModel[i] ← 0
12:     return userModel
```

Essentially, the user model is a bitstring where each bit position is set to 1 if the majority of bits in that position in the history are also set to 1. In the case of a tie (for example, 5 swipes have it set to 1 and 5 swipes have it set to 0) the bit is set to 0. This arbitrary decision could be avoided by enforcing an odd number of swipes retained in the history, however in practice we observed few cases where features were evenly split. Finally, once a new swipe is observed, it is added to the history and the oldest swipe in the history is removed (that is, the history acts as a circular buffer).

5.2.4 Authentication Decision

Since both the new swipe and the user model are represented as equal length bit-strings, calculating a distance between them is simple. We chose to use the hamming distance, the number of positions which differ between the two strings. If the hamming distance exceeds a threshold value, the swipe is considered anomalous and the user is not authenticated.

Since we are using the hamming distance and only one bit is set in each feature at a time, the size of each feature in bits is not important. For example, the starting x coordinate feature is eight bits, while the x-axis acceleration feature is one bit. However, this does not mean that the starting x coordinate has more weight. Rather, a one bit change in either feature will have equal effect.

5.2.5 Discussion

The goal of this algorithms design was to keep authentication as lightweight and simple as possible. The majority of the complexity is isolated in the feature extraction — both the user model construction and authentication decision are simple bitwise operations. The price of this simplicity is in the discretization of features. In order for this algorithm to function well, the discretization should be fine-grained enough that different users will occupy different ranges for enough features to make them distinguishable, but coarse enough that the same user will consistently occupy the same range.

The main task then is to determine the correct ranges for each feature. For a small user population this is possible, either by empirical observation or by a search of the possible ranges looking for the optimal values. When the user population grows, however, it's possible that the feature ranges do not sufficiently capture the variation between users. Our next model addresses these shortcomings as explained below.

5.3 Dynamic Range Version

This algorithm provides a response to the fragility of static ranges for features when the user population grows. For a large user population, determining static ranges

which capture the variation between users is difficult. Instead of each feature having static discretization that is applied to every user, the features have a dynamic range per user which is based on the user's history of swipes. From the history, the algorithm calculates an expected range of values (interpreted as a standard deviation window around the mean). If enough of the features fall in this expected range, the user authenticates. We hypothesized this dynamic range would allow the model to adapt more tightly to each user, since it is based off of the user's unique history.

There are also significant changes to the feature set. A number of features which we judged to not be effective were removed, and a number of new features were added. In particular, the accelerometer is significantly modified and the gyroscope is added. We hypothesized that extracting more information about the physical movements of the device would capture characteristic patterns in the user's physical movement, which we expect to be characteristic of them. The new accelerometer and gyroscope features attempt to capture more fine grained information about physical movement by treating each reading as a vector (with an angle and magnitude), rather than just as a indication of direction as the previous algorithm does. Additionally we consider features locating these vectors in time (such as when the smallest and largest vectors were), capturing any characteristic information about how the user's movement changes through the swipe.

5.3.1 Feature Definitions

This version takes a different approach to calculating features than the Bit Vector Version. Rather than discretizing a value into multiple buckets, the value is left as is. Version two also leverages the gyroscope for feature extraction. All the features are listed below.

Like the Bit Vector Version, the starting and ending x and y coordinates are four features.

starting x coordinate $\leftarrow$ touch[0].x

starting y coordinate $\leftarrow$ touch[0].y

ending x coordinate $\leftarrow$ touch[n_t].x

ending y coordinate $\leftarrow$ touch[n_t].y

Assume we have the following function to compute distances between two indices a and b in the touch array:

$$\text{distance(a, b)} = \sqrt{\text{touch}[a].\text{x} - \text{touch}[b].\text{x})^2 + (\text{touch}[a].\text{y} - \text{touch}[b].\text{y})^2} \qquad (5.5)$$

Then we can define the length of a swipe as the total distance (in pixels) traversed by the swipe, while the magnitude is the distance of the linear interpolation from the starting to ending points.

$$\begin{aligned} \text{length} &\leftarrow \sum_{i=1}^{\text{n}_t} \text{distance}(i, i-1) \\ \text{magnitude} &\leftarrow \text{distance}(\text{n}_t, 0) \end{aligned}$$

The peak point is calculated in the same way as for the Bit Vector Version.

$$\begin{aligned} \text{peak size} &\leftarrow d_p \\ \text{peak location} &\leftarrow p_i \\ \text{peak time} &\leftarrow t_p \end{aligned}$$

Direction and time are also the same as Bit Vector Version.

$$\begin{aligned} \text{direction} &\leftarrow \tan^{-1}\left(\frac{\text{touch}[\text{n}_t].\text{x} - \text{touch}[0].\text{x}}{\text{touch}[\text{n}_t].\text{y} - \text{touch}[0].\text{y}}\right) \\ \text{time} &\leftarrow \text{touch}[\text{n}_t].\text{t} \end{aligned}$$

For each sensor three features are calculated: the sum total of the sensor readings for the x, y, and z axis.

$$\begin{aligned} \text{x-axis acceleration} &\leftarrow \sum_{i=0}^{\text{n}_a} \text{accel}[i].\text{x} \\ \text{y-axis acceleration} &\leftarrow \sum_{i=0}^{\text{n}_a} \text{accel}[i].\text{y} \\ \text{z-axis acceleration} &\leftarrow \sum_{i=0}^{\text{n}_a} \text{accel}[i].\text{z} \\ \text{x-axis gyroscope} &\leftarrow \sum_{i=0}^{\text{n}_g} \text{gyro}[i].\text{x} \\ \text{y-axis gyroscope} &\leftarrow \sum_{i=0}^{\text{n}_g} \text{gyro}[i].\text{y} \\ \text{z-axis gyroscope} &\leftarrow \sum_{i=0}^{\text{n}_g} \text{gyro}[i].\text{z} \end{aligned}$$

The next three features are simply the number of intervals for each sensor, where an interval is the gap between one recording and the next. Since the amount of time

between sensor readings is not strictly uniform, these features are not a function of the total time of the swipe.

$$\begin{aligned}
\text{touch intervals} &\leftarrow \text{number of touch intervals} \\
\text{accel intervals} &\leftarrow \text{number of accel intervals} \\
\text{gyro intervals} &\leftarrow \text{number of gyroscope intervals}
\end{aligned}$$

For a zero-indexed array of length n, the number of intervals is n. So, in the above features, the number of touch, accel and gyro intervals are n_t, n_a, and n_g respectively. For the next set of features, we will define a function to compute the magnitude of a vector using the Pythagorean Theorem. The function takes an array and and index into the array:

$$\text{vector\_size}(array, i) = \sqrt{\text{array}[i].\text{x}^2 + \text{array}[i].\text{y}^2 + \text{array}[i].\text{z}^2} \tag{5.6}$$

The next features treat each element of the accel and gyro arrays as a three dimensional vector. Each vector has a magnitude and an angle. If we calculate these values for each reading, we can further derive the mean magnitude and angle.

$$\begin{aligned}
\text{mean accel vector magnitude} &\leftarrow \frac{\sum_{i=0}^{\text{n}_a} \text{vector_size}(\text{accel}, i)}{\text{n}_a} \\
\text{mean gyro vector magnitude} &\leftarrow \frac{\sum_{i=0}^{\text{n}_a} \text{vector_size}(\text{gyro}, i)}{\text{n}_g} \\
\text{mean accel vector angle} &\leftarrow \frac{\sum_{i=0}^{\text{n}_a} \cos^{-1}\text{accel[i]} \cdot [1,0,0]}{\text{n}_a} \\
\text{mean gyro vector angle} &\leftarrow \frac{\sum_{i=0}^{\text{n}_g} \cos^{-1}\text{gyro[i]} \cdot [1,0,0]}{\text{n}_g}
\end{aligned}$$

Similar to above, we can also derive the largest and smallest vector magnitudes.

$$\begin{aligned}
\text{largest accel vector} &\leftarrow \max(\{\text{vector_size}(\text{accel}, i)\, \forall\, i \in [0...\text{n}_a]\}) \\
\text{smallest accel vector} &\leftarrow \min(\{\text{vector_size}(\text{accel}, i)\, \forall\, i \in [0...\text{n}_a]\}) \\
\text{largest gyro vector} &\leftarrow \max(\{\text{vector_size}(\text{gyro}, i)\, \forall\, i \in [0...\text{n}_a]\}) \\
\text{smallest gyro vector} &\leftarrow \min(\{\text{vector_size}(\text{gyro}, i)\, \forall\, i \in [0...\text{n}_a]\})
\end{aligned}$$

Furthermore we can find the indices of the largest and smallest values in the accel and gyro arrays.

$$\begin{aligned}
\text{largest accel vector location} &\leftarrow \text{indexof(largest accel vector)} \\
\text{smallest accel vector location} &\leftarrow \text{indexof(smallest accel vector)} \\
\text{largest gyro vector location} &\leftarrow \text{indexof(largest gyro vector)} \\
\text{smallest gyro vector location} &\leftarrow \text{indexof(smallest gyro vector)}
\end{aligned}$$

For the touch array, consider the longest time between sensor readings.

$$\begin{aligned}
\text{longest interval time} &\leftarrow \max(\{\text{touch}[i].\text{t} - \text{touch}[i-1].\text{t}, \forall\, i \in [1...\text{n}_t]\}) \\
\text{longest interval location} &\leftarrow \text{indexof(longest interval time))}
\end{aligned}$$

Since the touch screen reports Cartesian coordinates along with a timestamp, we can calculate the velocity for each point. From this we extract the highest velocity (in pixels per millisecond) and the index in the touch array of this high velocity point.

$$\begin{aligned}
\text{highest velocity} &\leftarrow \max\left(\left\{\frac{\text{distance}(i, i-1)}{\text{touch}[i].\text{t} - \text{touch}[i-1].\text{t}}\, i \in [1...\text{n}_t]\right\}\right) \\
\text{highest velocity location} &\leftarrow \text{indexof(highest velocity)}
\end{aligned}$$

To illustrate these calculations an example swipe is given at the top of Table 5.3. Note that in this specific example, for simplicity, the gyroscope reports the same values as the accelerometer. The rest of the table gives the feature decomposition for the swipe.

5.3.2 User Model

The user model consists of all the features for the past fifteen swipes. A range of expected values is then calculated for each feature based on this history. Given $history[x]$ as an array comprising all the measurements for a single feature x in the history, then this range is defined as:

$$mean(history[x]) \pm k * std(history[x])$$

In other words, the range is k standard deviations around the mean, where k is a configurable number. Like with the Bit Vector Version, the history acts as a circular buffer. When a new swipe from the user is recorded it is added to the queue and the oldest swipe is removed.

sensors	accelerometer	(1, 0, 2, 0.1)	(2, 1, 2, 0.2)	(3, 4, 3, 0.3)
	gyroscope	(1, 0, 2, 0.1)	(2, 1, 2, 0.2)	(3, 4, 3, 0.3)
	touch screen	(10, 10, 0.1)	(15, 12, 0.15)	(20, 20, 0.3)
features	starting x coordinate	10		
	starting y coordinate	10		
	ending x coordinate	20		
	ending y coordinate	20		
	length	16.70		
	magnitude	14.14		
	peak size	0.16		
	peak location	1		
	peak time	0.2		
	direction	0.79 rad		
	time	0.3		
	x-axis acceleration	6		
	y-axis acceleration	5		
	z-axis acceleration	7		
	x-axis gyroscope	6		
	y-axis gyroscope	5		
	z-axis gyroscope	7		
	touch intervals	2		
	accel intervals	2		
	gyro intervals	2		
	mean accel vector magnitude	3.69		
	mean gyro vector magnitude	3.69		
	mean accel vector angle	1.03		
	mean gyro vector angle	1.03		
	largest accel vector	5.83		
	smallest accel vector	2.24		
	largest gyro vector	5.83		
	smallest gyro vector	2.24		
	largest accel vector location	2		
	smallest accel vector location	0		
	largest gyro vector location	2		
	smallest gyro vector location	0		
	longest interval time	0.15		
	longest interval location	2		
	highest velocity	107.70		
	highest velocity location	1		

Table 5.3: feature decomposition of a swipe for Dynamic Range Version

5.3.3 Authentication Decision

To reach an authentication decision, the algorithm first transforms the incoming data streams (accelerometer, gyroscope and touch) into its feature representation according to the equations above. For each feature, it checks whether the value for the swipe fits within the range specified by the user model. The algorithm keeps track of how many of these values fall outside the range. If this number is below some threshold, then the user authenticates.

The following below shows the process of authentication for the Dynamic Range Version. Note that k, *threshold* and the size of the *history* array are all configurable:

Algorithm 2 Dynamic Range Version Authentication

```
1:  procedure AUTHENTICATE
2:      history[][] : array of arrays of features
3:      swipe[] : array of features
4:      count ← 0
5:      k ← 1.3
6:      threshold ← 15
7:      for i = 0; i < swipe.length; i++ do
8:          min ← mean(history[i]) − k ∗ std(history[i])
9:          max ← mean(history[i]) + k ∗ std(history[i])
10:         if min <= swipe[i] <= max then
11:             count++
12:     return count > threshold
```

5.3.4 Discussion

As stated previously, the motivation of this design is to give more flexibility to the features by allowing the range of accepted values to vary across users. The calculation above attempts to capture the range of the user's behaviour for that feature. The configuration of the k value is key, since it directly controls how broad the range is. Additionally, the number of past swipes included in the history is an important factor.

A smaller history size means that future behaviour must match past behaviour very tightly, whereas a larger history size tends to accept more — which can apply to attack swipes as well as legitimate swipes.

The introduction of means and standard deviations to the user model introduces outliers as a problem. A single outlier can skew both the mean and standard deviation significantly, which can cause the mean to move to some region of the space that the user's values do not occupy (and so causing the user to not authenticate) or causing the standard deviation to broaden to such a level that nearly all input will authenticate, including attack swipes.

There are several approaches one can take to mitigating the effect of outliers. The algorithm can refuse to add certain swipes to the history if they cause the mean or standard deviation of some features to move significantly from their previous values, however what constitutes a significant movement may be difficult to configure. The algorithm could also calculate a distance (such as the euclidean distance) between all the swipes in the history and ignore the top n largest distances. However, that approach assumes that the number of outliers is exactly n, otherwise either an outlier is still included in the history (if the true number of outliers is greater than n) or a legitimate non-outlier is removed from the calculation (if the true number of outliers is less than n). In any outlier removal scheme, however, more parameters are added to the algorithm which increases the configuration complexity.

The last note is the addition of features. Most of the new features are extracted from the accelerometer and gyroscope. This is because, from inspection of the data, we came to believe that these sensor inputs were being under-utilized in the algorithm design and so more information could be pulled out of them.

5.4 Mean Distance Version

The next iteration of the algorithm was developed after repeated small scale pilot testing of the Dynamic Range Version. In this testing, we observed that the accelerometer and gyroscope features were not contributing to the success of authentication as much as we believed possible. In response to this we significantly reworked these features. In particular, we wanted to capture more information about how the readings from

the sensors changed over the course of the swipe. Additionally, the touch features were behaving well for some users but performing sporadically for others. We hypothesized that basing the feature level authentication window on the mean distance, rather than directly on the value, could add flexibility to the user model by allowing it to adapt more to each user.

5.4.1 Feature Extraction

The full set of features for this version are presented here. There are no new touch features, however several touch features have been removed. The remaining features are given below:

T1	starting x coordinate	$\leftarrow$	$\text{touch}[0].\text{x}$
T2	starting y coordinate	$\leftarrow$	$\text{touch}[0].\text{y}$
T3	ending x coordinate	$\leftarrow$	$\text{touch}[\text{n}_t].\text{x}$
T4	ending y coordinate	$\leftarrow$	$\text{touch}[\text{n}_t].\text{y}$
T5	peak size	$\leftarrow$	d_p
T6	peak time	$\leftarrow$	t_p
T7	direction	$\leftarrow$	$\tan^{-1}\left(\frac{\text{touch}[\text{n}_t].\text{x} - \text{touch}[0].\text{x}}{\text{touch}[\text{n}_t].\text{y} - \text{touch}[0].\text{y}}\right)$
T8	time	$\leftarrow$	$\text{touch}[\text{n}_t].\text{t}$
T9	length	$\leftarrow$	$\sum_{i=1}^{\text{n}_t} \text{distance}(i, i-1)$
T10	magnitude	$\leftarrow$	$\text{distance}(\text{n}_t, 0)$
T11	longest interval time	$\leftarrow$	$\max\left(\{\text{touch}[i].\text{t} - \text{touch}[i-1].\text{t}, \forall\, i \in [1...\text{n}_t]\}\right)$
T12	highest velocity	$\leftarrow$	$\max\left(\left\{\frac{\text{distance}(i, i-1)}{\text{touch}[i].\text{t} - \text{touch}[i-1].\text{t}} \forall\, i \in [1...\text{n}_t]\right\}\right)$

The accelerometer and gyroscope features have been significantly reworked. First, the swipe is divided into three segments, each containing an equal number of sensor readings (in the case where the length is not evenly divisible by 3, the final segment has fewer readings). Then, within these segments, for each axis we calculate the slope and axis of the sensor readings. When the sensor readings are interpreted as a series of coordinates (where the x-axis is the index in the array, and the y-axis is the sensor reading) one can calculate the slope of a simple linear regression through

these points, as well as the area underneath the points using the trapezoidal rule. The two equations (defined as ψ and μ) are given below give the formula for deriving these measurements. Note that each equation takes $array$ of the length n, which will be a subarray corresponding to the values for one axis of one segment of the accelerometer or gyroscope array. ρ represents the Pearson correlation, and σ the standard deviation.

$$\psi = \rho(array[0, ..., n], [0, ..., n]) \cdot \frac{\sigma(array[0, ..., n])}{\sigma([0, ..., n])} \tag{5.7}$$

$$\mu = \frac{array[0] + array[n] + 2 \cdot \sum_{i=0}^{n} array[i]}{2} \tag{5.8}$$

The three segments we are going to use are $[0, ..., \frac{\mathrm{n}_a}{3}]$, $[\frac{\mathrm{n}_a}{3}, ..., 2 \cdot \frac{\mathrm{n}_a}{3}]$, and $[2 \cdot \frac{\mathrm{n}_a}{3}, ..., \mathrm{n}_a]$. We represent these ranges as α, β and γ, respectively. Since we deal with each axis independently, we will notate all of the n-axis values for a segment of an array as $array[segment].n$. So, for example, an array consisting of all the x-axis measurements for the first segment of the accelerometer array is denoted accel[α].x.

Using the above equations, we can represent each feature for the accelerometer and gyroscope. First take the accelerometer. For each segment, and for each axis, we calculate ψ and μ. For example, consider the first segment (α). Here we would calculate ψ and μ for the x, y, and z-axis with ψ(accel[α].x), μ(accel[α].x), ψ(accel[α].y), μ(accel[α].y), ψ(accel[α].z), and μ(accel[α].z). We would repeat these measurements for β and γ, and then repeat them all for the gyroscope. An explicit listing is given in the next section.

5.4.2 User Model

The user model consists of an expected distance from a swipe to a exemplar vector of features derived from a history of the past fifteen swipes. To begin, for each swipe, we create seven vectors which are concatenations of related features. There are three vectors for the accelerometer, three for the gyroscope, and one for the touch features. For later reference these vectors are labelled V1 through V7. For the accelerometer the three vectors consist of the ϕ and μ measurements for each of the three axes.

They are:

V1 first segment accelerometer $\leftarrow$ [ψ(accel[α].x), μ(accel[α].x), ψ(accel[α].y), μ(accel[α].y), ψ(accel[α].z), μ(accel[α].z)]

V2 second segment accelerometer $\leftarrow$ [ψ(accel[β].x), μ(accel[β].x), ψ(accel[β].y), μ(accel[β].y), ψ(accel[β].z), μ(accel[β].z)]

V3 third segment accelerometer $\leftarrow$ [ψ(accel[γ].x), μ(accel[γ].x), ψ(accel[γ].y), μ(accel[γ].y), ψ(accel[γ].z), μ(accel[γ].z)]

Similarly for the gyroscope:

V4 first segment gyroscope $\leftarrow$ [ψ(gyro[α].x), μ(gyro[α].x), ψ(gyro[α].y), μ(gyro[α].y), ψ(gyro[α].z), μ(gyro[α].z)]

V5 second segment gyroscope $\leftarrow$ [ψ(gyro[β].x), μ(gyro[β].x), ψ(gyro[β].y), μ(gyro[β].y), ψ(gyro[β].z), μ(gyro[β].z)]

V6 third segment gyroscope $\leftarrow$ [ψ(gyro[γ].x), μ(gyro[γ].x), ψ(gyro[γ].y), μ(gyro[γ].y), ψ(gyro[γ].z), μ(gyro[γ].z)]

And finally the touch features:

V7 touch features $\leftarrow$ [T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12]

These feature groupings are how a swipe will be represented for authentication. The algorithm retains 15 swipes in its history. The history is represented as a 2 dimensional array illustrated in Table 5.4. Each index in the array is an array of the feature grouping vectors for the previous 15 swipes.

Now that we have a data representation for the history, we can construct the user

	history[0]	history[1]	history[2]	...	history[6]
1st swipe	$V1_1$	$V2_1$	$V3_1$	...	$V7_1$
2nd swipe	$V1_2$	$V2_2$	$V3_2$	...	$V7_2$
3rd swipe	$V1_3$	$V2_3$	$V3_3$	...	$V7_3$
...	...	...	...	...	...
15th swipe	$V1_{15}$	$V2_{15}$	$V3_{15}$	...	$V7_{15}$

Table 5.4: 2 dimensional history array for Mean Distance Version

model which is illustrated in the code below.

Algorithm 3 Mean Distance Version User Model Construction

```
1:  procedure MEAN DISTANCE VERSION USER MODEL
2:      history
3:      meanVectors
4:      distancesFromMean
5:      meanDistances
6:      for i = 0; i < 15; i++ do
7:          history[6][i] = z-score(history[6][i], history[6])
8:      for i = 0; i < 6; i++ do
9:          meanVectors[i] = mean(history[i])
10:     for i = 0; i < 6; i++ do
11:         for j = 0; j < 15; j++ do
12:             distancesFromMean[j][i] = distance(history[j][i], meanVectors[i])
13:     for i = 0; i < 6; i++ do
14:         meanDistances[i] = mean(distancesFromMean[i])
```

The first step in the above code is to normalize the touch features by converting them to their z-score with respect to the history. The z-score measures how many standard deviations from the mean a value is (with positive meaning it is to the right of the mean, and negative meaning it is to the left). This is because we will be considering all the touch features as a vector when calculating a euclidean distance, and so we require each value in the vector to have the same scale. Next for each index in the history (that is, for each history of feature grouping vectors) we calculate the mean vector of the history. This is so that, in the next step, we can calculate the

euclidean distance from each feature grouping vector in the history to the mean vector for that grouping. This results in a two dimensional array with the same structure as the history array, however with euclidean distances instead of the feature vectors themselves. Finally, for each feature grouping, we calculate the mean distance. Like in Versions 1 and 2, the history acts as a queue. When a new swipe from the user is recorded, it is added to the queue and the oldest swipe is removed.

5.4.3 Authentication Decision

It is these mean distances which we will use to make the actual authentication decision, which is represented in the code below. Assume that *swipe* represents the swipe being authenticated against and the array variables defined in the code above are available.

Algorithm 4 Mean Distance Version Authentication Decision

```
1:  procedure MEAN DISTANCE VERSION AUTHENTICATION
2:      swipe
3:      distThresholds
4:      authThreshold
5:      count
6:      swipe[6] = z-score(swipe[6], history[6])
7:      for i = 0; i < 6; i++ do
8:          dist = distance(swipe[i], meanVectors[i])
9:          if dist > distThresholds[i] then
10:             count++
        return count < authThreshold
```

Like with user model construction, the touch features are first normalized using the z-score. Then the distance from the swipes feature grouping vectors are compared to the mean vector for that feature grouping. If the euclidean distance is above a certain threshold, the count is increased. If, after going through all the feature groupings, this count exceeds a certain amount, the authentication fails.

5.4.4 Discussion

The additional complexity added into the user model and authentication decision give this version of the algorithm more flexibility than previous versions. This is because the authentication decision is not made based on the values of the feature itself, but rather based on the distribution of distances which the system has observed from the user in the past. This means that if the user has been highly variable in the past, then the algorithm will be more tolerant of variable input. If the user has been highly consistent in the past, then the algorithm will expect values tightly clustered around the mean.

This additional flexibility can break the system in a couple of different ways. Consider the case where a user has highly variable input. This can lead to a couple different outcomes. First, highly variable input can cause the standard deviation of the history distances to grow to such a level that it will accept practically everything, effectively removing any security provided by the system. Secondly, the user may have a complex feature value distribution, for example consider a bimodal distribution which is tightly clustered around two points. In this case, the mean will fall somewhere between the two clusters and the standard deviation (depending on how the k multiplier is set) may not stretch far enough to cover a large portion of the user's input. Though the security of the system is maintained, the ability of the user to authenticate is significantly compromised.

Now consider the case where a user has tightly clustered input. However, over time the values undergo concept shifts which change their distribution. Initially, the algorithm comes to a tight model of user behaviour, and is able to sustain tight security bounds while allowing the user access. However, the user puts the device down and does not interact with it for some time. When the user begins using the device again, the values are slightly outside the tight bounds defined by the previous history. Initially, the algorithm will reject virtually all swipes entered by the user. Though the algorithm will eventually learn and adapt to the new behaviour, it is possible that the user will put the device down and undergo another concept shift before the algorithm has fully adapted. This pattern of behaviour would lead to a perpetual cycle of non-authentication for the user and poor results.

There are a couple approaches one can take to dealing with the first problem, that of highly variable input. The simplest is for input which may cause the standard deviation to grow above some pre-defined threshold can be automatically rejected (meaning, not appended to the history). This ensures that there is a maximum range of acceptable values, putting a lower bound on the security of the system, however it hampers the ability of the algorithm to learn from new input and may exacerbate the problem of a complex (for example, bimodal) distribution. In a related way, input could be included in the history so that the algorithm can learn from it in the future, however if the ranges exceed a certain bound they can be reduced to some maximum value.

5.5 Analysis

The roadmap for algorithmic design was to begin with as simple an algorithm as possible and, from there, add complexity only when it was shown to be necessary from testing. This is borne out in the progression of algorithm design in the three version detailed above.

There are a few general trends which cut across all three algorithm versions which we can pull out for discussion. These are:

1. *outliers.* An outlier swipe, if it is included in the history, will always skew the user model. The problem then becomes how to identify outliers, and what to do with them once they are detected. Unfortunately, outlier detection is a tricky process in an anomaly detection system. This is because detecting a true outlier — that is, a swipe which is performed by the user, but from which the algorithm should not learn information about normal user behaviour, is equivalent to doing authentication. After all, the process of anomaly detection is that of characterizing outliers and labelling them as anomalous. For this reason, the best approach is a heuristic one — for example, assume that a user will have no more than two outliers in their past behaviour, and so remove the two furthest out swipes from consideration.

2. *flexibility tradeoff.* All three versions of the algorithm are based on constructed

a range of acceptable values for a feature. This naturally leads to a tradeoff between how forgiving the system is for the user (i.e, how much variability it will recognize as the user's swipe) and how secure the system is against attackers. A larger range necessarily means the forgiveness increases and the security decreases, while a smaller range necessarily means the forgiveness decreases and the security increases.

3. *quality of user data*. No amount of algorithmic invention can pull a signal out of data where no signal exists. If the user's behaviour has no underlying consistency to it, any attempt to find consistency is an exercise in futility. For this reason, there may be some subset of users for whom behavioural authentication will not be a viable option. Conversely, some data may be of such high quality that the signal is trivial to pull out. For this class of user, virtually any feature representation and authentication algorithm will have exceptional performance. The main motivation for algorithmic tinkering is the average user's where the signal is obscured by various amounts of noise. In this case more refined algorithms can focus in on the pattern. The goal should not be to increase the performance of the worst performing users, nor should the performance of the best performing users be an indication of the algorithms overall performance.

Each of the three algorithm versions have different approaches to representing a swipe and making an authentication decision. In this way, each version has built in assumptions about what is unique about users' behaviour and what is consistent about swipes performed by the same user. In order to determine which assumptions are the most correct, we will compare their performance in a user study conducted according to the principles laid out in Chapter 3. This is the subject of the next chapter.

Chapter 6

User Study Methodology and Results

At the outset we discussed the importance of establishing a framework for evaluation prior to designing an algorithm to solve a problem. We did this in Chapter 3, especially in the list of requirements we formulated. In this chapter, we present in detail the design and results of our user study, which we derived directly from these principles.

6.1 User Study

In Chapter 3, we developed several requirements necessary for effective evaluation. For reference, we list these requirements here again:

- Result Measurement. Results should be based on empirical measurement of the performance of the system in a user study.
- Result Reporting Format. Among whatever other results researchers wish to present, the UAR and AAR should be included. These measurements should be given for multiple configuration settings, and plotted for comparison. Additionally, researchers should establish a target AAR and compare their schemes to other ones based on the corresponding UAR.
- User Population. The user population should be at least a comparable size to populations in the related literature (typically 20 to 30 people) and should be demographically varied.
- Usage Scenarios. The participants usage of the device should correspond to how they would use it in the field. In particular, an excessive number of tasks should not be performed in a short period of time and users should not be biased to behave in one way.

- Attack Scenarios. The study should have well defined attack scenarios and results for the attack scenarios should be included. At a minimum, two attackers should be considered: first, an attacker who naively tries to authenticate to the system, and second, an attacker who has knowledge of the users behavioural pattern and attempts to imitate it.

6.1.1 Design

We decided to perform two laboratory studies over the course of our research for several reasons. Since this is the first set of studies we are performing, a laboratory study allows quicker access to results because they are available immediately after the user's session. Additionally, we desired tightly controlled conditions. This is because our knowledge of the system's performance is initially very limited, and so the extra variability introduced from a field study would not be well understood.

Moreover we decided that, in the beginning, we did not want to test the online performance of a specific version of our algorithm, but rather wanted to collect large samples of swipes from a diverse user population. We chose this approach because we did not want the results we gathered to be tied to one specific version of our interface or our algorithm — rather, we wanted to be able to use the user data and simulate it through multiple versions of our algorithm. We also did not want users to get frustrated with the performance of the system, which may affect the way they would swipe if the system performed poorly.

This meant that, when designing the apparatus (the application which would collect data from the participants about their swipes), we needed an interface which would elicit swipes from the user similar to the swipes we wanted to use for authentication. As stated in Chapter 4, navigational swipes to transition between screens are a common interface component on mobile applications. We decided to replicate this by having the participant swipe through a number of text screens. When the application loaded, it would show the participant text and instruct them to navigate to the next screen. At the bottom of the screen were standard navigational dots, indicating that the user could swipe to the left or to the right to continue. The participants

would continue through 10 text screens, at which point they were instructed to continue swiping as they were doing but without the text changing. Instead, two figures appeared on the screen. Originally, the two position of the two figures was meant to be a rough indication of the consistency of the user's swipes. However, during pilot testing we discovered that the figures were not accurately representing consistency, and so when the study began participants were instructed that the position of the figures had no purpose or that the purpose was only meaningful to the experimenter. In this way they functioned solely as additional screens for collecting swipe data.

The exact sequence of the first study can be found in Appendix B. In total, between 45 and 50 swipes were collected from each participant. After every five swipes participants were instructed to put the device down and perform a distractor task, either playing with a ball or writing a sentence on a piece of paper. Additionally, at various points the experimenter would attempt to imitate the participants swipe a total of ten times. The experimenter would also demonstrate a swipe to the participant and ask the participant to imitate it, again for a total of 10 times. Twenty-seven users participated in this study.

In order to ensure that any findings we concluded from the first study were genuine and not merely artifacts of our user population, as well as to collect additional data, we performed a second smaller scale user study with seven users. In this study users used a mock payment prototype we developed which incorporated a swipe authentication algorithm tuned based on feedback from the first study. Users were shown a receipt, which they swiped away to complete a payment. Participants performed a similar total number of swipes as in the first study (45 to 50). In addition to the distractor tasks mentioned above, participants also used a mock PIN screen application developed by the researchers. Participants were assigned a PIN, and at various points both the experimenter and participant performed shoulder-surfing attacks (that is, imitation attacks) according to the same form as imitation attacks for swipes. The purpose of collecting this information on PINs was to compare PINs — the standard authentication mechanism for mobile devices — to swipes along several key metrics: entry time, success rate, and imitation resistance.

6.1.2 Studies Compared to Requirements

The studies were designed to flow naturally out of the requirements stated previously in Chapter 3. First, the results are empirical: they are based solely on measurement from participants using the apparatus. This includes results for the authentication rate of the users themselves, as well as the success of attacks. In the detailed results section below, results are presented in terms of UAR, AAR, and input time — in line with the second requirement. Across both user studies 34 individuals participated (27 participants in the first study and 7 participants in the second study), fitting in with the third requirement. From this, one participant in the first study and one participant in the second study had data recording issues during the session which compromised their accuracy and so were not included in the analysis below.

Since we had two studies, we had two usage scenarios. Both were closely modelled on how the a user would interact with the application in the real world. In the first study, users swiped on an application with a screen displaying text and some images. The format of the text screen, and especially the standard navigational dots included at the bottom, was meant to elicit a navigational swipe — one of the common interaction modalities with touch screen devices. In the second study, we sought to create a typical usage flow where authentication is inserted: paying for a good. In this case the swipe is less explicitly navigational; however, by still retaining the concept of moving the application forward through standard use case the navigational component is still present. Furthermore, by employing distractor tasks and collecting a moderate number of swipes from each user, we avoided introducing an artificial effect from the user swiping repeatedly many times in a row. Repeating the same action over and over again in quick succession can lead the participant to unnatural behaviour. For example, the participant might swipe more quickly because they are focused on getting through a large number of swipes.

It is important to note that, because of the nature of a laboratory study, there are several aspects of typical usage which we were not able to capture. In particular, the many distractions introduced by being in a real environment, such as noise, walking, and visual stimulation were not present. Though this does diminish the ecological validity of the study, we believe the tradeoff is justified in an early study where tighter

control and observation are necessary.

Finally we will elaborate on our attack scenarios in light of requirement 5 above. Our attack scenarios are well defined: for an informed attacker, the attacker (at various points the researchers and the participants) sat beside the authenticating user and observed the user entering the credential (swiping, or entering a PIN) multiple times. This represents a best-case scenario for the attacker: they have unhindered observation of multiple authentication attempts. An example of when this may occur is on a long plane or train ride, where someone is sitting closely next to you and can observe multiple authentications. For our uninformed (or random) attacker, we used user confusion: testing swipes from all other users against a behavioural model constructed and trained for a particular user. These attacker models were consistent across both studies.

6.2 Results and Analysis

In this section we present the results obtained from the user study. The results are presented and analyzed along several axes to give us a complete understanding of the data. A full table of results can be found in Appendix A. Throughout the text the versions are referred to as "Bit Vector", "Dynamic Range" and "Mean Distance".

6.2.1 ROC curves and Machine Learning

All three versions of the algorithm we presented in Chapter 5 have an internal, configurable threshold which controls the size of the space of swipes acceptable to the algorithm — for example Bit Vector has a threshold size for the hamming distance, Dynamic Range has a maximum number of features which can fall outside a defined range, and Mean Distance computes a maximum size for the distance from a mean for each feature. It is insufficient to simply compare the performance for one configuration of each algorithm because it could be the case that, under a different combination of configurations, performance would be significantly different.

This threshold which adjusts the size of the authenticated input space inherently controls a tradeoff between the UAR and the AAR. Consider a swipe to be an n-dimensional vector (as it is in a feature representation). Then the task of an anomaly

detection algorithm is to draw a "box" around acceptable input. If a new swipe falls into this box, then it is authenticated (i.e, it is not anomalous). The configuration controls the size of the box. If the box is bigger, it is likely to encompass more of the user's input — however, it is also more likely to encompass attacker input which looks similar (note that in this instance similarity is well defined — it is the euclidean distance between two feature representations). Hence, the larger the box and the more accepting of user input, the more accepting of attacker input the algorithm is likely to be as well.

Receiver Operating Characteristic (ROC) curves are often used in machine learning scholarship as a way to compare the extent of this tradeoff for different algorithms under different configurations. An ROC plot measures the False Positive (in our case, the URR) rate along the x-axis and the True Positive (in this case, the ARR) rate along the y-axis. For each configuration of an algorithm, a point is placed on the graph corresponding to (URR, ARR). All the points for all the configurations of an algorithm constitute a kind of curve representing the tradeoff between URR and ARR as the threshold is changed.

In order to judge relative performance, a diagonal line $y = x$ is plotted as well. This represents the performance of a random anomaly detector, which will reject an equal proportion of user and attacker swipes. The more an algorithm's points are able to "curve" above this diagonal line towards the top left corner (the top left corner being the point where URR is 0 and ARR is 1), the better the detector is performing.

In a typical ROC curve, the detector is trained and then is immutable while it is evaluated against the testing set. Since our algorithms perform online learning, we allow the detector to continue learning while it is evaluating the testing set. This means that the curves we present below may exhibit properties not typical of ROC curves used in other contexts; however, they remain useful for comparatively evaluating our algorithm versions.

At this point, we also recall the discussion of ROC reporting given in Chapter 3. In particular, it is important to, along with ROC reporting, give a target ARR and judge schemes based on the corresponding UAR. In this case, we establish a baseline ARR of 0.8 (meaning, roughly, that 80% of attacks are rejected). Though we compare

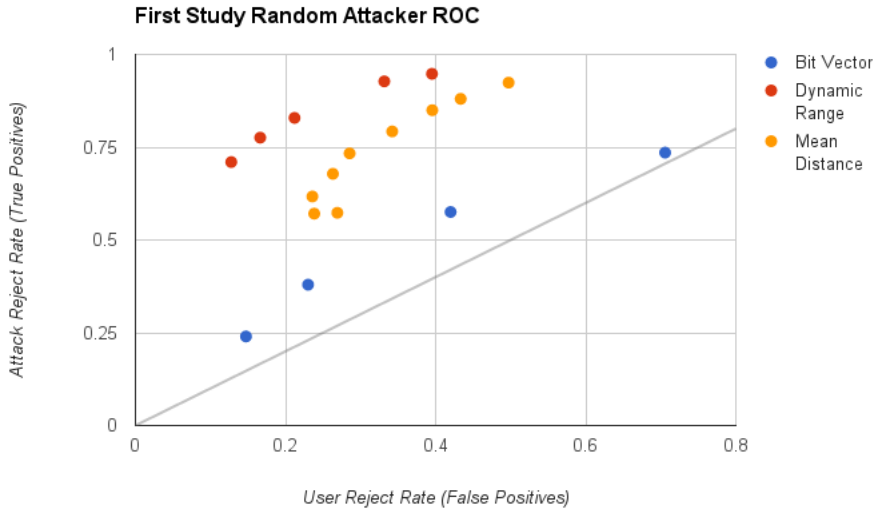

Figure 6.1: ROC curve comparing three versions of the algorithm for random data in the first study

the ROC for each algorithm to get a sense of overall performance and the tradeoffs between the rates, it is this judgment that we hold to be definitive of performance.

6.2.2 Comparisons for all versions For Study 1

The ROC curves for the first study are shown in Figures 6.1 and 6.2, representing a random attacker and an informed attacker, respectively. In both figures, Bit Vector occupies the space closest to the diagonal, implying that it is the worst performing. Dynamic Range occupies the space furthest from the diagonal, indicating the best performance. Mean Distance performs in between Bit Vector and Dynamic Range. The most curious feature of these graphs is the cluster of points for Mean Distance around 0.25 URR. This means that adjusting the parameters of Mean Distance (in this case, the size of the standard deviation window) had a non-linear effect on the performance. In other words, when the parameter was tightened, it did not always have the effect of increasing the URR and lowering the ARR.

Next we consider the 0.8 ARR baseline to make a direct comparison. These are shown in Tables 6.1 and 6.2 for a random and informed attacker, respectively. This table shows clearly that Dynamic Range has the highest UAR for the desired level of

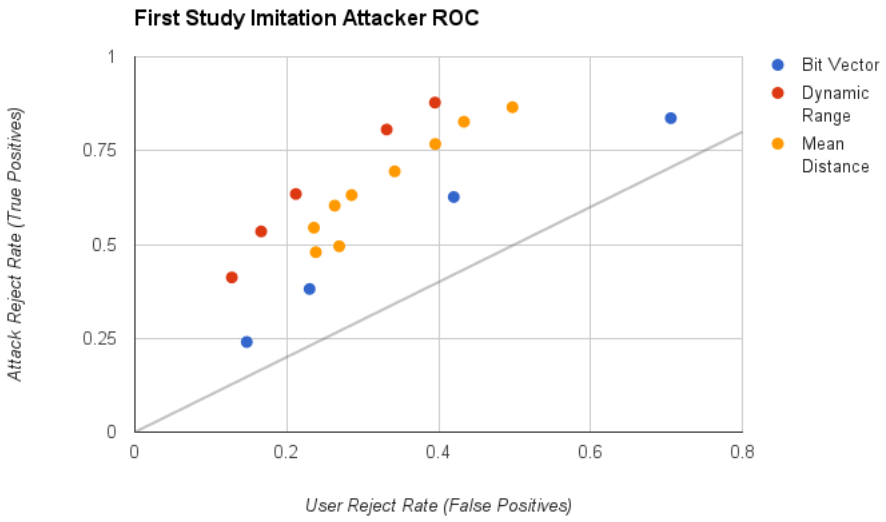

Figure 6.2: ROC curve comparing three versions of the algorithm for imitation data in the first study

Version	UAR at 0.8 ARR
Bit Vector	0.29
Dynamic Range	0.79
Mean Distance	0.66

Table 6.1: Baseline comparison for random attacker

Version	UAR at 0.8 ARR
Bit Vector	0.29
Dynamic Range	0.57
Mean Distance	0.42

Table 6.2: Baseline comparison for imitation attacker

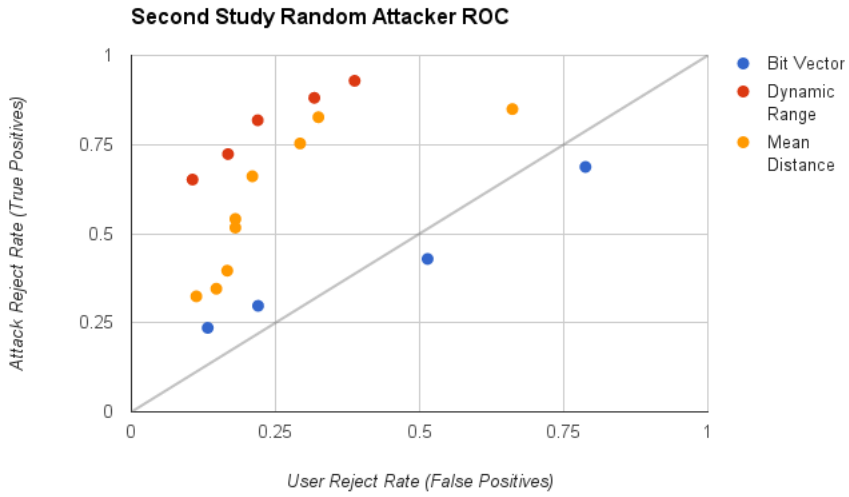

Figure 6.3: ROC curve comparing three versions of the algorithm for random data in the second study

security, followed by Mean Distance. Bit Vector is able to achieve a high ARR only by also rejecting many legitimate authentication attempts, resulting in a low UAR.

6.2.3 Comparisons for all versions For Study 2

Figures 6.3 and 6.4 show the ROC curves derived from the results of the second study (again, for random and informed attackers). A similar story emerges from these results as from the results from the first study. In Figure 6.3 it is clear that Dynamic Range performs the best, Bit Vector performs the worst, and Mean Distance performs in between. Mean Distance has a larger variance of performance, beginning close to Bit Vector and peaking close to the performance of Dynamic Range. Looking at Figure 6.4 paints a more confusion picture. Bit Vector again performs the worst, however Dynamic Range and Mean Distance occupy much the same space, particularly on the left side of the figure.

Comparing the results in Tables 6.3 and 6.4 may lead to more clarity. Focusing only on the performance which achieves our acceptable level of security shows that Dynamic Range is able to achieve a higher UAR than either Bit Vector or Mean Distance under these constraints. Interestingly, Dynamic Range performs much better

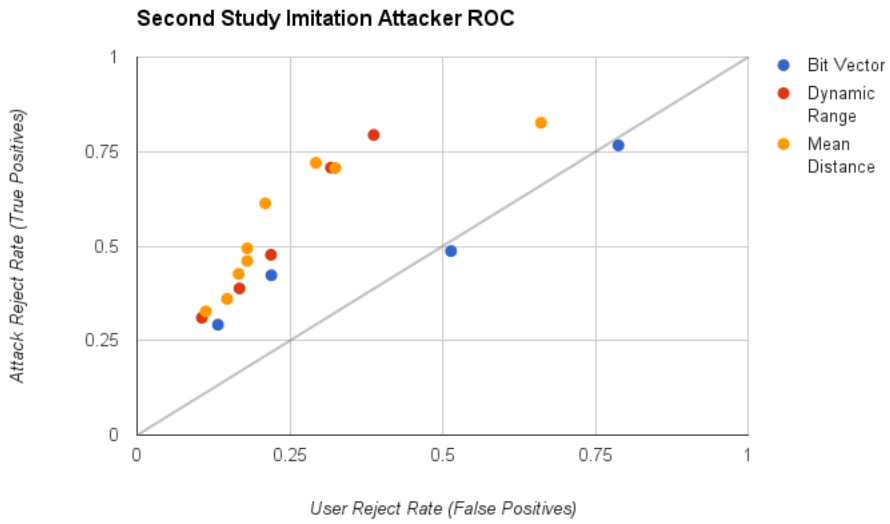

Figure 6.4: ROC curve comparing three versions of the algorithm for imitation data in the second study

Version	UAR at 0.8 ARR
Bit Vector	N/A
Dynamic Range	0.78
Mean Distance	0.0.68

Table 6.3: Baseline comparison for random attacker

Version	UAR at 0.8 ARR
Bit Vector	N/A
Dynamic Range	0.61
Mean Distance	0.34

Table 6.4: Baseline comparison for imitation attacker

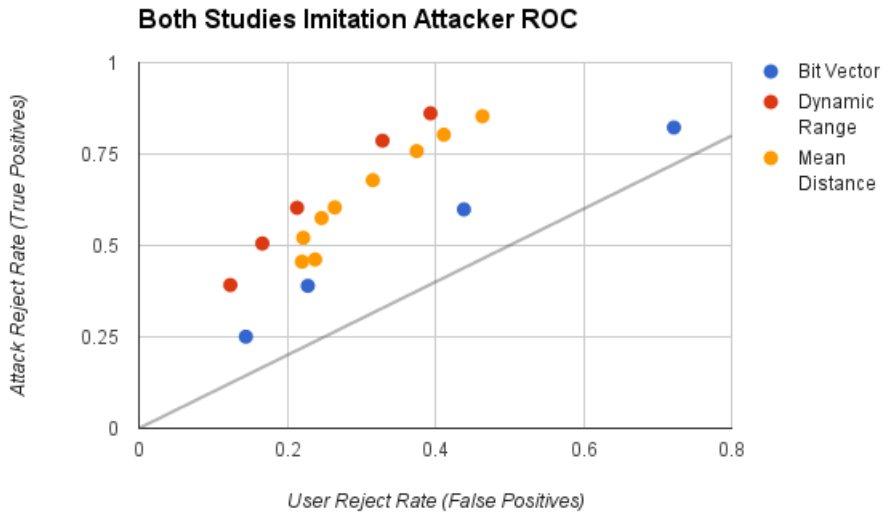

Figure 6.5: ROC curve comparing three versions of the algorithm for imitation data against an informed attacker in Table 6.4 than Mean Distance.

6.2.4 Comparisons For all versions For Both Studies

We now apply ROC curves to compare the performance of the three versions of the algorithm to each other, combining the results from the first and second study. As we see above, both the first and second study tell a similar story about which version performs the best. Though the interface presented to the participant during data collection differs between the two studies, the swipe collection and representation is the same. Figure 6.5 shows the plot under an informed attacker. As we can by now expect, the points representing Bit Vector occupy the space closest to the diagonal. This indicates performance slightly better than random. Dynamic Range occupies the space furthest from the diagonal, indicating the best performance. Between the two is Mean Distance, which also contains a cluster of points around 0.2 URR whose performance varies between being as good as Mean Distance and being little better than Bit Vector. Figure 6.6 shows the plot for a random attacker. The story is very similar to that of Figure 6.5 (and of all the previous figures). Bit Vector performs close to the diagonal. Dynamic Range performs furthest out, and Mean Distance performs in between the two.

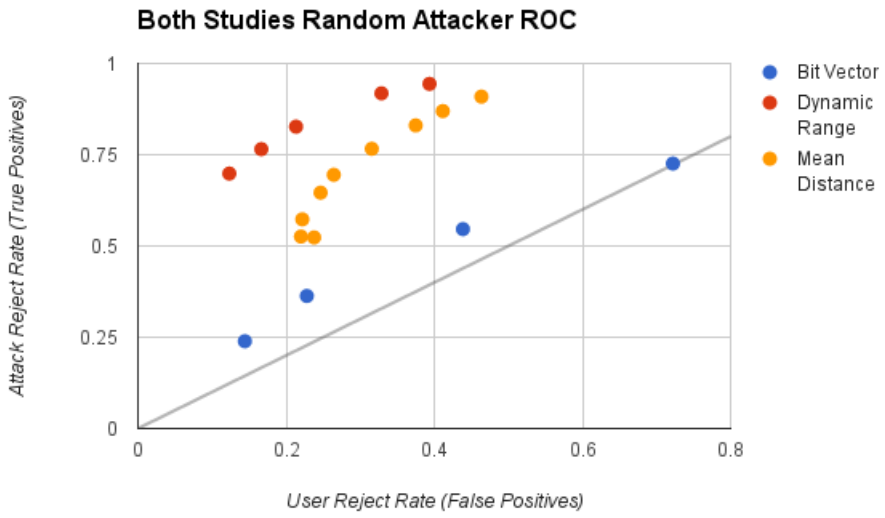

Figure 6.6: ROC curve comparing three versions of the algorithm for random data

Version	UAR at 0.8 ARR
Bit Vector	N/A
Dynamic Range	0.79
Mean Distance	0.63

Table 6.5: Baseline comparison for random attacker

Version	UAR at 0.8 ARR
Bit Vector	0.28
Dynamic Range	0.67
Mean Distance	0.59

Table 6.6: Baseline comparison for imitation attacker

We now turn again to our 0.8 ARR baseline to judge performance. The results for a random attacker are given in table 6.5 and for an imitation attacker in 6.6. First consider 6.5. In this table, we can quickly and easily see that, given the security constraint we have put in place, Dynamic Range imposes the least usability cost on users, followed by Mean Distance. Bit Vector at no point is able to achieve near rate of security we desire, so it is not considered.

Now consider table 6.6. Again, Dynamic Range imposes the smallest usability cost, followed by Mean Distance. Bit Vector is able to meet the level of security required by the baseline, however it does so at a very high cost: nearly three quarters of authentication attempts would end by prompting the user for additional credentials.

A clear picture emerges from looking at the results from the first study, second study, and the combination of the two studies. Dynamic Range unambiguously performs the best under almost every scenario analysed (the exception being imitation attacks for the second study, where Dynamic Range and 3 perform similarly). Bit Vector performs the worst, little better than a random authentication decision, and Mean Distance performs somewhere between Bit Vector and Dynamic Range. This becomes clearer by inspecting the comparison tables. In the tables for random attackers (Tables 6.1, 6.3, 6.5) Dynamic Range achieves 0.78 and 0.79 UAR when we desire an ARR of 0.8. This means that, against this attacker model, if we desire 8 out of 10 attacks to fail, roughly 8 out of 10 user authentication attempts will succeed. Against an imitation attacker, this figure drops to 0.67 (meaning, roughly 7 in 10 legitimate authentication attempts will succeed) in Table 6.6. The next section will examine in more detail the differences between these two attacker models.

6.2.5 Performance against Attacker Models

It is also instructive to compare the same algorithms performance against our two attacker models. In this case, rather than comparing two different algorithms against each other over the same data set, we are comparing the same algorithm to itself over different data sets. From this we can gain insight into the differences between the two attacker models and their ramifications for security analysis. In this section we consider both the first and second studies together, we do not perform separate

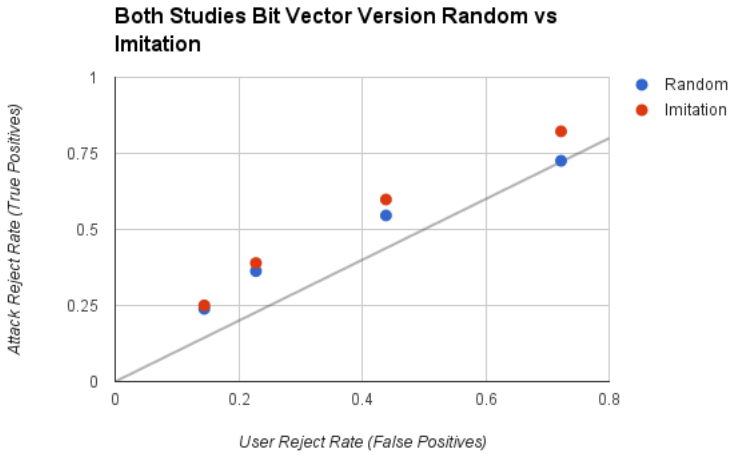

Figure 6.7: ROC curve comparing version 1 for two different attacker datasets

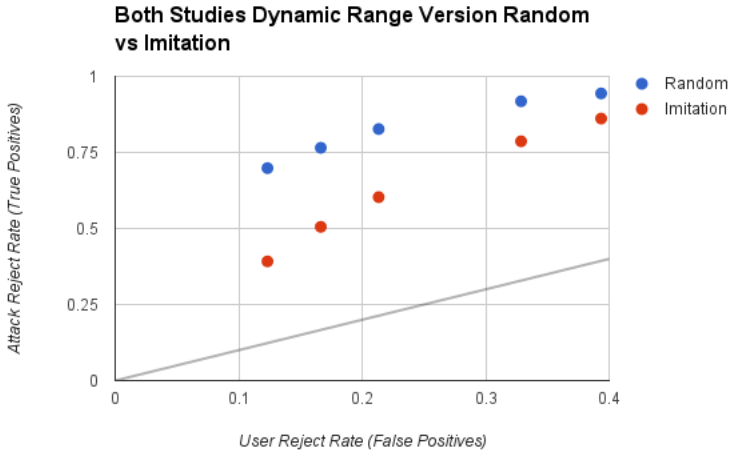

Figure 6.8: ROC curve comparing version 2 for two different attacker datasets

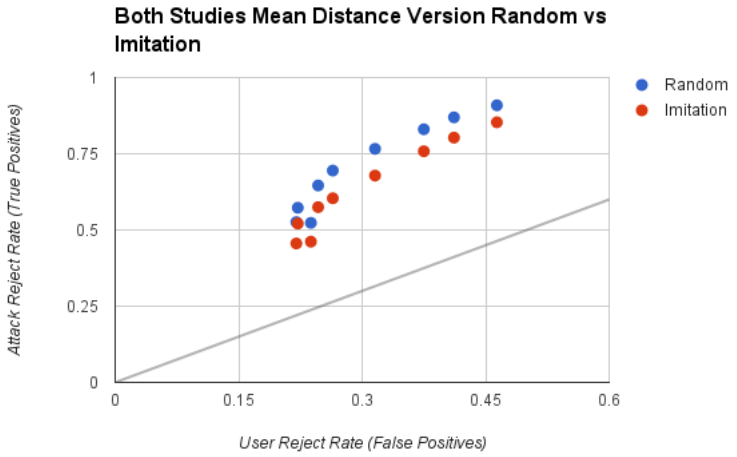

Figure 6.9: ROC curve comparing version 3 for two different attacker datasets

analysis for the two studies as we did in the previous section.

Figures 6.7, 6.8 and 6.9 contrast the performance of each version against the two attacker datasets. In all three charts, the blue represents performance against a random attacker and red represents performance against an informed attacker. Figure 6.7 provides the ROC curve for Bit Vector. For the first two points, the algorithm performs similarly for both datasets. Continuing on, however, the algorithm performs slightly better against imitation attacks (meaning, it correctly rejects more imitation attacks) than against random attacks.

Figure 6.8 provides the ROC plot for Dynamic Range. Under all configurations, Dynamic Range is able to achieve higher performance against random attacks than against imitation attacks. This is especially true from 0 to 0.3 URR. However, when we configure URR to grow past 0.3, the performance of the two versions begins to converge. Figure 6.9 provides the same plot for Mean Distance. Like Dynamic Range, the algorithm achieves higher performance against random attacks than against imitations, though the distance between the two is not as large. Interestingly, the distance between the two curves is almost the same at every point, implying that the performance difference is consistent across different URR levels.

In general, we expected to see the algorithm perform better (again, meaning rejecting a higher proportion of attacker swipes) against a random attacker than against

	UAR	AAR
Inc. Corrections	0.986	1.0
Exc. Corrections	0.942	1.0

Table 6.7: UAR and AAR for PIN-based authentication

an informed attacker. For Bit Vector, this did not match up with the results. However, given Bit Vector's poor performance on the whole, it is perhaps not surprising that it is aberrant. One possible explanation could involve the algorithmic design. Since Bit Vector relies on static feature discretizations, if an attacker is consistently off by a small amount which puts the features in a different range, it would lead to consistently poor results. For the random attacks, however, the attacks are less "consistent", and so are more likely to fall in a variety of discretization ranges. Dynamic Range and Mean Distance tell a more consistent story. In both versions, the algorithms are able to correctly reject a random attacker more consistently than an informed attacker. On balance, this supports our hypothesis that an informed attacker is a stronger threat model than a random attacker. Moreover, it illustrates that a security analysis which only takes into consideration a random attacker will overstate its performance against an important class of attackers.

6.2.6 Comparisons to PIN authentication

Next we move on to comparing the performance of swipe-based authentication to the performance of PIN authentication. PIN authentication and attacks were part of the second study. In particular, we are interested in the effectiveness of imitation attacks against the two systems. Table 6.7 gives the results for UAR and AAR for PIN entry from users participating in the second study. As expected, UAR is very high, with very few authentication attempts being submitted with the wrong credentials. However, along with a high UAR comes an exceedingly high AAR. Every imitation attempt performed by the participants was successful. This implies that the participants were even more successful at attacking than they were at authentication. It is possible that explicitly asking the participants to observe and repeat a pattern encouraged them to focus more, and thus make less mistakes, than when participants are normally authenticating.

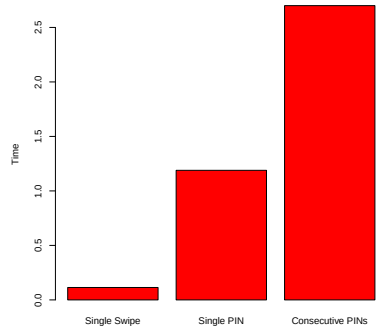

Figure 6.10: Comparison of input times to enter a swipe vs a PIN

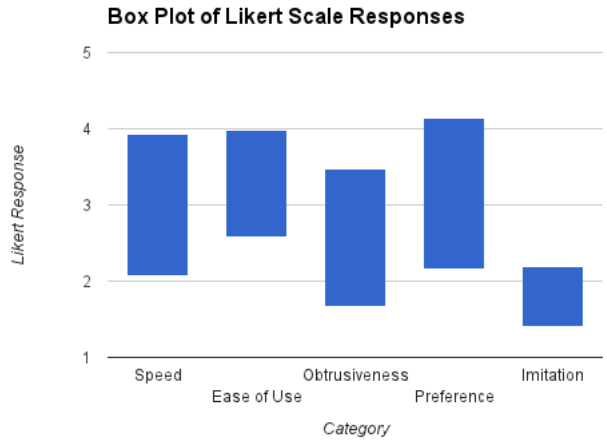

Figure 6.11: Box plot showing the average and standard deviations of Likert-scale responses

Finally we can compare the total input time as a rough heuristic for usability. Figure 6.10 shows the amount of time it takes to input a single swipe, a single PIN, and two consecutive PINs. A single swipe occurs very quickly, on average 0.144 seconds, which is 7 to 8 times faster than a PIN at 1.189 seconds. Additionally, we can see the effect of an incorrectly entered digit on the input time of a PIN. With an incorrectly entered digit, the input time of a PIN more than double to 2.7 seconds. This high cost of an error has been identified by previous authors [74] as an important usability issue.

Question	Average	Standard Deviation	Significance
Speed	3	0.926	0.5
Ease of Use	3.29	0.7	0.1608
Obtrusiveness	2.57	0.904	0.1281
Preference	3.14	1.0	0.3579
Imitation	1.8	0.4	0.00129

Table 6.8: Values for Likert Scale Responses

6.2.7 User Perception

As part of the user study, participants completed a questionnaire on their perception of swipe based authentication. After using a mock payment authentication prototype, the seven users who participated in the second phase of the user study answered a questionnaire on their experience with the system. In total they answered five five-point Likert scale questions comparing their perception of swipe-based authentication to PIN authentication. These questions compared user's perception of swipe and PIN authentication according to the categories of speed, ease of use, obtrusiveness, general preference and imitation resistance. The full questionnaire is available in Appendix B. We report on these results here.

Figure 6.11 provides a visual interpretation of the user's responses. For each category, the plot presents the range around the average to one standard deviation. Since this data is based only on the responses of seven users, it cannot be held as conclusive. However, it can provide us some useful direction for further research.

The first question compares users' perception of the speed of authentication. Responses in this category were relatively scattered, spanning a range from 4 to 2 with a high standard deviation. The average response was 3. Some users noted in their responses that they would consider swipe authentication to be faster than PIN authentication if the algorithm was more successful at recognizing them — that is, if there were consistently able to authenticate on the first try. This implies that, given algorithmic improvements, the user's perception of the speed of authentication would increase substantially.

The second question compares the users' perception of the general ease of use. The average response was 3.2, with a smaller standard deviation than the previous

Authors	URR	AAR	anomaly detection
De Luca et al [21]	19%	21%	yes
Feng et al [24]	18.92%	14.02%	no
Lin et al [52]	6.85%	6.85%	no
Bo et al [10]	27.64%	24.99%	yes
Frank et al [27]	13%	13%	no
Our Work	21%	20%	yes

Table 6.9: Results in the Literature

question. The users also considered swipes to be slightly less obtrusive than PINs (2.5 response on average). When asked whether they prefer to use swipes to PINs, the responses edged slightly towards swipes (3.2 average response). In contrast to these more mixed responses, user perception of the security of swipes as compared to PINs were quite consistent. Overall, users perceived that it was easier to observe and imitate PINs at an average response of 4.2.

6.2.8 Comparison to Literature

Due to the large number of approaches take to evaluation methodology and reporting results, comparing our work to that in the literature is not a straightforward task. Out of the authors discussed in the background section, there are five authors who evaluate a scheme based in some way on touch screen input. These authors are given in Table 6.9. For comparison our results for a uninformed attacker model are given in the last row. Out of these, two (De Luca et al [21] and Bo et al [10]) report numbers for an anomaly detection algorithm, while the others (Feng et al [24], Lin et al [52], Frank et al [27]) report numbers only for binary classifiers. All results from these five authors use only a random attacker model, with no testing of an informed attacker model.

The most direct comparison is between those authors using an anomaly detection approach and our uninformed attacker model. De Luca et al [21] report an AAR of 21% (just above our results of 20%) and a URR of 19% (just below our result of 21%), giving largely similar results to our own. The authors analyze a touch gesture laid overtop of an authentication mechanism, such as a pattern unlock or slide-to-unlock

feature using Dynamic Time Warping to compute a distance between a sample authentication and a reference set. Under the criteria described in Chapter 2, this is classified as a secondary authentication mechanism. The distinguishing feature of secondary mechanisms is that, since they are layered overtop of an existing authentication mechanism, they are necessarily explicit. Given that explicit authentication imposes a friction cost on users, an implicit approach (such as our task-based mechanism) is preferred for its lower security cost, especially when the performance is similar.

Bo et al [10] also analyse a touch gesture and report results for both anomaly detection and binary classification. For a single gesture under anomaly detection, Bo et al perform worse for both AAR and URR than our system. The authors further report results considering multiple consecutive swipes and under binary classification, which significantly improves performance, but which does not align with the requirements we have outlined.

As stated above, Feng [24], Lin [52] and Frank [27] report results for a single touch action, however only under binary classification and without considering informed attackers. These results are better for both AAR and URR than our own, however that is not surprising given knowledge of the attacker's swipes.

Overall, our performance is consistent with the best results available in the literature which are closest to the requirements laid out above for accurate evaluation. Though we do not report numbers for binary classification or multiple consecutive swipes, this also suggests that our approach is a promising foundation for building an implicit, swipe based authentication mechanism which is secure and imposes little authentication burden on users.

6.2.9 Analysis and Conclusions

From the results illustrated in the previous section, we can pull out several key observations and conclusions. We list the key findings here:

- *Comparative performance of V 1, 2 and 3.* First, we observe that, under both attacker models, the relative performance of the three versions is the same: Bit Vector performs the worst, Dynamic Range performs the best, and Mean

Distance performs slightly worse than Dynamic Range. Though any inference about the cause of the performance differences is at some level speculation, several potential reasons can be identified. The first is that Dynamic Range incorporates the most *domain specific* information in the form of its 36 features. Mean Distance, by using the regression line slope and trapezoidal area of the timeseries, moves away from highly contextual measurements — particularly for the accelerometer and gyroscope. It is possible that highly contextual measurements are the best performing. Additionally, Mean Distance introduces more flexibility in that the authentication threshold is based on the mean of the distances to an exemplar vector. This is in contrast to Dynamic Range, where the threshold is more static (rather than a mean derived from the history). Its possible that this additional flexibility this introduced was proportionately more useful to the attacker than to the legitimate user.

- *Importance of attacker models.* In Chapter 1 and in this Chapter, we asserted that it is important to measure the performance of authentication systems in the presence of attackers. Specifically, we considered an informed attacker who had observed the authentication process. Our results illustrate the importance of this evaluation methodology. If one did not consider an informed attacker and only used the results of a user confusion analysis to judge the security of an authentication scheme (or worse, used user confusion as a proxy for imitation resistance), one would be misled about the mechanism's security performance. This is illustrated by the comparative performance of Dynamic Range and Mean Distance against both our attacker models. In both cases, the algorithm performs worse in the presence of an informed attacker as compared to an uninformed attacker. Bit Vector is an aberration to this, however Bit Vector performs quite poorly against both attackers, and in any case the point still stands that there is a difference between the two.

- *Comparative performance against PINs.* The difference between attacker models is especially pertinent to the evaluation of PIN authentication. Against an uninformed attacker, PIN authentication is relatively strong — in theoretical

terms, a password space of 10^4. However, this security dissolves in the presence of an informed attacker, with our study reporting a 100% success rate for an informed attacker.

- *Usability.* Usability is a difficult concept to capture in a measurement. The usability of the system is impacted by a number of factors. First, a system is not usable if it is exceedingly difficult for the user to authenticate. That is, if the UAR is below a certain amount, users will be frustrated with frequent authentication failures. The amount of usability that is lost per point-decrease in UAR is beyond the scope of this study. However, we can make some observations on other areas of usability.

Chapter 7

Conclusion

Authentication on mobile devices is an area of research which cannot be ignored. The wealth of personal and sensitive information on mobile devices is highly valuable to an attacker, while devices' frequent use in public means they are often lost or susceptible to being stolen. Moreover, the constraints of a small screen, frequent access, and difficult keyboard input makes standard forms of authentication, such as the password, untenable. In this environment, researchers need a solution which offers security, particularly security against imitation attacks, as well as usability.

7.1 Contributions

We proposed behavioural biometrics, and in particular task based behavioural biometrics, as a solution to the mobile authentication problem. Behavioural biometrics have the potential to increase usability when they are implemented transparently — that is, when the behaviour that is being observed is incorporated into the normal task flow of a user. Behavioural biometrics also have the possibility of achieving high levels of security against imitation attacks, since an attacker must not only imitate the user's actions at a surface level, but must also imitate how the user performs those actions. Additional, some aspects of how the user performs an action may be very difficult to perceive, such as small timing variations and exact hand positioning.

Upon reviewing the literature, we recognized large discrepancies between the various evaluation methodologies employed by researchers. These discrepancies made it difficult to effectively compare the performance of authentication systems. To rectify this, we proposed in Chapter 3 a framework for effective evaluation.

To illustrate the correct approach to solving and evaluation this problem, we developed and refined a behavioural authentication algorithm over three different iterations. Each of these versions was designed to be as lightweight as possible while

providing the necessary security properties. We also ran a user study to evaluate the effectiveness of our algorithm versions, compared to each other as well as to PIN authentication. This study showed different performance levels between the three versions, with the Dynamic Range Version performing the best. It also highlighted the importance of attacker models by showing how the Dynamic Range Version and the Mean Distance Version performed worse against imitation attacks than against random attacks. Finally, it highlighted the insufficiency of PIN authentication against imitation attacks.

7.2 Limitations

There are several limitations to our work. First, our studies, at 34 participants total, does not achieve the scale necessary to make final and conclusive statements. We intend rather for the work to be indicative of the direction research should take. Second, our work consists solely of laboratory studies. These studies have known limitations, especially when trying to observe "normal" or implicit behaviours. We attempted to mitigate the laboratory effect by embedding our data collection within use cases which were not explicitly authentication and by introducing distractor tasks, however it is impossible to completely remove the effect of observation. Additionally, the small time scale of our studies (approximately one hour sessions) is unable to observe any drifts in user behaviour that may happen over extended periods of time.

Task-based behavioural biometrics themselves also have a few weaknesses. In particular, the application or device developer must be able to work the task into the system's normal workflow. This may not be possible in all situations. For example, an application which displays sensitive information immediately upon launching does not have an obvious point to insert a task. It is possible to artificially insert a screen which the user must dismiss by swiping it away (similar to "slide to unlock" functionality), however this turns the task from being completely implicit to semi-explicit, and is artificially inserted into the application rather than being a natural navigation modality, so has a negative (if small) impact on usability.

7.3 Future Work

Since our work is intended to be a roadmap and framework for upcoming research in the field, it is important to outline a few avenues research should take. There are two ways future research could complement our efforts. The first is by supplementing them with a long term field study. This would give insight into more typical usage outside of a laboratory context, as well as information on how the algorithms adjust to long term shifts in user behaviour. The second is to expand the number of participants. This would allow for stronger claims on algorithmic performance.

More experimental research could examine other possible tasks on mobile devices. Though swipes comprise a large part of user input, there are a rich variety of input modalities to explore. Actions such as dragging, scrolling, and pressing are all potential targets for behavioural biometrics. Research should establish the amount of entropy and consistency in these tasks. Additionally, the authentication properties of sequences of tasks is an interesting research question.

Finally, our efforts have been solely focused on mobile devices. However, there is no reason why other platforms would not be able to gain similar advantages from task based behavioural authentication. Reappropriating these insights back into a desktop context could help make usability advances. Additionally, users increasingly encounter a large number of other computing contexts which require authentication, particularly with the trend towards smart devices and the internet of things. As more things come online, more things have security concerns, and more things need authentication as a first line defence. Applying our insights might help make this reality more secure and more usable.

Glossary

Anomaly Detection

A system which attempts to detect abnormal behaviour by building a model of expected behaviour.

Attacker Acceptance Rate

A measure of how many times the attacker correctly authenticates. Equivalent to the false negative rate.

Attacker Reject Rate

A measure of how many times the attacker fails to authenticate. Equivalent to the true positive rate.

Authentication

The process of verifying the identity of an agent using a system.

Behavioural Biometric

A subject's behaviour used to identify them for authentication.

Biometric

A physical measurement which is used to identify a subject for authentication.

Continuous Authentication

An authentication system which observe and characterize user behaviour at all times.

Credential

The response presented to an authentication task.

Entropy

A measure of the randomness, or information, contained in a credential.

Imitation Attacker

An attacker who has witnessed the target user authenticating, for example by shoulder surfing.

Imitation Resistance

A feature of an authentication system where observation of the user inputting the credential does not lead directly to reliable attacks.

Implicit Behavioural Biometric

A task-based authentication system where the user action is a typical action for the user in the course of using the system.

Intrusion Detection

A system which attempts to detect attacks occuring in a monitored environment.

Random Attacker

An attacker who has not witnessed the target user authenticating.

Secondary Behavioural Biometric

A task-based authentication system where the user action is a separate authentication action, such as inputting a password.

Shoulder-Surfing

An attack where the attacker observes a user inputting a credential, for example by looking over their shoulder.

Stable Observable

A feature of behaviour which can be modelled without excessive variation for the same user.

Task-Based Behavioural Biometric

An authentication system which observes and characterizes the user performing a specific action.

Threat Model

A formalized characterization of how attackers may attack a system.

User Acceptance Rate

A measure of how many times the legitimate user correctly authenticates. Equivalent to the true negative rate.

User Reject Rate

A measure of how many times the legitimate user fails to authenticate. Equivalent to the false positive rate.

Appendix A

Tables of Results

Table A.1: Results Table for Figure 6.2

version	threshold	URR	ARR
1	6	0.1472958333	0.2402095833
	5	0.2300290529	0.3814644513
	4	0.4199231963	0.6265624938
	3	0.7057668956	0.8366319404
2	17	0.3951317164	0.8782339325
	18	0.3315800745	0.8063385076
	20	0.2119676068	0.6345928649
	21	0.1662513194	0.5347416744
	22	0.1276683945	0.4122388539
3	2.0	0.2631002715	0.6035231953
	2.4	0.2357323734	0.5446910313
	1.8	0.2853792145	0.6316054888
	1.4	0.3421135932	0.6948580588
	2.8	0.238235425	0.4795169214
	1.0	0.3955015576	0.7677064628
	3.2	0.2690475151	0.495337284
	0.8	0.433336184	0.8270746987
	0.6	0.4971864682	0.8658311052

Table A.2: Results Table for Figure 6.4

version	threshold	URR	ARR
1	6	0.1323333333	0.2921683333
	5	0.2197064683	0.4228758
	4	0.5137503038	0.486928095
	3	0.7877091368	0.76666666
2	17	0.3871721345	0.7941176471
	18	0.3172592318	0.7078431373
	20	0.2190696003	0.477124183
	21	0.1675638265	0.3882352941
	22	0.1059967309	0.3104575163
3	2.0	0.1802694406	0.4608776844
	2.4	0.1662099265	0.4267507003
	1.8	0.1802694406	0.4942110177
	1.4	0.209796026	0.6133053221
	2.8	0.1472718708	0.3605975724
	1.0	0.2925842955	0.7204948646
	3.2	0.1125696552	0.327264239
	0.8	0.3243790582	0.7067693744
	0.6	0.6609138627	0.8265934498

Table A.3: Results Table for Figure 6.1

version	threshold	URR	ARR
1	6	0.1472958333	0.2400416667
	5	0.2300290529	0.3795440688
	4	0.4199231963	0.5754974215
	3	0.7057668956	0.7354222346
2	17	0.3951317164	0.947800871
	18	0.3315800745	0.9274633496
	20	0.2119676068	0.8290738833
	21	0.1662513194	0.7757756288
	22	0.1276683945	0.7100299247
3	2.0	0.2631002715	0.6783073858
	2.4	0.2357323734	0.6172042006
	1.8	0.2853792145	0.7333711488
	1.4	0.3421135932	0.7925863524
	2.8	0.238235425	0.5709922963
	1.0	0.3955015576	0.8497095022
	3.2	0.2690475151	0.5732498702
	0.8	0.433336184	0.8803176387
	0.6	0.4971864682	0.9240131951

Table A.4: Results Table for Figure 6.3

version	threshold	URR	ARR
1	6	0.1323333333	0.2356666667
	5	0.2197064683	0.297551905
	4	0.5137503038	0.4290929267
	3	0.7877091368	0.6872286567
2	17	0.3871721345	0.9288961039
	18	0.3172592318	0.8810525677
	20	0.2190696003	0.8181507143
	21	0.1675638265	0.7232674476
	22	0.1059967309	0.6517510136
3	2.0	0.1802694406	0.516976215
	2.4	0.1662099265	0.396121578
	1.8	0.1802694406	0.541297262
	1.4	0.209796026	0.6609424582
	2.8	0.1472718708	0.3454340866
	1.0	0.2925842955	0.753100568
	3.2	0.1125696552	0.3241147736
	0.8	0.3243790582	0.8267014508
	0.6	0.6609138627	0.8495454431

Table A.5: Results Table for Figure 6.6

version	threshold	URR	ARR
1	6	0.1443033333	0.2391666667
	5	0.227964536	0.363145636
	4	0.4386886178	0.5462165225
	3	0.7221553439	0.725783519
2	17	0.3935398	0.9440199175
	18	0.328715906	0.9181811932
	20	0.2133880055	0.8268892495
	21	0.1665138208	0.7652739926
	22	0.1233340618	0.6983741425
3	2.0	0.2465341053	0.6460411516
	2.4	0.221827884	0.5729876761
	1.8	0.2643572597	0.6949563714
	1.4	0.3156500798	0.7662575736
	2.8	0.2200427142	0.5258806544
	1.0	0.3749181052	0.8303877153
	3.2	0.2377519432	0.5234228509
	0.8	0.4115447588	0.8695944012
	0.6	0.4636601122	0.9091196447

Table A.6: Results Table for Figure 6.5

version	threshold	URR	ARR
1	6	0.1443033333	0.2506013333
	5	0.227964536	0.389746721
	4	0.4386886178	0.598635614
	3	0.7221553439	0.8226388843
2	17	0.3935398	0.8614106754
	18	0.328715906	0.7866394336
	20	0.2133880055	0.6030991285
	21	0.1665138208	0.5054403984
	22	0.1233340618	0.3918825864
3	2.0	0.2465341053	0.5749940931
	2.4	0.221827884	0.5211029651
	1.8	0.2643572597	0.6041265946
	1.4	0.3156500798	0.6785475115
	2.8	0.2200427142	0.4557330516
	1.0	0.3749181052	0.7582641432
	3.2	0.2377519432	0.461722675
	0.8	0.4115447588	0.8030136339
	0.6	0.4636601122	0.8533931755

Table A.7: Results Table for Figure 6.7

attacker model	threshold	URR	ARR
random	6	0.1443033333	0.2391666667
	5	0.227964536	0.363145636
	4	0.4386886178	0.5462165225
	3	0.7221553439	0.725783519
imitation	6	0.1443033333	0.2506013333
	5	0.227964536	0.389746721
	4	0.4386886178	0.598635614
	3	0.7221553439	0.8226388843

Table A.8: Results Table for Figure 6.8

attacker model	threshold	URR	ARR
random	17	0.3935398	0.9440199175
	18	0.328715906	0.9181811932
	20	0.2133880055	0.8268892495
	21	0.1665138208	0.7652739926
	22	0.1233340618	0.6983741425
imitation	17	0.3935398	0.8614106754
	18	0.328715906	0.7866394336
	20	0.2133880055	0.6030991285
	21	0.1665138208	0.5054403984
	22	0.1233340618	0.3918825864

Table A.9: Results Table for Figure 6.9

attacker model	threshold	URR	ARR
random	2.0	0.2465341053	0.6460411516
	2.4	0.221827884	0.5729876761
	1.8	0.2643572597	0.6949563714
	1.4	0.3156500798	0.7662575736
	2.8	0.2200427142	0.5258806544
	1.0	0.3749181052	0.8303877153
	3.2	0.2377519432	0.5234228509
	0.8	0.4115447588	0.8695944012
	0.6	0.4636601122	0.9091196447
imitation	2.0	0.2465341053	0.5749940931
	2.4	0.221827884	0.5211029651
	1.8	0.2643572597	0.6041265946
	1.4	0.3156500798	0.6785475115
	2.8	0.2200427142	0.4557330516
	1.0	0.3749181052	0.7582641432
	3.2	0.2377519432	0.461722675
	0.8	0.4115447588	0.8030136339
	0.6	0.4636601122	0.8533931755

Appendix B

Research Materials

B.1 Demographic Questionnaire

1. Age:
2. Sex:
3. Education Level (check all that apply)

Completed High School Completed Some University or College Completed Some post-graduate degree

4. In what field did/do you study?

5. Have you participated in a touch screen related usability study before? If yes, describe it.

6. Do you own a touch screen phone or device: Yes No
7. Do you own a tablet PC: Yes No
8. What touch-based OS system do you use on a regular basis? (check all that apply)

   Windows (with touch) IOS Android Linex Blackberry X Other
9. Do you protected your device(s) with a password, pin or other: Yes No
10. What type of password system do you commonly use?

11. Have you ever gesture-based passwords? Yes No

B.2 Participant Usability Impression Questionnaire

1. After using this application do you think swipe gesture passwords is faster than regular touch screen passwords? Why or why not?

   No Not Sure Maybe Probably Yes

2. Would you use this method over your current password method? Why or why not?

   No Not Sure Maybe Probably Yes

3. Do you think you'd easily be able to repeat this password in a few weeks?

   No Not Sure Maybe Probably Yes

4. In what situations do you think swipe-authentication should be used (if any?)

5. Add any suggestions you have for improving the application.

Bibliography

[1] Adams, A., and Sasse, M. A. Users are not the enemy. *Communications of the ACM 42*, 12 (1999), 40–46.

[2] Apple. ios security. https://www.apple.com/business/docs/iOS_Security_Guide.pdf, 2015. 2015-10-16.

[3] Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., and Smith, J. M. Smudge attacks on smartphone touch screens. *WOOT 10* (2010), 1–7.

[4] Barkhuus, L., and Rode, J. A. From mice to men-24 years of evaluation in chi. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2007), Citeseer.

[5] Barrett, D. One surveillance camera for every 11 people in britain, says cctv survey. http://www.telegraph.co.uk/technology/10172298/One-surveillance-camera-for-every-11-people-in-Britain-says-CCTV-survey.html, 2013. 2015-10-16.

[6] Ben-Asher, N., Kirschnick, N., Sieger, H., Meyer, J., Ben-Oved, A., and Möller, S. On the need for different security methods on mobile phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (2011), ACM, pp. 465–473.

[7] Berndt, D. J., and Clifford, J. Using dynamic time warping to find patterns in time series. In *KDD workshop* (1994), vol. 10, Seattle, WA, pp. 359–370.

[8] Biddle, R., Chiasson, S., and Van Oorschot, P. C. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR) 44*, 4 (2012), 19.

[9] Biggio, B., Akhtar, Z., Fumera, G., Marcialis, G. L., and Roli, F. Security evaluation of biometric authentication systems under real spoofing attacks. *IET biometrics 1*, 1 (2012), 11–24.

[10] Bo, C., Zhang, L., Li, X.-Y., Huang, Q., and Wang, Y. Silentsense: silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th annual international conference on Mobile computing & networking* (2013), ACM, pp. 187–190.

[11] Bonneau, J. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 538–552.

[12] Bowyer, K. W., Hollingsworth, K. P., and Flynn, P. J. A survey of iris biometrics research: 2008–2010. In *Handbook of iris recognition*. Springer, 2013, pp. 15–54.

[13] Brooks, D. Assessing vulnerabilities of biometric readers using an applied defeat evaluation methodology.

[14] Brostoff, S., and Sasse, M. A. Are passfaces more usable than passwords? a field trial investigation. In *People and Computers XIVUsability or Else!* Springer, 2000, pp. 405–424.

[15] Burge, M. J., and Bowyer, K. *Handbook of iris recognition*. Springer Science & Business Media, 2013.

[16] Cha, S.-H., and Tappert, C. C. Automatic detection of handwriting forgery. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on* (2002), IEEE, pp. 264–267.

[17] Cha, S.-H., Tappert, C. C., Gibbons, M., and Chee, Y.-M. Automatic detection of handwriting forgery using a fractal number estimate of wrinkliness. *International Journal of Pattern Recognition and Artificial Intelligence 18*, 07 (2004), 1361–1371.

[18] Chakka, M. M., Anjos, A., Marcel, S., Tronci, R., Muntoni, D., Fadda, G., Pili, M., Sirena, N., Murgia, G., Ristori, M., et al. Competition on counter measures to 2-d facial spoofing attacks. In *Biometrics (IJCB), 2011 International Joint Conference on* (2011), IEEE, pp. 1–6.

[19] Coli, P., Marcialis, G. L., and Roli, F. Vitality detection from fingerprint images: a critical survey. In *Advances in Biometrics*. Springer, 2007, pp. 722–731.

[20] Damousis, I. G., and Argyropoulos, S. Four machine learning algorithms for biometrics fusion: A comparative study. *Applied Computational Intelligence and Soft Computing 2012* (2012), 6.

[21] De Luca, A., Hang, A., Brudy, F., Lindner, C., and Hussmann, H. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), ACM, pp. 987–996.

[22] Dell'Amico, M., Michiardi, P., and Roudier, Y. Password strength: An empirical analysis. In *INFOCOM, 2010 Proceedings IEEE* (2010), IEEE, pp. 1–9.

[23] Derawi, M. O., Nickel, C., Bours, P., and Busch, C. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on* (2010), IEEE, pp. 306–311.

[24] Feng, T., Liu, Z., Kwon, K.-A., Shi, W., Carbunar, B., Jiang, Y., and Nguyen, N. K. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for* (2012), IEEE, pp. 451–456.

[25] Findling, R. D., and Mayrhofer, R. Towards face unlock: on the difficulty of reliably detecting faces on mobile phones. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia* (2012), ACM, pp. 275–280.

[26] Florencio, D., and Herley, C. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 657–666.

[27] Frank, M., Biedert, R., Ma, E., Martinovic, I., and Song, D. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on 8*, 1 (2013), 136–148.

[28] Furnell, S., Clarke, N., and Karatzouni, S. Beyond the pin: Enhancing user authentication for mobile devices. *Computer fraud & security 2008*, 8 (2008), 12–17.

[29] Gafurov, D. A survey of biometric gait recognition: Approaches, security and challenges. In *Annual Norwegian Computer Science Conference* (2007), Citeseer, pp. 19–21.

[30] Gafurov, D., Snekkenes, E., and Bours, P. Spoof attacks on gait authentication system. *Information Forensics and Security, IEEE Transactions on 2*, 3 (2007), 491–502.

[31] Gallagher, S. Opm breach included five times more stolen fingerprints. http://arstechnica.com/security/2015/09/opm-breach-included-five-times-more-stolen-fingerprints/, 2015. 2015-10-09.

[32] Graziano, D. Getting to know the galaxy s5s fingerprint scanner. http://www.cnet.com/how-to/getting-to-know-the-galaxy-s5s-fingerprint-scanner/, 2015. 2015-10-14.

[33] Greenberg, S., and Buxton, B. Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), ACM, pp. 111–120.

[34] Hadid, A., Ghahramani, M., Kellokumpu, V., Pietikainen, M., Bustard, J., and Nixon, M. Can gait biometrics be spoofed? In *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), IEEE, pp. 3280–3283.

[35] Hadid, A., Heikkilä, J. Y., Silvén, O., and Pietikäinen, M. Face and eye detection for person authentication in mobile phones. In *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on* (2007), IEEE, pp. 101–108.

[36] Harbach, M., von Zezschwitz, E., Fichtner, A., De Luca, A., and Smith, M. Itsa hard lock life: A field study of smartphone (un) locking behavior and risk perception. In *Symposium on Usable Privacy and Security (SOUPS)* (2014).

[37] Herley, C., and Van Oorschot, P. A research agenda acknowledging the persistence of passwords. *Security & Privacy, IEEE 10*, 1 (2012), 28–36.

[38] Jafri, R., and Arabnia, H. R. A survey of face recognition techniques. *journal of information processing systems 5*, 2 (2009), 41–68.

[39] Jain, A. K., Ross, A., and Prabhakar, S. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on 14*, 1 (2004), 4–20.

[40] Jakobsson, M., and Liu, D. Your password is your new pin. In *Mobile Authentication*. Springer, 2013, pp. 25–36.

[41] Jakobsson, M., Shi, E., Golle, P., and Chow, R. Implicit authentication for mobile devices. In *Proceedings of the 4th USENIX conference on Hot topics in security* (2009), USENIX Association, pp. 9–9.

[42] Jermyn, I., Mayer, A. J., Monrose, F., Reiter, M. K., Rubin, A. D., et al. The design and analysis of graphical passwords. In *Usenix Security* (1999).

[43] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and Lopez, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 523–537.

[44] Khan, H., Atwater, A., and Hengartner, U. A comparative evaluation of implicit authentication schemes. In *Research in Attacks, Intrusions and Defenses*. Springer, 2014, pp. 255–275.

[45] Khan, H., Atwater, A., and Hengartner, U. Itus: an implicit authentication framework for android. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), ACM, pp. 507–518.

[46] Killourhy, K. S., Maxion, R., et al. Comparing anomaly-detection algorithms for keystroke dynamics. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on* (2009), IEEE, pp. 125–134.

[47] Kinnunen, T., Wu, Z.-Z., Lee, K. A., Sedlak, F., Chng, E. S., and Li, H. Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (2012), IEEE, pp. 4401–4404.

[48] Komanduri, S., Shay, R., Kelley, P. G., Mazurek, M. L., Bauer, L., Christin, N., Cranor, L. F., and Egelman, S. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), ACM, pp. 2595–2604.

[49] Kose, N., and Dugelay, J.-L. On the vulnerability of face recognition systems to spoofing mask attacks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 2357–2361.

[50] Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., and Srivastava, J. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM* (2003), SIAM, pp. 25–36.

[51] Lee, C.-H., Soong, F. K., and Paliwal, K. *Automatic speech and speaker recognition: advanced topics*, vol. 355. Springer Science & Business Media, 2012.

[52] Lin, C.-C., Liang, D., Chang, C.-C., and Yang, C.-H. A new non-intrusive authentication method based on the orientation sensor for smartphone users. In *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on* (2012), IEEE, pp. 245–252.

[53] Maltoni, D., Maio, D., Jain, A. K., and Prabhakar, S. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[54] Matera, M., Rizzo, F., and Carughi, G. T. Web usability: Principles and evaluation methods. In *Web engineering*. Springer, 2006, pp. 143–180.

[55] Mazurek, M. L., Komanduri, S., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., Kelley, P. G., Shay, R., and Ur, B. Measuring password guessability for an entire university. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 173–186.

[56] McCarney, D. Password managers: comparative evaluation, design, implementation and empirical analysis. *Carleton University* (2013).

[57] Micallef, N., Just, M., Baillie, L., Halvey, M., and Kayacik, H. G. Why aren't users using protection? investigating the usability of smartphone locking. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2015), ACM, pp. 284–294.

[58] Moss, S. Neighbourhood watch: how domestic cctv is sweeping the uk. http://www.theguardian.com/world/2014/dec/19/neighbourhood-watch-domestic-cctv-sweeping-uk, 2014. 2015-10-16.

[59] Myagmar, S., Lee, A. J., and Yurcik, W. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)* (2005), vol. 2005, pp. 1–8.

[60] Pashalidis, A., and Mitchell, C. J. A taxonomy of single sign-on systems. In *Information security and privacy* (2003), Springer, pp. 249–264.

[61] Rathgeb, C., and Uhl, A. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security 2011*, 1 (2011), 1–25.

[62] Riva, O., Qin, C., Strauss, K., and Lymberopoulos, D. Progressive authentication: Deciding when to authenticate on mobile phones. In *USENIX Security Symposium* (2012), pp. 301–316.

[63] Schaub, F., Deyhle, R., and Weber, M. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia* (2012), ACM, p. 13.

[64] Shekhar, S., Patel, V. M., Nasrabadi, N. M., and Chellappa, R. Joint sparse representation for robust multimodal biometrics recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 36*, 1 (2014), 113–126.

[65] Shi, E., Niu, Y., Jakobsson, M., and Chow, R. Implicit authentication through learning user behavior. In *Information Security*. Springer, 2011, pp. 99–113.

[66] Shi, W., Yang, J., Jiang, Y., Yang, F., and Xiong, Y. Senguard: Passive user identification on smartphones using multiple sensors. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on* (2011), IEEE, pp. 141–148.

[67] Somayaji, A. B. *Operating system stability and security through process homeostasis*. PhD thesis, The University of New Mexico, 2002.

[68] Staggers, N., and Norcio, A. F. Mental models: concepts for human-computer interaction research. *International Journal of Man-machine studies 38*, 4 (1993), 587–605.

[69] Support, A. Use touch id on iphone and ipad. https://support.apple.com/en-ca/HT201371, 2015. 2015-10-14.

[70] Tari, F., Ozok, A., and Holden, S. H. A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. In *Proceedings of the second symposium on Usable privacy and security* (2006), ACM, pp. 56–66.

[71] Thorpe, J., and van Oorschot, P. C. Human-seeded attacks and exploiting hot-spots in graphical passwords. In *USENIX Security* (2007), vol. 7.

[72] Tresadern, P., Cootes, T. F., Poh, N., Matejka, P., Hadid, A., Levy, C., McCool, C., and Marcel, S. Mobile biometrics: Combined face and voice verification for a mobile platform. *IEEE pervasive computing*, 1 (2013), 79–87.

[73] Vildjiounaite, E., Makela, S.-M., Lindholm, M., Kyllonen, V., and Ailisto, H. Increasing security of mobile devices by decreasing user effort in verification. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on* (2007), IEEE, pp. 80–80.

[74] Von Zezschwitz, E., Dunphy, P., and De Luca, A. Patterns in the wild: a field study of the usability of pattern and pin-based authentication on mobile devices. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services* (2013), ACM, pp. 261–270.

[75] Wang, F., and Han, J. Robust multimodal biometric authentication integrating iris, face and palmprint. *Information technology and control 37*, 4 (2015).

[76] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. Passpoints: Design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies 63*, 1 (2005), 102–127.

[77] Yan, J. J., Blackwell, A. F., Anderson, R. J., and Grant, A. Password memorability and security: Empirical results. *IEEE Security & privacy 2*, 5 (2004), 25–31.

[78] Yazji, S., Chen, X., Dick, R. P., and Scheuermann, P. Implicit user re-authentication for mobile devices. In *Ubiquitous Intelligence and Computing*. Springer, 2009, pp. 325–339.