Evaluating the Efficacy of Implicit Authentication Under Realistic Operating Scenarios

by

Hassan Khan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contribution

The content in Chapter 3 and Chapter 6 was co-authored with fellow PhD student Erinn Atwater, who contributed to the development of Itus. The content in Chapter 4 and Chapter 5 was co-authored with Prof. Daniel Vogel. The rest of this thesis contains original content and was authored under the supervision of Prof. Urs Hengartner.

Abstract

Smartphones contain a wealth of personal and corporate data. Several surveys have reported that about half of the smartphone owners do not configure primary authentication mechanisms (such as PINs, passwords, and fingerprint- or facial-recognition systems) on their devices to protect data due to usability concerns. In addition, primary authentication mechanisms have been subject to operating system flaws, smudge attacks, and shoulder surfing attacks. These limitations have prompted researchers to develop implicit authentication (IA), which authenticates a user by using distinctive, measurable patterns of device use that are gathered from the device users without requiring deliberate actions. Researchers have claimed that IA has desirable security and usability properties and it seems a promising candidate to mitigate the security and usability issues of primary authentication mechanisms.

Our observation is that the existing evaluations of IA have a preoccupation with accuracy numbers and they have neglected the deployment, usability and security issues that are critical for its adoption. Furthermore, the existing evaluations have followed an ad-hoc approach based on synthetic datasets and weak adversarial models. To confirm our observations, we first identify a comprehensive set of evaluation criteria for IA schemes. We gather real-world datasets and evaluate diverse and prominent IA schemes to question the efficacy of existing IA schemes and to gain insight into the pitfalls of the contemporary evaluation approach to IA. Our evaluation confirms that under realistic operating conditions, several prominent IA schemes perform poorly across key evaluation metrics and thereby fail to provide adequate security.

We then examine the usability and security properties of IA by carefully evaluating promising IA schemes. Our usability evaluation shows that the users like the convenience offered by IA. However, it uncovers issues due to IA's transparent operation and false rejects, which are both inherent to IA. It also suggests that detection delay and false accepts are concerns to several users. In terms of security, our evaluation based on a realistic, stronger adversarial model shows the susceptibility of highly accurate, touch input-based IA schemes to shoulder surfing attacks and attacks that train an attacker by leveraging raw touch data of victims. These findings exemplify the significance of realistic adversarial models.

These critical security and usability challenges remained unidentified by the previous research efforts due to the passive involvement of human subjects (only as behavioural data sources). This emphasizes the need for rapid prototyping and deployment of IA for an active involvement of human subjects in IA research. To this end, we design, implement,

evaluate and release in open source a framework, which reduces the re-engineering effort in IA research and enables deployment of IA on off-the-shelf Android devices.

The existing authentication schemes available on contemporary smartphones fail to provide both usability and security. Authenticating users based on their behaviour, as suggested by the literature on IA, is a promising idea. However, this thesis concludes that several results reported in the existing IA literature are misleading due to the unrealistic evaluation conditions and several critical challenges in the IA domain need yet to be resolved. This thesis identifies these challenges and provides necessary tools and design guidelines to establish the future viability of IA.

Supervisor

- Urs Hengartner

Committee & Examiners

- Danial Vogel
- Ian Goldberg
- Mahesh Tripunitara
- Paul van Oorschot

Acknowledgements

I thank Almighty God for giving me the ability to accomplish this. My father has been my role model and instrumental in helping me become who I am today. His active involvement throughout my early and university education has provided me with a solid foundation, which prepared me well for the research challenges that I faced. He has ensured that I have access to the best possible resources to successfully accomplish my goals. My mother's love, support and prayers made this journey much easier. My wife, Asma, has been incredibly understanding throughout this degree. She has sacrificed several special occasions to my deadlines, yet she always provided unwavering love and support. My life has been truly enriched by her presence. Mustafa, my brother, has provided me with a support system without which I would not have been able to continue my endeavors. Afoo and Ibrahim's pictures and videos has kept me cheerful throughout. Awais, my friend, has always been around. His sincerity and willingness to help is unparalleled.

I would like to thank my advisor, Urs Hengartner. Urs was flexibile in letting me pursue my research interests, yet thoroughly engaged with my work. He guided, mentored and challenged me during my research by nudging me to approach my research with clarity and focus. He was always available to provide assistance on the research challenges that I encountered. I am truly indebted to him for the way he has shaped me into a researcher. I could not have asked for a better advisor.

I am grateful to Daniel Vogel for his "informal co-supervision" on some of the research problems that I worked on. His insightful discussions and suggestions have greatly improved the quality this thesis. I am grateful to Paul van Oorschot for agreeing to be my external examiner and for his invaluable critique and constructive feedback. I am thankful to Ian Goldberg for his comprehensive feedback and Mahesh Tripunitara for agreeing to be on my examining committee. I would also like to thank members of CrySP for their support and feedback particularly Erinn, Tariq, Jalaj, Tao and Cecylia. Finally, I am thankful to my friends and mentors, Khayam and Fauzan, for introducing me to the world of scientific research.

This research was supported by the Government of Ontario through Ontario Trillium Scholarship, Google's Focused Research Award and Cheriton Graduate Scholarship.

Dedication

To Abu, for making all my accomplishments possible..

To Ami, for love & prayers..

To Asma, for loving support..

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

User authentication is the first line of defence as a preventive control to stop unwanted access to smartphones. This control works effectively when the authentication scheme is both secure and usable. However, an overwhelmingly large proportion of smartphone users does not configure the existing authentication mechanisms due to their usability issues [EJP+14, HVZF+14]. Bring your own devices (BYODs) that contain confidential corporate data and require complex password policies add an additional layer of complexity to this problem [Ste14]. Implicit authentication (IA) uses behavioural biometrics to continuously and transparently authenticate smartphone users. Researchers have suggested to use IA as a middle ground for smartphone users who do not use primary authentication mechanisms due to their usability issues or second line of defence for the scenarios where additional security is desired [FBM+13, LZX13]. In this thesis, we advance research in the area of implicit authentication through deployability, usability, and security and performance evaluations of IA schemes.

1.1 Authentication on Smartphones

In order to prevent unauthorized access on smartphones, primary authentication mechanisms (such as PINs, passwords, and fingerprint- or facial-recognition systems) are used. A major issue with these mechanisms is their all-or-nothing access approach where users have to pass an authentication challenge in order to access any resource on the device. This approach is unsuitable for smartphones where 40% of the frequently accessed apps (such as games, weather, navigation, news and music apps) do not contain confidential data [HRS+12]. Furthermore, unlike desktops or laptops, smartphone sessions are short

and frequent and PIN entry for every short session is inconvenient for users [HRS+12]. Due to these usability issues of primary authentication mechanisms, several independent studies show that between 40–50% of smartphone owners do not configure them on their devices [EJP+14, HVZF+14, Loo16].

In addition to the usability issues, primary authentication mechanisms that rely on a secret that the device owner knows have been subject to operating system flaws [Thr16], smudge attacks [AGM+10] and shoulder surfing attacks [LZX13]. Operating system flaws enable an adversary to bypass a primary authentication mechanism altogether [Thr16]. Smudge attacks leverage the oily residue of the finger left on the touchscreen during the secret entry to significantly reduce the input space for a bruteforce attack on PINs and pass-locks for an adversary [AGM+10]. Similarly, authentication mechanisms that rely on who the device owner is have also been subject to novel attacks and both fingerprint- and facial-recognition schemes have been successfully bypassed by researchers [And16, Zdn16].

Organizations may employ mobile device management (MDM) solutions to enforce password policies on BYODs to protect corporate data [Ste14]. These solutions require a device-level authentication scheme, such as a PIN or fingerprint recognition, while complex passwords may be required to further establish identity for corporate email, messaging, or calendar apps. Research efforts have demonstrated that the constrained keyboards on smartphones are a bottleneck when users are authenticating using complex passwords [SDW12]. While MDM solutions can enforce password policies, they aggravate the usability issues. According to a prediction by Gartner, 20% of enterprise BYOD programs will fail due to overly restrictive policies of MDMs [Gar16].

Despite these usability and security limitations, primary authentication mechanisms are the first line of defense against misuse on mobile devices. IA has security limitations of its own as it is subject to false accepts (type II errors) and detection delays (delay in classification decision due to the unavailability of behavioural data). Therefore, IA is not an alternative for primary authentication mechanisms and we strongly advocate the use of latter as the first line of defence. However, similar to Frank et al. [FBM+13] and Li et al. [LZX13], we suggest that primary authentication mechanisms can be complemented using IA schemes to provide a second line of defense for scenarios where additional protection is required (such as BYOD environments). Furthermore, IA schemes can be used to provide protection to device owners who do not use primary authentication mechanisms due to their usability issues.

1.2 Implicit Authentication

IA schemes provide authentication by using distinctive, measurable patterns of device use that are gathered from mobile device users without requiring deliberate actions [CRS13]. For example, a key input behaviour-based classifier learns to recognize a device owner's keystroke patterns and then monitors for anomalies in real-time key events to detect misuse. As a result, when an adversary gains access to the victim's device and enters some text, IA may be able to detect the misuse and lock the device or alert the device owner via email.

IA schemes are useful in many scenarios, such as for:

- *Enterprises:* may use IA to provide additional security for their corporate apps on BYODs, where the company may be unable to force users to use explicit authentication or enforce password policies on their device due to usability issues or device limitations.
- *Banks, mail service providers and OSNs:* may allow their users to save passwords on the device to reduce authentication overhead every time users access their apps. An IA scheme can protect the identity and data of the consumers on mobile devices where the users are only authenticated in case the IA scheme detects a misuse.
- *Security-conscious app developers:* can use IA to protect user data in their apps (e.g., browser, communication, and photo gallery apps).

To provide IA support on smartphones, many IA schemes have been proposed by the research community that rely on a diverse set of behavioural features including a user's device usage patterns [BDG14, KJB+14, LCPD10, JSGC09, SYJ+11, ZWWZ13], touchscreen input behaviour [BZL+13, DLHB+12, FLK+12b, FYY+14, FBM+13, LZX13, LCL13, SLS13, ZFS13], text entry patterns [CF07, DZZ14, FZCS13, GMCB14, HCP09, MCGCN, ZSKF09], and gait patterns [FMP10, GHS06, JXBJ+12, KWM10, MLV+05, MM13]. More details on how these IA schemes operate are provided in Chapter 2.

1.3 Scope and Threat Model

In this thesis, we advance the research in the area of IA on smartphones. We specifically only consider the scenario where IA is used for user-to-device authentication and not for user-to-remote-site authentication. This scope is also applicable to the other authentication

schemes (e.g., PINs and passwords) discussed in this work. Moreover, while similar implicit factors might be used to establish identity of users on the web [BHvOS15, CPH11] or to fingerprint devices [VGSPJ16], such use of implicit factors is out of scope.

We use the standard threat model used for IA schemes [BZL+13, LZX13, XZL14]. An adversary attempts to gain unauthorized access to a victim's device, which employs an IA scheme to continuously authenticate the device user. The victim has either not configured a primary authentication scheme (such as a PIN) or the adversary has bypassed it completely through known mechanisms like shoulder surfing or smudge attacks [AGM+10]. The adversary is aware of the presence of an IA scheme on the victim's device. Finally, an adversary may be uninformed or informed about the victim. An uninformed adversary may be a curious stranger/thief who found/stole a device, while an informed adversary might be an inquisitive friend, co-worker, or family member. The difference between these two types of adversary is that the latter may have additional knowledge about the behaviour of the victim (e.g., he may know that the victim always uses his right hand for swiping). Finally, we do not consider authentication secret capture attacks through network interception (which is precluded through our user-to-device authentication scenario) and malware.

1.4 Open Challenges

The usability and security limitations of primary authentication mechanisms have prompted researchers to develop IA schemes. The focus of the majority of the existing IA literature is on proposing novel behavioural biometrics and features to improve the detection accuracy of IA schemes. Several critical factors in terms of usability, security and deployability of IA have been neglected in the existing literature. In this section, we provide an overview of the neglected issues and highlight their significance.

Primary authentication mechanisms operate in a consistent manner. If a user knows the secret or possesses the required physiologic biometric, she will be able to accurately establish her identity in a predictable time interval. On the other hand, IA relies on behavioural biometrics, which may vary or the data available to establish a user's identity may not be sufficient. For a wide-spread adoption of IA, it is critical to demonstrate that some IA schemes provide acceptable performance across all evaluation metrics that matter for IA. However, this is challenging since to the best of our knowledge, an exhaustive set of evaluation metrics for IA has not been provided in the contemporary IA literature. Therefore, the first challenge is the identification of a comprehensive set of evaluation metrics for IA. Once evaluation metrics are defined, an unbiased evaluation of IA schemes needs to

be performed on real-world datasets to determine whether they provide acceptable performance. Another challenge is to elicit users' preferences to define an acceptable standard for the performance of IA. If the performance evaluation of IA shows that it meets an acceptable standard, it will make a stronger case for IA adoption.

Existing IA literature has assumed without empirical evidence that since IA authenticates without requiring explicit input, it is more usable. However, despite the reasonably high detection accuracy of some IA schemes, these schemes are still subject to false accepts (type II errors), false rejects (type I errors) and detection delays, which could introduce new usability issues and affect users' security perceptions. If IA is unsure about the user's identity, it naturally resorts to an interrupt-authenticate approach in which the current task is pushed to the background and the user has to explicitly authenticate to establish their identity [FYY+14, LZX13]. This interrupt-authenticate approach for false rejects is quite different from consistent authentication of primary authentication schemes and it is unclear how it affects usability. Similarly, it is not obvious whether the usability-security trade-off offered by IA overcomes the perception of security given the risks of false accepts and delay in detection of an intruder. Finding answers to these usability aspects and security perceptions of IA will determine whether IA is a potential candidate for deployment to mitigate the usability issues.

To further the evaluation of IA, we aim to investigate the threat of mimicry attacks on prominent IA schemes. IA schemes rely on distinct patterns of device usage, which can be observed over-the-shoulder by an adversary or the usage pattern can be recorded by a malicious application on the device. For the former case, the adversary can try to mimic the behaviour of the victim to gain access to the device and for the latter case, the adversary can train himself to behave like the victim. In both cases, it is important to understand how successful such mimicry attacks are on IA schemes. The resistance provided by IA schemes against these mimicry attacks will determine the protection level that they offer in deployments.

Finally, we explore the deployment issues of IA schemes that may impede its adoption. Since the majority of IA schemes employ behavioural features that can only be gathered at the system level, IA deployment either requires platform vendor support (to be included at the application framework level or at the kernel level) or it requires rooting of the device. However, providing IA support at the system level has flexibility and extensibility issues of its own. These flexibility issues arise due to the fact that different apps have different characteristics and a generic system-level IA scheme may not be suitable for different apps. The difference in app characteristics can also severely reduce the accuracy of IA schemes by as much as 20% [FYY+14, KH14]. In terms of extensibility, system-level IA mechanisms would need to be managed by the platform developers or some central authority. However,

IA is a relatively new area that still experiences radical revisions due to the research findings in such areas as the use of novel sensors or wearable devices for IA [MMMC+14, SSTA14], and the research findings on the usability and acceptance of IA schemes. The platform developers in this case are more inclined toward accepting mature contributions, which will lock out many developers. We investigate how these deployment issues can be mitigated for the deployment of IA on off-the-shelf devices.

1.5 Thesis Statement

The major goal of this research effort is to perform a realistic evaluation of the deployability, security and usability related issues of IA. Based on the findings of these evaluations, we also propose strategies to make IA more practical so that it can be adopted. Our thesis statement is:

Evaluating an authentication scheme based only on accuracy numbers determined from synthetic datasets and weak adversarial models and neglecting the scheme's deployment, usability and security concerns can lead to misleading results, as illustrated for the case of implicit authentication on smartphones.

Four main research objectives of this thesis are described below:

Objective 1: Identify the key evaluation metrics that should be used to evaluate the efficacy of IA schemes. Perform an evaluation of existing IA schemes on this extended evaluation criteria to identify the IA schemes that appear most promising for extensive deployability, security and usability evaluations.

Objective 2: Determine how the inaccuracy and the detection delay of IA schemes impacts their usability and security perceptions. Ascertain that the usability and security perceptions of IA warrant its adoption.

Objective 3: Identify realistic attacks on the most promising IA scheme identified and perform an evaluation of those attacks to evaluate its security.

Objective 4: Identify the forestalling obstacles to the deployment of IA and address those by providing a flexible and extensible IA deployment mechanism.

1.6 Main Contributions of this Research

This research contributes original ideas and knowledge to the IA domain. We identify an extensive evaluation criteria for IA and used that to determine which schemes work best in terms of detection accuracy, detection delay and processing complexity under different operating conditions in terms of attack scenarios and availability of training and classification data. We conduct an experiment to mount two realistic attacks on touch input based IA schemes to determine the susceptibility of these schemes to targeted mimicry attacks. We conduct the first extensive lab and field study to gain insights into usability and security perceptions of IA. Finally, we propose an effective deployment scenario for IA on smartphones and provide a framework to prototype and readily deploy IA schemes.

The main contributions of this research are:

1. We evaluate six diverse IA schemes on real-world datasets using eight evaluation metrics. Our results show that while the majority of IA schemes provide reasonable accuracy with low detection delay, IA schemes based on touchscreen input behaviour outperform others by providing near real-time misuse detection with high accuracy. We also find that some IA schemes perform well in terms of detection accuracy but frequently do not have enough data available for classification.

2. We perform the first ever study to evaluate the usability and security perceptions of IA. We find that in terms of performance, the interrupts due to false rejects impose overhead for individual authentications, but IA increases amortized task performance without affecting the error rate. For usability perceptions, while we found no significant difference between IA and explicit authentication using the system usability scale (SUS) survey, more users agreed that IA was more convenient. On the other hand, annoyance is a potential issue with IA in terms of usability and the detection delay and false accepts were a security concern for users. We also found that that the majority of our participants were interested in adopting or trying IA with possibility of adoption.

3. We propose two realistic attack scenarios from malicious insiders such as shoulder surfing attacks and offline training attacks (where an attacker trains on a victim's raw touch data). We perform the first evaluation of touch input based IA schemes against these realistic attack scenarios to show that it is surprisingly easy to bypass these schemes. We show the accepted assumption that shoulder surfing attacks on touch IA are infeasible due to the hidden nature of some features is incorrect. We outline

methods and provide the necessary apparatus for malicious insiders to perform offline training attacks without installing a malicious app on their victims' devices.

4. We identify the deployment challenges to IA schemes and show there is a need for a framework that supports: different behavioural classifiers, given that different apps have different requirements; app developers using IA without becoming domain experts; and real-time classification on resource-constrained mobile devices. We develop Itus, an IA framework for Android that allows the research community to improve IA schemes incrementally, while allowing app developers to adopt these improvements at their own pace.

1.7 Overview of Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides necessary background and an overview of related work on authentication in smartphones, usability evaluation of authentication schemes on smartphones and it surveys existing IA proposals and frameworks. Chapter 3 addresses *Objective 1* of Section 1.5. It outlines an extensive evaluation criteria for IA evaluations and provides results of an evaluation of six diverse IA schemes on the outlined criteria. Chapter 4 addresses *Objective 2* and presents the results of our usability and security perception evaluation of IA. To address *Objective 3*, Chapter 5 investigates the efficacy of two types of attacks on three prominent touch input based IA schemes. To meet *Objective 4*, Chapter 6 identifies the deployment challenges to IA, builds a case for app-level deployment of IA and presents a framework that can be used to deploy IA at the app level. Finally, Chapter 7 discusses whether our findings support our thesis statement. It also outlines future research directions and offers concluding remarks.

1.8 Related Publications

Significant parts of the research presented in this thesis have been peer-reviewed and published in academic conferences. I am the primary author on the following papers based on the work from this thesis. As indicated in the statement of contribution, parts of this work were done in collaboration with fellow PhD student Erinn Atwater, Prof. Daniel Vogel, and my advisor, Prof. Urs Hengartner.

H. Khan, U. Hengartner, and D. Vogel. Targeted mimicry attacks on touch input implicit authentication. In the proceedings of the 14$^{\text{th}}$ Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), 2016.

H. Khan, U. Hengartner, and D. Vogel. Usability and security perceptions of implicit authentication: convenient, secure, sometimes annoying. In the proceedings of the 11th Symposium on Usable Privacy and Security (SOUPS), 2015.

H. Khan, A. Atwater and U. Hengartner. Itus: an implicit authentication framework for Android. In the proceedings of the 20th Annual International Conference on Mobile Computing & Networking (MobiCom), 2014.

H. Khan, A. Atwater and U. Hengartner. A comparative evaluation of implicit authentication schemes. In the proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID), 2014.

H. Khan and U. Hengartner. Towards application-centric implicit authentication on smartphones. In the proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile), 2014.

Chapter 2

Background

In this chapter, we first outline the primary authentication mechanisms available on modern smartphones and their usability and security issues. We then discuss the relevant solutions that have been proposed to mitigate the limitations of primary authentication mechanisms including the state of the art in the IA domain. Finally, we provide a brief overview of classification in machine learning and define the relevant terminology that is used throughout the rest of this thesis.

2.1 Smartphone Authentication in Practice

User authentication on smartphones can be performed using one or more of the following factors: the knowledge factors, the ownership factors and the inherence factors [ARKR13]. The knowledge factors rely on something that the user knows, such as a text-based password, a personal identification number (PIN), or a graphical password, such as Android's pattern-lock. The ownership factors employ something that the user has, for example, a smart watch or trusted Bluetooth devices that are in proximity of users' devices. The inherence factors leverage something the user is, such as facial- or fingerprint-recognition.

Authentication schemes based on the knowledge factors are the oldest and the most common form of authentication on smartphones. The proximity-based schemes that rely on a paired smart watch and a Bluetooth device have been introduced more recently (2014 and 2015, respectively) and they have been advocated as promising approaches to reduce the authentication overhead [Dro16, Sch16]. It should be noted that these schemes are not primary authentication mechanisms and in case the device owner is not wearing the smart

watch or is not in the proximity of a trusted Bluetooth device, they fall back to the primary authentication mechanism. Finally, while the majority of recent high-end smartphones are now equipped with a fingerprint scanner, it is absent on the mid- or low-end models.

2.1.1 Acceptance of authentication methods

Several research efforts have investigated the adoption of authentication schemes on smartphones. We provide a brief overview of the findings of these research efforts to gain an insight into the (un)locking behaviour of smartphone users.

Harbach et al. [HVZF+14] conducted an online survey with 260 participants to understand their behaviour towards authentication. They found that only 42.7% of the respondents reported that they configured authentication on their devices and 57% of the respondents who did not configure authentication cited inconvenience as the reason for not using a locking mechanism. Furthermore, 46.8% of the respondents who locked their devices somewhat or fully agreed that unlocking their phones can be annoying.

Egelman et al. [EJP+14] performed 28 qualitative interviews to understand users' behaviours and attitudes towards security on smartphones. They also complemented their qualitative interviews by conducting an online survey of 2,518 smartphone users to corroborate their findings. The survey results showed that 42% of the respondents reported that they did not lock their smartphones. Furthermore, 33.6% of the respondents who did not lock their devices chose to do so because they considered it inconvenient.

Harbach et al. [HDLE16] performed a month-long field study to understand the locking behaviour of Android users. They found that half of the 71 participants did not lock their devices. When these participants were asked if they would use authentication if it was quicker, 62% of them agreed while 16% were neutral. In another study, Harbach et al. [HDLME16] performed an online survey of smartphone unlocking behaviour with 8,286 participants in eight countries. Similar to the findings of the previous studies, they found that convenience was a major reason for a 60% of the respondents chose to not lock their devices. Their survey also found that there are differences in locking behaviour between people in different countries; however, the causes of the observed differences were not investigated.

A common finding across the aforementioned surveys is that an alarmingly large number of users do not configure authentication on their devices due to the usability issues. In the following section, we look at the research efforts that further investigate the usability issues with the widely used authentication mechanisms on smartphones.

2.1.2 Usability issues

Usability and security perceptions of authentication mechanisms based on knowledge have been extensively evaluated. Schaub et al. [SDW12] investigated the usability of text-based passwords with 80 participants and show that while carefully chosen passwords provide good security, the constrained virtual keyboards on smartphones are a bottleneck when users are authenticating through text-based passwords. Von Zezschwitz et al. [VZDLH14] investigated the usability of the creation and entry of alphanumeric passwords on mobile devices for web-based services. To this end, they performed a lab-based study with 24 participants to show that it is indeed cumbersome to enter alphanumeric passwords using the virtual keyboards and therefore, users chose simple and thus weaker passwords for frequently used services.

Ben-Asher et al. [BAKS+11] conducted a survey with 465 participants to evaluate their satisfaction with PINs to identify the need for different security methods on mobile phones. They found that 71% of respondents reported that they would use a PIN as the preferred authentication method on their devices. While they gave diverse options to the participants (signature, gesture, iris, fingerprint and facial recognition), they did not investigate the pattern-lock. A usability evaluation of PIN and pattern-lock schemes was performed by Von Zezschwitz et al. [VZDDL13]. They conducted a three-week field study with 31 participants for a quantitative and qualitative comparison of both schemes. Their findings suggest that the PIN outperforms the pattern-lock in terms of input speed and error rate. However, the participants still favored the pattern-lock and it was rated better in terms of ease-of-use and likeability.

Some research efforts have used prototypes to evaluate the usability of biometric authentication schemes that have not seen large-scale adoption [BAKS+11, TSK+12]. We discuss more relevant efforts that have evaluated the biometrics that are available on modern smartphones (i.e., Apple's Touch ID and Android's Face Unlock).

Bhagavatula et al. [BUI+15] performed a within-subject lab study with ten participants, and an online survey with 198 participants to evaluate the usability of Touch ID and Face Unlock. Their participants reported that both schemes were easy to use for common usage scenarios. During the controlled lab study, they found that Touch ID was robust to hands covered in moisture although there were unlocking issues for dirty hands. Similarly, they found that Face Unlock failed under poor lightening conditions. An overwhelming majority of participants perceived Touch ID more convenient than a PIN whereas a PIN was preferred over Face Unlock by the participants. De Luca et al. [DLHVZH15] conducted an MTurk survey with 383 participants to understand users' perceptions of Touch ID and Face Unlock. Their respondents reported that they considered these biometrics more secure

as compared to PINs. However, 47% of Touch ID's users and 36% of Face Unlock's users complained about the slow speed and the inconvenience of these schemes due to external factors such as oily hands and bad lightening conditions, respectively.

In addition to the usability issues specific to each scheme, a major issue with the current deployment for primary authentication mechanisms is their all-or-nothing access approach. Smartphone users have to pass an authentication challenge in order to access most resources on the device. This approach is unsuitable for smartphones where 40% of frequently accessed apps (such as games, weather, navigation, news and music apps) do not contain confidential data [HRS+12]. Furthermore, unlike desktops or laptops, smartphone sessions are short and frequent and authentication for every short session is inconvenient for users [HRS+12]. In a field study, Harbach et al. [HVZF+14] demonstrated that smartphone users considered unlock screens unnecessary in 24% of the situations and they spent up to 9% of the time they use their smartphones to deal with the unlock screens. Their field study provides the first conclusive evidence on the inconvenience resulting from the inflexibility of the all-or-nothing access approach to authentication on smartphones.

2.1.3 Security issues

Primary authentication mechanisms have been subject to several attacks. For PINs, passwords and pattern-lock, researchers have demonstrated that due to usability and memorability constraints, users tend to choose weak secrets [ATOY13, MKV+13, VZDLH14]. This enables an adversary to launch a brute force attack. The authentication schemes based on knowledge are also susceptible to shoulder surfing attacks [HVZF+14]. Aviv et al. [AGM+10] demonstrated the efficacy of leveraging the oily residue smudges on the smartphone's touchscreen to fully or partially identify the pattern-lock with a 68% and a 92% success rate, respectively. After partial identification, adversaries can significantly reduce the input space for a brute force attack.

Smartphones have multiple sensors, which can be effectively used as a side-channel to infer PINs and passwords. Cai and Chen [CC11] developed TouchLogger for Android, which infers PINs on Android using the motion sensor (accelerometer) side-channel. Their evaluation showed that TouchLogger can infer 71.5% of the typed digits. Owusu et al. [OHD+12] evaluated the accelerometer side-channel to infer text-based passwords on Android. They showed that accelerometer measurements can be used to extract 6-character passwords in as few as 4.5 median trials. Miluzzo et al. [MVBC12] developed TapPrints to infer taps and keystrokes on iOS and Android devices. TapPrints leverages accelerometer and gyroscope sensors to infer tap locations across the display and keystrokes with up to 90% and

80% accuracy, respectively. Kune and Kim [FKK10] investigated the information leakage from the audio feedback of the pressed keys. They showed that by using a Hidden Markov Model, they were able to substantially reduce the search space for a brute force attack.

Operating system flaws may enable an adversary to bypass a primary authentication mechanism altogether. Operating system flaws have been exploited in several versions of Android and iOS [Thr16]. Apple's Touch ID and Android's Face Unlock have also been subject to novel attacks. Researchers have demonstrated that the Face Unlock scheme of Android 4.0 could be easily tricked with a static photograph and the improved Face Unlock scheme of Android 4.1 (with liveness check) could also be defeated with minor photo editing [And16]. For Apple's Touch ID, researchers have leveraged a high resolution camera, a laser printer and a few household supply items to create a fake finger to bypass Touch ID [Zdn16, ZXL+12].

2.2 Mitigating Usability and Security Issues

In the previous section, we identified several usability and security issues with contemporary authentication mechanisms. We now provide a summary of approaches that are available on modern devices to mitigate these issues.

Several approaches have been introduced in Android Smart Lock [Dro16] that reduce the authentication overhead. In addition to unlocking the device when trusted Bluetooth devices are connected, the Android Smart Lock unlocks the device when: a user is in a trusted location, a trusted NFC tag is nearby, a trusted voice is used to say a known phrase ("Ok Google"), or when a user is carrying the device on herself (on-body detection using the accelerometer sensor). These approaches reduce the authentication overhead. However, a primary authentication mechanism is used when the smart unlock criteria are not met (i.e., the device is in an unfamiliar location or is not connected to a trusted Bluetooth device).

Android has introduced support for multi-user accounts since 2014 with Android 5.0. The multi-user accounts can be used to create multiple profiles with different authentication configurations and access to apps. This enables a user to create a guest profile and use it to access the non-sensitive apps without authenticating. This approach has the potential to mitigate the issues arising from the all-or-nothing access model. Third party apps can also be used to password protect sensitive apps. AppLock is one such third party app with between 10-50 million downloads on the Android Play store [App16]. However, since these apps operate in the user space, an adversary can "force stop" or uninstall them to gain

access to the protected app. The wide adoption of these apps indicates that there is a need for a secure app-aware protection mechanism.

Bring your own device (BYOD) demonstrates one approach to deal with the security issues with the primary authentication schemes. Employees use BYODs to access corporate files, business apps, email, calendar, and the corporate intranet, which results in storage of corporate data on their devices [Tho12]. While permitting BYODs is beneficial for organizations, security limitations of primary authentication mechanisms pose a challenge for the protection of corporate data on BYODs from non-owners. Various Mobile Device Management (MDM) solutions have been developed to address BYOD security challenges. MDM solutions leverage the device administration APIs to enforce password policies. MDM solutions enable enterprises to set policies about password type, password timeout, password expiration and password length.

One limitation of MDM solutions is that they defeat the purpose of BYODs. Employees want to use BYODs to have the convenience and comfort of their own devices. However, the device-wide restrictions imposed by MDM solutions (such as longer and complex passwords) reduce the appeal of BYODs and aggravate the usability issues. According to a prediction by Gartner, 20% of enterprise BYOD programs will fail due to overly restrictive policies of MDM solutions [Gar16].

2.3 Alternate Authentication Proposals

We now provide a brief overview of two alternate authentication proposals from the research community: authentication schemes based on graphical passwords and gestures. IA, another alternate authentication proposal, is more relevant to our work and we discuss it in detail in Section 2.4.

2.3.1 Graphical passwords

Many graphical password schemes have been proposed for personal computers as alternatives to text-based passwords. We only discuss graphical passwords on mobile devices and refer interested readers to the comprehensive overview of published research in the area by Biddle et al. [BCVO12]. Graphical password systems have been categorized into three categories: recall-based systems, recognition-based systems and cued-recall systems [BCVO12]. Recall-based systems require the users to recall and reproduce a secret drawing either on a canvas or on a grid. Recognition-based systems require the users to memorize a set of

images during password creation and then recognize their images from a superset including decoy images. Cued-recall systems require the users to remember and identify specific locations within an image.

Dunphy et al. [DHA10] performed a lab-based study with 16 participants to evaluate a recognition-based scheme on phones with physical keyboards. They evaluated two implementations of their scheme: a high entropy implementation where 36 images were displayed across four screens in a 3x3 grid and a low entropy implementation where 24 images appeared across six screens in a 2x2 grid. They noted 70% and 84% success rate for the high entropy and the low entropy version, respectively. In terms of authentication time, the high entropy and the low entropy schemes on average took 19.8 seconds and 21 seconds, respectively. They also demonstrated that their construction was more resistant to shoulder surfing attacks than PINs.

Chiang and Chiasson [CC13] evaluated one notable graphical password scheme from each major category on smartphones and tablets. Their initial experiments with 31 participants indicated that recall-based systems had accuracy issues whereas recognition-based and cued-recall based systems were disliked because participants had to memorize unfamiliar images. Based on their findings, they proposed a touchscreen multi-layered drawing system, which is a recall-based scheme that uses a grid of detached cells to improve accuracy. In a usability evaluation with 90 participants, their proposed scheme received favorable response.

Schaub et al. [SWKW13] evaluated five graphical password schemes covering all three major categories with 60 participants. Their quantitative results indicated that PINs provided comparable or better performance than graphical schemes in terms of authentication time and login success rate. The qualitative analysis indicated that the usability of graphical password systems was comparable to PINs. Similar to the findings of Dunphy et al. [DHA10], their evaluations indicated that the graphical passwords were more resistant to shoulder surfing attacks.

The graphical password proposals outlined in this section report positive user experiences and resistance to shoulder surfing attacks. However, in terms of authentication time and login success rate, these authentication proposals were not exceptional and were outperformed by PINs. Therefore, it cannot be conclusively established that these schemes will mitigate the usability issues with contemporary authentication mechanisms.

2.3.2 Gesture-based authentication

Authentication proposals based on gestures can be broadly classified into predefined and free-form gestures on the touchscreens and device waving gestures. A brief description of each follows.

Sae-Bae et al. [SBAIM12] evaluated a predefined set of five-finger touch gestures and showed that touch gestures are distinct for each user (based on biometric features such as hand size and finger length). They evaluated their scheme to show that it had over 90% accuracy and the overall feedback from participants was positive in terms of usability. Shazad et al. [SLS13] proposed a predefined gesture scheme where users perform one of ten predefined gestures to authenticate themselves. Their goal was to build a system that could resist shoulder surfing attacks. To this end, in addition to the gesture, their scheme employed the touch behaviour of the user including the touch coordinates, the finger velocity, the device acceleration, and the stroke time. Their evaluations showed that by using a single touch gesture, they were able to achieve an EER of 4.8%.

Sherman et al. [SCY+14] explored the security and memorability of free-form gestures for authentication on smartphones. They conducted experiments where users generated free-form gestures, repeated the generated gestures, and then later recalled their gestures. Their experiments showed that signatures and shapes were the best remembered gestures and that free-form gestures were resistant to shoulder surfing attacks. Yang et al. [YCLO16] performed a field study with 91 participants to compare the usability of free-form gestures with text-based passwords. They found free-form gestures to be more usable than text-based passwords. More specifically, password creation and entry times for free-form gestures were less than those for text-based passwords. They also found that free-form gestures had similar memorability as text-based passwords.

Yang et al. [YGD+15] proposed OpenSesame — a scheme that uses smartphone waving gestures for authentication on smartphones. They evaluated four waving gestures and obtained unique patterns of handwaving actions of the users. Experimental results based on 200 distinct handwaving actions showed a 15% FAR with a corresponding 8% FRR. They conducted qualitative interviews in which participants reported that OpenSesame provided good user experience. Hong et al. [HWY+15] proposed a similar scheme with ten waving gestures. In an evaluation with eight participants their scheme provided a 4% FAR with a corresponding 7% FRR.

Our survey of literature on predefined gestures shows that they are subject to a high number of authentication errors. Furthermore, while the original papers on the predefined gestures do not report login times, given that they use multi-touch gestures (five-fingers

for Sae-Bae et al. [SBAIM12]), login times and the need to use both hands could be a potential issue. Similarly, while the password creation and entry time for free-form gestures are lower than for the text-based passwords, they are outperformed by PINs. Finally, the device waving gestures also suffer from accuracy issues. Our brief survey of gesture-based schemes indicates that these schemes do not provide a clear advantage over contemporary authentication mechanisms. We now turn our attention to IA, which circumvents these issues by verifying the identity of its user without requiring explicit input.

2.4 State of the Art in IA

We review the existing literature on IA in this section. To this end, we first provide a brief overview of classification in machine learning and define the relevant terminology. We then survey existing behaviour-based IA proposals, which employ behavioural patterns across the following five categories: (i) device usage patterns; (ii) gait patterns; (iii) text input patterns; (iv) touch input patterns; and (v) multi-modal behavioural patterns. Finally, we review the IA literature that investigates the usability, security, and deployability issues.

2.4.1 Classification in machine learning

IA schemes require learning distinct patterns of device use. The real-time device usage pattern is compared with that of the device owner to detect anomalous behaviour. Machine learning is used to solve the user authentication problem, which given an observed device usage pattern and usage patterns of the device owner determines whether they match. This is a supervised learning problem where the classifier is provided with labeled usage pattern instances. A classifier may require negative training samples comprising of instances of usage pattern from non-owners. The outcome of the classifier is a binary decision which indicates whether the two patterns matched or not.

The pattern that the classifier has to match contains a finite number of predefined non-overlapping features. Each IA scheme defines the features that it will employ to learn the distinct behaviour of the device owner. For example, a keystroke behaviour-based IA scheme may use the following features to define an instance of usage pattern: the key pressed, the key hold time, and the touch pressure on the touchscreen of the device. Instances of usage pattern from the device owner are used as positive training samples for the classifier. Once a classifier is trained using training data, it generates a training model. The training model is then used to determine whether a previously unseen pattern matches

the device owner's pattern or not. There are four possible outcomes of the classifier for the user authentication scenario:

- *True accept (TA):* A true accept is when a real-time usage pattern of a device owner is correctly classified. A true accept grants access to the legitimate user of the device.
- *True reject (TR):* A true reject is when a real-time usage pattern of a non-owner is correctly classified. A true reject correctly rejects an access attempt of an adversary.
- *False accept (FA):* A false accept is when a real-time usage pattern of a non-owner is incorrectly classified. A false accept grants an adversary access to the device.
- *False reject (FR):* A false reject is when a real-time usage pattern of a device owner is incorrectly classified. A false reject incorrectly rejects access attempt by a legitimate user of the device.

We also define two terms that are used to evaluate accuracy of IA schemes. The *equal error rate (EER)* of an IA scheme is defined as the operating point where the rate of true accepts is equal to the rate of true rejects. The *accuracy* of an IA scheme is defined as:

$$Accuracy = \frac{TAs + TRs}{FAs + FRs + TAs + TRs}, \tag{2.1}$$

where TAs, TRs, FAs and FRs are the sum of TA, TR, FA and FR of an experiment, respectively.

2.4.2 Device usage behaviour-based schemes

IA schemes based on the device usage behaviour ("device usage IA") identify the device user based on the way they interact with various services on the device. These schemes rely on the premise that smartphone owners exhibit habitual behaviour. For example, in their seminal paper, Jakobsson et al.[1] [JSGC09] suggested that smartphone owners generally visit specific places during a particular time-of-day on a particular day-of-week. They also suggested that smartphone owners are more likely to call or text a known number than an unknown number and doing so indicates that the phone is in possession of its owner.

[1] Jakobsson et al. [JSGC09] have suggested that their proposed approach can be used for both user-to-device and user-to-remote-site authentication.

Table 2.1: Device usage behaviour-based schemes.

Scheme	Data sources	Classifier	Accuracy
Jakobsson et al. (2009) [JSGC09]	GPS, call logs, text logs, browser history	Majority score	≈10% EER
Li et al. (2010) [LCPD10]	Call logs, Bluetooth, time of use	Neural Networks [HDBDJ96]	13.5% EER
Buthpitiya et al. (2014) [BDG14]	GPS, call logs, text logs, Bluetooth, WiFi, calendar	Majority score	93% TAR @ 20% FAR
Kayacık et al. (2014) [KJB+14]	Running apps, WiFi, location, CPU load, battery, light, noise	Majority score	95% TRR @ 1% FRR
Neal et al. (2015) [NWS15]	Running apps, WiFi, Bluetooth	kNN [CH67]	≈18% EER
Fridman et al. (2015) [FWGK15]	Running apps, WiFi, GPS, browser history, stylometry	SVM [CV95]	5% EER

Features employed: Device usage IA models users' interaction by leveraging data from various sources on the device including device operations, connectivity services, personal data management services, ambient sensors and location services. Some of the features employed by these schemes include:

- *Device operations:* calls or texts sent or received from known and unknown contacts; running apps; time of device use; battery level
- *Connectivity services:* WiFi and Bluetooth scanning results and connection
- *Personal data management services:* browser history; calendar data
- *Ambient sensors:* ambient noise and light sensors
- *Location services:* GPS location and nearby cell towers

Existing schemes: Several device usage IA schemes have been proposed. In Table 2.1, for each device usage IA scheme, we provide its feature list, classifier and accuracy. The "accuracy" column in Table 2.1 shows that these schemes report between 5–18% EER. It also shows that there is a variance among reported accuracy for several schemes that use similar features (for instance, see Buthpitiya et al. [BDG14] and Kayacık et al. [KJB+14]). We suspect that the accuracy variations are caused by the biased evaluation due to the unavailability of real attack data. Device usage IA schemes are evaluated by creating synthetic attack data by splicing a user's (U_{victim}) behaviour stream with another user's

Table 2.2: Gait pattern-based schemes.

Scheme	Approach	Device placement	Accuracy
Frank et al. (2010) [FMP10]	Fix-length	Front pocket	0% EER
Kwapisz et al. (2010) [KWM10]	Fix-length	Front pocket	85% TAR @ 3% FRR
Juefei-Xu et al. (2012) [JXBJ+12]	Cycle-based	Front pocket	3.6% EER
Muaaz & Mayrhofer (2013) [MM13]	Cycle-based	Back pocket	30% EER
Muaaz & Mayrhofer (2014) [MM14]	Fix-length	Front pocket	18% EER

($U_{adversary}$) behaviour stream to mimic that $U_{adversary}$ is in possession of the device of U_{victim}. This results in bias, since the spliced stream of $U_{adversary}$ will contain calls or texts to unknown numbers with respect to U_{victim} and this bias will result in over-reporting of the accuracy for these IA schemes.

A limitation of device usage IA is that it requires a large amount of training data to create a distribution of users' behaviour. For instance, Jakobsson et al. [JSGC09] and Fridman et al. [FWGK15] used two and four weeks of training data to create a model of user behaviour, respectively. Finally, we note that several of these schemes rely on data from multiple sources. However, for our classification purposes, we do not consider these schemes as multi-modal since unlike the multi-modal schemes discussed in Section 2.4.6, these schemes only rely on patterns of habitual device use.

2.4.3 Gait pattern-based schemes

IA schemes based on the gait pattern ("gait IA") leverage the distinct style of walking of individuals to identify them. Existing gait recognition schemes employ techniques from the machine vision domain for gait-based identification of video data [WTNH03] or use on-body sensors [FMP10]. The latter is relevant to our work. Most modern smartphones are equipped with accelerometer and gyroscope sensors and these on-board sensors can be used to learn gait patterns to authenticate the device users.

Features employed: Human walk is cyclic in nature and it consists of one or more gait cycles, where each gait cycle consists of two steps [MM13]. In order to extract features, a gait IA scheme follows one of two possible approaches for feature extraction: cycle-based and fix-length approach. In the cycle-based approach, data from motion sensors is used to identify and extract complete gait cycles. Template-based classification schemes are then used to identify the user. In the fix-length approach, data from motion sensors is divided

into small fixed length segments and stochastic or machine learning based techniques are used for user identification.

Existing schemes: In Table 2.2, for each gait IA scheme, we have listed the approach used, device placement and the reported accuracy. Table 2.2 shows that the scheme proposed by Frank et al. [FMP10] has perfect accuracy. However, their results are based on an evaluation with only eight participants. Juefei-Xu et al. [JXBJ+12] proposed a pace independent gait identification scheme and Muaaz and Mayrhofer [MM14] leveraged gyroscope data to provide orientation independent gait identification. These gait IA schemes have been evaluated in restricted and controlled environments where smartphones were placed in specific pockets of the participants (front pocket [FMP10, JXBJ+12, KWM10, MLV+05, MM14] or back pocket [MM13] of trouser). Therefore, it is difficult to determine whether these schemes will provide comparable accuracies in an uncontrolled environment.

2.4.4 Text input behaviour-based schemes

Text input behaviour based on key hold timings and inter-key intervals has been widely studied for physical keyboards on personal computers (see Banerjee and Woodard [BW12] and references therein). These approaches have been mapped to smartphones with hard (physical) keyboards with reasonable success. Modern smartphones are equipped with soft keypads, which enable users to input data by tapping on a software-based keyboard rendered on the touchscreen. This tap interaction with the touchscreen provides novel features such as touch pressure and the width of the finger. Text input behaviour-based IA ("text IA") schemes have been revamped to leverage features available on touchscreens.

Features employed: The feature set of a text IA scheme depends on the type of keyboard on the smartphone. We list the features used by touch IA separately for each type of keyboard.

- *Hard keyboards:* Key hold time (time elapsed between a key press event and the corresponding key release event); inter-stroke time (time elapsed between a key release event and the next key press event); error rate (number of corrections made)
- *Soft keyboards:* Key hold time; inter-stroke time; error rate; touch pressure; touch area; other spatial features
- *Gesture keyboards:* Feature set is similar to the touch input behaviour-based IA schemes (see Section 2.4.5)

Table 2.3: Text input behaviour-based schemes.

Keyboard	Scheme	Features	Classifier	Window size	Accuracy
Hard	Clarke & Funnel (2007) [CF07]	Key hold time, inter-stroke time	Neural Network	6	20.6% EER
	Hwang et al. (2009) [HCP09]	Key hold time, inter-stroke time, typing cadence	Manhattan distance [DHS12]	4	4% EER
	Zahid et al. (2009) [ZSKF09]	Key hold time, inter-stroke time, error rate	Genetic algos. [RNI95]	8	6% FAR @ 3% FRR
	Maiorana et al. (2011) [MCGCN]	Key hold time, inter-stroke time	Manhattan distance	26	14.7% EER
Soft	Feng et al. (2013) [FZCS13]	Key hold time, inter-stroke time, touch pressure	Naive Bayes [RNI95]	40	4% FAR @ 0.7% FRR
	Giuffrida et al. (2014) [GMCB14]	Key hold time, inter-stroke time, accelerometer, gyroscope	Manhattan distance	9	0.5 % EER
	Draffin et al. (2014) [DZZ14]	Touch location, touch pressure, touch duration, touch area	Neural Network	15	14% FAR @ 2% FRR
	Buschek et al. (2015) [BDLA15]	Touch pressure, touch area, hold time, jump time and distance, drag angle and drag distance	SVM, kNN	8	10% EER
Gesture	Burgbacher & Hinrichs (2014) [BH14]	Ten touch features derived from touch location, touch pressure and touch area	SVM	N/A	5% EER

Existing schemes: In Table 2.3, for each keystroke behaviour-based IA scheme, we have provided the list of features that it uses, the classifier that it employs, the number of keystroke events that it uses for classification and its accuracy. The "window size" column represents the number of keystrokes that are used to make a classification decision. For smartphones with hard keyboards, key hold times and inter-stroke times of users are employed to create their behaviour profile [CF07, MCGCN]. Hwang et al. [HCP09] also took into account the differences in the typing cadence of users (artificial rhythms with cues and natural rhythm without cues) to improve the accuracy of text IA. Zahid et al. [ZSKF09] used the error rate as an additional feature and performed multiple pre-processing steps

using genetic algorithms for accuracy improvements.

Researchers have leveraged features from touchscreens (e.g, the touch area and pressure) to improve the detection accuracy of text IA [DZZ14, FZCS13]. Giuffrida et al. [GMCB14] also employed features derived from accelerometer and gyroscope sensors, which capture the force of the key press, to further increase the accuracy of their scheme for virtual keypads. Buschek et al. [BDLA15] proposed spatial features (in addition to the existing temporal features) for virtual keyboards. The spatial features took into account: (i) *drag:* the drag that is observed between a key press and the corresponding key release event; and (ii) *jump:* the distance between two subsequent touches. They performed evaluations to show that their proposed spatial features are more unique than the existing temporal features.

Gesture keyboards allow users to enter words by sliding a finger across the individual letters of the word (from the first letter to the last). Burgbacher and Hinrichs [BH14] utilized spatio-temporal features from typing gestures to implicitly authenticate the device user. They used the following subset of features from a popular touch input behaviour-based IA scheme (Touchalytics [FBM+13]): the centroid of the subsequence, the median absolute velocity, the median absolute acceleration, the median fingertip area, the duration, the end-to-end distance, the mean resultant length and mean direction Their evaluation showed that they are able to achieve an average EER of 5%.

Soft keyboards are widely used to input text on smartphones. A survey of existing text IA schemes shows that the proposals by Feng et al. [FZCS13] and Giuffrida et al. [GMCB14] provide the highest accuracy for soft keyboards. However, the former requires a large number of key input events to achieve the reported accuracy, which may not be available for some text entry sessions. A limitation of all text IA schemes is that some sensitive device usage sessions may not require any text input at all. For instance, browsing a photo gallery does not require text input and a text IA scheme would be useless for such sessions.

2.4.5 Touchscreen input behaviour-based schemes

Touchscreen input behaviour-based IA ("touch IA") schemes operate on the premise that finger movement patterns of smartphone users are distinct enough to identify them. The capacitive touchscreens on smartphones provide rich biometrics for every touch interaction. More specifically, for every touch point in a swipe, the smartphone operating system provides: (i) the touch location (x and y coordinates), the touch area, the touch pressure and the time stamp. This raw data is used to model the touch input behaviour of the user for their identification.

Table 2.4: Touch input behaviour-based IA schemes.

Scheme	Features	Classifier	Window size	Accuracy
Frank et al. (2013) [FBM+13]	31 features across all touch feature categories	SVM, kNN	7	5% EER
Li et al. (2013) [LZX13]	Nine features related to swipe duration, direction, curvature, length, pressure and area	SVM	7	8% EER
Bo et al. (2013) [BZL+13]	Touch features (pressure, area, duration, position) correlated with accelerometer and gyroscope data	SVM	3	≈1% EER
Zhao et al. (2013) [ZFS13]	Performs trace geometry comparison in image space	Cross-correlation [DHS12]	6	6.07% EER
Lin et al. (2013) [LCL13]	Touch position, touch pressure, touch area	Weighted kNN	10	16.6% EER
Feng et al. (2014) [FYY+14]	Swipe's length, stroke location, stroke curvature, stroke size, stroke speed, running app	DTW [DHS12]	8	8% EER
Xu et al. (2014) [XZL14]	38 features across all touch feature categories	SVM	5	0.98% EER
Lu et al. (2015) [LL15]	Swipe's coordinates, touch area and pressure, direction, angle, duration, distance, and velocity	SVM, kNN	10	5% EER
Roy et al. (2015) [RHM15]	Time stamp, vibration, rotation, touch pressure, touch area, touch location	HMM [DHS12]	9	6% EER

Features employed: The raw touch input data can be used to extract features across the following categories (see Frank et al. [FBM+13] for more details):

- *Location:* start and end x and y coordinates of the swipe
- *Time:* the duration of the swipe and the inter-stroke delay
- *Speed:* the velocity and the acceleration of the finger on the screen
- *Direction:* swipe direction and end-to-end line direction

- *Length:* direct end-to-end distance and the length of the trajectory
- *Curvature:* deviation from end-to-end line and moving curvature of the swipe
- *Contact:* the touch pressure and the touch area

Existing schemes: A summary of touch IA schemes including their features, employed classifier, the number of touch events that it uses for classification, and reported accuracy is provided in Table 2.4. Table 2.4 shows that the majority of touch IA schemes provide significantly better accuracy as compared to device usage, text input and gait IA schemes. Moreover, to improve accuracy, Bo et al. [BZL+13] also employ features extracted from accelerometer and gyroscope sensors to capture the reaction of the device to touch input events.

Feng et al. [FZD+15] investigated the effect of screen size, physical activity and application context of a smartphone on the accuracy of touch IA. Their evaluations show that a larger screen size provides more potential methods of interacting with the device, which increases the accuracy of a scheme. Similarly, they show that the application and physical activity context can be leveraged to increase the accuracy of a scheme. Buschek et al. [BDLA16] evaluate the effect of the placement and size of UI elements (such as buttons) on touch IA. They show that small, compactly shaped targets near screen edges yield the most descriptive touch targeting patterns. They also show that thumb touches are more unique than index finger ones. Their findings suggest that touch IA should analyze GUI layouts and infer hand postures to improve their accuracy.

Several touch IA schemes in Table 2.4 provide less than 5% EER with ten or fewer touch events. Since touch input is the predominant form of interaction on smartphones, these schemes have sufficient data available to classify a user with reasonable accuracy for most of the scenarios. Therefore, touch IA seems to be a promising approach for IA.

2.4.6 Multi-modal schemes

We now look at multi-modal IA schemes that combine behavioural data across several categories (i.e., touch behaviour combined with physical movement patterns) to increase the accuracy [SYJ+11].

Features employed: Multi-modal IA schemes may combine unrelated features from various categories. For instance, Shi et al. [SYJ+11] proposed a scheme that combines the touch input behaviour with the device usage behaviour.

Table 2.5: Multi-modal IA schemes.

Scheme	Data Source	Classifier	Accuracy
Shi et al. (2011) [SYJ+11]	Location, voice, touch input, accelerometer	Naive Bayes	97% TAR @ 3.6% FAR
Zhu et al. (2013) [ZWWZ13]	User interaction with apps is correlated with accelerometer data	k-Means	71% TAR @ 13% FAR
Murmuria et al. (2015) [MSBF15]	Touch input, power consumption and physical movement	Transduction [Vap98]	7% EER
Micallef et al. (2015) [MJB+15]	Ambient noise and light, WiFi, accelerometer and magnetometer	J48 [RNI95]	Not reported
Sitova et al. (2016) [SSY+16]	Keystroke, tap, and hand movement, orientation, and grasp	SVM	15% EER

Existing schemes: Shi et al. [SYJ+11] proposed combining the touch input behaviour with the location and the voice biometric. Murmia et al. [MSBF15] combined the touch input behaviour with power consumption and the physical movement and Sitova et al. [SSY+16] combined the touch input behaviour with the text input behaviour and hand movement, orientation and grip patterns.

Some research efforts have focused on leveraging multi-modal implicit factors to reduce the authentication overhead. Riva et al. [RQSL12] built a prototype to use face recognition, proximity, phone placement, and voice recognition to progressively authenticate. They employed contextual information to carefully choose when to authenticate a user. Their prototype on a Windows phone reduced the number of required authentications by 42% as compared to the number of authentications required during normal interaction. However, their scheme uses a physiological biometric (face recognition) in addition to the implicit factors. Micallef et al. [MJB+15] evaluated ambient noise, ambient light, WiFi, and accelerometer and magnetometer patterns to reduce the number of authentications. Their evaluations showed that based on the implicit factors they are able to reduce the number of authentications by 71%.

While multi-modal schemes provide more data sources that may assist in IA, these schemes have lower reported accuracy than touch IA. Furthermore, similar to device usage IA, the synthetic attack data is used to evaluate the accuracy numbers against some data sources [MSBF15, SYJ+11].

2.4.7 Usability evaluations of IA

The usability issues related to IA have been ignored except for the work by Clarke et al. [CKF09] and Crawford and Renaud [CR14]. Clarke et al. developed a prototype on a personal computer for an IA scheme that employed a combination of face, voice and keystroke biometrics to continuously authenticate users. They evaluated their prototype using 27 participants and found that 92% of the participants considered it more secure in comparison to the traditional forms of authentication. The participants were also asked to rate the convenience on a 5-point Likert scale and although the responses were mixed, a slight skew towards the system being convenient existed. While Clarke et al. are the only authors who provide a usability evaluation of the IA scheme that they proposed, their evaluation was limited because: (i) it was not a strictly behavioural biometric-based scheme since they used a combination of physiological (facial recognition) and behavioural biometrics (voice and keystroke data); and (ii) participants evaluated the prototype on a personal computer instead of a mobile device.

More related to our work is the recent work by Crawford and Renaud [CR14] in which they determined the security perceptions of IA by conducting an in-lab study with 30 participants. They provided a smartphone with a pseudo-IA scheme and asked the participants to perform tasks that required different levels of security. The participants were divided into three groups:

- **G1:** the participants started with a low device confidence level. If a participant wanted to perform a task of medium/high security level, she may increase the device confidence by providing a matching keystroke or voice biometric or by explicitly authenticating
- **G2:** participants were always successfully implicitly authenticated (0% FR rate). This group was used to get the perceptions of distrustful participants.
- **G3:** participants always failed implicit authentication (100% FR rate). This group was used to get the perceptions of frustrated participants.

Crawford and Renaud found that 73% of participants felt IA was more secure than EA and 90% indicated that they would consider adopting it. While Crawford and Renaud provide the only in-depth study on the security perceptions of IA, it has some limitations including: (i) no usability evaluation is performed; (ii) annoyance due to FRs is not quantified; and (iii) security perceptions due to FAs and detection delays are not evaluated

2.4.8 Security evaluations of IA

In the existing literature, behavioural biometrics have been subjected to four types of attacks: (i) attacks based on generative algorithms; (ii) crowd sourcing attacks; (iii) shoulder surfing attacks; and (iv) offline training attacks. A brief description of each and a summary of published attacks on behavioural biometrics follows.

Generative algorithms based attacks employ general population statistics and have been proposed for handwriting recognition, keystroke and touch IA. For handwriting recognition, Ballard et al. [BLM07] showed that a generative model based on concatenative synthesis exceeds the effectiveness of forgeries rendered by skilled humans. Serwadda and Phoha [SP13a] analyzed keystroke data from over 3000 PC users and observed statistical traits. These traits were then fed to their generative algorithm to increase the FAR of a keystroke classifier from 15% to 90%. A generative algorithm based attack on touch IA has been proposed by Serwadda and Phoha [SP13b]. They showed that a robotic device equipped with generic traits across touch data poses a major threat to touch IA schemes as it increases their EER from 5% to 50%. In their model, the attacker required a mechanical robot to mount the attack, which may be impractical or suspicious in a work environment. Furthermore, their evaluation showed that their generic attack failed for up to 40% of the victims because their touch behaviour was different from the inferred generic behaviour.

Crowd sourcing based targeted mimicry attacks have been demonstrated for speaker and gait verification systems. Panjwani and Prakash [PP14] proposed a method to crowdsource search for candidate mimics for speakers in a given target population. They showed that while the probability of finding a successful match is only 3%, MTurk workers are easier and cheaper to locate and recruit than mimicry artists. Gafurov et al. [GSB07] used a database of 760 gait sequences from 100 subjects to show that while trained forgery attacks were unsuccessful for the gait biometric, closest matching subjects from the database could be used to increase the EER up to 80%. Given that an inter-user overlap exists across touch and typing behaviour [SP13a, SP13b], similar crowd sourcing attacks might be possible on these schemes.

Shoulder surfing attacks have been evaluated for IA proposals that employ cognitive abilities [AGSN15] and eye movement patterns [ERLM15]. Shoulder surfing attacks have also been evaluated for explicit authentication schemes that employ user defined touch gestures [SBAIM12, SLS13, SCY+14]. These research endeavors indicated that shoulder surfing attacks were not a threat for their respective proposals. It is not clear whether such attacks will be successful on touch IA.

Finally, offline training attacks that train the attackers to mimic their victims have been demonstrated for gait patterns, handwriting and keystroke biometrics. Kumar et

al. [KPJ15] performed offline training attacks on gait IA. They used raw accelerometer data of victims to train attackers by providing them feedback on how to change their gait behaviour using a treadmill. Their experiments showed that their attack increased the average FAR from 5.8% to 43.6%. However, their attack failed for the cases where physical characteristics of a victim (e.g., height and weight) were remarkably different than the attacker. For handwriting recognition, Ballard et al. [BLM07] showed that some users — who are better forgers than others — can be trained using a naive method to successfully attack the handwriting biometric. Tey et al. [TGG13] used the keystroke data of the victim to train attackers to mimic two keystroke features on PCs. Their evaluations showed that with the full knowledge of the keystroke patterns of the victims, 14 of their best attackers (out of 84 attackers) were able to achieve a 99% bypass success rate. The resistance of text IA and touch IA on smartphones against these attacks needs to be established.

2.4.9 Frameworks for IA deployment

The research focus of the IA proposals that we discussed in the previous sections is to show the effectiveness of the behavioural biometrics that they employ for IA. The original papers of these schemes do not consider the deployment issues of these schemes on smartphones. Clarke et al. [CKF09] proposed the design of a framework to support continuous and transparent authentication using facial, voice and keystroke biometrics. Crawford et al. [CRS13] proposed a similar system that supports multi-modal combination of these biometrics. Clarke et al. [CKF09] and Crawford et al. [CRS13] only provide design guidelines for transparent and continuous authentication systems and stop short of providing an implementation against their design that can be used to provide IA support on smartphones.

Damopoulos et al. [DKP14] propose the design of a framework for deploying defense mechanisms both on the smartphone and the cloud. Their system employs data from a variety of event sensors and after event processing on the device, it detects misuse on the device or in the cloud. The focus of their scheme is in providing a synergistic framework for the cooperation between the smartphone and the cloud for misuse detection. However, since IA can be performed on the smartphone in real-time with reasonable battery consumption [XZL14], a hybrid model may not provide a significant advantage.

2.5 Rationale for Thesis

Our survey of IA revealed that much of the published work has focused on proposing novel behavioural biometrics or improving the accuracy of existing IA schemes. Furthermore,

some critical factors have been neglected in the evaluations (e.g., the delay in misuse detection and the availability of data for classification), which determine the level of protection provided by these schemes. We also observe significant variance in the reported accuracies of similar schemes, which emphasizes the need for an evaluation on independently collected, unbiased datasets. Without a comprehensive evaluation, we cannot ascertain the effectiveness of any scheme for real world usage.

Our survey of the literature on the security evaluation of IA points out some pressing issues. Mimicry attacks on device usage IA are easily possible since an adversary can identify patterns when they gain access to the victim's device. Gait IA has been subjected to offline training and mimicry attacks [GSB07]. However, shoulder surfing and offline training attacks have not been investigated for text and touch IA. The resistance offered by the highly accurate touch IA against these realistic attacks needs to be established to ensure its suitability as a defense mechanism.

In Section 2.4.7, we identified the limitations of existing usability evaluation studies on IA. We observed that false rejects, which might be a usability issue, were not considered during evaluations. Similarly, IA limitations that may have affected security perceptions (false accepts and detection delay) were not communicated to the participants. Finally, the existing evaluations were conducted in a lab, which may have failed to capture user perceptions during the normal device use. These limitations emphasize the need for a comprehensive usability and security perception evaluation of IA.

Finally, the frameworks that have been proposed for IA deployment do not address the deployment challenges identified in Section 1.4. Furthermore, the proposed frameworks only outline design guidelines and fail to provide an implementation that can be used for rapid prototyping and deployment of IA. There is a need for a system that circumvents the deployment challenges to IA on off-the-shelf devices.

Given these unaddressed challenges in IA, we begin our research effort with an extensive evaluation of IA schemes (Chapter 3). We conduct a thorough usability (Chapter 4) and security analysis (Chapter 5) of the IA schemes that perform better in our extensive evaluations. These evaluations provide a clear understanding of the strengths and weaknesses of IA from perspectives that have largely been ignored. We also propose and evaluate a framework to address some of the unique challenges to the IA deployment (Chapter 6).

Chapter 3

A Comprehensive Evaluation of IA

In Section 2.4, we provided an overview of the existing IA schemes and their reported accuracies. We also noted significant variance across the reported accuracies of similar IA proposals. This variances may arise due to the differences in underlying machine learning classifiers or their parameter values, number of participants and the quality of the evaluation dataset. For example, some IA schemes based on touchscreen input behaviour [FLK+12a, ZFS13] provide exceptional accuracies when they are evaluated on datasets collected by those individual efforts. However, Feng et al. [FYY+14] showed that on data collected in an uncontrolled environment, the accuracy of these approaches reduces significantly. Similarly, due to the unavailability of real-world datasets, it is not possible for these individual research efforts to accurately report the training and detection delay in an uncontrolled environment.

A majority of existing IA proposals also fall short of providing performance benchmarks (in terms of CPU and memory overhead) on smartphones. Consequently, it is difficult to understand the impact on user experience due to overhead on power-constrained smartphones by these schemes. In addition to unreported performance numbers (in terms of detection delay and computational cost), many IA schemes use behavioural features, for which it is non-trivial to estimate the frequency or availability of such data. For example, an IA scheme based on the device owner's gait pattern may be useful for authentication, but is not useful if the device owner is stationary most of the time. Therefore, there is a need not only for datasets that allow IA schemes' evaluation in realistic scenarios, but an analysis of real-world behavioural patterns that may influence the appropriateness of deploying one scheme over another. While evaluation metrics have been defined for user authentication on the web [BHVOS12] and physiological biometrics [JRP04], these metrics

are unique to their respective authentication mechanisms and may not effectively capture the usability and security properties of IA.

To the best of our knowledge, a comparative evaluation of different IA schemes has not been performed. Serwadda et al. [SPW13] perform a benchmark evaluation of three touch IA schemes using ten classification algorithms to evaluate which classifiers work best. While their analysis provides interesting insights, it fails to provide a comparative evaluation of IA schemes that employ different behavioural biometrics (i.e., across the five IA categories discussed in Section 2.4). Furthermore, except Feng et al. [FYY+14], none of the other authors have provided a comparison with other schemes. We believe this is due to the effort required to implement another scheme and to collect data to perform empirical evaluations. Therefore, in addition to providing a comparative evaluation of IA schemes, by making the implementation and datasets publicly available, we will enable future researchers to quantify the efficacy of their approach with other related schemes.

In this chapter, we evaluate and compare six IA schemes using four independently collected datasets from multiple geographic locations, comprising over 300 participants. The IA schemes evaluated in this work use diverse behavioural biometrics including touch input patterns, text input patterns, gait patterns, and device usage patterns. We also select two schemes that combine touch patterns with the device's micro-movements as a reaction to touch input and context information. This diversity allows us to better scrutinize different aspects of these individual IA schemes. We evaluate these IA schemes on eight criteria: 1) accuracy, 2) data availability, 3) training delay, 4) detection delay, 5) CPU and memory overhead, 6) uniqueness of behavioural features, 7) vulnerability to mimicry attacks, and 8) deployment issues on mobile platforms.

To achieve the comprehensive evaluation goal, we expand our first research objective (see *Objective 1* in Section 1.5):

- To quantify and compare the accuracies of these IA schemes on independently collected datasets from uncontrolled environments
- To use real-world traces to measure training and detection delays for these IA schemes
- To determine the performance overhead of these IA schemes on mobile devices
- To determine the frequency of data availability for different behavioural features employed by these IA schemes
- To release open source implementations of these IA schemes for performance benchmarking.

3.1 IA Schemes Evaluated

In this section, we expand the brief description of six IA schemes from Chapter 2 that we comprehensively evaluate. To meet our objective of diverse IA schemes, we chose an IA scheme based on device usage patterns [SNJC10], an IA scheme based on gait patterns [FMP10], an IA scheme based on touch patterns [FBM+13], an IA scheme based on text input patterns [FZCS13], an IA scheme based on touch input and the corresponding device's micro-movement patterns [BZL+13], and an IA scheme based on touch and the corresponding usage context patterns [FYY+14]. A description of each scheme follows.

3.1.1 Device usage IA scheme by Shi et al. (Shi-IA) [SNJC10]

Shi et al. [SNJC10][1] proposed an IA scheme based on device usage behaviour that uses good and bad events to determine an authentication score for the user. Good/habitual behaviour is determined by a phone call/text to a known number, a visit to a familiar website, and presence at habitual locations around a certain time-of-day. Similarly, bad behaviour is a phone call/text to an unknown number, a visit to an unfamiliar website, and presence at previously unseen locations. Passage of time since the last good event is also treated as a negative event and results in gradual decay of the authentication score. For empirical evaluations, the authors used data gathered from 50 participants, trained on 2 weeks of their usage data and evaluated on the remaining data. Their results indicated that 95% of the time an adversary can be detected using 16 or fewer usages of the devices with negligible false rejects (1 in 165). We choose Shi-IA as a representative device usage IA scheme because at the time of this writing it provided the best accuracy among device usage schemes.

3.1.2 Gait IA scheme by Frank et al. (Gait-IA) [FMP10]

Frank et al. [FMP10] proposed a time-delay embedding approach to gait recognition. Time-delay embedding is employed to reconstruct the state of an unknown dynamical system from observations of that system taken over time. The authors first extracted features using time-delay embeddings and then performed noise-reduction over those features using principal component analysis (PCA) [Jol05] on a short embedding of training data. PCA produced a projection from the time-delay embedding space to a lower dimension model space. These resulting features were then employed in an approximate nearest neighbor

[1] This work is an expanded version of Jakobsson et al. [JSGC09]

classifier [AMN+98]. Empirical evaluation on walking data from 25 individuals (with the device in the front trouser pocket) resulted in 100% detection accuracy. We chose Gait-IA for empirical evaluations as its implementation and dataset has been made publicly available, which makes it easier to reproduce their results for verification purposes. Furthermore, Gait-IA has reported perfect accuracy.

3.1.3 Touch IA scheme by Frank et al. (Touchalytics) [FBM+13]

Touchalytics operates by extracting 31 features from the raw data of a single swipe. These features capture the user behaviour in terms of the touch location on the screen, the length, direction and duration of a swipe, the velocity and acceleration of a swipe, and the finger pressure and the area covered by a swipe. The extracted features are then used to classify a user using an SVM or kNN classifier. The authors evaluated Touchalytics on a dataset of 41 participants and showed that it is able to provide an EER of $\leq 3\%$ with a window size of 13 swipes. For our empirical evaluations, we chose Touchalytics as a representative touch IA scheme due to its high accuracy and because its implementation is publicly available.

3.1.4 Text IA scheme by Feng et al. (Keystroke-IA) [FZCS13]

Soft keyboards are the predominant form of text entry on the smartphones. Feng et al. [FZCS13] proposed a text IA scheme for soft keyboards. Their proposed scheme uses the key hold time, the inter-stroke time and the touch pressure features. Empirical evaluations on data collected from 40 users using a Bayesian classifier with a window size of 20 keystrokes provided a FAR of 20% with a corresponding FRR of 4%. With a window size of 40 keystrokes, they achieved a FAR of 4% with a corresponding FRR of 0.7%. We choose Keystroke-IA as a representative text IA scheme because at the time of this writing it provided the best accuracy among text IA schemes.

3.1.5 Touch & device's micro-movements (SilentSense) [BZL+13]

The authors of SilentSense observed that a combination of the touch input behaviour and the corresponding reaction of a device (in terms of micro-movements) can be used to create a more robust model of a user's touch input behaviour. SilentSense operates by combining interacting features from touch behaviour (such as pressure, area, duration, and position) for different touch actions (including fling, scroll, and tap) with the reaction of device features (acceleration and rotation, which are captured using accelerometer and

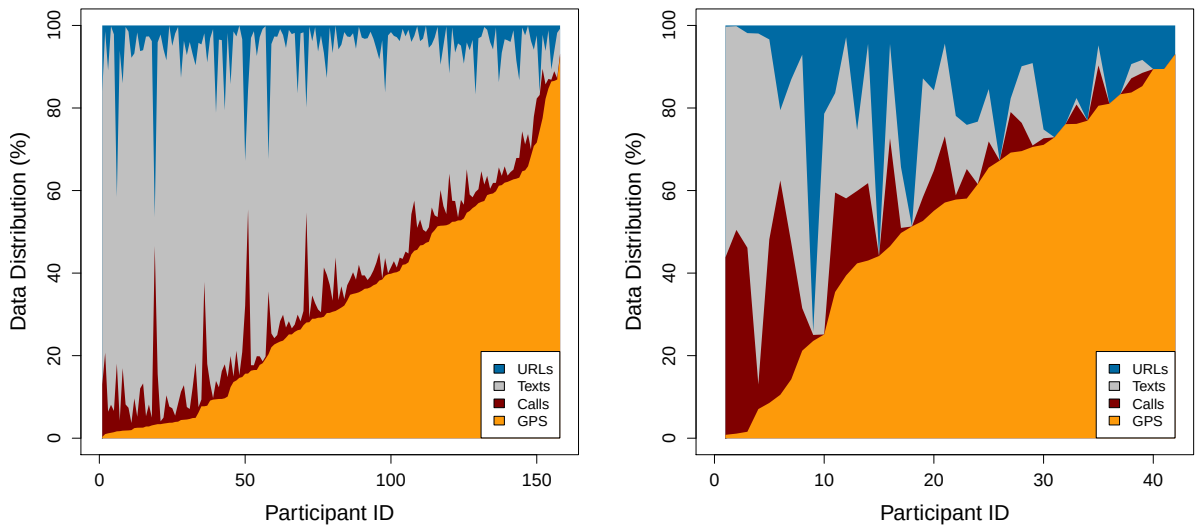
gyroscope, respectively). For the scenarios where the user is walking, the micro-movement patterns are perturbed, since the sensory data generated during walking skews the sensory data generated due to the reaction of the device to touchscreen interactions. To deal with the walking scenario, SilentSense extracts four features including: (1) vertical displacement of each step; (2) current step frequency; (3) mean horizontal acceleration for each step; and (4) standard deviation of vertical acceleration for each step. The authors evaluated SilentSense on a dataset containing data from 10 users and 90 guests. Their evaluations showed that with an SVM classifier, they were able to achieve an EER of $\leq 1\%$ by using a window of three touch strokes. We chose SilentSense for our evaluations because of its unique feature set and its near perfect detection accuracy.

3.1.6 Touch IA & application context (TIPS) [FYY+14]

The author of TIPS showed that on real-world datasets accuracy degradation is observed due to variations in usage behaviour. For example, data generated for the same user for different applications (maps vs. browser) is different enough to cause accuracy degradation. To mitigate this degradation, they proposed a multi-stage filtering hierarchy consisting of four levels: (1) foreground application; (2) direction of swipe; (3) swipe length; and (4) swipe curvature. During the one week training period, TIPS collected 2000 gestures from 23 smartphone users. After generating the templates by performing multi-stage filtering, TIPS achieved an EER of $\leq 10\%$ using a window of eight swipes. Although TIPS uses features similar to Touchalytics [FBM+13], we chose TIPS to evaluate the impact of intelligent use of contextual information.

3.2 Datasets

For the empirical evaluation of the IA schemes, we used real-world and independently collected datasets that capture the natural behaviour of the participants. We used two real-world datasets that broadly capture data from devices while users are using them (e.g., location, wireless connections including network, bluetooth and WiFi, contacts, battery status, call logs, text logs, phone orientation, gyroscope and accelerometer readings, and running apps). These datasets were used to evaluate Shi-IA and Gait-IA. However, these datasets do not include touch or text input data. We therefore used a third real-world dataset that captures swipe data and used it for evaluating Touchalytics, SilentSense and TIPS. Ideally we would gather day-to-day freehand text input from participants. For privacy reasons, however, we cannot use a user's real-world communications for text input



(a) Netsense dataset

(b) WatApp dataset

Figure 3.1: Distribution of URL, text, call and GPS records collected from different participants in the Netsense and WatApp datasets, sorted by fraction of GPS data. Percentages are derived from the number of discrete events collected from each participant.

data. We therefore had users type predefined email and SMS strings to evaluate Keystroke-IA. In this section, we provide data collection goals, experimental design, and the process used for collecting the four evaluation datasets. Our methodology of data collection and reuse of other datasets was reviewed and approved by the ORE of our university.

3.2.1 Netsense dataset [SLM+13]

University of Notre Dame researchers created the Netsense dataset by providing 200 first-year Notre Dame students with Android smartphones. These devices were modified to log many events including contacts, texts, voice calls, Wi-Fi scanning results and current access point, Bluetooth scanning results and connections, browser history, running apps, battery status, location, email, and port traffic. While the purpose of their study was to understand social ties, many of these features overlap with the features used by device usage IA schemes [JSGC09].

Data Statistics We contacted Striegel et al. [SLM+13] and requested a chunk of their dataset. They provided us with data that was logged between 2012-11-01 09:34:35 and 2012-11-30 12:49:50. This chunk of the dataset contained data belonging to 158 participants. For our study, we extracted the location, call history, text history and browser history data. For 158 users, we extracted 125846, 15003, 244627 and 4817 location events, call events, text events and webpage access events, respectively. The data distribution across participants is plotted in Figure 3.1(a). We note that this dataset is not labeled; i.e., there is no way to label the data for instances when the device was voluntarily given to someone for use by the owner or when it was deliberately misused by a non-owner.

3.2.2 WatApp dataset

While the Netsense dataset is useful for our study, we wanted to collect labeled data. To this end, we instrument WatApp[2] (an Android App widely used by University of Waterloo students to get information about current weather, searchable maps, class schedules, and events) to log events on participants' devices. In addition to logging the same data as Netsense, WatApp logs gyroscope readings and accelerometer readings. The sensitive fields are one-way hashed to preserve the privacy of participants. Furthermore, to establish the ground truth, we ask participants to label the intervals for which they are *absolutely certain* that the device was in their possession.

To advertise for participants, we used our university-wide mailing list to advertise for people who would be interested in a study on "Mobile Misuse Detection". Participants were expected to install WatApp on their smartphones for ten weeks. Participants had the option to opt-out any time they wanted by disabling the data collection mode. Furthermore, if they wanted WatApp to not log data, they were provided with the option to pause data collection for an indefinite amount of time. We paid the participants $5 for each week of participation (up to $50 in total for ten weeks of participation).

Data Statistics Our app was downloaded and installed by 74 participants and 42 of those participants completed the study. In total, we logged 1371908 events over ten weeks. For 42 users, we extracted 121525, 15962, 28958 and 36178 location events, call events, text events and webpage access events, respectively. Data distribution across participants is plotted in Figure 3.1(b). We suspect that the differences in data distribution between the Netsense and WatApp datasets are due to the fact that participants of the Netsense project were provided with free unlimited phone plans.

[2] `http://play.google.com/store/apps/details?id=watapp.main`

3.2.3 Touch input dataset

Frank et al. [FBM+13] and Serwadda et al. [SPW13] have made their touch datasets publicly available and we wanted to use those to generate reproducible benchmarks. However, their datasets have a few limitations: (i) both datasets have been generated using only two apps; (ii) for collection of both datasets, predefined tasks were given to the participants to perform; and (iii) participants used the data collection apps in a lab. We argue that due to these reasons, the data gathered by Frank et al. and Serwadda et al. falls short of capturing in-the-wild behaviour of participants.

Our goal was to collect a dataset that captured the natural behaviour of the participants when they used the touchscreens of their smartphones. We did not want the participants to perform predefined tasks. We also wanted to study touchscreen input behaviour across a diverse set of apps. Therefore, to capture data that satisfied our data collection goals, we instrumented four Android apps: a browser app[3], a maps/ navigation app[4], a launcher app[5] and a comic viewer app[6]. The apps that we chose belong to diverse categories and helped us in understanding user behaviour across different apps. To advertise for participants, we used our university-wide mailing list for people who would be interested in a study on smartphones apps. Participants were expected to install these apps on their smartphones for ten weeks. We did not ask the participants to explicitly perform any tasks and participants were to use these apps as per their needs. This allowed us to capture participants' in-the-wild behaviour. We paid the participants $5 for each week of participation (up to $50 in total for ten weeks of participation).

For data collection, every time a participant interacted with the touchscreen on one of the provided apps, we recorded: 1) time stamp in milliseconds; 2) x and y co-ordinates of the touch point, 3) finger pressure on the screen; 4) area covered by the finger on the screen; 5) values from the accelerometer sensor; 6) finger orientation; 7) screen's orientation; 8) smartphone's orientation sensor's value (roll, pitch and azimuth); and 9) accelerometer sensor values. These values were temporarily stored on the participant's device and then batch transmitted to a server. Before every data transmission, we established the ground truth (only the participant used the apps) by asking the participants to label the intervals for which they were *absolutely certain* that the device was in their possession.

Data Statistics Our apps were downloaded and used by 61 participants. In total, we logged about 2.49 million touch points comprising over 53,000 swipes in ten weeks.

[3] `http://code.google.com/p/zirco-browser/`
[4] `http://code.google.com/p/osmand/`
[5] `http://code.google.com/p/android-launcher-plus/`
[6] `http://code.google.com/p/andcomics/`

Table 3.1: Statistics of touch points dataset.

App.	Num. of touchpoints	Num. of Swipes	Sessions	Mean (Median) swipes per session
Launcher	642442	19740	4417	4.46 (2)
Browser	1164011	20139	826	24.3 (16)
Maps	236878	4664	365	12.7 (8)
Comics	445538	8928	272	32.8 (16)
Total	**2488869**	**53471**	**5880**	**9.09**

The details of swipes, their distribution across apps and distribution across user sessions is provided in Table 3.1.

3.2.4 Text input dataset

We wanted to collect text input behaviour of participants during their normal usage sessions; however, this was difficult in a privacy preserving manner. Therefore we presented users with text strings that are used in everyday communication. We chose text strings from existing publicly available SMS [CK13] and email corpora [KY04]. We developed an Android app that presented a participant with each string of data that they were expected to input using the virtual keypad on their smartphone. Once users entered all the strings, the logged keystroke data was transmitted to our server. To advertise for participants, we used our university-wide mailing list for people who would be interested in a study on "The need for auto-complete and auto-correct on smartphones". To avoid any bias, we did not tell participants about the real purpose of this study before the conclusion of the study. Finally, we did not restrict the participants to complete the study in a limited number of sessions nor asked them to complete it in a lab. We paid $10 to each participant for completing this study.

Data Statistics We presented participants with 13 strings. These strings contained 43 words and 268 characters in total. We required every participant to input each string four times to collect 1072 keystrokes from each participant. Our app was installed and used by 40 participants. The mean time taken to complete the study was eight minutes.

3.3 Evaluation Setup

While most of the evaluation metrics that we use are independent of the underlying implementation language, we wanted to measure processing complexity on real Android devices.

By using Java as our implementation platform, we were able to measure these statistics easily. Therefore, despite the availability of Matlab source code for Touchalytics [FBM+13], we re-implemented it in Java. We re-used the publicly available C++ implementation of Gait-IA [FMP10] via the Android Native Development Kit. We note that evaluating the Gait-IA scheme as a native app will result in relatively better results for processing overhead metrics. For the evaluation of other metrics, we used automated scripts on a desktop machine

For our evaluations, we used the recommended parameter values of IA schemes from their original papers. If a paper does not specify a recommended value (e.g., the decay parameter for Shi-IA), we first evaluated the proposed scheme while keeping the classifier threshold to a constant value to determine the best operating point of the tuning parameter for which a recommended value is not provided. To evaluate Shi-IA, we used the Netsense and WatApp datasets. For Gait-IA, we used sensor readings from the WatApp dataset. Keystroke-IA uses the Keystroke dataset for training and classification purposes. Finally, the Touchalytics, SilentSense and TIPS schemes all use the Touchscreen Input dataset.

We constructed non-overlapping training and test sets for each of the participants, using negative instances from other users. In practice, it is recommended that IA classifiers come prepackaged with such data to be used as negative instances, allowing robust classifiers to be trained on-device. In our work, the negative training sets of a user for the text and touch input datasets were constructed by employing usage data from 20 other users. For the Netsense and WatApp datasets, we used one day of data from 14 other users to construct two weeks of negative training data. Frank et al. [FMP10] recommend using a continuous block for training their Gait-IA classifier; consequently, we employed the largest single block of continuous data for training. For Touchalytics, Keystroke-IA, and SilentSense, we used half of the data for training, and the remaining data for testing. In the case of TIPS, we used a 30/70 ratio for training and testing, respectively. This variation in partition ratios is due to us following the convention established in the respective original papers, and due to the heterogeneity of the different types of data used by the different schemes in this work.

3.4 Evaluation Results

3.4.1 Accuracy evaluation

The accuracy of an IA scheme is its most critical evaluation metric. Ideally, the scheme should have no false rejects (for a seamless user experience of the device owner) and 100%

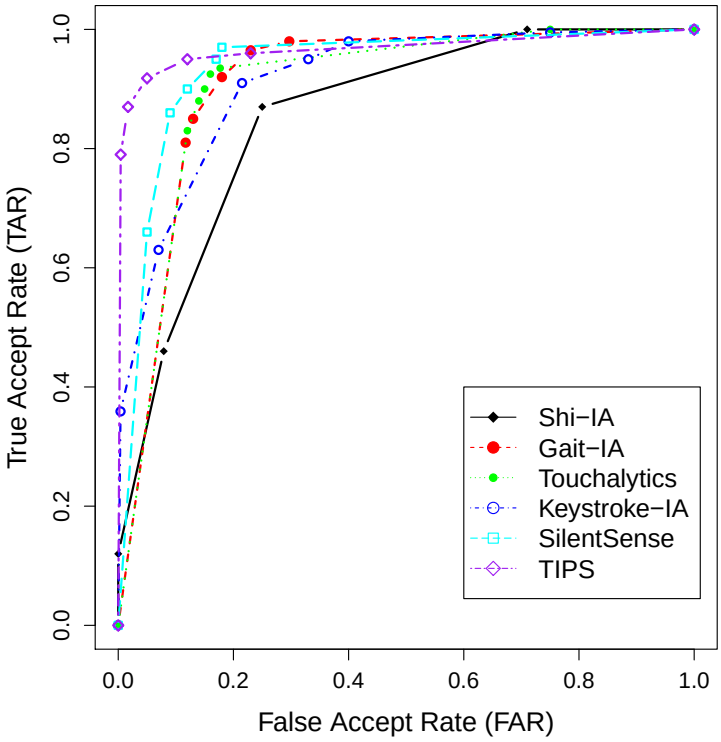


Figure 3.2: Accuracy of the IA schemes.

true reject rate (to detect and prevent misuse by an adversary). To understand the accuracy of these classifiers, we plot the ROC curve using the True Accept Rate (TAR) and the False Accept Rate (FAR). To understand the trade-off between TAR and FAR, we threshold the authentication score. Thresholding provides a means to set different values for various parameters of the underlying machine learning algorithm to obtain different values of TARs and corresponding FARs. Thresholding of Shi-IA is performed over the computed authentication score. Gait-IA and Touchalytics, which use ANN [AMN+98] and k-NN for classification, are thresholded over the distance function score and over k, respectively. Keystroke-IA implementation uses a Bayesian Network classifier [FGG97] and is thresholded over the p score. Our implementation of SilentSense uses LIBSVM [CL11] with a gaussian radial-basis function (rbf) as kernel. For thresholding, we tune the γ and C parameters to the rbf. TIPS uses Dynamic Time Warping [BC94] to compute a similarity score and we threshold the similarity score. The results of the accuracy evaluation, averaged across all users, for the six classifiers are provided in Figure 3.2.

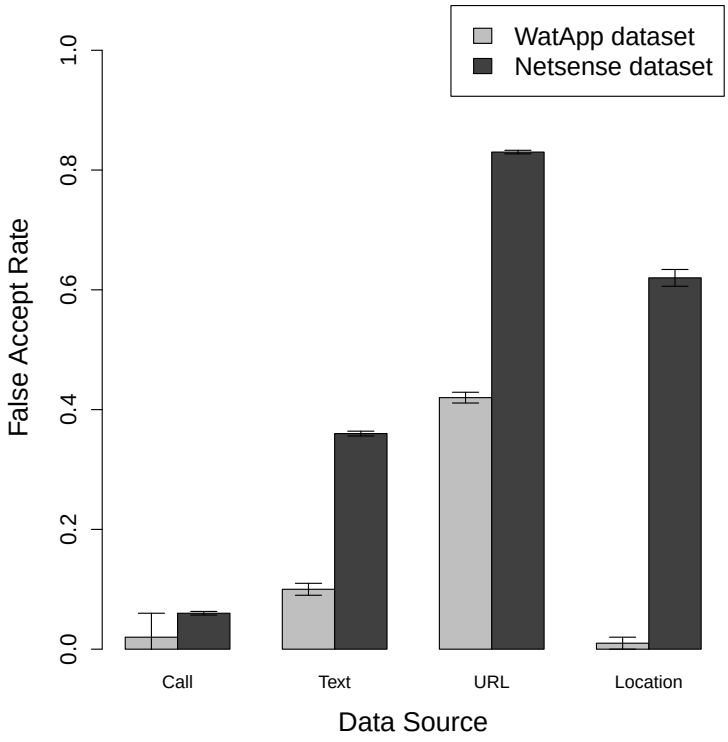


Figure 3.3: FAR for individual data sources for WatApp and Netsense datasets.

As shown, the TIPS scheme outperforms the others in almost all cases. In particular, it is able to achieve a TAR of 79% with a FAR of only 0.43%. TIPS and SilentSense together Pareto dominate all other schemes when the FAR is under 25%. Shi-IA generally underperforms the other schemes, although it has the distinction of being the only IA scheme to achieve a TAR of 100% with a FAR of less than 100% (specifically, 71%). Empirically, this may be due to the fact that Shi-IA uses location information as a discriminator, while the datasets are mostly taken from students living in tightly grouped geographic areas. Consequently, these results may be different for other types of users (e.g., people who travel often). We explore this further using the two device usage datasets.

The Netsense dataset only contains data from Notre Dame students who live in four residence halls on the campus. On the other hand, during the data collection campaign for the WatApp dataset, we placed no restrictions on the participants. Therefore, the WatApp dataset has participants that belong to relatively diverse geographical areas. Furthermore, since the participants of the Netsense dataset are co-located, they have more overlap in

terms of people they communicate and interact with. To understand how the performance of Shi-IA differs among the two datasets, Figure 3.3 plots the FAR against each data source within two datasets. It shows significantly more false accepts for the Netsense dataset than the WhatApp dataset for text, browser history (URL) and location data sources. More specifically, we observe twice as many and three times more false accepts due to texts and URLs for the Netsense dataset, respectively. Similarly, for the location data source, we observe orders of magnitude more false accepts for the Netsense dataset. These results signify the importance of an unbiased dataset for the evaluation of IA schemes.

3.4.2 Data availability

If an IA scheme employs data from a behavioural source that does not have enough data available to make a classification decision for a significant number of usage sessions, the IA scheme would be ineffective despite its high detection accuracy. For example, while Gait-IA outperforms Keystroke-IA in terms of accuracy (see Figure 3.2), Gait-IA will not be useful if the device user is stationary and is not generating enough data for classification purposes. We leverage our real-world traces to determine the availability of data for these IA schemes. To compute the data availability we assume that IA is to be performed only once during a session (and not performed repeatedly after a predefined interval of time). We note that an IA scheme may save past authentication scores and re-use them in case data is unavailable. For example, Gait-IA may compute authentication score prior to the device usage when accelerometer data is available and then reuse this score to authenticate future sessions. However, for a fair comparison, to compute the data availability we only consider data that has been generated during a device usage session.

From the Netsense and WatApp datasets, we calculate the total number of usage sessions (delimited by screen-on events) and the sessions in which enough behavioural features are available to perform a classification decision for Shi-IA, Gait-IA and Keystroke-IA. For text input availability, exact text data is not available and so we assume enough data is available whenever the keyboard is displayed on the screen during the session; note that this will lead to some overreporting of text input data availability against sessions with insufficient number of keystrokes. Since the Netsense and Watapp datasets do not log touch input, for Touchalytics, SilentSense and TIPS, we report data availability against the four apps used in the touch input dataset. This will also result in some overreporting of data availability; however, since touch interaction is the primary input mechanism on modern devices, we expect our results to hold for other apps.

As seen in Figure 3.4, data derived from touch interaction is almost always available, so IA schemes making use of it are thus most likely to be usable. SilentSense additionally

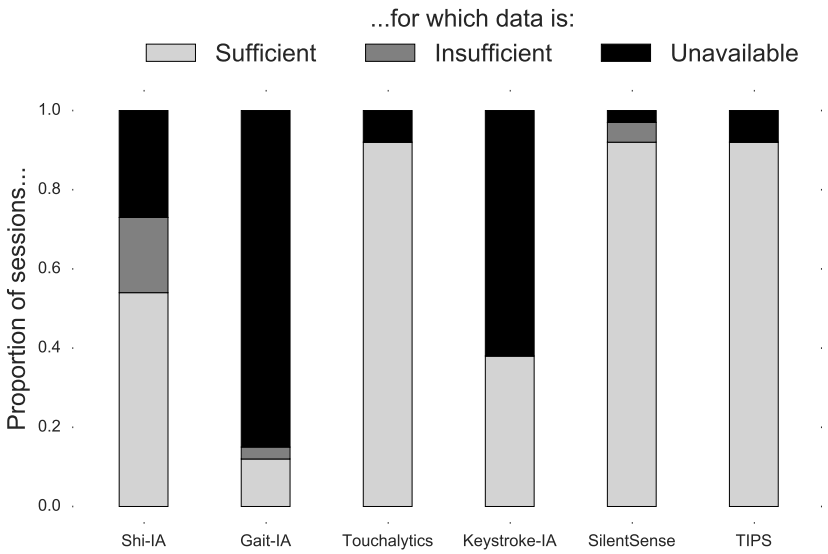


Figure 3.4: Data availability on real-world datasets.

makes use of accelerometer data; when the device is resting on a stable surface this data will not be as meaningful as when the device is being held, but it is still available for training and classification. Availability of data for Shi-IA is highly dependent upon the users' context and is discussed further in Section 3.5. Gait information was generally the most difficult to find, with enough information available in only 13.1% of sessions.

3.4.3 Training delay

An IA scheme that could employ data from a few sessions to robustly train itself would be highly desirable. While the IA scheme may explicitly request a user to provide training data (for example, a text IA scheme asks a user to input a set of strings), most of the existing schemes rely on collecting data during normal usage for training purposes. We utilize the datasets as described in Section 3.3 to determine the training delay for each of the six schemes evaluated in this work. To measure training delay, we set all the tuning parameters including the classification threshold to a constant value and then train the classifier by incrementally increasing the size of training data to the classifier. For IA schemes that employ classifiers that require negative training instances (e.g., Touchalytics

Table 3.2: Minimum training delay to achieve accuracy rates of $\geq 70\%$, $\geq 80\%$, $\geq 90\%$. 95% confidence intervals are provided in parentheses. Note that Shi-IA uses the contents of logs as a whole and as such has no concept of an "event".

	Accuracy $\geq$ 70%		**Accuracy $\geq$ 80%**		**Accuracy $\geq$ 90%**	
	Events	**Time (sec)**	**Events**	**Time (sec)**	**Events**	**Time (sec)**
Shi-IA	N/A	1.7 weeks	N/A	3.2 weeks	N/A	N/A
Gait-IA	1434	159 (32)	1832	205 (47)	2338	287 (59)
Touchalytics	67	106 (10)	165	280 (30)	275	464 (49)
Keystroke-IA	1352	594 (55)	2028	839 (108)	3380	1101 (360)
SilentSense	86	139 (14)	204	346 (36)	272	460 (49)
TIPS	738	1391 (224)	1295	2443 (378)	1611	3034 (445)

and SilentSense), we use equal amounts of out-of-class training instances from 20 out-of-class sources. For every training session, we measure the accuracy of the classifier by running it on the test dataset. Using this process, we find the minimum number of events and the amount of time required to collect these events to obtain an accuracy of $\geq 70\%$, $\geq 80\%$, and $\geq 90\%$. These results are provided in Table 3.2.

Training delays are closely correlated with data availability rates. When gait information is available—which is frequently not the case, as discussed previously—Gait-IA takes the least amount of time to accumulate enough information to train a model with high accuracy. Touchalytics and SilentSense take only a few minutes extra, as touch input is a frequent event. Keystroke-IA data takes longer as high accuracy requires the user to type strings that cover a fair amount of high frequency bigrams (as the training data is derived from inter-stroke timings). TIPS, despite having the best TAR and FAR overall, requires approximately one hour of data collection to achieve $\geq 90\%$ accuracy. Shi-IA requires several weeks' worth of data, as it relies on user behaviour patterns repeating over large periods of time.

3.4.4 Detection delay

While the data availability metric determines whether enough data is available across sessions, we evaluate detection delay for these IA schemes to measure the sensitivity of these schemes to misuse attempts. Ideally, we would like the detection delay to be as low as possible to prevent the adversary from accessing confidential data on the device. We measure detection delay in terms of time elapsed from the start of misuse to the time when the IA scheme detects the misuse. For detection delay evaluation, we play back negative instances and look for those that are correctly classified as true rejects by the IA scheme

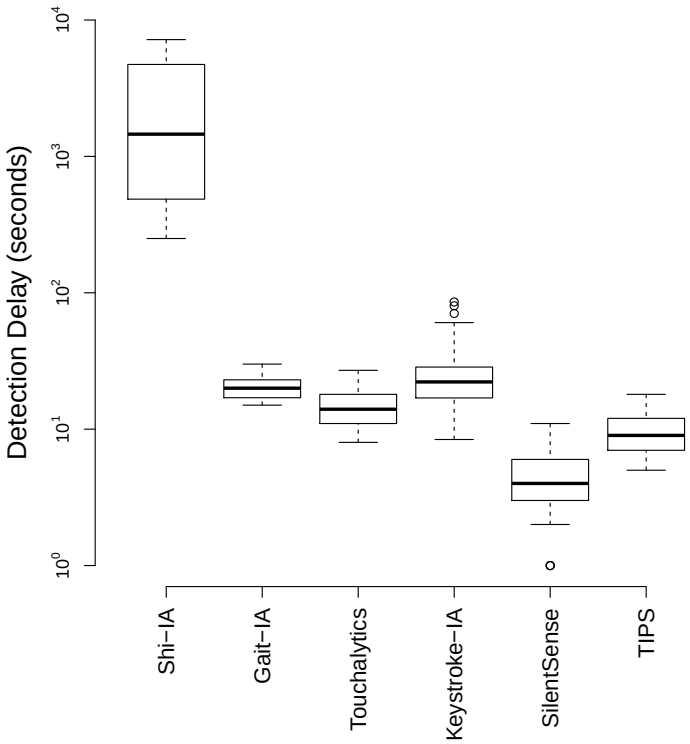


Figure 3.5: Detection delay for true rejects (note log scale).

(i.e., we ignore data that results in false accepts).

The detection delay results are shown in Figure 3.5. SilentSense generally detects non-owners the fastest, in the range of 2–11 seconds. Other schemes generally detect non-owners in less than 30 seconds, with the exception of Shi-IA. Shi-IA takes more than 15 minutes on average before enough data is available for it to reject a non-owner from the device. This result is significantly longer than the average session length, and a malicious user would likely be able to export data from the device before even realizing that an IA scheme is in use.

3.4.5 Processing complexity

Since the target for these IA schemes is mobile platforms, it is critical for the IA schemes to have low processing complexity. For complexity evaluations, we measure the performance overhead in terms of elapsed CPU time and heap size of the IA scheme for feature

collection, training and classification operations. We divide the performance overhead into these operations to distinguish the one-time (training) and run-time (feature collection and classification) costs. An efficient IA scheme would have a reasonable one-time cost and minimal run-time cost.

For execution time calculation, we chose an HTC Nexus 1 and an LG Nexus 4. The Nexus 1 has Android OS v2.1 on a 1 GHz processor with 512 MB of RAM. The Nexus 4 has Android OS v4.2 on a Quad-core 1.5 GHz processor with 2 GB of RAM. We loaded the training and test sets for this experiment from a file on the app's local storage. The size of the training set used for this experiment corresponds to the number of samples required to achieve 70% accuracy (see Table 3.2). We repeated each experiment 15 times and report averages. For the experiment, both devices were configured in the *"airplane mode"* and had only the factory provided services running in the background. Execution time results for both devices are provided in Table 3.3.

The Nexus 4 generally performs operations faster than the Nexus 1, but with marginally higher memory overhead. In our experience, these small differences are generally due to changes in the Android API. SilentSense initialization and training take several seconds due to the SVM classifier used; it also loads negative instances from disk at initialization. Shi-IA takes 1–2 seconds to extract features from data as it filters call, SMS, and browser logs. All schemes are able to perform classification in tens of milliseconds in the worst case.

3.4.6 Uniqueness of behavioural features

Jain et al. [JRP04] list distinctiveness as one of the key properties of a biometric-based authentication system, which requires any two persons to be sufficiently different in terms of the characteristics measured. While the presence of false accepts in Figure 3.2 indicates that none of the behavioural features employed in the IA schemes evaluated in this work is distinct, nevertheless they should provide sufficient discriminatory information among a *sufficiently* large set of users to provide an acceptable FAR. To gain insight into this, we simulate N non-owners attempting to access a protected device, and measure the rate at which someone is able to successfully bypass IA. By varying the number N, we gain some sense of the device owner's uniqueness in a crowd of that size. For each value of N, this simulation is run using 4-fold cross-validation for each user and the results are averaged.

Figure 3.6 shows the results from this simulation. All of the IA schemes tested appear to exhibit similar growth patterns in IA bypass rate as the number of users increases. While TIPS and Shi-IA exhibit the most uniqueness overall, SilentSense is also quite resilient when faced with 10 or fewer adversaries. Keystroke-IA does not appear to be distinctive

Table 3.3: Performance evaluation of the IA schemes. 95% confidence intervals are provided in parentheses.

		CPU (ms)				Heap(kB)
		Init.	**Feature Extraction**	**Training**	**Classification**	**Runtime**
Nexus 1	**Shi-IA**	677 (26)	1758 (31)	13053 (87)	58 (4)	790 (6)
	Gait-IA	5 (≃0)	7 (≃0)	764 (42)	93 (7)	9532 (81)
	Touchalytics	5 (≃0)	<1 (≃0)	65 (2)	2 (≃0)	59 (1)
	Keystroke-IA	21 (2)	<1 (≃0)	<1 (≃0)	<1 (≃0)	3 (≃0)
	SilentSense	1162 (81)	<1 (≃0)	10384 (91)	<1 (≃0)	18 (1)
	TIPS	5 (≃0)	<1 (≃0)	35 (2)	1 (≃0)	92 (2)
Nexus 4	**Shi-IA**	575 (24)	1406 (22)	10964 (74)	51 (3)	817 (5)
	Gait-IA	4 (≃0)	5 (≃0)	522 (31)	75 (6.8)	9775 (94)
	Touchalytics	3 (≃0)	<1 (≃0)	15 (≃0)	1 (≃0)	67 (6)
	Keystroke-IA	12 (≃0)	<1 (≃0)	<1 (≃0)	<1 (≃0)	3 (≃0)
	SilentSense	972 (67)	<1 (≃0)	5937 (329)	<1 (≃0)	21 (≃0)
	TIPS	3 (≃0)	<1 (≃0)	8 (0.86)	<1 (≃0)	96 (2)

even in scenarios with few non-owners present, suggesting that it would be wise to pair these features with other, non-keystroke-derived attributes when creating IA schemes.

3.4.7 Vulnerability to mimicry attacks

While a detailed analysis of vulnerability to mimicry attacks across six IA schemes is beyond the scope of this thesis, in this section we consider the informed adversary threat scenario. We evaluate mimicry attacks on Touchalytics and SilentSense in Chapter 5 and provide a discussion on the possibility of mimicry attacks on other schemes. In Section 1.3, we outlined our attack model and the two types of adversaries — informed and uninformed adversaries. In this section, we consider how effortlessly informed adversaries can defeat an IA scheme.

We argue that in accordance with Kerckhoffs's principle, the IA mechanism (including its features and computation of anomaly score) is public knowledge but feature values for individual users are secret. Consequently, if an adversary can estimate the feature values for an IA scheme easily and mimic those feature values, he can steal data from the device. From the approaches that we evaluate, Shi-IA is the most vulnerable to mimicry attacks. Even an uninformed adversary can scan the device for call/text logs and browser history

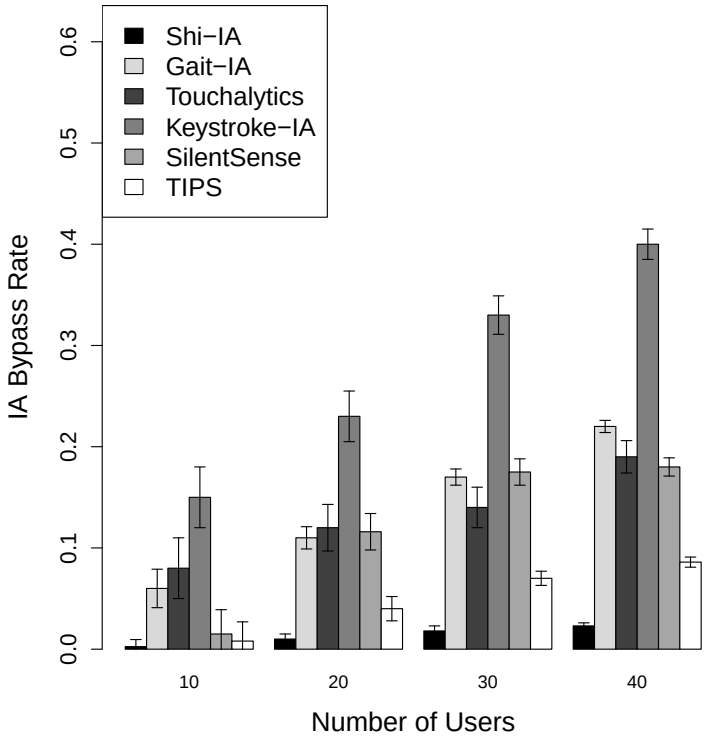


Figure 3.6: Relationship between IA bypass rate and number of users.

and then mimic it to ensure that the device does not lock him out. An informed adversary would attempt to stay in the same vicinity as the device owner to get an even better authentication score. During the literature survey (in Section 2.4.8), we noted that while shoulder surfing attacks on gait IA schemes were not possible [GSB07], offline training attacks were a serious threat for gait IA [KPJ15]. Keystroke-IA relies on features that are difficult to estimate by looking over the victim's shoulder therefore, shoulder surfing attacks on Keystroke-IA may be difficult to mount.

A more serious attack surface for these IA schemes exists in that many of the features employed by these schemes can be collected by any app without requiring any special Android permissions (except Shi-IA, which requires the permissions mentioned in Section 3.4.8). Consequently, an adversary might persuade the victim to install a Trojan app on his device in order to log his behaviour. The adversary can then train himself to mimic the victim. Tey et al. [TGG13] mounted this attack on a keystroke-based authentication scheme for traditional keyboards.

3.4.8 Ease of deployment on mobile platform

Finally, we look at the deployment related issues for these IA schemes on the popular iOS and Android platforms. We understand that sufficient changes might be introduced by the OS providers in future versions to mitigate the deployment limitations of these IA schemes; nevertheless, we provide an overview of the deployment issues on contemporary mobile platforms.

The features used by Gait-IA can be collected without requiring any permissions. Features employed by Shi-IA can be collected using non-root permissions. More specifically, on Android five permissions including `ACCESS_FINE_LOCATION`, `READ_SMS`, `READ_CALL_LOG`, `READ_HISTORY_BOOKMARKS` and `READ_CONTACTS` can be used to implement Shi-IA. Feature extraction for touch and text IA schemes is more complicated. Due to security and privacy concerns, iOS and Android only allows a foreground app to receive input events (touch and keystroke events). Therefore, IA schemes that employ these features (including Touchalytics, Keystroke-IA, SilentSense, and TIPS) can only be deployed either on rooted devices or deployed per app instance [KH14]. We discuss these deployment challenges and possible mitigation strategies in Chapter 6.

3.5 Discussion and Open Challenges

This section discusses lessons from our findings of comparative evaluation of IA schemes in Section 3.4, and our experience in implementing the schemes on Android devices.

Practical IA is possible with low overhead and in near-real-time. Our results on Nexus 1 and Nexus 4 devices given in Table 3.3 show there are IA schemes that can run feature extraction and classification in only milliseconds. Even the worst case training scenarios take only ten seconds, which is performed one-time only and can be done in a background thread. In case of misuse by a non-owner, the majority of these IA schemes are able to detect misuse in under 30 seconds. In terms of accuracy, touch IA approaches provide $\geq$ 90% true accepts with $\leq$ 10% false accepts and might potentially be a good candidate for secondary authentication. In Chapter 4, we conduct a usability study to determine whether these accuracy rates are acceptable for users.

Features should be chosen in a complementary and context-aware manner. Sources for behavioural features must be chosen carefully, and take the intended deployment context into account. Touch input data is almost always available (see Figure 3.4) but should be augmented with a secondary source (such as text input data) for better coverage. Taking into account user context information — e.g., whether the user is walking

or stationary, which app the user is interacting with — is important for classifying data from onboard sensors (TIPS), but does not necessarily make a good discriminator by itself (Shi-IA). No individual source of behavioural data provides a silver bullet for IA.

Devices may not need to be rooted to make use of IA. Android does not allow background applications to gather input events (touch and text input events) due to security concerns. Therefore, IA schemes that rely on input events (e.g., touch and text IA schemes) require root privileges on the device in order to collect data. On the other hand, Shi-IA and Gait-IA do not require root privileges and only require Android permissions. Input event data can be collected by individual apps without any additional permissions, which opens the door for IA protection at the app level instead of the device level. For example, enterprises can bundle IA schemes within their apps to protect confidentiality of their corporate data. While providing IA at the app level mitigates the restrictions imposed by Android, it also imposes significant development overhead. We leverage this observation to address some deployment challenges to IA in Chapter 6.

Using a realistic threat model and evaluating in an uncontrolled environment is necessary when evaluating an IA scheme. Some IA proposals are accompanied by unrealistic evaluations, by having users perform a repeated task in a lab setting to generate data. When these schemes are then applied in real-world settings, the assumptions made in the lab may prove false and the scheme's performance will suffer accordingly. Feng et al. [FYY+14] demonstrate that on real-world datasets, many existing touch-based IA schemes have significantly higher EER than reported in the original papers. Our findings are similar for the IA schemes that had their datasets publicly available [FMP10, FBM+13]. Furthermore, a recent Symantec study finds that 68% of non-owners who attempted to access private data on an unguarded smartphone did so on the spot, which would make location filtering an unhelpful IA feature [Wri12]. A similar study by Lookout-Sprint [Loo16] found that 44% of users were primarily concerned with their devices being accessed by family and friends, as opposed to strangers. Since such adversaries may have multiple overlapping features (e.g., location and contacts), IA schemes that rely on such features will not be very effective. Therefore, it is critical to provide protection against a realistic threat model that captures these security and privacy concerns of smartphone users.

3.6 Conclusion

In this chapter we provided a comparative evaluation of six IA schemes that employ different behavioural features. Our empirical evaluations show that IA can be performed with reasonable accuracy and low complexity with acceptable detection delay on contemporary

mobile devices. More specifically, our evaluations show that in addition to adequate data availability for training and classification, touch IA schemes outperform other schemes in terms of accuracy and detection delay. We also analyzed real-world traces to show that while text and gait IA schemes provide reasonable performance, there was not enough data available for a significant proportion of sessions to make a classification decision. In terms of evaluation of IA schemes by the research community, our findings emphasize the significance of evaluation on uncontrolled datasets. Furthermore, an overview of features employed by device usage IA indicates that there is a need to consider a realistic threat model.

Finally, while our evaluation quantifies the performance of IA schemes, it does not provide any empirical evidence on the human side of IA. For instance, the finding that most touch IA schemes correctly identify over 95% misuse attempts with less than 10% false rejects in under ten seconds, does not convey whether it is *acceptable* for users from a security standpoint. In the following chapter, we bridge this gap by conducting experiments with human subjects on the usability and the acceptability of the security properties of IA.

Chapter 4

Usability and Security Perceptions of IA

The high accuracy reported by touch IA schemes has resuscitated the potential of IA to mitigate the usability issue with primary authentication mechanisms. Touch IA schemes have gained traction in technology media news with claims like: *"Identifying someone by the way they tap and swipe on a touchscreen might be the more natural, unobtrusive future of smartphone biometrics"* [New16]. However, these claims have not been substantiated. The review of the existing IA literature in Chapter 2 showed that the focus of the majority of IA research is on improving the accuracy whereas the usability evaluation has largely been ignored. Existing IA literature has assumed without empirical evidence that since IA authenticates without requiring explicit input, it is more usable. For instance, when Jakobsson et al. introduced the term IA with a device usage IA scheme, they postulated that *"this is a meaningful approach, whether used to increase usability or increase security"* [JSGC09].

Given that IA does not require explicit input to authenticate the device user for every session, intuitively it seems that IA should reduce the amount of time spent on authentication. However, despite the reasonably high detection accuracy of some IA schemes, these schemes are still subject to false rejects, false accepts and detection delays (see Section 3.4.1), which could introduce new usability issues and affect users' security perceptions. If the IA detection model is unsure about the user's identity, it naturally resorts to an interrupt-authenticate approach in which the current task is pushed to the background and the user has to explicitly authenticate to establish their identity [FYY+14, LZX13]. This interrupt-authenticate approach for false rejects is quite different from consistent

explicit authentication (EA). It remains unclear how it affects usability in terms of annoyance and task performance in terms of time and error. Similarly, it is not obvious whether the usability-security trade-off offered by IA overcomes the perception of security given the risks of false accepts and delay in detection of an intruder. Finding answers to these usability aspects and security perceptions of IA will determine whether IA provides a competitive advantage in terms of usability over EA.

Our research effort aims to answer the research questions that were not explored in the previous usability evaluations of IA (see the existing literature on usability evaluations of IA in Section 2.4.7). More specifically, the existing efforts on usability and security perceptions evaluations of IA by Clarke et al. [CKF09] and Crawford and Renaud [CR14] fail to: (i) evaluate IA in the wild; (ii) perform a through usability evaluation given FRs; (iii) evaluate task performance given the interrupt-authenticates due to FRs; and (iv) evaluate security perceptions due to FAs and detection delays.

We expand our second research objective (see *Objective 2* in Section 1.5) to addresses the aforementioned limitations of the existing research:

- To conduct a field study to quantitatively and qualitatively evaluate the usability of IA given FRs
- To determine the security perceptions of IA given all IA schemes are subject to FAs and detection delays
- To identify whether the usability and security properties of IA are acceptable for users and whether they are interested in adopting IA

We begin by outlining the aims for our comprehensive user study, which meet the aforementioned objectives.

4.1 Study Goals

We divide our goals to investigate IA usability and perceived security into seven questions. Later, we organize our study results around these seven questions.

Our main goal regarding IA usability was to test established usability metrics and commonly accepted usability assumptions, as captured by the following research questions:

U1 Does IA decrease the overall task completion time and the authentication overhead when compared to EA?

U2 Do the interrupt-authenticates in IA increase the error rate of the primary task?

U3 Are fewer but less predictable authentication interrupts of IA less annoying or tolerable as compared to EA or no authentication at all?

U4 Does IA score higher on the system usability scale [Bro96] as compared to EA?

U1 and U2 address standard usability metrics for time and error as they may be affected by the interrupt-authenticate model of IA. These metrics have never been evaluated directly with IA, but they have been used to measure performance impact of similar task interruptions with personal computers [BK06] and they have been implied as benefits of IA in previous work [JSGC09]. Furthermore, with desktop systems, Bailey and Konstan [BK06] found that interruptions increased errors overall so it was an important metric to include. By answering U3, we will test levels of annoyance caused by FRs and through U4, we test claims of higher perceived usability for IA compared to the primary authentication baselines of EA [CKF09, CR14] and no authentication.

Our main goal for the security perceptions of IA was to explore the following research questions:

S1 Are the security properties of current IA schemes (such as the FA rate) acceptable to users?

S2 Is the perception of IA security better than common current authentication schemes?

S3 Are smartphone users interested in adopting IA?

S1 has never been explored in the IA literature. S2 has been explored in previous studies [CKF09, CR14] and we attempt to validate these prior findings. In addition to evaluating the overall perceived level of security, we elicit the perceived level of security against different types of adversaries, different device states and different types of tasks. Finally, answering S3 provides an indication of IA deployment potential from a human-centric perspective since it essentially combines security perceptions and usability.

4.2 Study Design

We use a two-part study for our evaluations. The first part is a lab-based experiment where each participant performs simulated tasks with IA and with their current authentication

scheme. This provides highly controlled, quantitative results. Measuring annoyance and other subjective feedback caused by IA interruptions is more ecologically valid when evaluated with real tasks over a longer time period, so the second part is a three-day field study where participants used IA on their own smartphone. For experimental control, both parts use a pseudo-IA scheme (described below). Our methodology was reviewed and approved by the ORE of our university.

4.2.1 Participants

The in-lab study was completed by 37 participants and 34 of those same participants completed the field study. Three participants dropped due to technical issues (two participants had device encryption enabled and one participant reported a broken device). We recruited these participants using multiple sources including: (i) an advertisement on Craigslist and Kijiji in November of 2014, under the "other jobs" section; and (ii) on the university-wide mailing list. The title of the advertisement was "Participate in a research study on the efficacy of a novel authentication scheme on smartphones" and it stated that the study was about the evaluation of a novel authentication scheme and adults who owned and used an Android-based smartphone for over six months could participate. Those interested were requested to fill out an online screening survey (provided in Appendix A), which collected information about their age, gender, profession, security preferences, smartphone make and model, amount of time they have used a smartphone, and email address. Participants were paid $35 ($10 for each of two in-lab sessions and $15 for the field study).

Participant demographics, current authentication schemes, and authentication preferences are summarized in Table 4.1. Current authentication scheme by age group is provided in Figure 4.1. Overall, our participant pool has good diversity by profession, age, and current authentication scheme. For our research questions, this kind of diversity is important. Similar to the prior studies [EJP+14, HVZF+14], the top reason our participants gave for not using any authentication scheme was inconvenience. Furthermore, about half of the participants who used some authentication scheme agreed that it was inconvenient or annoying at times. The annoyance was split by current authentication scheme: PIN users were significantly more likely (53% more) to find their authentication scheme inconvenient as compared to the pattern lock users (Fisher's Exact Test, $p = 0.028$).

In terms of current authentication, 14 participants used no authentication, eleven used Android's Pattern Lock, nine used a four-digit PIN and three used other schemes (two participants used a password and one participant used a longer PIN conforming to his company policy). We use the participants' current authentication scheme as an independent between-subject variable, which we refer to as *Use*. Where relevant, we summarize

Table 4.1: Demographics and security preferences of the study participants.

n = 37	
Gender:	56% Female
	43% Male
Occupation:	32% Employed
	30% Grad student
	24% Undergrad student
	13% Unemployed/retired
IT experience:	22% Studied/worked in IT
Current authentication scheme:	38% None
	24% PIN
	30% Pattern-lock
	8% Other
Sharing habits:	51% Never
	41% Rarely (once a month)
	5% Occasionally (once a week)
	3% Daily
Top reasons for not using any authentication:	8/14 It's a hassle/takes time
	5/14 Nothing to hide
	3/14 Never thought about it
Top reasons for using authentication:	19/23 Protected if lost/stolen
	18/23 Protected when unattended
	12/23 Someone casually picking it
	10/23 Unwanted disclosures
Protecting against:	18/23 Strangers
	12/23 Coworkers
	8/23 Friends/roommates
	7/23 Spouse/own children
Thoughts on authentication:	13/23 It is inconvenient sometimes
	10/23 It is easy
	3/23 It takes time

results using groups and subgroups formed by this variable. Specifically, *DontUseAuth* is the group of participants who reported that they do not use any authentication and *UseAuth* refers to the group of participants who reported that they use some EA scheme. We further separate *UseAuth* into two common EA schemes: *UsePIN* for the subgroup of *UseAuth* participants who reported using a PIN and *UsePAT* for the subgroup of *UseAuth* participants using a pattern lock.

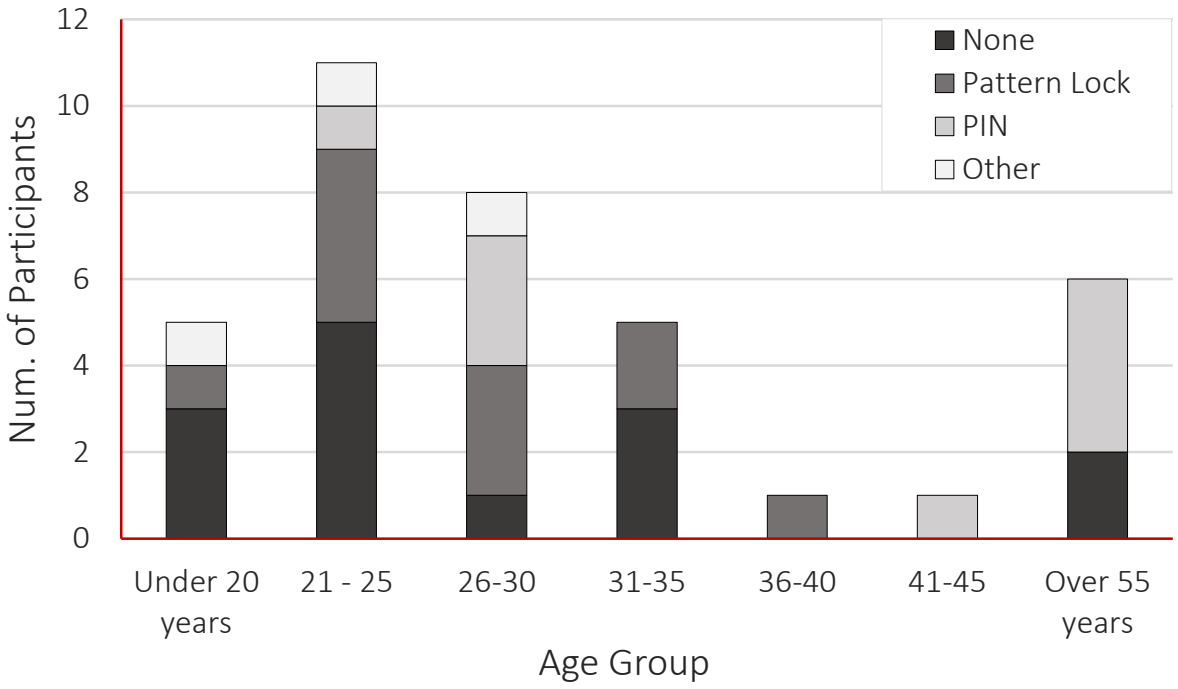


Figure 4.1: The "Age Group" – "Current Authentication Scheme" distribution for study participants.

4.2.2 Apparatus

We developed two Android apps, Explicit Authentication and Implicit Authentication, that executed on the devices of the participants during both parts of the study. For the lab-based experiment, the apps presented a series of tasks to the participants. The apps contained authentication screens to authenticate the participants using a PIN, Android's pattern-lock or a six character password. We used the Android Open Source Project's UI and implementation[1] for the pattern-lock and used a UI identical to that of Android for PIN and password screens. We simulated authentication on the participants' devices using our apps (explained in Section 4.2.3) to accurately measure the time spent on authentication. For the field study, the Implicit Authentication app executed as a background service to simulate FRs. Although IA was simulated, participants were told that all biometric data remained solely on their device.

Deception: In both parts of the study we used a pseudo-IA scheme that ostensibly

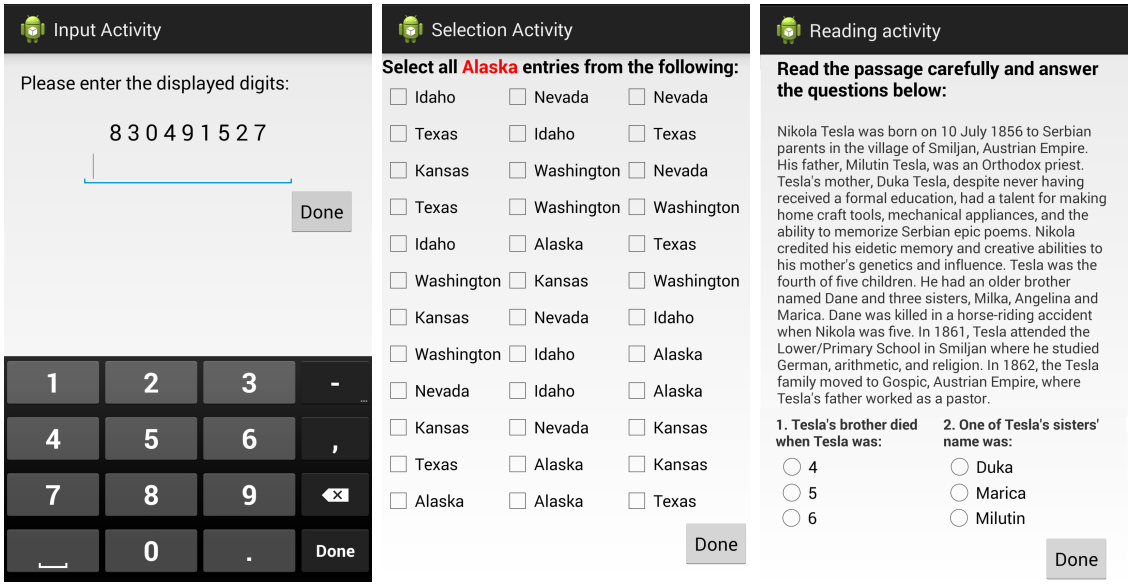
[1] http://code.google.com/p/android-lockpattern/

employed the touch behaviour biometric. The pseudo-IA scheme interrupt-authenticated users by triggering authentication screens during their interaction with the device to simulate FRs. Our pseudo-IA scheme was configurable to simulate different FR rates and detection delays. We used a pseudo-IA scheme because: it was not possible to intercept touch input events for IA without rooting the device due to the constraints imposed by Android [KAH14b]; and a pseudo-IA scheme enabled us to strictly control the frequency and the detection delay of FRs. Participants were told that the IA scheme on their device was fully operational and that one of the in-lab sessions served as training for the IA algorithm (details provided in Section 4.2.3). Furthermore, to reduce the chance of participants discovering this deception by testing it with other users, we asked them not to share their devices during the field study arguing that sharing would interfere with the brief re-training phases required by our early-stage IA scheme. There are limitations of using a pseudo-IA and we discuss these limitations in Section 4.5.

Configuration parameters selection: A challenging aspect was the selection of IA configuration parameters. By choosing different parameter values, we can change the behaviour of IA schemes. For instance, there is a negative correlation between FAs and FRs, and increasing detection delay reduces the number of FRs. While there is no recommended operating threshold, we inspected different operating thresholds because: (i) high accuracy input behaviour-based IA schemes have deployment constraints as compared to low accuracy schemes that can be readily deployed [KAH14a]; (ii) studies report a high degree of variability for FR rate between users [DZZ14]; (iii) variability in FR rate for an individual user may be caused by the type of activity that the user is performing [FYY+14]. We chose a representative and realistic FR rate range based on previous work [KAH14a]. We evaluated three FR rates: 5%, 10% and 20% that have a corresponding FA rate range between 3%–18% and 5–30 seconds of detection delay for different IA schemes. We discuss operating threshold configurations specific to each experiment in Section 4.2.3 and Section 4.2.4.

4.2.3 Part 1: Controlled lab experiment

The purpose of the controlled lab experiment is to: (i) introduce the participants to IA; (ii) perform an A/B testing of IA with non-IA (*UseAuth* or *DontUseAuth*); (iii) demonstrate an ostensible TR; (iv) elicit initial feedback on usability and security perceptions of IA; (v) collect data to evaluate the performance metrics of U1 and U2; and (vi) test the pseudo-IA scheme on the participants' devices without any EA scheme configured (this was not possible in the field study due to data security and privacy threats without EA).



(a) Input Activity (b) Selection Activity (c) Reading Activity

Figure 4.2: Apps' screens showing different activities for the lab-based experiment. App's screens presented here have been modified to show complete activities.

Task

Our two apps presented a series of experiment tasks on the participant's own smartphone. In the experiment, each *task* represented a device usage session. The participant waited for the device to ring (with vibrate) to indicate it was time to perform a task. For the Explicit Authentication app, the participant turned the screen on, performed authentication if required, completed an activity, and turned the screen off. For the Implicit Authentication app, instead of the authentication at the beginning, the participant was interrupted and authenticated in the middle of an activity (frequency and timing of interrupt-authenticates are discussed in the Design section). There was no time limit to complete a task. To simulate longer breaks between real device usage sessions, the participant waited a random time between 8–15 seconds before performing the next task.

We chose a subset of activities from the primary activities proposed by Bailey and Konstan [BK06]. We chose those activities that were abstract representations of common mobile activities and were diverse in terms of difficulty and cognitive load, enabling us to inspect error rate and interrupt-authenticate overhead. A description of these activities by increasing level of difficulty due to higher mental loads on working memory (based on the

rankings of Bailey and Konstan [BK06]) are provided below (see also screen captures in Figure 4.2):

- *Input Activity:* entering a sequence of characters. Our activity required participants to enter a nine digit number displayed on the screen into an input field. For the input number, we carefully chose permutations that did not overlap with the local area codes and did not have any consecutive or repeating integers. This activity is representative of common smartphone activities like entering search queries, composing texts, and entering emails.

- *Selection Activity:* selecting multiple items from a list of items. We used a list of words with selection checkboxes arranged in a 12-row x 3-column table. Thirty-six words were randomly chosen from a base set of six words. Participants had to select each word in the table that matched a given target word (taken from the base set). This activity is representative of common smartphone activities such as choosing a number to dial, choosing an app to launch or scanning the results of a search.

- *Reading Activity:* reading and comprehending information. Participants read a 7–10 sentence narrative passage from Wikipedia and then answered two multiple choice questions regarding its content. This activity is representative of common smartphone reading and comprehension activities such as reading emails or web browsing.

Design

Participants completed the lab-based experiment in two sessions held on different days. Each session lasted between 45–60 minutes including introduction, pre-survey, experiment tasks, post-survey, and interview. In each session, experimental tasks were completed under one of two within-subject conditions: the *IA* condition when they used IA and *non-IA* when they did not use IA. The order of the IA and non-IA sessions was counterbalanced across participants.

Each session had 30 task trials with each task showing one activity. There were ten instances of each of the three activity types. Since the activity types were of varying difficulty level, we did not use simple random sampling to select their order since some orderings could introduce a confounding effect (e.g., the first 10 tasks are all difficult activities). Instead, we constrained the random presentation order by creating five blocks of six tasks where the tasks have two variations of each activity. We then permuted the order of tasks within each block using the first 5 rows of a 6x6 Latin square. This counterbalanced the varying difficulty levels of activity types. The same order of blocks

and tasks was used across IA and non-IA sessions across all participants to create an unbiased comparison and to make sessions directly comparable.

For the non-IA session, participants were assigned the same authentication scheme that they used currently on their device (which could be an EA scheme or no authentication). For the interrupt-authenticate caused by a simulated FR in the IA session, *UseAuth* participants used their current authentication scheme while *DontUseAuth* participants were assigned a scheme that they preferred to use. Although we could have assigned a random authentication scheme to *DontUseAuth* participants, this could have introduced negative bias from dislike or inexperience with the assigned scheme. In both sessions, we were not interested in the memorability of the secret. Participants could write down their secret or reset it in between tasks if they wished.

While the input activity naturally generated tapping data, we rendered the reading activity and the selection activity in such a way that participants had to swipe to scroll to see their content. This led them to believe that their interactions were used as a biometric. We also used deception in terms of training by telling the participants who tested IA in their first session that the data from the first few tasks was used for training. The participants who tested IA in the second session were told that the data from the first session was used for training.

We used a 20% FR rate (six interrupt-authenticates in total, twice for each type of activity) and a detection delay between 5–10 seconds. A lower detection delay (2–4 seconds) was used for the shorter input activity.

Procedure and data collection

The shortlisted participants were asked to bring their devices to the lab. We started the first session by showing a two-minute video introducing the apps and activities[2]. Participants were introduced to IA using a three-minute video before the IA session, which explained the operations of touch behaviour-based IA and the associated FAs, FRs and operating threshold[3]. A researcher was available during these video demonstrations to answer any questions. After the briefing, participants downloaded and installed the app for the session through Google Play Store. They were then asked to set the current authentication scheme to 'None' and turn on 'airplane mode' on their devices. This eliminated notifications or interruptions during the experimental tasks and enabled our app to control all authentications.

[2] http://youtu.be/qDQm_0ad6Pw
[3] http://youtu.be/HUR2-bxBtI8

After device setup, participants completed the state-trait anxiety inventory (STAI) survey (provided in Appendix D) [Spi83] to provide us with their current state of anxiety. The participants were then asked to launch the app to configure an authentication secret and then complete the main experiment tasks. After completing all tasks, they provided another measurement on anxiety by completing the STAI survey again. The participants were asked to complete a post-survey (provided in Appendix B) for the non-IA and the IA sessions. This survey consisted of 12 questions regarding usability and security perceptions of the authentication schemes that they tested. The participants used the survey to rate their perceived level of security (overall and for different adversaries, device states, and different tasks) and usability (in terms of convenience, annoyance, time consuming and tiring). Participants who tested any authentication scheme during a session were asked to complete the system usability scale (SUS) survey [Bro96] after the session. The SUS survey was modified (provided in Appendix C) to explicitly inform the participants that it was evaluating the authentication scheme, not the apps. We also changed the word 'system' to 'method' and we dropped the question 'I found the various functions in this system were well integrated' because it was not applicable to our evaluations. Finally, a semi-structured interview (provided in Appendix E.1) of 10–15 minutes gained insight into survey answers. The interviews were recorded and later transcribed.

4.2.4 Part 2: Field study

The field study was conducted after the lab study with the same participants. The main purpose of the field study was to gather realistic data on potential annoyance due to FRs. In addition, we wanted to subject participants to different operating thresholds to determine a tolerable one in terms of the frequency of FRs.

Task

The task for the field study was for participants to use their device as usual and handle simulated IA FRs (experienced as interrupt-authenticate screens) as they occurred. After each interruption the participant also provided brief feedback through an in-situ pop-up.

Each FR interrupted the current smartphone app with an authentication screen requiring an explicit authentication. These were the same simulated authentication screens used in the lab experiment. The background service in the Implicit Authentication app monitored two events: the `ACTION_SCREEN_ON` event to keep track of when the users turned on their screens and the `ACTION_USER_PRESENT` event to know when the users were present

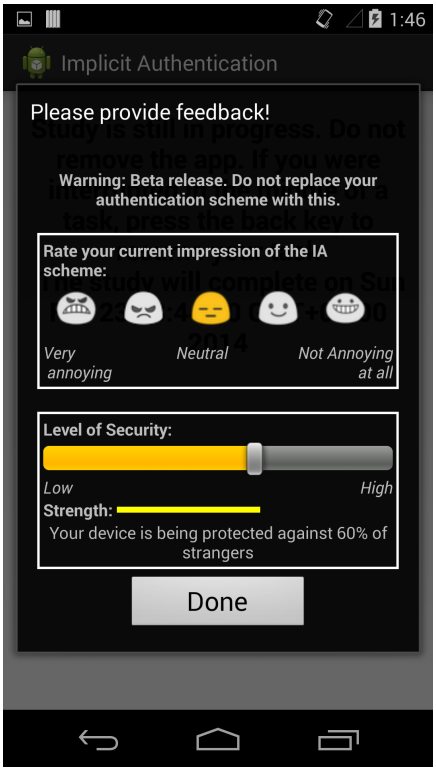


Figure 4.3: The in-situ feedback pop-up after interrupt-authenticate.

on their devices after dismissing the lock screen. An interrupt-authenticate was triggered after k `ACTION_USER_PRESENT` events (k was controlled during the study, details are in the Design section).

We were also interested in measuring the annoyance of each FR. After an interrupt-authenticate, we performed experience sampling with a simple in-situ feedback screen (Figure 4.3). It asked the participants about their current annoyance on a 5-point Likert scale ("Very annoying" – "Not annoying at all"). The feedback screen also displayed the current operating threshold and the associated security strength of that threshold (in terms of the proportion of strangers that the IA scheme would likely protect against).

Design

We conducted the field study for three days to measure annoyance for different operating thresholds. FRs were simulated after every k `ACTION_USER_PRESENT` events and we

randomly chose a value of k for each day to reflect high, medium and low accuracy corresponding to 20%, 10% and 5% FR rates, respectively. Since we were unable to determine when a participant interacted with the touch screen after an `ACTION_USER_PRESENT` event, we simulated a FR by choosing a random delay between 15–30 seconds. If the participants turned off the screen before the delay timeout, they were authenticated in the next session with a reduced delay. The delay value is decreased by five seconds each time down to a minimum delay value of ten seconds to ensure that the participants with short sessions also experienced FRs. We did not simulate a FR for the sessions when the call state of the device was ringing or off-hook.

We allowed participants to reduce the operating threshold if they wished. To ensure that the participants did not set the operating threshold to zero, the participants' adjusted value was only effective for 30 minutes and after that it was reset. This mitigated the possibility of participants killing the background service if interrupts became too irritating (participants felt they had some control) and gathered data to study the potential need and utility for users to control the trade-off between usability and security. The briefing video explained the trade-off when setting different threshold values. Participants could use the interface to increase the operating threshold; however, their action had no operational effect. The assigned security strength corresponded to the best achievable accuracy for the SilentSense scheme (see Section 3.4.1). We told participants that the IA scheme automatically adjusted the threshold value after brief re-training phases but they could change it depending on their desired level of protection. Participants were informed about this behaviour and that they could dismiss the interrupt-authenticate by pressing the home button but we asked them to avoid doing so except in extreme cases.

Procedure and data collection

After the second in-lab study session, participants were briefed about the background service executing on their devices and the interface of the in-situ feedback pop-up. The Implicit Authentication app was programmed to reject the user on its first use. The researcher used this feature to perform an ostensible demonstration of a true reject on participants' device to lead them to believing that the IA scheme was behaving as expected. After the completion of the three day usage period of the pseudo-IA scheme, the participants were instructed to contact us through emails to arrange for an in-person semi-structured interview (provided in Appendix E.2) of 10–15 minutes and to collect the remuneration. The participants were also debriefed about the deception at the end of this interview.

During the field study, we logged the `ACTION_SCREEN_ON` and `ACTION_USER_PRESENT` events. From the in-situ feedback, we logged the level of annoyance of IA and the adjusted

value of operating threshold.

4.3 Results

The quantitative and qualitative results of the controlled lab experiment and field study are presented together organized by the research questions raised in Section 4.1. A discussion is provided after the results for each research question.

For the in-lab study, participants completed all tasks in 25 minutes on average (*median = 23, sd = 4*). During the IA session, participants witnessed 222 FRs in total. For the field study, on average our app logged 104 `ACTION_SCREEN_ON` events (*median = 57, sd = 65*) and 63 `ACTION_USER_PRESENT` events (*median = 42, sd = 28*) per participant per day. In total 10,608 `ACTION_SCREEN_ON` events and 6,420 `ACTION_USER_PRESENT` events for 34 participants were logged across three days. During the field study, participants also provided feedback against 693 FRs (98, 214, and 381 for low, medium and high operating threshold, respectively) and dismissed 42 authentications and feedbacks.

For qualitative analysis of the semi-structured interviews, the researchers coded all participant responses using the grounded theory approach [GS09] with meetings to achieve consensus. For test statistics, we use a t-test when comparing continuous data between subjects (such as between *UseAuth* and *DontUseAuth* or between *UsePIN* and *UsePAT*). We use a paired t-test when comparing continuous data for the within subjects condition (IA and *Non-IA*). We use a chi-square test for participants' responses to categorical Likert scale questions. We use $p < 0.05$ to indicate whether the test result is statistically significant. For multiple comparisons (such as in Section 4.3.1), we use the adjusted p-value based on Bonferroni correction and set the significance cut-off at α/n, where n is the number of multiple comparisons [Hol79].

4.3.1 Usability evaluation of IA

U1: Effect of IA on overall task completion time and authentication overhead

Overall task completion time is the total time to complete all 30 tasks including EA authentications or IA interrupt-authenticates if present. We calculate the increase or decrease in this time for each individual participant for their IA session compared to their non-IA session. This relative measure compensates for inter-participant differences due to reading level, motor skills, etc. The time differences are aggregated by *UseAuth* and *DontUseAuth*

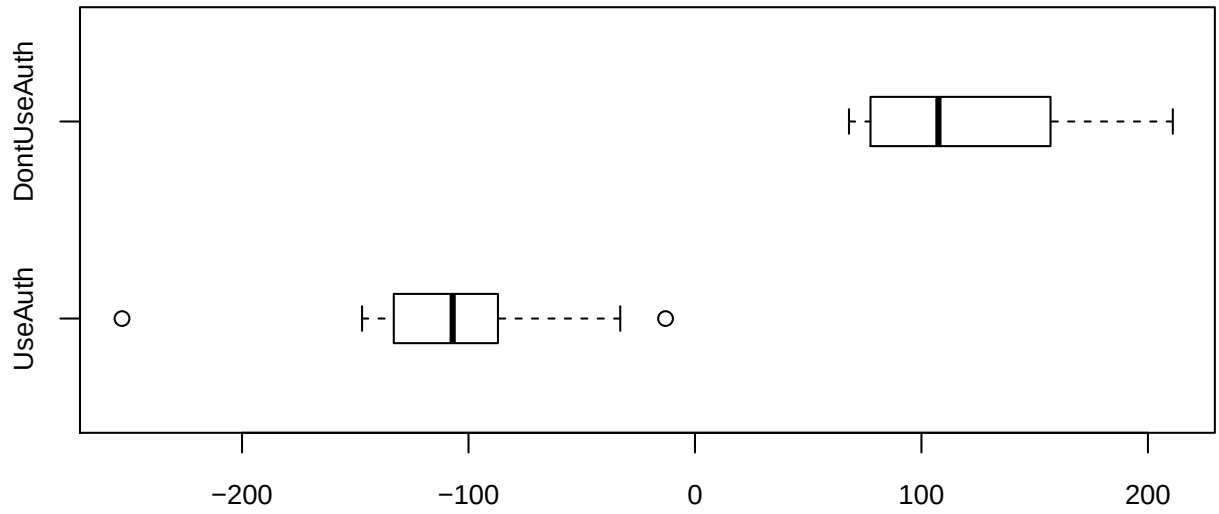


Figure 4.4: Change in the overall task completion time for the non-IA session as compared to the IA session.

participants (see Figure 4.4). Intuitively, the IA session should take less time as compared to the non-IA session for *UseAuth* participants due to fewer authentications, and more time when compared to the non-IA session for *DontUseAuth* participants. For *UseAuth* participants, the overall task completion time on average decreased by 100 seconds (*median = -103; mean = -101; sd = 59*). A paired t-test between the completion times of the IA and non-IA session for the *UseAuth* participants indicates that they are significantly different (t = *-3.6*, p = *0.01*). The overall task completion time for *DontUseAuth* participants increased by 120 seconds on average (*median = 107; mean = 122; sd = 52*) for the IA session. A paired t-test between the completion times of the IA and non-IA session for the *DontUseAuth* participants indicates that they are significantly different (t = *5.2*, p = *0.014*).

We also evaluate the interrupt-authenticate overhead, defined as additional time taken for IA interrupted tasks compared to non-interrupted tasks. For our tasks, interrupt-authenticate overhead is the difference between the average completion times of an activity, with each activity type analysed separately. Figure 4.5 shows that on average, *DontUseAuth* participants had an interrupt-authenticate overhead of 8, 15, and 20 seconds for the interrupted input, selection, and reading activities, respectively. Similarly, on average, the *UseAuth* participants had an interrupt-authenticate overhead of 7, 11, and 16 seconds for the input, selection, and reading activities, respectively. A t-test for interrupt-authenticate for each activity reveals that the difference is not significant for the input activity between

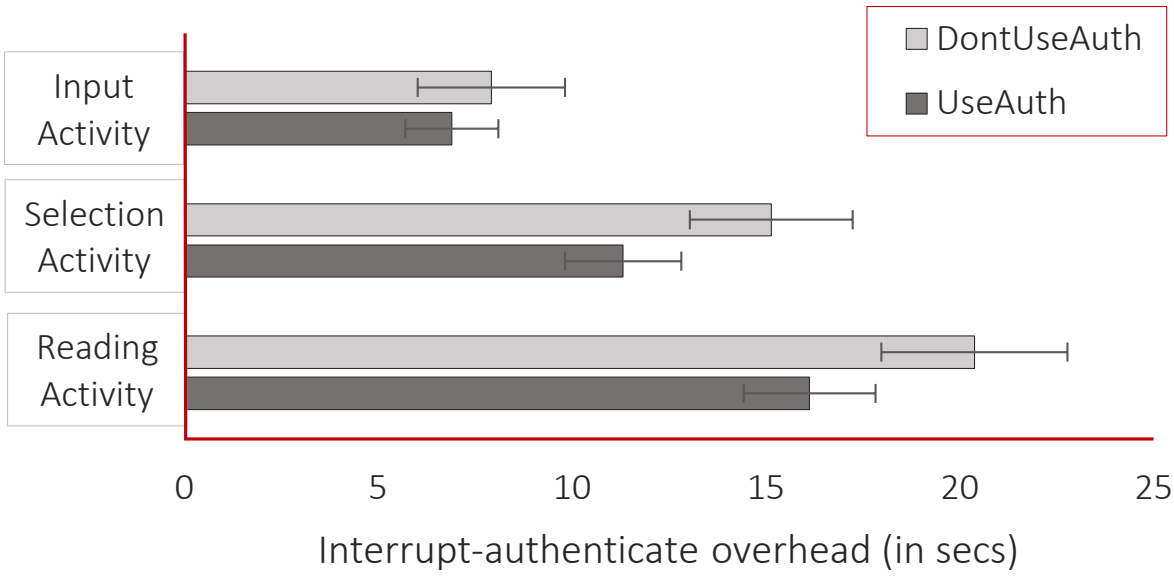


Figure 4.5: Interrupt-authenticate overhead for different activities (error bars represent 95% confidence intervals).

IA interrupted tasks and non-interrupted tasks ($t = 1.6$, $p = 0.11$), but the difference is significant for the selection ($t = 7.8$, $p = 0.002$) and the reading activity ($t = 10.5$, $p = 0.002$).

Discussion: While these results indicate that IA imposes an interrupt-authenticate overhead for the individual interrupted tasks, the total completion time decreased by 7.1% for *UseAuth* participants because they did not authenticate for every task. For *DontUseAuth* participants, we observe an 8.8% increase in the total completion time due to interrupt-authenticates. It should be noted that the performance gains (or losses) will be more pronounced when the number of device usage sessions increases. The interrupt-authenticate overhead was primarily due to the unpredicted or sudden "lock-out" and the context switch as pointed out by the participants:

> *"It pops-out very suddenly... in between the tasks at times. I got tensed because I was worried about completing the task without making the pop-up appear... getting pop-up in the middle of task was quite distracting"* (P10)

> *"I generally lost my train of thought when it popped up that authentication"* (P37)

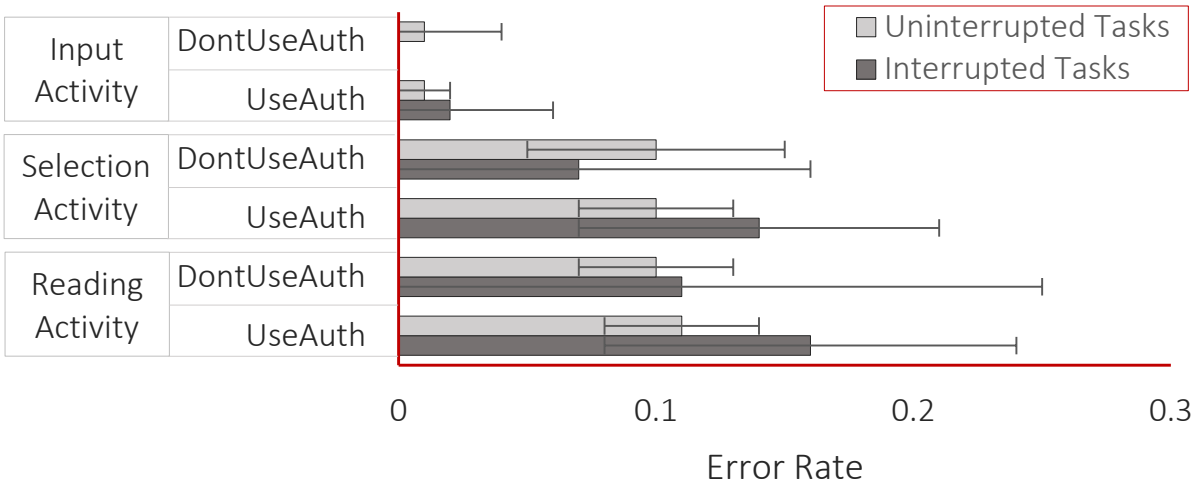


Figure 4.6: Error rate between interrupted tasks from the IA session and corresponding uninterrupted tasks from the non-IA session (error bars represent 95% confidence intervals).

U2: Effect of IA on the task error rate

We classified errors using simple correctness checks built into the apps. An error occurred when: entered numbers mismatched in the input activity; incorrect answers were provided to a question in the reading activity; or a target word was missed or a non-target word was selected in the selection activity. We calculated the error rate separately for the 222 interrupted tasks from the IA session and for the 222 uninterrupted tasks located at the same task index from the EA session (see Figure 4.6).

A t-test indicates that the differences in error rates across uninterrupted and interrupted tasks are not statistically significant for input (t = *-1.0*, p = *0.69*), selection (t = *1.5*, p = *0.32*) and reading activities (t = *-1.8*, p = *0.30*). There is no evidence that the interrupt-authenticate model increases the error rate.

Discussion: Our results agree with Bailey and Konstan's [BK06] findings for personal computers where interruptions did not increase the error rate of interrupted tasks. However, they also found that the expectancy of interruptions caused more errors overall (due to a higher load on the cognitive resources). While our participants complained about the unpredictability of interrupt-authenticates, a paired t-test reveals that there is no significant difference between the error rates of the IA and non-IA session (t = *0.84*, p = *0.4*).

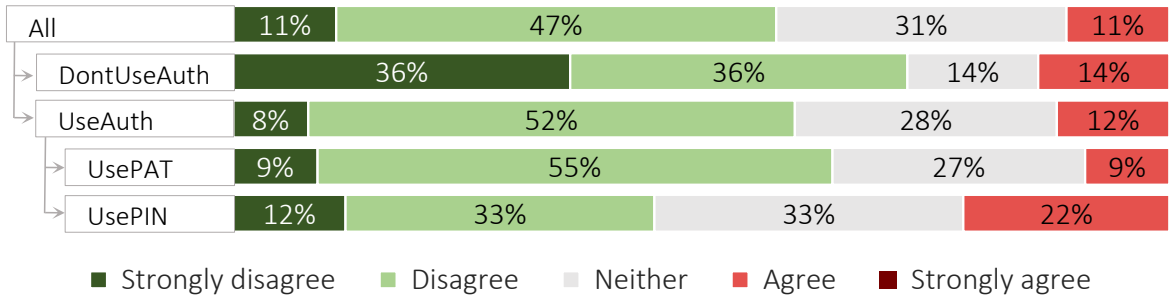


Figure 4.7: Responses for "Do you agree with the statement 'I think this method is annoying'?".

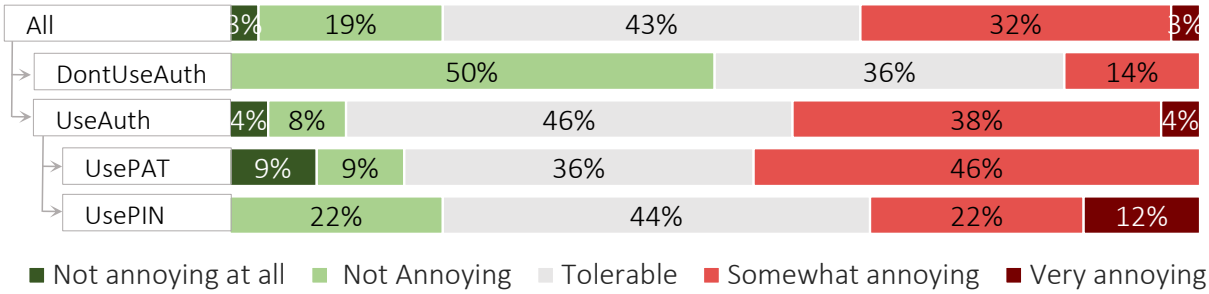


Figure 4.8: Responses for "How annoying were the interruptions for authentication?".

U3: Effect of fewer but less predictable authentication interrupts on annoyance

After the in-lab sessions, participants answered survey questions regarding annoyance. The first question asked if they thought the overall experience of IA was annoying (see Figure 4.7). Overall, 58% did not say IA was annoying, 11% considered IA as annoying while the rest were neutral. Furthermore, significantly fewer *UseAuth* participants (12% less) thought IA was not annoying compared to *DontUseAuth* participants ($\chi^2(1) = 5.1, p = 0.02$). There is also a significant difference in annoyance for participants based on the type of EA currently used: significantly more *UsePIN* participants (22% more) found IA to be annoying compared to *UsePAT* participants ($\chi^2(1) = 4.09, p = 0.04$). We suspect that IA's lower perceived level of protection by *UsePIN* participants (discussed in Section 4.3.2) and consequent low utility is responsible for this.

The second question asked participants how annoying the IA *interrupt-authenticates* were (see Figure 4.8). Overall, 35% of the participants found them to be annoying (32%

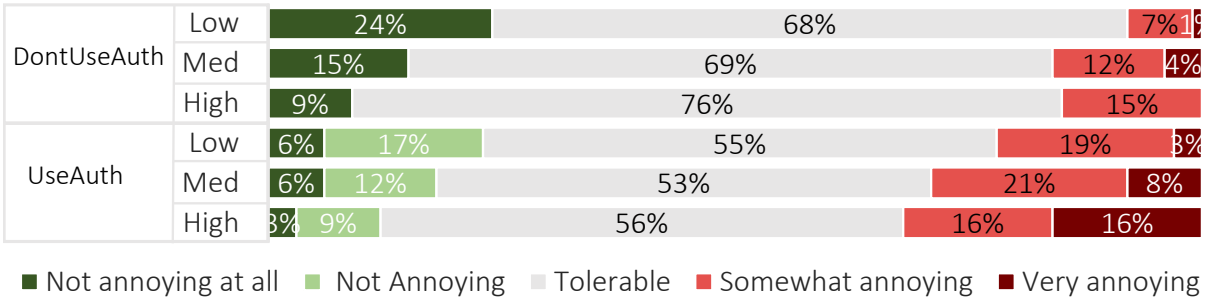


Figure 4.9: Annoyance against three operating thresholds from the in-situ feedback survey of the field study.

somewhat annoying and 3% very annoying), 44% found them to be tolerable and 21% found them to be not annoying. Furthermore, significantly more *UseAuth* participants (28% more) found interruptions to be annoying as compared to *DontUseAuth* participants ($\chi^2(1) = 9.4, p = 0.002$).

During the field study, we subjected participants to three different operating thresholds corresponding to 5%, 10% and 20% FR rate with a goal to determine an *acceptable threshold* (operating thresholds with "tolerable" or better annoyance ratings). The feedback of participants for annoyance across these FR rates is provided in Figure 4.9. *DontUseAuth* participants found interrupt-authenticates to be more acceptable for different thresholds as compared to *UseAuth* participants. More specifically, for low, medium and high FR rates, interrupt-authenticates for *DontUseAuth* participants were significantly more likely (14%, 13% and 17% more) to be acceptable as compared to *UseAuth* participants ($\chi^2(1) = 11.4, p < 0.001$), ($\chi^2(1) = 8.2, p = 0.004$) and ($\chi^2(1) = 13.2, p < 0.001$), respectively. Figure 4.9 also illustrates responses in terms of the proportion of interrupt-authenticates that are annoying between low-medium and medium-high thresholds, while differences between medium-high thresholds are negligible. More specifically when *DontUseAuth* participants were subjected to the low threshold, interrupt-authenticates were significantly more acceptable (8% more) as compared to the medium threshold ($\chi^2(1) = 4.7, p = 0.03$). On the other hand, for *DontUseAuth* participants the difference in terms of proportion of acceptable interrupt-authenticates between medium-high threshold was insignificant — 84% vs. 85% ($\chi^2(1) = 0.07, p = 0.78$), respectively. These observations for inter-threshold level correlations across *DontUseAuth* participants were also true for *UseAuth* participants.

Discussion: Although the majority of participants were not annoyed with IA, it is clear that interrupt-authenticates can cause moderate levels of annoyance, more so for users who currently use EA. During the qualitative interviews, we asked the participants

for the cause of this annoyance and 10/37 participants indicated the unpredictability of the interrupt-authenticate as the reason:

> *"I think it is a little bit annoying because there is a little stress [when] you don't know what will happen"* (P15)

> *"I am ready to enter a password before I start doing anything whereas for implicit authentication it catches me off-guard"* (P14)

4/37 participants were more annoyed due to a perceived error at the part of IA:

> *"I thought the interruptions were very annoying, I felt that I was doing the same thing and it triggered multiple times."* (P8)

> *"It is the unpredictability of it... I know that I have to enter my PIN every time and this becomes annoying... it would be frustrating because you don't know what was wrong that you did, with PIN you know because it is something wrong that you entered "* (P14)

> *"It sure was annoying. I use my phone a lot when I am watching TV and at times my device turns off due to inactivity. That's my fault and [it] is understandable but when this interrupts me, I think that the operating system is faulty or something"* (P37)

For the field study, eight *UseAuth* participants had to use IA in addition to their EA scheme despite their preference to replace their current EA scheme with IA. Two of these participants mentioned that the cause of annoyance was "redundant authentications":

> *"I felt like it was a lot of work with two PINs. At times I would confuse which one was which and then had to re-enter it. That made it more annoying"* (P20)

> *"I already use a password and after that it was quite annoying to use a second one. It seemed redundant"* (P32)

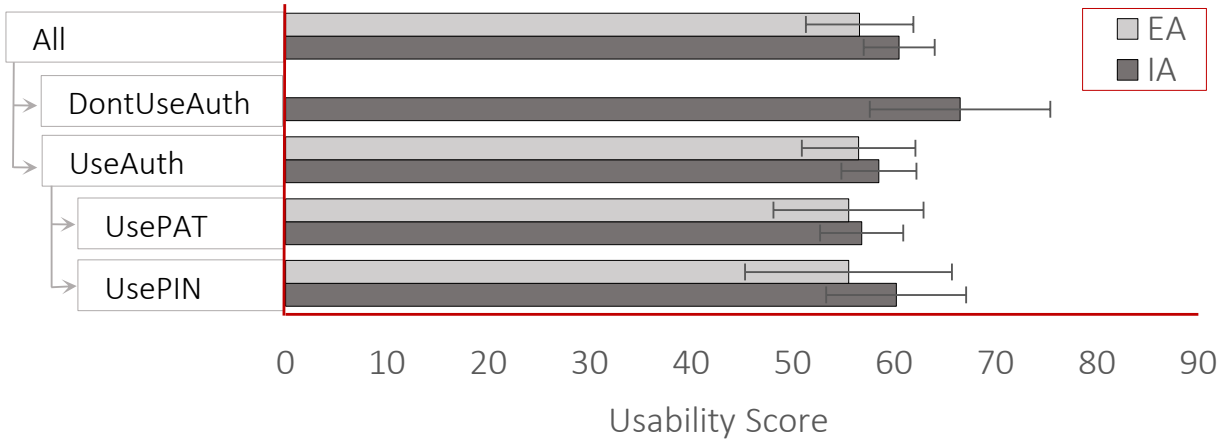


Figure 4.10: Average system usability scale (SUS) score for IA and non-IA sessions (error bars represent 95% confidence intervals).

U4: Overall usability of IA

The SUS scores for IA and EA are provided in Figure 4.10. Since we used nine questions from SUS, we report the usability scores out of a total of 90. A higher SUS score indicates that a system is more usable. A score is unavailable for the non-IA session of the *DontUseAuth* participants because they did not authenticate in that session. A t-test does not indicate any significant differences for the SUS scores between EA and IA (t = *-2.4, p = 0.31*). Similarly, a t-test does not indicate any significant differences between *DontUseAuth* and *UseAuth* participants for the IA session (t = *4.7, p = 0.1*).

An increase in anxiety of users may result in usability issues. To investigate the effect of the FRs on anxiety of participants, we calculate the change in the STAI anxiety score after each lab session. A comparison of the change in the STAI anxiety score between IA and non-IA sessions is provided in Figure 4.11. A negative change in the STAI anxiety score indicates the session did not increase the anxiety level of the participant and vice versa. A pair-wise t-test does not indicate any changes in anxiety across the participants for the IA and the non-IA session (t = *10.6, p = 0.82*).

Discussion: IA did not outperform EA on SUS and we do not observe any significant differences between the STAI anxiety scores. However, there are interesting differences for the individual SUS questions between IA and EA. Figure 4.12(a) shows significantly more participants (26% more) thought that IA was easier to use compared to EA *($\chi^2(1) > 100$, $p < 0.001$)*. Supporting comments for the ease of use:

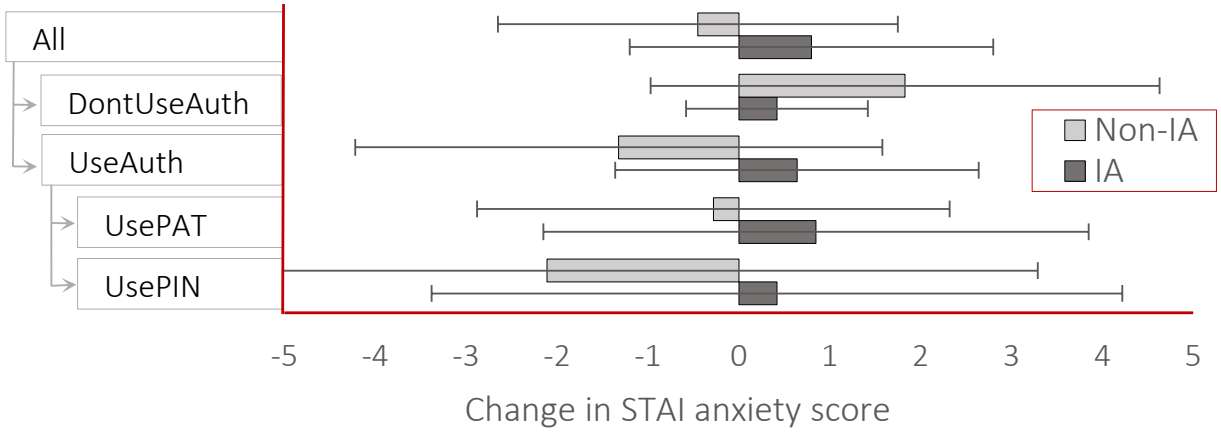


Figure 4.11: Average change in STAI anxiety score for IA and non-IA sessions (error bars represent 95% confidence intervals).

> *"I like that it really can be easy if your phone is effective at recognizing you and doesn't ask you to enter the password"* (P21)

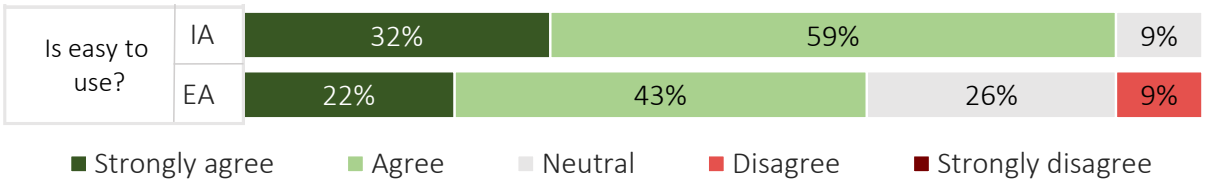
> *"Actually I found it pretty easy to use and it wasn't bothersome"* (P21)

As would be expected, significantly more participants (24% more) thought that IA was more inconsistent than EA $(\chi^2(1) = 64,\ p < 0.001)$ (Figure 4.12(b)). Furthermore, significantly more participants (10% more) thought that they need to learn more about IA $(\chi^2(1) = 15,\ p < 0.001)$. In Section 4.3.2, we discuss the learnability issue in detail.

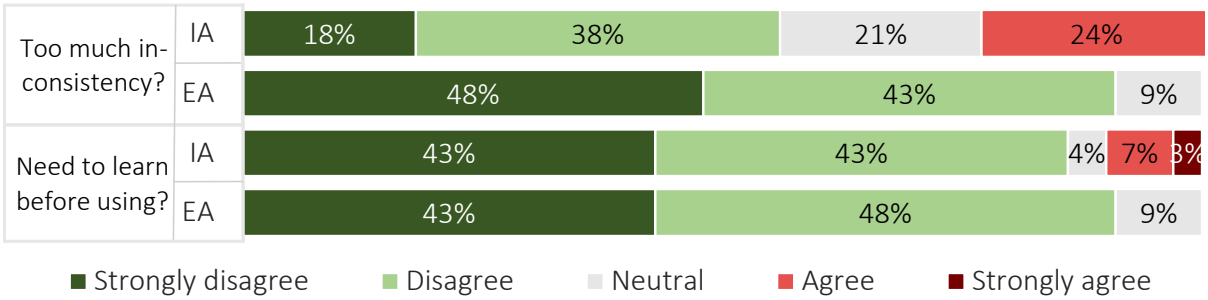
4.3.2 Security perceptions of IA

S1: Perceptions of IA security properties

Participants were made aware of the security properties of IA including FAs, FRs and the detection delay through the briefing video and the lab-based experiment. We then asked participants how satisfied they were with the overall level of protection that was provided by IA (see Figure 4.13). Overall, 81% of the participants were satisfied (22% very satisfied and 59% were satisfied) with IA, 8% were not satisfied and the rest were undecided. *DontUseAuth* participants were significantly more (12% more) likely to be satisfied with the level of protection that was provided by IA as compared to *UseAuth*



(a) Usability issue for which the participants favored IA



(b) Usability issues for which the participants favored EA

Figure 4.12: Responses to the individual SUS questions.

participants ($\chi^2(1) = 9.5$, $p = 0.001$). Two *UsePIN* participants were not satisfied while three *UsePAT* participants were undecided about the overall level of protection that was provided by IA.

We also asked participants regarding their perceived level of protection against different adversaries, device states and tasks (see Figure 4.14). Overall, 12%, 6%, 3% and 15% of the participants were not satisfied with the level of protection that was provided against coworkers, spouse, friends and strangers, respectively. In terms of different device states, 21%, 6% and 3% of the participants were not satisfied with the provided level of protection if their device was lost in a public location, unattended at work and unattended at home, respectively. Finally, 33%, 12% and 9% of the participants were not satisfied if IA was protecting their device while there was a banking app, email app and photo gallery app on their device, respectively.

Discussion: Overall, we found that participants were satisfied with the level of security that was provided by IA. However, 18/37 participants showed some concerns regarding FAs, detection delays and possible mimicry attacks in IA. We now shed some light on the concerns based on the participants' comments. The non-zero FA rate was a concern for 8/37 participants:

"I'm not sure what will happen when it is lost. It will depend on who picks it

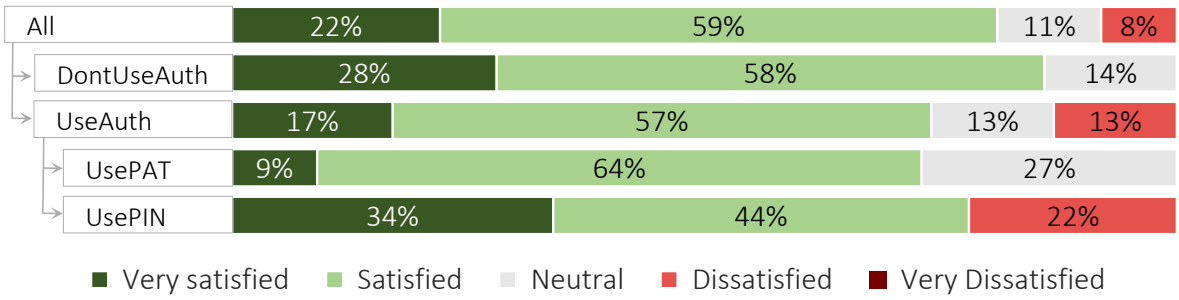


Figure 4.13: Responses for "How satisfied are you with the overall level of protection that is provided?".

> *up and I may get unlucky if his behaviour is the same as mine"* (P30)
>
> *"No one can use the phone without entering the PIN but here someone 'can' use it"* (P25)
>
> *"Other folks might use the phone the same way you do and you won't know"* (P21)

10/37 participants were displeased with the delay in detection and expressed concerns that an adversary may have enough time to look at private data before a TR:

> *"You know for some of these reading activities, I was not authenticated until I pressed somewhere on the screen... can't imagine what would happen if these were my messages or emails"* (P37)
>
> *"Like I mentioned before, you can see the screen right away before it can lock you out and that will make it a little unsettling for me"* (P29)

4/37 participants were apprehensive about the feasibility of a mimicry attack. Their concern was that someone (in particular their friends and family members) might be able to observe their behaviour and then defeat IA:

> *"I think implicit provides a better level of authentication initially but if someone spends too much time with me they might be able to learn it"* (P7)
>
> *"One thing that I noticed about implicit authentication is and that is my impression of it is that if someone watches you for a longer time then he might be able to bypass it"* (P31)
>
> *"Maybe there is a way for them to be aware of your pattern"* (P26)

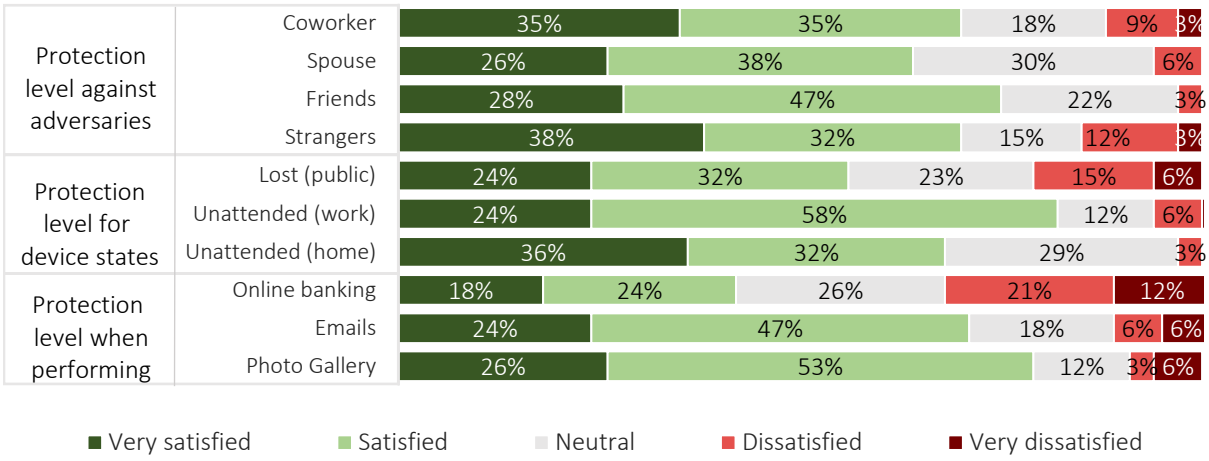


Figure 4.14: Security perception responses according to different adversaries, device states and tasks.

S2: Perceptions of IA security vs. current method

We asked participants how secure they thought IA was compared to the authentication method that they currently used (see Figure 4.15). All *DontUseAuth* participants perceived IA to be more secure and 87% of *UseAuth* participants thought that IA was at least as secure as their current method or more secure. Only 13% of *UseAuth* participants perceived IA to be less secure due to the security concerns discussed in the previous section.

Discussion: Clarke et al. [CKF09] and Crawford and Renaud [CR14] found that 92% and 73% of their participants considered IA to be more secure as compared to the traditional authentication schemes, respectively. On the other hand, only 48% of our *UseAuth* participants thought that IA was more secure as compared to their authentication schemes. Our results are not consistent with the previous findings. In the original papers, Clarke et al. [CKF09] and Crawford and Renaud [CR14] do not mention briefing the participants regarding the IA limitations. We suspect that this difference in results is due to the increased knowledge of our participants about the limitations of IA.

S3: Willingness to Adopt IA

We asked participants how willing they were to use IA with four choices: (i) Yes, I would replace my current scheme with IA; (ii) Yes, I would use it in addition to my current authentication scheme; (iii) I may use it; and (iv) No, I will not use it. The purpose of introducing

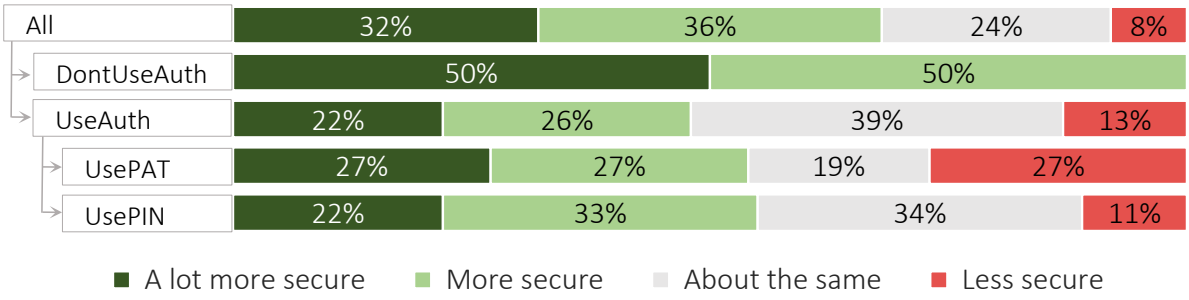


Figure 4.15: Responses for "How secure is this method as compared to your current authentication method?".

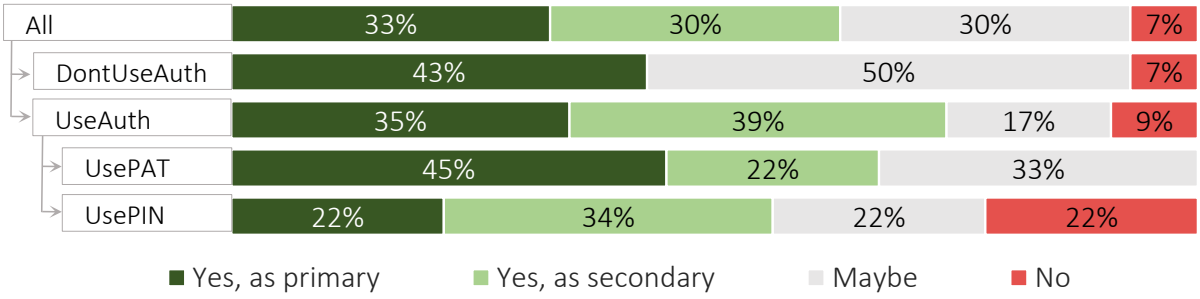


Figure 4.16: Responses for "Would you use IA?".

a spectrum of answers was to understand the various types of authentication needs that IA might be able to satisfy. The response of participants is provided in Figure 4.16. Overall 63% of participants were interested in using IA either as a primary (33%) or a secondary (30%) authentication mechanism. On the other hand, 30% of participants were not sure whether they would use IA and 7% did not want to use IA. A further breakdown across authentication preferences indicates that *DontUseAuth* participants were significantly more likely (37% more) to be unsure about IA adoption as compared to *UseAuth* participants $(\chi^2(1) = 19,\ p < 0.001)$. Interestingly, we found 4/7 participants who were not adequately satisfied with IA's level of protection (S2) were still interested in using it.

Discussion: Although 63% of our participants were willing to use IA as a primary or secondary authentication method, this is less than the 90% willingness to adopt IA that Crawford and Renaud [CR14] found. This difference is likely due to Crawford and Renaud providing a binary yes-or-no option rather than the spectrum of answers we provide. The interview provides rationale for the participants' choices. 6/14 *DontUseAuth* participants

were interested in adopting IA because they thought that it provided convenience and protection which was better than no authentication:

> *"It seemed easier than entering a password and more secure than not using anything"* (P30)

> *"Instead of forcing me to enter the password every time, it offers me not to enter a password which is my current preferred level of security and it provides additional security on top"* (P14)

On the other hand, seven *DontUseAuth* participants who said that they may use IA wanted to test it before making up their minds, for example:

> *"I would give it a shot for a month and if I see that it is getting a lot better I will like using it."* (P12)

> *"I have only used it once so I am not sure. I will have to use it for a longer duration and would like to test it on other people too"* (P7)

One *DontUseAuth* participant who did not choose to use IA did so because she had nothing to protect on her device:

> *"If I had work related data or anything else on my device that needed protection, I would use it. Right now I don't have anything that needs protection"* (P36)

UseAuth participants who chose to replace their EA scheme with IA did so because they felt it was more convenient: saves time (6/8) or has fewer authentications (2/8). They seemed to understand the associated risks but thought the trade-off was reasonable:

> *"Because in past months I never had a non-approved access to my phone and in the seven months I have entered my PIN thousands of times and it will be less annoying to use IA"* (P11)

> *"Even with the disadvantages [perceived low level of protection], I think I like the less number of authentications, given that I carry my phone with me"* (P8)

UseAuth participants who said they would use IA as a secondary authentication did so to have an additional layer of security (4/9) or to test it further (5/9):

"I would be interested since I work at the community center and at times I have to leave it at places and then I have to worry about it." (P6)

"It would be beneficial for spouse because they would know your password by asking it or see you type a lot but they won't know your pattern [behaviour]" (P14)

"I think just because of my unfamiliarity with it, passwords, I am accustomed with it, but perhaps the more I use it, the more I will trust it" (P32)

"I am so used to typing something in that I think it will take me a while to feel comfortable that authentication has occurred as oppose to you know when you turn it on and enter it" (P19)

"I feel like, like I said before, I would use it in parallel for a while and if it works I will use it after the trip period" (P13)

Finally, *UseAuth* participants who said no to using IA had concerns related to its detection delay (1/2) or they felt that EA was more suitable for them (1/2):

"Not the best for adoption because people would start looking at my photo gallery before it would lock them out" (P16)

"I think the current system that I have is enough to deter strangers and for the cases when the phone gets stolen" (P28)

4.4 Discussion

Our results suggest barriers for IA adoption and deployment along with associated design implications.

4.4.1 Mitigating the effects of interruptions

Overall, participants ranked IA higher than EA in terms of ease of use and the majority of participants reported that IA and interrupt-authenticates were at least tolerable. However, interrupt-authenticates did increase task completion time and several participants, especially those already using EA, felt the interruptions were annoying. Their comments suggest this was largely due to the unpredictability of interrupt-authenticates and the context-switch. Mitigating the negative effects of these necessary interrupt-authenticates remains a challenge for IA. Two participants suggested authentication without interruption by using the front camera of the device to perform facial-recognition. Similarly, a participant who was interested in using IA as a secondary mechanism to protect from misuse by friends or family members suggested IA took a picture of the perpetrator and email it to her. Two participants also suggested that instead of instantly locking the user out, the IA scheme could display an authentication screen in a smaller window on the screen and allow the user to choose the best moment to context-switch and authenticate:

> *"PINs were really annoying a lot of times. I would forget what I was about to say. I was wondering if there was a mechanism where it could indicate that it was locking on me in three, two, one and show me a small screen on the side to authenticate in parallel."* (P8)

It is important to understand that interruptions serve a purpose beyond authentication. Supporting the findings of Crawford and Renaud [CR14], some "annoyed" participants indicated that while interrupt-authenticates were annoying, they were necessary to indicate that IA was working and arguably contribute to a perceived sense of security:

> *"I guess it was frustrating that it kicked me out, but I could deal with that... and if it did ask for the PIN just knowing that the phone will be secure, it comes down to that"* (P22)

> *"Yeah the interruptions are annoying and I guess then I have to say to myself, practically it is not good but as soon as I see it, I know that it is protecting me"* (P5)

Balancing the need for interruptions with potential annoyance is a design challenge. The alternate authentication methods discussed above could be one approach, but the visual design of the authentication screen (e.g., choice of colour and language) as well as the timing of the interruption (e.g., postponing for non-sensitive tasks, slow fade in) are critical choices. Agarwal et al. [AKH16] have investigated some of these alternate authentication methods in their recent work.

4.4.2 Opaque deployment of IA

Our IA deployment was hidden as a background service, so participants were essentially unaware of its operation until a FR interrupt-authenticate occurred. While these interruptions currently serve to indicate that IA is working (see above), as IA detection algorithms evolve and FR rates decrease, these interruptions will become very infrequent and thus IA will be more opaque. Furthermore, operating at a relaxed operating threshold also reduces the number of FRs and the consequent visibility of IA. The background operations (opacity) of IA will raise concerns like:

> *"So looking at it... I see there is no lock. Sometimes I felt... the lock exists or not? When PIN is used, I know [it] every time. Now there is no way to discriminate if someone has hacked my phone and removed the lock. PIN is a secure feeling that the phone is safe"* (P31)

> *"The main problem for me is that if I am unaware that what I am doing is authenticated then I don't know if my device is secure. With this [IA] there really is no way of knowing that I have been authenticated when I open an app... How do I know it is not a fluke? Maybe it is no longer running and protecting, how do I know?"* (P19)

The concerns of users regarding the background deployment of IA have never been raised in existing literature. Since these issues arise due to their inability to tell whether IA is protecting their device, simple UI changes may be able to address these. For example, an indicator on the status bar can be used to indicate the current status of IA scheme. While such an indicator can keep the users up-to-date and act as a deflector against potential adversaries, it may also notify adversaries and enable them to launch highly focused attacks to gain access to the target data before being locked out. For IA deployment, the design and control of an IA status indicator needs to be studied closely and respective trade-offs need more exploration. To this end, the existing related research on the security cues for SSL on browsers can be leveraged (see Clark and Oorschot [CvO13] and references therein).

4.4.3 Operating threshold customization

The in-situ feedback screen (see Figure 4.3) provided participants with an option to adjust the operating threshold during the field study, and we asked them about their experiences with this functionality. 17/34 participants indicated that they found the customization

capability to be useful. A common explanation was that they reduced the operating threshold when texting or when at home. 5/34 participants indicated that they always set it to high to get maximum protection (4/5 belonged to *DontUseAuth*). 12/34 participants never adjusted the threshold during the field study and relied on the value chosen by the IA scheme.

These specific results have some limitations since any operating threshold customization in our app was temporary — the threshold was set to a predetermined value each day of the three-day field study. Nevertheless, participant comments indicate that there is a need to explore the various customization options for IA (such as the trade-off between FA and FR; and the detection delay and FR). For example:

> *"I think threshold selection bar would be a useful function. I feel having that is more choice and useful."* (P21)

The threshold customization interface needs to communicate the security and annoyance trade-off for the chosen threshold so users can make informed decisions.

4.5 Limitations

Our study has reasonable limitations due to the inclusion of human subjects: the scope is limited to people willing to participate; it contains self-reported and subjective views; participants might be inclined to provide favorable responses to the researchers; and the known limited duration of the field study might have made participants more optimistic about their annoyance. Since these are not easily preventable, we focus on limitations specific to our study:

1. We use a pseudo-IA scheme to strictly control FR rates and to circumvent restrictions on Android event data collection. As a result, it was possible for participants to witness unexpected behaviour of IA (for instance, they may get a FR or a TA for what they felt was the exact same sequence of touch input).

2. In the field study, all participants of the *UseAuth* group evaluated IA as a secondary authentication scheme (including those who indicated they would replace their EA scheme with IA). This resulted in multiple authentications during a single session (the system EA first, then an IA interrupt-authenticates), which may have contributed to feelings of annoyance. However, the deployment of IA as a primary authentication scheme in the field was not possible due to security and privacy issues.

3. When a FR occurred in the field study, participants had to authenticate and provide in-situ feedback. Although we designed the feedback pop-up to be simple to complete, it may have increased annoyance. Participants had the option to dismiss the authentication and the feedback pop-up in extreme situations. Although only 6% of the pop-ups were dismissed, but this may have slightly skewed results by underreporting annoyance.

4. In this study, we did not compare IA with biometric-based EA schemes, like fingerprint- or facial-recognition schemes. These alternative biometric-based authentication schemes have the same limitations of EA schemes (discussed in Chapter 2).

5. The pseudo-IA scheme may have authenticated a participant after at most 30 seconds of activity in some cases. This introduced a considerable lag between the input event and the corresponding FR. Since, we were unable to extract input events from the Android OS, this scenario was not preventable.

6. We report the security perceptions of participants regarding IA. We note that similar to participants' views regarding usability, their views on the perceived level of security are self-reported and subjective. The purpose of soliciting security perceptions was to understand whether IA satisfies the diverse security requirements of our participants (see Table 4.1).

Limitations 1 and 2 are attributed to the pseudo-IA scheme, but this is a reasonable trade-off for the advantages of using pseudo-IA for strict control of FR rates and elimination of confounds from performance idiosyncrasies of specific IA algorithms. Limitation 3 is a trade-off for benefits from gathering in-situ feedback. Limitation 4 must be considered in light of the fact that our study compares IA to no authentication and dominant forms of EA (PIN and pattern-lock): these arguably form lower and upper baselines for usability and security perception. For limitation 6, if users perceive IA as more secure than it is (or vice versa), further briefing is required to ensure that participants perceive the security properties of IA correctly. Furthermore, for enterprise deployment scenarios of IA, it would be better to ask IT managers for better informed perceptions.

4.6 Conclusion

Our two-part study on IA usability and security perceptions provides empirical evidence for the "human side" of IA. In terms of performance, the interrupt-authenticate model

may impose overhead for individual authentications, but it increases amortized task performance without affecting the error rate. For usability perceptions, there is no significant difference between IA and EA for SUS and 26% more of our participants agreed that IA was more convenient than EA. However, annoyance is a potential issue with IA with 35% of the participants who found interrupt-authenticates annoying. For security perception, detection delay and FAs were issues for 27% and 22% participants respectively, and 11% of our participants were concerned about the feasibility of mimicry attacks. Yet, participants who currently use explicit authentication perceived IA to be more secure, or at least as secure as their current authentication method. Perhaps most encouraging is that 63% of our participants were interested in adopting IA and a further 30% were interested in trying IA out with possibility of adoption. Based on insights gained from post-study interviews, we propose design implications that may reduce annoyance and increase security perception even more.

Our findings provide supporting evidence for earlier work's [JSGC09] postulation: IA is indeed a meaningful approach with a reasonable trade-off in terms of usability and security. However, participants voiced their concerns regarding the possibility of mimicry attacks. In the following chapter, we further our discussion on a realistic threat model (from Section 3.4.7) and we subject touch IA to mimicry and offline training attacks.

Chapter 5

Mimicry Attacks on Touch IA

In Section 3.4.7, we discussed the possibility of mimicry attacks on IA schemes. We discussed the vulnerability of device usage IA schemes to mimicry attacks and reviewed the existing literature that presented successful mimicry attacks on gait IA [KPJ15], and text IA (on personal computers) [TGG13]. In Section 2.4.8, we also presented a generative algorithm based attack on touch IA, which used a robotic device and generic traits across touch input data [SP13b]. However, we noted: (i) the impracticality of their attack in a work or home environment since an attacker is required to carry a mechanical robot; and (ii) their generic attack failed for up to 40% of the victims because their touch behaviour was was different from the inferred generic behaviour.

In this chapter, we investigate practical attacks on touch IA. These attacks consider a more realistic threat model than the one considered in the conventional methodology of IA evaluation. The accuracy evaluation in existing touch IA proposals follows a conventional methodology for classifier evaluation. Touch data is collected from users and a classifier is trained for each user by employing a subset of their data as positive training samples and other users' data as negative samples. To calculate accuracy statistics, the remaining data from other users is used as synthetic attack data. This methodology is followed in the existing literature and we followed the same during the evaluation of IA schemes in Chapter 3.

Using random attackers who have no knowledge of their victims' behaviour fails to capture more realistic attack scenarios such as shoulder surfing and offline training where attackers have access to their victims' raw touch data. Only Bo et al. [BZL+13] and Frank et al. [FBM+13] acknowledge their evaluations omit these realistic attack scenarios. Both argue adversaries could not learn invisible features such as touch pressure and swipe

acceleration only by shoulder surfing, and Frank et al. [FBM+13] rule out attacks using a victim's raw touch data because malware is needed to gather the data. These arguments have been considered *"sound without doubt"* by others [SP13b] and to the best of our knowledge, these attacks have never been evaluated.

We argue that shoulder surfing and offline training are realistic for malicious insiders like friends, family, and colleagues — insiders that are recognized threats [MBK+13]. Unlike random attackers, malicious insiders are able to observe their victims' behaviour, giving them an advantage for shoulder surfing attacks. Moreover, an attacker may launch sophisticated mimicry attacks after gathering the victim's raw touch data by asking that victim to perform a task on the attacker's device. This eliminates the need for malware or sophisticated logging. Our evaluation of the security perceptions of IA indicated that shoulder surfing attacks by insiders was a concern for potential early adopters of IA (see Section 4.3.2). However, previous work provides no evidence that touch IA protects against these targeted mimicry attack scenarios. We expand our third research objective (see *Objective 3* in Section 1.5) to cover shoulder surfing and offline training attacks for the threat scenarios where an adversary is able to observe or obtain raw touch data of their victim. We begin our realistic evaluation of these attacks by outlining the threat model and a detailed description of the attack scenarios.

5.1 Threat Model and Attack Scenarios

For touch IA, we use the standard threat model used for IA (outlined in Section 1.3) [BZL+13, LZX13, XZL14]. An adversary attempts to gain unauthorized access to a victim's device, which employs a touch IA scheme to continuously authenticate the device user. The victim has either not configured a primary authentication scheme (such as a PIN) or the adversary has bypassed it completely through known mechanisms like shoulder surfing or smudge attacks [AGM+10]. Furthermore, the adversary is aware of the presence of a touch IA scheme on the victim's device.

Accuracy numbers reported in the IA literature evaluate this threat model against a *random attacker model*. This means that data from random attackers with no knowledge of their victims' behaviour is used as synthetic attack data. While this tests scenarios where attackers have possession of a stranger's device, it does not cover attacks by malicious insiders seeking to mimic their victim's behaviour. Smartphone users are concerned about insider threats from friends, family members, and colleagues [MBK+13]. IA evaluations should also consider this threat. We evaluate two malicious insider mimicry attacks: shoulder surfing and offline training.

Shoulder surfing attacks: Malicious insiders may observe their victims' interactions. It is impractical for users to conceal all touch input behaviour by shielding the device screen or holding the device at extreme angles. While the "invisible nature of features" argument [BZL+13, FBM+13] is true for a subset of features such as touch pressure, it may not hold for features such as touch location and swipe duration. Thus, adversaries may attempt to mimic these observable features. It is unclear whether mimicking observable features by shoulder surfing provides an advantage.

Offline training attacks: Malicious insiders can gain access to the raw touch data of their victims through various techniques. Since any foreground app is able to capture touch input events, malicious insiders can recommend an instrumented app from the official app store to their victims, which in turn collects and transmits raw touch data to the malicious insiders. Malicious insiders may ask their victims to visit a webpage where HTML5 `TouchEvents` are used to skim raw touch data. TapPrints uses similar methods to infer tap locations and corresponding keystrokes by sensing the accelerometer and gyroscope sensors in the background [MVBC12]. Finally, malicious insiders have the convenient option of obtaining the raw touch data of their victims by asking them to perform a task (e.g., read an article or view photos) on the insiders' device. This eliminates the need to install or access anything on the victims' devices. Once the insiders gain access to the raw touch data, they can use it to train and mimic their victims' behaviour.

5.2 Schemes and Data Evaluated

In this section, we first provide a brief overview of the three touch IA schemes that we evaluate against targeted attacks. We also describe our data collection setup which consisted of raw touch data and video recordings of "victims." Raw touch data is used to train IA classifiers and train attackers for offline mimicry attacks. Video recordings are used for shoulder surfing mimicry attacks. Note that we received approval from our university's ORE for all experiments involving human participants.

5.2.1 Schemes evaluated

We evaluate the Touchalytics [FBM+13], LXG [LZX13] and SilentSense [BZL+13] schemes. Touchalytics and SilentSense were described in Section 3.1.3 and Section 3.1.5, respectively. We chose Touchalytics because in addition to its low EER, to the best of our knowledge, it captures touch input behaviour using the most extensive feature set. We chose SilentSense

due to its low EER and its use of micro-movement features in addition to the touch features. A brief description of LXG follows.

LXG [LZX13]

LXG[1] derives following features from a swipe gesture: (1) coordinates of the first touch point; (2) touch area at the first touch point; (3) moving direction at the first touch point; (4) moving distance; (5) duration; (6) average moving direction; (7) average moving curvature; (8) average touch area; and (9) max-area portion. The authors also evaluate the tap gesture but they propose using it only as an auxiliary gesture due to its high EER. The authors evaluate LXG with the SVM classifier and they create separate training models for each of the four swipe directions. Their evaluation on a dataset of 75 participants indicates that LXG provides an EER of 8% with a window of eight swipes. We selected LXG because its small feature set complemented our selection of Touchalytics and enabled us to evaluate the impact of feature set size on the training effort of the attackers.

5.2.2 Data collection

We implemented two Android apps to collect raw touch data using the same tasks as Touchalytics [FBM+13]: a Wikipedia app collects up and down swipes while users read articles of their choice; and a "spot the differences" app collects left and right swipes while users navigate between two slightly different illustrations. Each participant used these apps on a LG Nexus 5 device while in our lab. No directions were given beyond explaining the basic tasks of reading articles and finding differences. Each participant interacted until at least 150 swipes in each direction were logged, the minimum number of training samples required by the IA schemes.

Logged data: For every touch interaction, we recorded: time stamp in milliseconds; touch point x and y coordinates; touch pressure; area covered by the finger on the screen; finger orientation; screen orientation; rotation values from the gyroscope sensor across three axes; and acceleration values from the accelerometer sensor across three axes. Accelerometer and gyroscope data were collected in a separate thread up to 100 Hz.

Videos for shoulder surfing attacks: We captured video of nine participants while they used the data collection apps. At least ten swipes in each direction were captured in two views, above the device and from the side. Each had an unobstructed view of the

[1] LXG are the initials of the last names of the authors.

Table 5.1: Statistics of collected data (n=55)

	Swipes	**Touch points**	**Accel. samples**	**Gyro. samples**
Total	34,889	1,138,199	2,863,809	2,553,233
Avg(Mdn) per user	528 (478)	17,245 (13,329)	43,391 (26,258)	38,685 (26,268)
Avg(Mdn) per swipe	-	32(16)	82(69)	73(62)

participant's finger on the touch screen. All videos were shot in 1080p format (1920x1080 pixels) with a frame rate of 29 FPS. The smartphone occupied 4–5% of the video frame. Given the open-ended task, the videos were between 23 and 44 seconds (avg 31 secs).

Data statistics: We recruited 55 participants (a subset of these also participated in the attacks experiment). On average, the participants took 26 minutes to submit data. In total, we logged about 35,000 swipes comprising over 1.1 million touch points, and over 2.5 million accelerometer and gyroscope sensor readings. Table 5.1 provides the details of collected data.

5.2.3 Parameter value selection

We fix two tunable parameters in our experiments, operating threshold and window size. Operating threshold defines the desired values for negatively correlated FA and FR entries. By increasing the operating threshold, FRs can be decreased at the cost of increased FAs (and conversely). Theoretically, at lower false accept rates (FAR), it should be difficult to launch successful mimicry attacks. Therefore, we set FAR for an arguably higher false reject rate (FRR) of 20% with corresponding FARs of 0.4% for Touchalytics, 4% for LXG, and 0.2% for SilentSense (see Section 5.2.4). The effect of the operating threshold is further investigated in Section 5.6.3. Window size defines the number of swipes used to calculate a user's authentication score. Larger window sizes increase confidence against classification scores at the cost of increased detection delay. We set the window size to eight swipes since the IA schemes we evaluate provide reasonable accuracy at eight swipes or fewer (see Section 2.4.5).

5.2.4 Evaluation baseline

To establish a baseline, we use our dataset to evaluate the three touch IA schemes against the random attacker model. We construct non-overlapping training and test sets for each user using negative instances from other users' data. Half of the data is used for training,

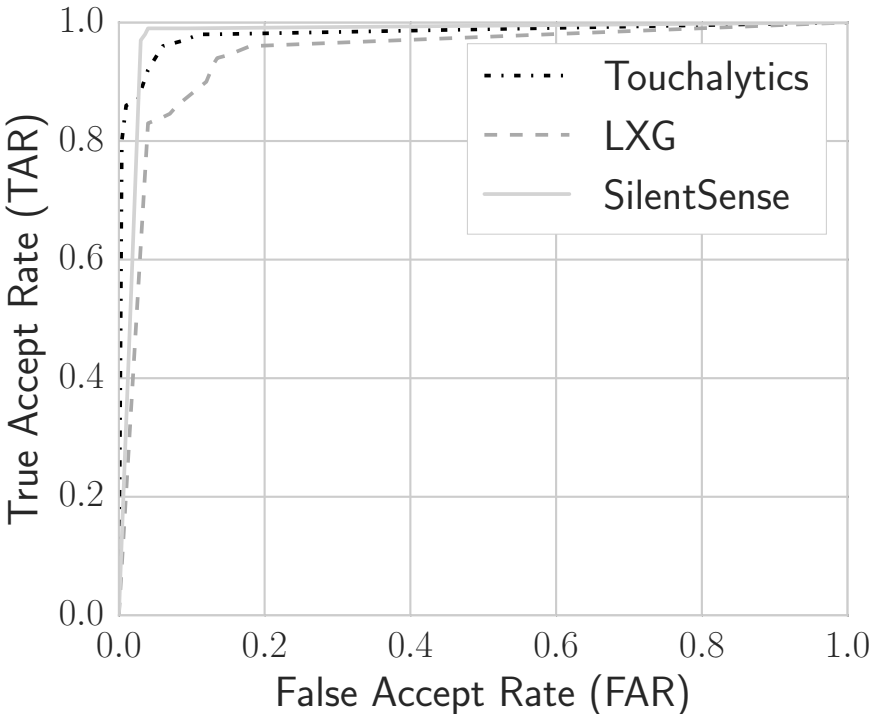


Figure 5.1: Accuracy of the IA schemes against the random attacker model with a window size of eight swipes.

and the remaining for testing. Figure 5.1 shows the ROC curves that plot true accept rate (TAR) against FAR using an SVM classifier. The ROC curves show an EER of 4% for Touchalytics, 9% for LXG, and 3% for SilentSense. These low EERs are very similar to the rates reported in the original papers and establish the efficacy of these schemes against the random attacker model.

5.3 Attack Design

In this section, we describe apps and tasks used for offline training attacks and for attack evaluation.

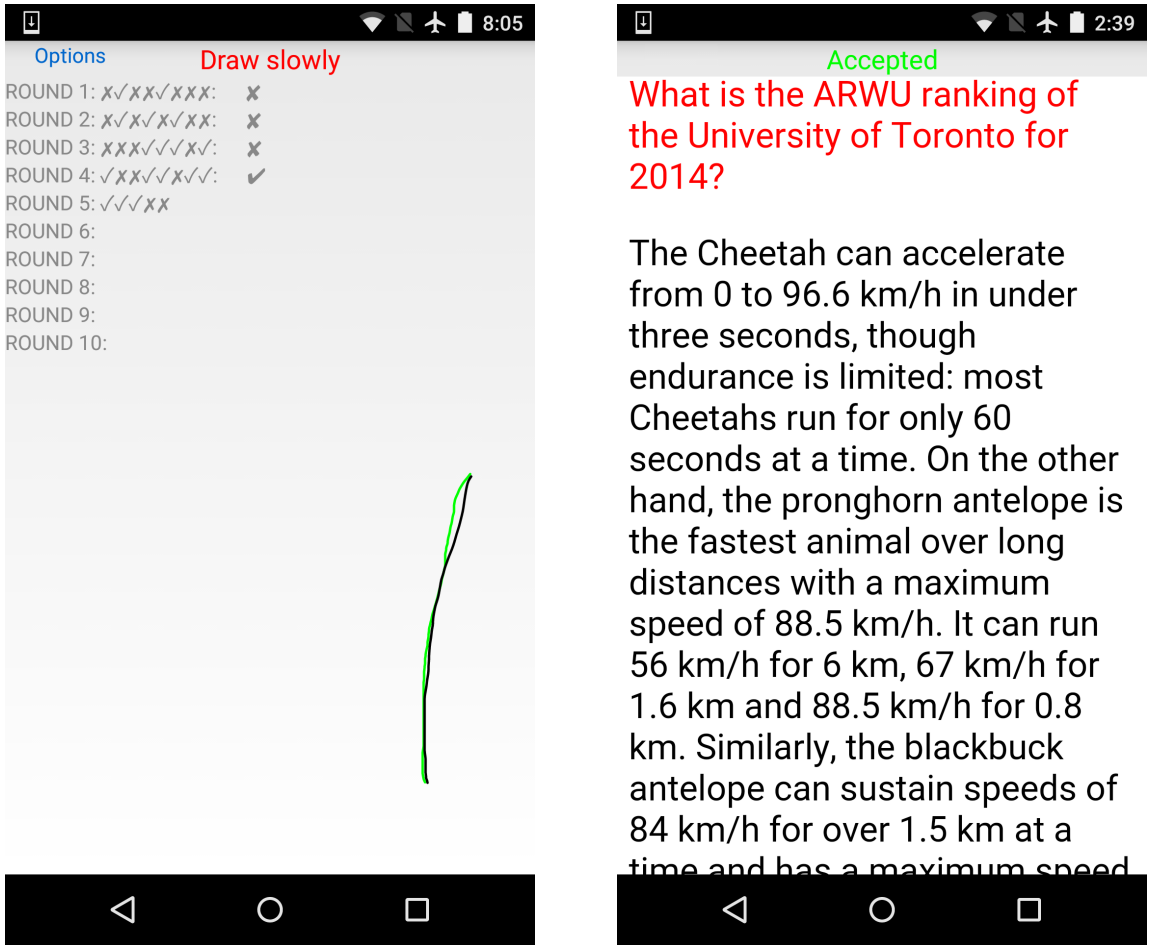
5.3.1 `Mimicker` for offline training attacks

Our `Mimicker` app trains an attacker to mimic a victim's behaviour using feedback and visualizations generated from that victim's raw touch data. The three main components of `Mimicker` are the target swipe selection module, the training interface, and the feedback module. The target swipe selection module chooses an optimal *target swipe* from actual victim swipes. The training interface displays the target swipe so the attacker can swipe along a similar path (the *mimic swipe*). If the mimic swipe is rejected by the IA scheme, the feedback module displays instructions about a single behavioural aspect to bring the mimic swipe closer to the target swipe (e.g., move start point towards right). The attacker continues adjusting their swipe based on these instructions until their mimic swipe is accepted.

Target swipe selection: Any victim's swipe classified as a true accept can be used as a target swipe. However, our goal is to present the attackers with an optimal swipe to increase their chances of success. We do this by selecting a victim's true accepted swipe with the highest similarity score with the rest of the victim's true accepted swipes. This simple heuristic provides an advantage to the attackers since their mimicry attempts can focus on the hypothesis space with the maximum concentration of true accepted swipes (i.e., the victim's most typical swipes). The target swipe selection module accomplishes this as follows: It creates a dataset, D, with positive samples from the victim's swipes and negative samples from other users' swipes. It uses the kNN classifier ($k = 5$) to find a subset of true accept swipes, S_{TA}, from D. For $n = |S_{TA}|$, it identifies the n nearest neighbours from D for every swipe that belongs to S_{TA}. The similarity score is the number of swipes in the n nearest neighbours that belong to S_{TA}. The similarity score of each swipe is recorded in the similarity rank table and the swipe with the highest similarity score is returned as the target swipe. We also evaluate using non-optimal target swipes in Section 5.6.4.

Training interface: Figure 5.2(a) provides a screenshot of the `Mimicker` user interface. The target swipe is displayed in green and the last mimic swipe is displayed in black. If the mimicry attempt is successful, "Accepted" is displayed at the top in green, otherwise the instruction from the feedback module is displayed in red. In the upper-left, the success or failure of recent attempts are displayed at swipe-level (using ✓and ✗) and window-level (using ✔and ✖). The success criteria for window-level are defined in Section 5.4.

Feedback module: The feedback module is responsible for comparing the mimic swipe with the target swipe to generate feedback for the attackers. The most obvious way to achieve this is to give feedback based on the classifier feature with the largest difference between target swipe and mimic swipe. However, this would not consider that



(a) Tracing phase

(b) Pseudo-attack phase

Figure 5.2: Screenshot of `Mimicker` interface.

some features are easier to mimic (e.g., first touch location is easier to mimic compared to midpoint velocity). To address this, the feedback module selects the first feature with absolute difference greater than a threshold from feature categories ordered by ease of adjustment: swipe location, swipe length, swipe duration, touch pressure, touch area, device rotation, device vibration, and swipe curvature. The feedback module focuses on features that are more adjustable for attackers and can be visualized or explained in simple instructions. Some features, like 50%-percentile pairwise velocity and median velocity at the last three points are hard to adjust for and difficult to comprehend. We emphasize this

means our bypass success rates form a lower bound by focusing on low-effort attackers.

5.3.2 Apparatus for attack evaluation

Personal data on smartphones can be broadly categorized as textual (e.g., emails, texts) and multimedia (e.g., images, videos). Our attack tasks reflect these categories. To further simulate a realistic mimicry attack scenario, the attacker has to multi-task by searching for interesting data while mimicking the victim. We introduce two tasks that capture the multi-tasking nature of real-world mimicry attacks. For attacks on textual data, the attacker is presented with a browser-like interface with a collection of paragraphs from Wikipedia where each paragraph discusses a different topic. A question precedes the paragraphs and the attacker has to swipe up or down to find the paragraph that contains the answer to the question and then find the answer within that paragraph. For multimedia data, the attacker is provided with several feline images along with a numeric label for each image in an image viewer app. The attacker is then provided with a description of a feline (e.g., a white kitten) and is asked to swipe left or right to report the numeric label of the image that matches the description. While the attacker has to tap the target app icon to launch it, we do not consider the tap gesture since it provides too few features to be discriminative and has a high EER [FBM+13, LZX13].

At launch time, the apps are trained using the SVM classifier on the victim's training model constructed using positive samples from the victim and negative samples from other users. The apps provide no feedback for individual swipes; however, in case of a reject, the apps display a popup to inform the attackers of their failure (simulating the point when an explicit authentication method should appear in a deployed IA scheme). If the attackers are successful, the apps allow them to complete the task. Finally, the apps record the raw touch, accelerometer and gyroscope data of the attackers along with the result of their mimicry attempt.

5.4 Attack Protocol

We now describe the protocol used to conduct attacks. The attack protocol was shaped by a pilot study with three volunteers from our lab. Relevant results from the pilot with subsequent changes to the final protocol are noted where applicable.

5.4.1 Participant recruitment and motivation

We recruited participants to be attackers in September of 2015 through a university-wide mailing list and using Craigslist and Kijiji under the "other jobs" section. The title of the advertisement was "Participate in a research study on mimicry attacks on a novel authentication scheme for smartphones" and it stated adults who owned and used a smartphone for over six months could participate. Each participant completed a demographic survey and was invited to our lab for the study.

In a real attack, malicious adversaries are motivated to snoop the devices of their victims to find valuable information. For our experiments, we motivate participants to mount best effort attacks with performance-based monetary rewards. All participants were paid \$10, but they could earn another \$6 based on performance. If they mounted a successful mimicry attack on the chosen victim in their first attempt, they received \$0.75. If they mounted a successful attack on the second or subsequent attempt, they received \$0.50.

5.4.2 Study procedure

The procedure began with each participant submitting raw touch data using the collection apps described in Section 5.2.2. This data forms a baseline to measure adjustments made by the participant during the attacks. The experimenter then briefed the participant (using a script and visual aids) with an explanation of touch IA, comprehensible features of their target IA schemes, the apparatus, tasks, and the performance-based rewards. We investigate the scenarios where the attackers have limited knowledge of touch IA in Section 5.6.2.

Each participant mounted shoulder surfing and offline training attacks as explained below. Attack type order was counterbalanced across participants. For each attack type, the participant was assigned four victims to mimic up and left swipes. In the pilot study, each attacker was assigned only two victims and mimicked four swipe directions, but there were no significant differences in attack success. By reducing to up and left swipes, we could double the number of victims for each attacker. Testing more attacker-victim pairs is most relevant to our work, and up and left swipes are predominant directions for viewing new content. All eight victims assigned to an attacker were unique to avoid carryover effects.

The four victims for each attack type were split into two groups: two were protected using either Touchalytics or LXG, and two were protected using SilentSense. We decided

to assign either Touchalytics or LXG to make it easier for the attackers to remember their target IA schemes' features. The assignment of Touchalytics or LXG, and the order of the target IA schemes, were both counter balanced across the attackers. Attackers trained and mounted their attack on one victim swipe direction at a time using the assigned touch IA scheme and attack type. Swipe direction and the corresponding attack task order was counterbalanced.

Shoulder surfing attacks: The shoulder surfing attack had two parts: watching videos of the victim and attacking by mimicking the victim's swipes while completing the attack tasks (see Section 5.3.2). The attacker was shown the victim's shoulder surfing video clips on a 50" television. The attacker was not allowed to hold a device while watching the clips. They were informed about the camera angles and told they could watch the clips from either angle as many times as they wanted. Once the participant indicated they were prepared, the video was closed and they were given the device to mimic the victim's swipes while performing the attack task. Participants were told that if their attack failed, they could watch the clips again before mounting another attack. In Section 5.6.1, we evaluate a scenario where the attack occurs one week after shoulder surfing.

Offline training attacks: The offline training attack had two parts: training using the `Mimicker` app and attacking by mimicking the victim's swipes while completing an attack task. Training and attack were performed on two different LG Nexus 5 devices to simulate switching to a victim's device in a real attack.

Training was completed in two phases, a *tracing phase* (Figure 5.2(a)) with feedback and the target swipe and a *pseudo-attack phase* (Figure 5.2(b)) with only feedback overlaid on the attack task. The participant was informed that they had to bypass the IA scheme for two consecutive windows during each training phase before proceeding to the actual attack. If they were not successful, they had to continue the current training phase for at least ten windows before given another opportunity to bypass the IA scheme. During the attack, no feedback was provided. Attackers had to set the device down in between the tracing and pseudo-attack phases.

The pseudo-attack phase was introduced after the pilot study where 25% of the attacks failed despite successful completion of training. This appeared to be caused by the abrupt change between the tracing phase and the attack task: the attackers did not always memorize the location of the target swipe during training since `Mimicker` displayed the swipe; switching the device after training disrupted their device holding posture; and, unlike training, the attacks involved performing a task in addition to mimicking. The pseudo-attack phase increases training quality for attackers: we argue that a real attacker can leverage similar training mechanisms by approximating the task they plan to attack.

Table 5.2: Demographics of the participants (n=32).

Gender:	56% Male
	44% Female
Occupation:	31% Employed
	63% Grad student
	6% Undergrad student
Age group:	41% 18–25 years
	31% 26–30 years
	28% 31–35 years
IT experience:	53% Studied/worked in IT

For the purpose of our experiment, no feedback was provided for one window (*experiment window*) between the two phases to measure the efficacy of introducing the pseudo-attack phase.

5.5 Attack Evaluation

The study was completed by 32 participants (demographics provided in Table 5.2). In total, 512 attacks were logged (256 for each attack type), which were mounted by 256 unique attacker-victim pairs in up and left directions. We logged 3656 mimic swipes for shoulder surfing and 2984 mimic swipes for offline training attacks. During training, 17,064 swipes were logged.

5.5.1 Attacker success

We measure the efficacy of attacks on IA schemes through the bypass success rate at the victim-level and the TRR at the window-level. The bypass success rate is defined as the ratio of successful attacks to all attacks of a particular attack type or a particular direction.

Figure 5.3 shows bypass success rate against each IA scheme for both attack types across all attacker-victim pairs. For shoulder surfing attacks, 75%, 78%, and 92% of the attacks successfully bypassed against Touchalytics, LXG, and SilentSense, respectively. For offline training attacks, 81%, 82%, and 91% of the attacks successfully bypassed against Touchalytics, LXG, and SilentSense, respectively. An independent samples t-test for bypass success rates between shoulder surfing and offline training attacks for each scheme indicates

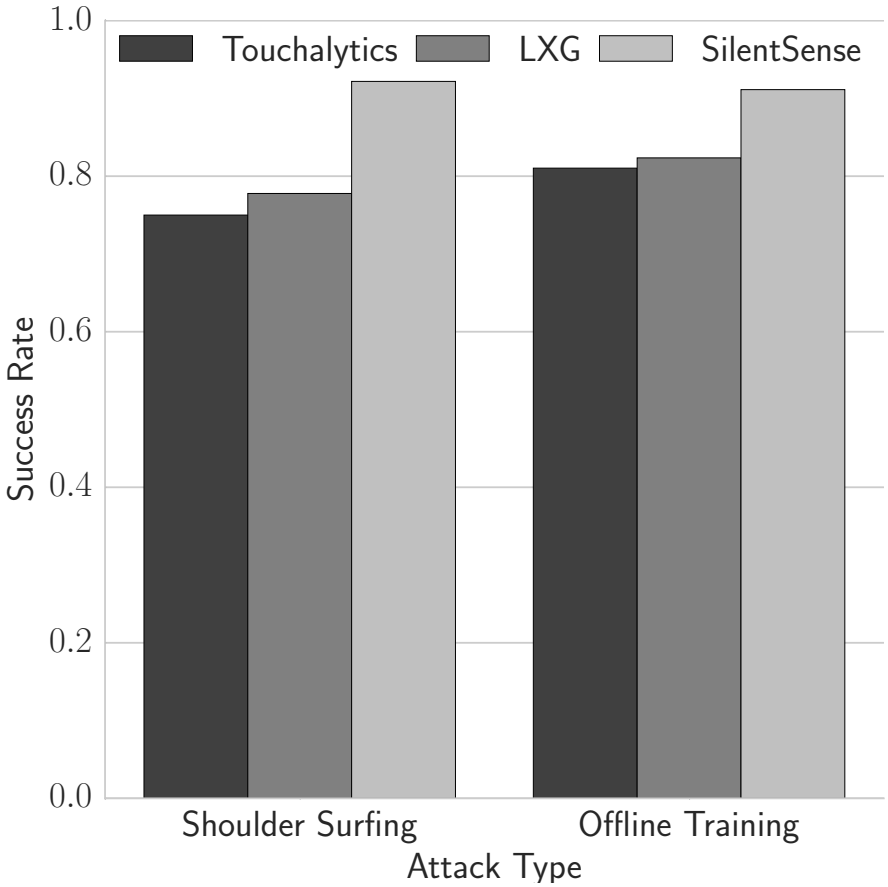


Figure 5.3: Bypass success rate for the three IA schemes across all attacker-victim pairs.

no significant differences. We note that the bypass success rate may be over-reported due to naturally occurring false accepts; however, for the random attacker model, our chosen operating threshold has negligible FAR. Compared to the random attacker model (Figure 5.1) at a FRR of 20%, we observe about 200x, 20x, and 450x increase in FAR for Touchalytics, LXG, and SilentSense, respectively.

To understand the performance of each scheme against mimicry attacks at the window-level, we calculate the average TRR across each attack window for all victim-attacker pairs for both attack types in Figure 5.4. Figure 5.4 shows that significantly lower TRRs are observed for offline training attacks for Touchalytics ($t = 2.12$, $p = 0.03$) and LXG ($t = 3.28$, $p = 0.001$). Lower TRRs are expected for offline training attacks because the attacks were mounted after the attackers received training. On the other hand, there are no

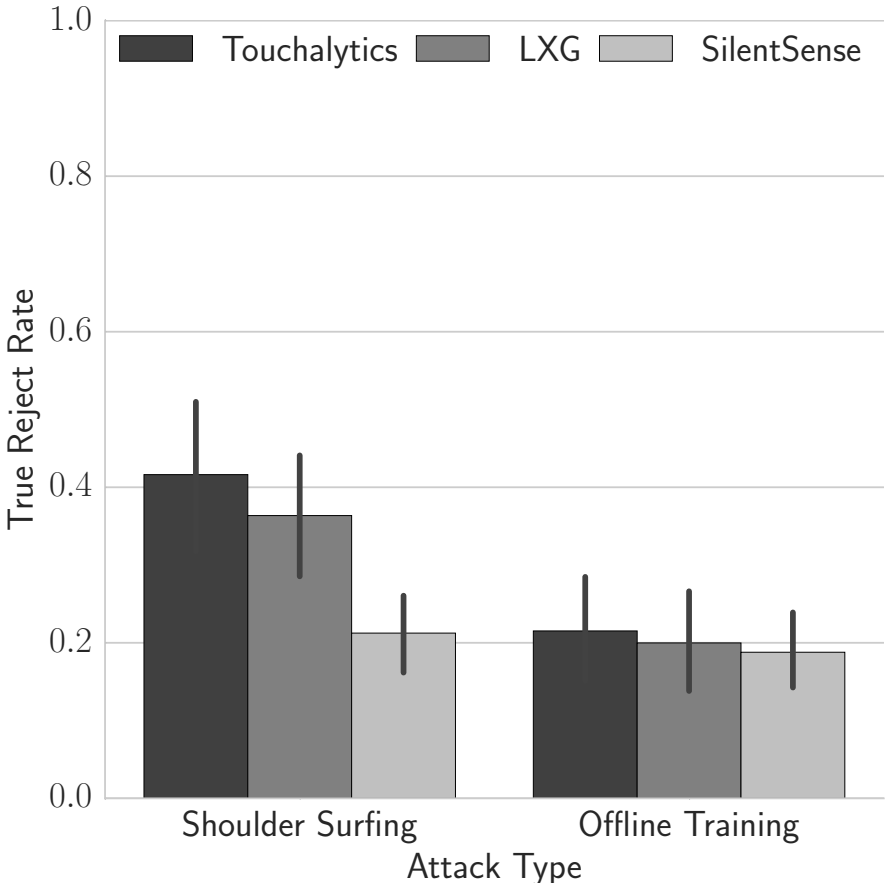


Figure 5.4: Average within window TRR for the three IA schemes across across all attacker-victim pairs. Error bars represent 95% confidence intervals.

significant differences between shoulder surfing and offline training attacks for SilentSense ($t = 0.56$, $p = 0.57$). Since the device reaction features of SilentSense rely on the device holding posture, the attackers were able to observe and mimic it during shoulder surfing. Consequently, the attackers performed better for SilentSense for shoulder surfing attacks when compared with the other schemes.

In terms of swipe direction, the bypass success rate for shoulder surfing attacks was 86% and 82% for up and left swipes. For offline training attacks, the bypass success rates were 85% and 87% for up and left swipes. An independent samples t-test indicates no significant differences between attack type for up *($t = -0.36$, $p = 0.71$)* and left swipes *($t = 1.03$, $p = 0.3$).*

Note that IA provides continuous authentication so attackers must continue mimicking a victim swipe after swipe. While our experiments require attackers to bypass their victim only once, after learning their victims' behaviour and bypassing IA for one window, they can mimic the behaviour on subsequent windows. This is shown in offline training, where attackers must bypass IA for three consecutive windows (two pseudo-attack windows and one attack window).

5.5.2 Attacker effort

To estimate attacker effort, we define three measures: shoulder surfing time (i.e., time spent viewing videos) captures attack preparation effort for shoulder shoulder surfing attacks; the number of windows used during training captures attack preparation effort for offline training attacks; and the number of windows until bypass (number of attempts required to mount a successful attack) captures attack execution effort for both types of attacks.

Shoulder surfing time: Figure 5.5 shows the cumulative distribution of time spent shoulder surfing before successful attacks. For 15% of successful attacks, the attackers were able to estimate the victim's behaviour by observing them only for half a minute. Similarly, 40% and 90% of the successful attackers required less than a minute and less than two minutes of shoulder surfing time. These results indicate attackers require trivial shoulder surfing time for successful attacks.

Number of training windows: Figure 5.6 shows the proportion of remaining attacker-victim pairs who still required training in a particular window and their distribution across successful (Win_S) and failed windows (Win_F). Recall that our protocol requires attackers to successfully mimic two consecutive windows in each phase. This means that 100% of all attackers will have at least two training windows in both phases in addition to the experiment window between the two phases. As an example for interpreting Figure 5.6, consider window 3 of the tracing phase. Here 33% of attacker-victim pairs remain because they were unsuccessful at mimicking the target swipe in both window 1 and window 2. Moreover 15% of attacker-victim pairs were successful at mimicking the target swipe while 19% failed in window 3.

Overall, 67% and 77% attacker-victim pairs only required two windows to complete the tracing and the pseudo-attack phase, respectively. A consistent decrease in the proportion of remaining attacker-victim pairs and Win_F from window 3 to window 7 in the tracing phase shows that `Mimicker` feedback was effective. However, after window 7, only 4% are able to successfully complete the training. Some attacker-victim pairs were unable to

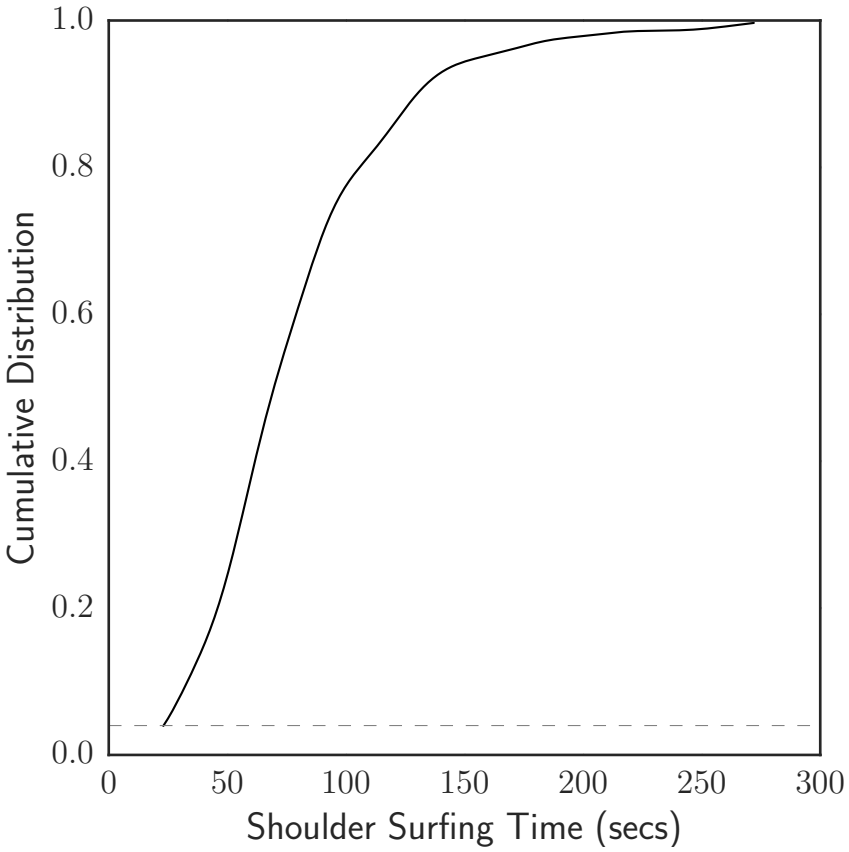


Figure 5.5: Shoulder surfing time for successful attacks.

complete training after ten windows: 11% did not proceed past the tracing phase and 4% did not proceed past the pseudo-attack phase.

Regarding the efficacy of our pseudo-attack training phase, observe that the special experiment window ('E') with only the attack task and no feedback has a 25% increase in Win_F. This window simulates the same abrupt jump from training to attack, and the result is the same as the 25% attack failure rate in the pilot. However, attacker-victim pairs corrected their behaviour when they received feedback: Win_F drops to 16% in Window 1 then to 10% in Window 2.

Number of windows until bypass: Figure 5.7 shows the cumulative distribution of the number of windows until bypass. This captures the attack execution effort in terms of number of attempts to successfully mount each type of attack. Recall attackers were

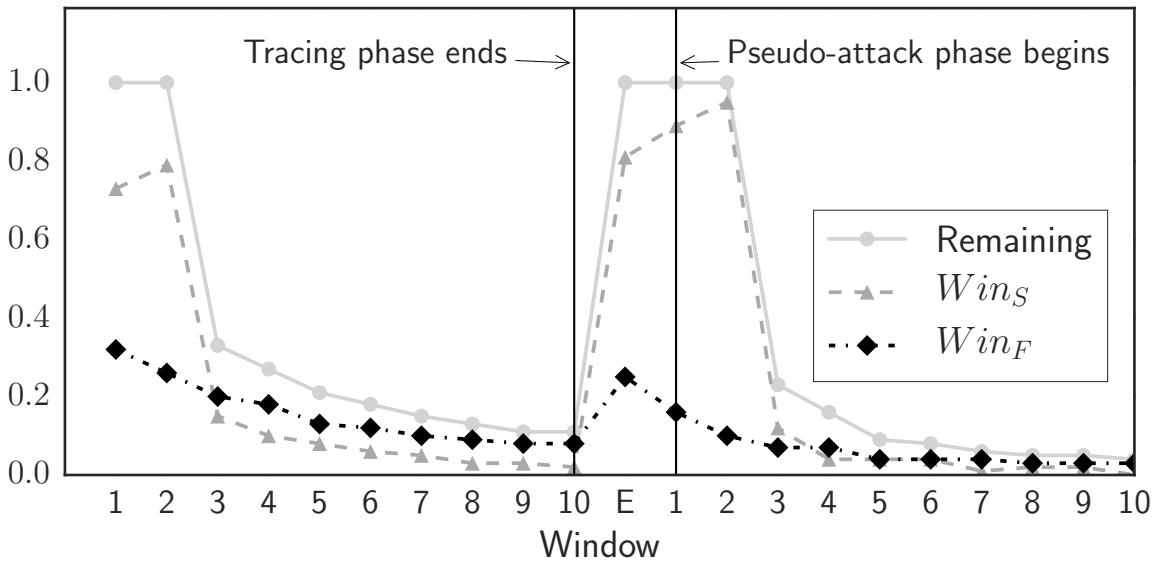


Figure 5.6: Proportion of attacker-victim pairs who required training for a window (Remaining) and their distribution across successful (Win_S) and failed (Win_F) windows. Window ('E') is the experimental window.

allowed to retry in case of a failed attack attempt and they could optionally shoulder surf or retrain before their next attempt. For the shoulder surfing attacks, about 73% of the successful attackers only required a single window and 93% required three windows or less to bypass the IA scheme. For offline training attacks, about 85% of the successful attackers bypassed the IA scheme in their first attempt and 90% required two attempts or less to gain access to the victim's device. There were no significant differences across the number of attempts to bypass for IA schemes or attack types.

Attacker effort and success: For shoulder surfing attacks, a Pearson correlation test indicates a slightly negative correlation between success rate of an attacker (the ratio of successful attacks among all attacks of a particular attack type or direction executed by this attacker) and the amount of time they shoulder surfed *(r = -0.07)*. We also observe a negative correlation between the success rate and the number of retries *(r = -0.19)* for the shoulder surfing attacks. These results indicate that the failed attempts are not due to the lack of effort by the attackers. A similar analysis cannot be performed for offline training attacks since `Mimicker` requires the attackers to attempt training for at least ten windows before giving up and only the attackers who successfully complete training mount attacks.

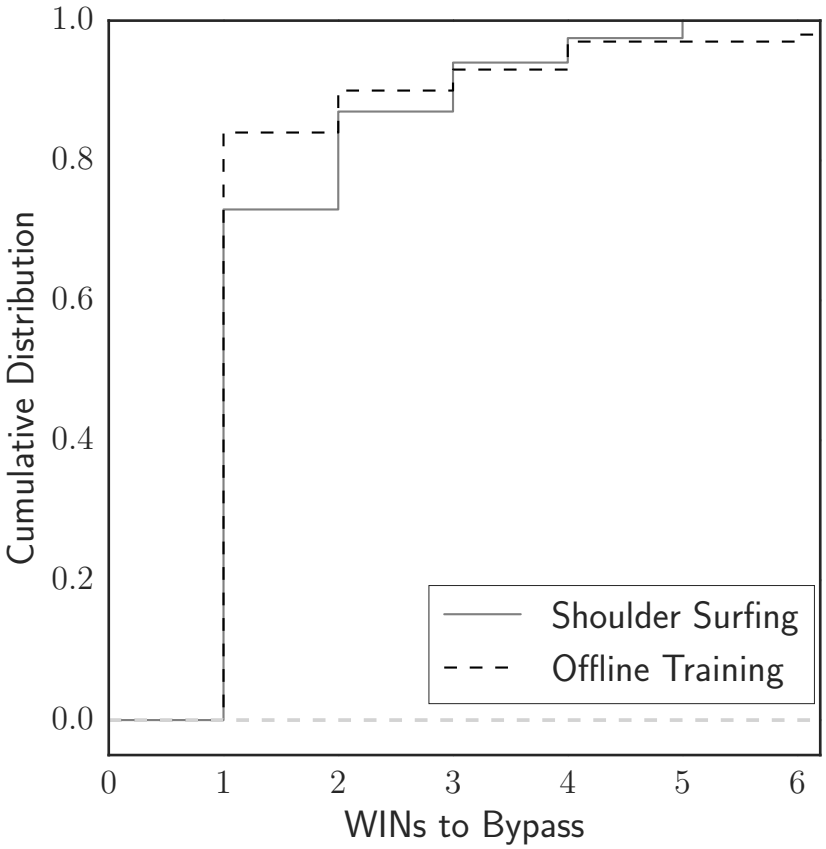


Figure 5.7: Windows until bypass for successful attacks.

5.5.3 Difficult or easy to mimic features

To measure how difficult or easy it was for the attackers to mimic IA schemes, we calculate mismatch and match scores for individual features for successful and failed attempts, respectively. We first calculate the absolute differences between individual features of the target and mimic swipes. The target swipe is explicit in offline attacks, but not when shoulder surfing. For shoulder surfing, the victims' swipe that is the nearest neighbour of the mimic swipe is selected as the target swipe. Mismatch and match scores are calculated by computing the normalized distribution of the occurrence of features in the top three most divergent and similar features, respectively. For divergence, we use the Kullback-Leibler (KL) information divergence measure [KL51]. A brief description of the KL divergence

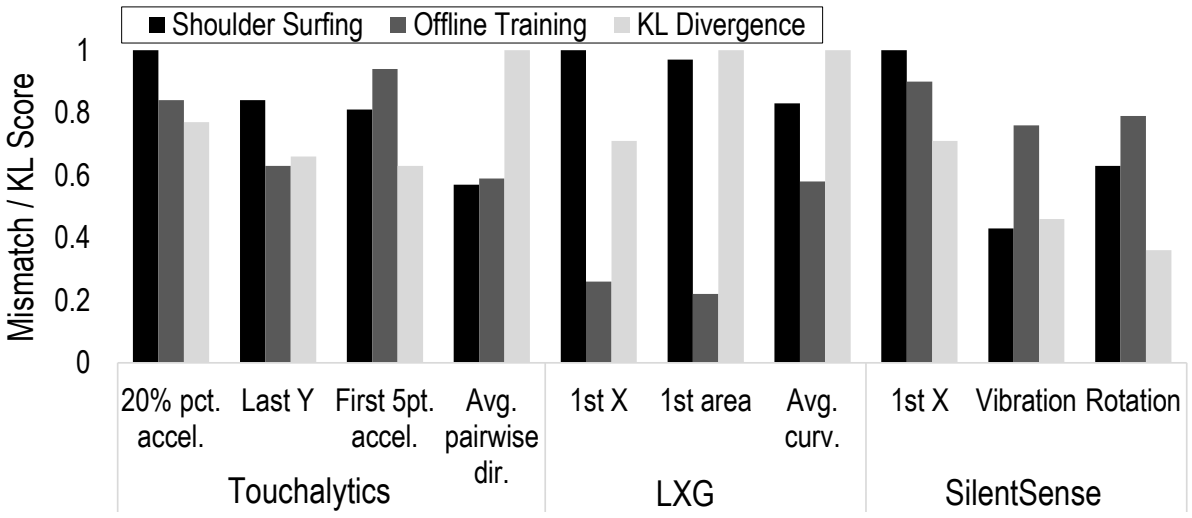


Figure 5.8: Difficult to mimic features across failed attempts for all the attackers. Mismatch score is the normalized distribution of the occurrence of a feature in top three most divergent features.

measure is provided below. For two Probability Mass Functions (PMFs) p and q of a discrete random variable X, KL divergence quantifies the difference between two PMFs as:

$$D(p\|q) = \sum_{i \in \Lambda} \log \left(\frac{p(i)}{q(i)} \right) p(i), \tag{5.1}$$

where Λ is the image of X and $p(i)$ and $q(i)$ respectively represent the probability of feature value i in p and q. To deal with the case when $D(p\|q) = \infty$, if $p(i) \neq 0, q(i) = 0$; we perform standard Laplace correction [CG96]. The Kullback-Leibler (KL) divergence [KL51] score for features is also calculated for the attacker-victim pairs on the raw input data from the device usage dataset. The KL divergence score provides a baseline for feature similarity and indicates the extent of adjustments made by the attackers for a feature. A higher KL divergence score indicates that the feature values are different across the participants.

Figure 5.8 and Figure 5.9 show features with high mismatch and match scores, respectively. Features related to swipe curvature, velocity, and acceleration have higher KL divergence and mismatch score for both attack types (see Figure 5.8). `Mimicker` does not provide feedback for acceleration or velocity related features since they are hard to comprehend, so we cannot fully conclude they are hard to mimic for offline training attacks. However, we can conclude these features are difficult to mimic for shoulder surfing attacks.

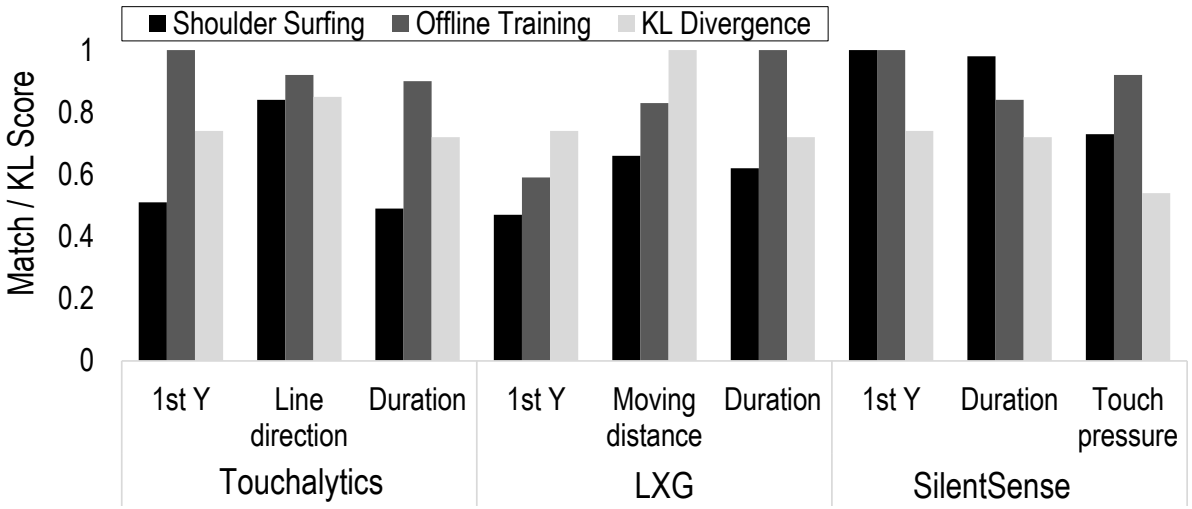


Figure 5.9: Easy to mimic features across successful attempts for all the attackers. Match score is the normalized distribution of the occurrence of a feature in top three similar features.

Figure 5.9 shows that swipe duration and touch pressure are easier to mimic. Some location based features (first and last touch coordinates) fall in both difficult and easy to mimic feature groups, especially for shoulder surfing attacks. We suspect that this is because some attackers are more critical observers than others.

Figure 5.8 shows that for almost all the features, the mismatch score is lower for offline training attacks. Since `Mimicker` provides feedback, attackers know how to adjust their behaviour to improve performance. This is quite pronounced for the location-related features such as '1st X', '1st Y', and 'Last Y' because `Mimicker` renders the target swipe.

There is an anomaly between LXG and SilentSense mismatch scores for the '1st X' location feature for offline training attacks. We suspect the higher mismatch score for SilentSense is due to its small feature set size and low KL divergence scores for vibration and rotation features. This indicates an overlap in behaviour across participants, increasing the chance that '1st X' is in the top three mismatched features.

5.6 Discussion

In this section we discuss constrained attack scenarios and the effect of key parameters.

Table 5.3: Effect of introducing one week delay between shoulder surfing and the attacks.

Attack follow up	Success Rate		
	Touchalytics	**LXG**	**SilentSense**
Immediate	80%	90%	95%
Delayed	80%	80%	90%

5.6.1 Basic shoulder surfing attacks

In our evaluation of shoulder surfing attacks, we assume attackers can record a video of their victims and watch it immediately before launching the attack. We were curious about the performance of a basic shoulder surfing attack using direct observation with hand written notes and a delay between observation and attack. To investigate this scenario, we repeated the shoulder surfing portion of the experiment with ten of the participant attackers, but inserted a one week delay between watching the videos and the attack. Each attacker viewed shoulder surfing videos of one victim they did not encounter in the main experiment. They were encouraged to make notes. One week later, the attacker returned to mount the attack on the victim. They were encouraged to consult their notes.

Table 5.3 compares mean bypass success rates across attackers for shoulder surfing attacks with delay and without delay (taken from their performance in the main experiment). Delayed attacks had a bypass success rate of 80%, 90%, and 90%, which an independent samples t-test did not find significantly different compared to attacks without delay. To gain insight into what attackers felt was important and what features they were mimicking, the notes were collected and their content categorized. Eight attackers drew the device screen with the corresponding location and curve of the swipe. Six of these attackers also noted the holding posture of the victims by drawing the holding hand and the location of the fingers of their victims. Two attackers wrote textual notes (such as: *"right bottom on the edge"*), seven noted the swipe speed (five noted it as slow/medium/fast; one marked a location on a continuous scale from slow to fast, and one wrote *"one-mississippi"*). Three attackers noted of the swiping finger or thumb of the victim. This small experiment suggests basic shoulder surfing is surprisingly effective, even when behavioural characteristics are mimicked at a coarse level.

5.6.2 Attacks with limited knowledge

In our evaluation, we assume the attacker has full knowledge regarding the victim's IA scheme. We were curious how performance is affected if the attacker has limited knowledge

Table 5.4: Bypass success rates for offline training attacks when attack attempts are replayed for different schemes.

Training Scheme	**Scheme for Attacks**		
	Touchalytics	LXG	SilentSense
Touchalytics	81%	84%	73%
LXG	76%	82%	71%
SilentSense	75%	78%	92%

about the IA scheme or its features. In a scheme-oblivious offline attack scenario, the attacker has the victim's raw data and a training app like `Mimicker`, but they do not know the exact IA scheme used by the victim (e.g., whether it is Touchalytics or LXG). For the feature-oblivious shoulder surfing attack scenario, the attacker has no knowledge of any of its features. We evaluate the performance of attacks for these scenarios using the data from our main evaluation.

Scheme-oblivious offline attacks

We simulate this attack scenario by mounting attacks on a different IA scheme than the one used for offline training. For example, if an attacker trained and successfully attacked a victim using `Mimicker` for LXG, would the attacker have been successful if they trained for LXG but that victim actually used Touchalytics? We accomplish this by replaying attack swipes logged during successful attacks in the main experiment on the same victim protected with a different IA scheme. Note there is some inherent overlap in schemes since they share some features, or capture touch behaviour across limited dimensions like location, pressure, area and speed. However, the specific model built by the scheme classifier could still be very different.

Table 5.4 shows that after training on Touchalytics, bypass success rates increased by 3% when the victim was actually protected by LXG and decreased by 8% when they used SilentSense. After training for LXG, bypass success decreased by 6% for Touchalytics and 11% for SilentSense. Finally, after training SilentSense, bypass success decreased by 17% for Touchalytics and 14% for LXG. There appears to be a greater drop when attackers train on Touchalytics/LXG and attack SilentSense (and vice versa). This is due to less overlap between Touchalytics/LXG and SilentSense (more touch features in the former, more device reaction features in the latter). Overall, 70% or higher bypass success rate for scheme-oblivious attacks indicates attackers may not even need to know the exact scheme used by their victims.

Feature-oblivious shoulder surfing attacks

We evaluate the success rate for feature-oblivious shoulder surfing attacks with an experiment using ten participants from our data collection (they were not attackers in the main experiment). The main experiment shoulder surfing attack protocol was followed except attackers were provided no details about touch IA and were simply told they would attack a device protected using a scheme that employs the touch input and device holding behaviour.

The results show a bypass success rate of 60%, 50% and 80% for Touchalytics, LXG, and SilentSense, respectively. Unsuccessful attackers were briefed on the features after they ended their attempted attacks. These attackers viewed the videos again and mounted attacks on the victims they failed to mimic. For these retry attempts after a briefing, bypass success rates were 75%, 80% and 85% for Touchalytics, LXG and SilentSense, respectively. These results indicate that attackers have a 50% or higher chance of defeating these schemes without any knowledge of the underlying features, but the chances of success increase with feature knowledge.

5.6.3 Effect of operating threshold

In the main experiment, we chose a FRR of 20%, which created FAR of 0.4%, 4%, and 0.2% for Touchalytics, LXG and SilentSense, respectively. To understand the effect of different operating thresholds on mimicry attacks, we evaluate attacks efficacy at FRRs from 20% to 35% and corresponding FARs. Note that a FRR of 35% is impractical since the IA scheme would reject the device owner for a third of their swipes making it quite unusable. We perform an offline evaluation by replaying the attack data through the IA schemes with different operating thresholds and log the bypass success rates for each threshold. A limitation with this simulated evaluation is that offline attackers are likely to perform better if they trained at the tested FRR. However, these results do provide a lower bound for bypass success rate.

The results in Table 5.5 show only a 7% decrease in success rate when the FRR is increased from 20% to 35% for shoulder surfing attacks. For offline attacks, the decrease in success rate is only 3%. There is a sublinear decrease in bypass success rate for a linear increase in FRRs, which is similar to the sublinear increase in FAR for linear increase in FRR (see Figure 5.1). We believe the lower relative decrease in the bypass success rate for offline training attacks is a result of the low intra-window TRR (see Figure 5.4).

Table 5.5: Bypass success rates for the attackers for different operating thresholds.

	FRR			
	20%	**25%**	**30%**	**35%**
Shoulder surfing	84%	81%	78%	77%
Offline training	86%	85%	83%	83%

5.6.4 Effect of different target swipes

The `Mimicker` target swipe selection module selects the 'best' swipe from a set of 150 swipes of the victim (see Section 5.3.1). We were curious how an attacker's bypass success rate was affected if they could not collect a large number of their victim's swipes and had to train using a suboptimal target swipe. To evaluate this, ten volunteers from the main experiment participated in a smaller experiment spanning three ten-minute sessions of offline training attacks, with each session spaced one week apart (to mitigate carry over effects). Each attacker was assigned two victims; one victim was protected with LXG and the other with SilentSense. For each session, the attackers performed offline training attacks using the best, average, and worst swipes (the order was counter balanced across attackers). The best, average and worst swipes correspond to the top, middle and bottom locations of the similarity rank table, respectively (see Section 5.3.1). These well-defined categories of swipes avoided a possible confound when randomly picking a target swipe.

The bypass success rates for the best, average and worst target swipes for LXG were 90%, 100% and 60%, respectively. For SilentSense, the bypass success rates were 100%, 100% and 80% for best, average, and worst swipes, respectively. A one way between subjects ANOVA score indicates a significant effect of target swipe on the intra-window TRR for the three target swipe types *(F(2, 27) = 9.18, p = 0.0009)*. Post hoc comparisons using the Tukey HSD test indicated that the mean for the intra-window TRR for the best *(M = 0.05, SD = 0.07)* and average *(M = 0.02, SD = 0.04)* swipes were significantly different than the worst swipe *(M = 0.47, SD = 0.44)*. However, the intra-window TRR for the best swipe did not significantly differ from the average swipe. These results indicate that while there is a reasonable chance ($\geq 70\%$) of success using any true accepted swipe as the target swipe, attackers can increase their chances of success by collecting more raw data to mine for an optimal target swipe.

5.6.5 Attacker- or victim-bound success?

Previous work has shown the performance of a generic attack against touch IA schemes is victim dependent [SP13b], meaning some victims are easier to attack than others. We

investigate if the same is true for mimicry attacks. A Kruskal-Wallis test comparing bypass success rates found no significant effect across victims, indicating no victims were easier to bypass than others. The same test comparing bypass success rates found no significant effect across attackers, indicating no attackers were better at mounting attacks than others.

5.7 Limitations

Like most human subjects studies, the scope is limited to people willing to participate. We discuss more specific limitations below.

We used similar devices for data collection and attacks. If an attacker collects a victim data using a device with a different form factor than the victim's device, a transformation would need to be applied to the features.

We used the same tasks to collect victim data and evaluate attacker performance. This setup represents a resourceful adversary able to obtain a victim's data using an app similar to the target. Even if an adversary used apps with different input behaviour, the results will likely be similar (previous work reports 5–8% increase in EER due to significantly different apps [KH14]).

The participants who volunteered for the additional experiments (reported in Section 5.6) did not receive remuneration for their participation. Consequently, these participants may had less motivation, which may have affected their performance.

For our experiments, we omitted difficult to comprehend features from briefings and the feedback module of `Mimicker`. We note that dedicated attackers may have increased their chances of success by training on the omitted features. However, our experimental setup demonstrates the vulnerability of these schemes against relatively effortless attacks.

Our protocol trained an attacker to mimic their victim's swipes one direction at a time excluding multi-touch gestures like pinch to zoom [FYY+14]. This simplified the attack tasks and expedited attacker training.

It might be difficult for the attacker to mimic multi-touch gestures or multiple swipe directions simultaneously. However, swipe is the predominant form of gesture and an attacker can view content using one directional swipe for most target applications (email and messaging apps using up swipes while gallery app using left swipes). Due to their infrequent use, Touchalytics also ignores multi-touch gestures. While efficacy of mimicry attacks on multi-touch gestures is an open research question, we do not address it in this work to conform to our realistic attack scenario. In terms of mimicry attacks on multiple

swipe directions, given the trivial shoulder surfing and offline training time required, the attacker can leverage `Mimicker` or observational notes to train for different directions in real time to mount these attacks.

5.8 Conclusion

We evaluate touch IA against two simple attacks by malicious insiders that effectively circumvent touch IA. We show that the widely accepted assumption that shoulder surfing attacks on touch IA are infeasible due to the hidden nature of many touch input features is incorrect. We also demonstrate how dedicated attackers can use an app like `Mimicker` to train themselves to mimic victims offline with very high success. Moreover, mimicry attacks appear practical even if the attacker has limited knowledge of the victim's IA scheme or limited logged examples of the victim's touch behaviour. Our findings suggest that touch IA does not provide adequate protection against malicious insiders. Consequently, touch IA cannot be used as a second line of defense against malicious adversaries that are able to bypass the primary authentication mechanisms through shoulder-surfing attacks. A focus for future research in IA should be the identification of a set of mimicry-resilient features across several behavioural biometrics that bolster IA security. We identify challenges and propose solutions to achieve this goal in the following chapter.

Chapter 6

Itus: A Framework for App-Centric IA Deployment

The last three chapters conclude that while IA seems promising in terms of usability and on a set of comprehensive evaluation criteria (see Chapter 3), there are challenges that need to be resolved. More specifically, in Chapter 3 we observe that app-specific differences not only influence the accuracy of IA schemes but also dictate the availability of data for training and classification purposes. In Chapter 4, we note that some users feel the need to improve the way IA was deployed for evaluation and how FRs were dealt with. Similarly, Chapter 5 suggests the need to identify a set of mimicry resilient features.

These findings indicate that until the aforementioned issues are investigated, wide-scale IA support will not be provided. However, IA deployments for specific use cases can be warranted. For instance, touch IA can protect corporate or personal data on a lost device since targeted mimicry attacks are not a threat for this scenario. Furthermore, the annoyance and deployment issues uncovered in Chapter 4 highlight the significance of active participation by human subjects for IA research, which requires a system for prototyping and deployment of IA. In this chapter, we investigate challenges in enabling IA deployment for specific use cases and research purposes.

We expand our fourth research objective (see *Objective 4* in Section 1.5) to first identify the bottlenecks to IA deployment. We suspect that some of the deployment issues arise from a device-centric IA deployment. We address these issues by proposing an app-centric approach to IA deployment. To further our objective of flexible and extensible IA deployment, we identify challenges involved in creating an IA framework that is flexible enough to be used by a general audience while extensible enough to assist in IA research. We proceed

to provide an open source implementation of a framework that meets our flexibility and extensibility objectives[1]. We begin our explorations by introducing app-centric IA.

6.1 A case for App-centric IA

Since contemporary IA schemes require the interception of user input events, some researchers have proposed including IA mechanisms at the platform level [CKF09, CRS13, LZX13, RQSL12], such that the operating system or app framework is responsible for providing IA to all apps on the system in an app independent manner. However, this approach has its own limitations in terms of flexibility and extensibility. We explore these limitations further in the following sections.

6.1.1 Flexibility issues

We investigate the flexibility limitations by comparing between two options on *where* to authenticate — at the device level or app level. For the comparison, we analyze how the outcome of *where* to authenticate impacts the decisions of *when* and *how* to authenticate.

When to authenticate?

A device-centric IA scheme may operate continuously in the background or get triggered when an app marked as sensitive is launched. However, some apps may not require IA for all usage scenarios. For example, consider a banking app that enables a customer to query his account or locate a nearby ATM. For this app, there is no need to authenticate a user when they are trying to locate an ATM. Similarly for a browser app, there is no need to authenticate a user when they are reading news. Since a device-centric approach is unaware of the task that the user is performing within an app, it cannot provide authentication control at the task level.

On the other hand, if an app can control when to authenticate, the banking app only authenticates a user when they are querying their account. Similarly, the browser app can detect when a user is reading news and it may decide not to authenticate them. But when the user switches to his social network website with saved credentials, IA may be performed. By delegating the decision of *when* to authenticate to the app, we can perform task-aware

[1] https://crysp.uwaterloo.ca/software/itus

IA, which reduces unnecessary authentication overhead. Another advantage of task-aware authentication control is its inherent support for multi-user scenarios. Smartphone owners share a mean of 12% of their smartphone apps and these apps are primarily entertainment applications such as games or web browsers [HRS+12]. For these multi-user scenarios, the non-owner is not supposed to access personal information of the owner on the device. An app-centric approach allows a non-owner to access content that does not leak personal information and denies access to the content that may leak personal information (e.g., access to a mail portal with saved credentials).

How to authenticate?

A device-centric approach may employ a specific behaviour-based classifier for IA. However, a classifier may not be suitable for a particular type of application. For example, during a session in which a user accesses his social network website on the browser using saved credentials, enough sample values may not be available for text IA to compute an authentication score. Similarly, a classifier based on touch IA may not have enough sample values for a messaging app (due to lack of swipes generated by the user). To cater for these behavioural differences, a device-centric approach may employ multiple classifiers; however, this will result in significant overhead in terms of feature sampling. On the other hand, an app-centric approach can leverage the knowledge of an app's nature to choose the appropriate IA scheme.

Another limitation of the device-centric approach due to its app-oblivious nature is the loss of valuable information that may be useful for classification. If the classifier has additional knowledge about a task that the user is currently performing, it may use that for robust classification. For example, in Section 6.2 we show that a user's touch behaviour is slightly different when he is finding POIs in maps as compared to browsing. Consequently, in the banking app, if the classifier is aware of the task being performed by the user (finding nearest ATM or querying his account), it may use this additional information to improve its accuracy by tuning its features (we demonstrate this in Section 6.2.2).

Discussion

The comparison of app- and device-centric approaches shows that due to the app-aware nature of the former, fine-grained authentication control is acquired and IA is performed only when required. This fine grained authentication control also reduces authentication overhead. The app-centric approach can use its knowledge about an app to determine the

appropriate IA scheme for that app. Finally, the app-centric approach can leverage its knowledge about an app's nature to improve the accuracy of the classifier by tuning its features.

6.1.2 Extensibility issues

In terms of flexibility, we note that different apps have different characteristics and a generic device-level behaviour-based classifier may not be suitable for different apps. To overcome some of these limitations, a library could be provided at the platform level which provides an interface to the developer to configure various parameters for IA. However, it is not certain that IA will get incorporated at the device-level given the unaddressed research challenges. This severely limits the possibility of limited IA deployments for specific use cases including protection of corporate data on lost devices.

Furthermore, a device-level library would need to be managed by the platform developers or some central authority. However, IA is a relatively new area that still experiences radical revisions in IA schemes due to the research findings in the use of novel sensors or wearable devices for IA [MMMC+14, SSTA14]. Similarly, the handling of authentication failures would require revisions due to research findings on the usability of IA schemes. The platform developers in this case are more inclined toward accepting mature contributions, which will lock out many app developers who want to deploy features specific to their use cases.

Finally, a device-level approach does not reduce the re-engineering effort of IA researchers. Since the platform developers are willing to accept mature contributions only, research prototypes are less likely to be reused. In Chapter 2, we noted a lack of synchronized effort to accelerate the research in the IA domain. For instance, several IA schemes had only minor differences in terms of employed features or machine learning classifiers, but each research effort came up with its own implementation. An app-level library can create an ecosystem that reduces the re-engineering effort. For instance, if an implementation of Touchalytics is available that provides the researchers values and low level interface of the features employed, it can reduce the re-engineering effort to determine mimicry resilient features.

6.2 Evaluation of App-centric IA

We empirically evaluate Touchalytics [FBM+13] on our touch input dataset in a device-centric and app-centric fashion to establish the limitations of the former approach. A detailed description of Touchalytics and the touch input dataset are provided in Section 3.1 and Section 3.2, respectively. We first apply Touchalytics in a device-centric manner such that it neither discriminates between data from different apps nor makes any assumptions about the nature of the app. We then repeat the experiment and compare the results by applying Touchalytics in an app-centric manner. Similar to our evaluations in Chapter 3, we set the classifier parameters to the values recommended in the original paper.

6.2.1 Device-centric Touchalytics

We evaluate device-centric Touchalytics using five-fold cross-validation and average the results across all the users. The accuracy evaluation of Touchalytics on our dataset provides a FAR of 17% with FRR of 8% on a window size of two swipes. The accuracy on our dataset is significantly lower when compared to accuracy reported in Touchalytics for the same window size (7% FAR with 7% FRR). We hypothesize that the variance in app nature and usage behaviour is responsible for this accuracy degradation. Since Touchalytics only used data from two apps (an app to read Wikipedia articles and a spot-the-difference game) in a lab setup and participants performed predefined tasks, their experimental setup did not capture this behaviour.

In order to evaluate our hypothesis, we use the Kullback-Leibler (KL) information divergence measure [KL51]. A brief description of the KL divergence measure is provided in Section 5.5.3. We calculate the intra-user and the inter-user KL divergence score across the Touchalytics' features for every app. The intra-user KL divergence score is calculated between two partitions of a user's data and the inter-user KL divergence score is calculated between a user's data and data samples from the rest of the users. If a feature is a good discriminating feature, it should have low intra-user KL divergence score and high inter-user KL divergence score.

Our results show that the majority of the features have low intra- and high inter-user KL divergence score for a subset of applications only. We show KL divergence scores for three representative features in Figure 6.1 and omit other qualitatively similar results. In Figure 6.1, end x-coordinate is a good discriminating feature for the launcher app, but for the maps app it has a relatively high intra-user KL divergence score. Looking at the raw data, we can see that this is due to the nature of the maps app. In the maps app,

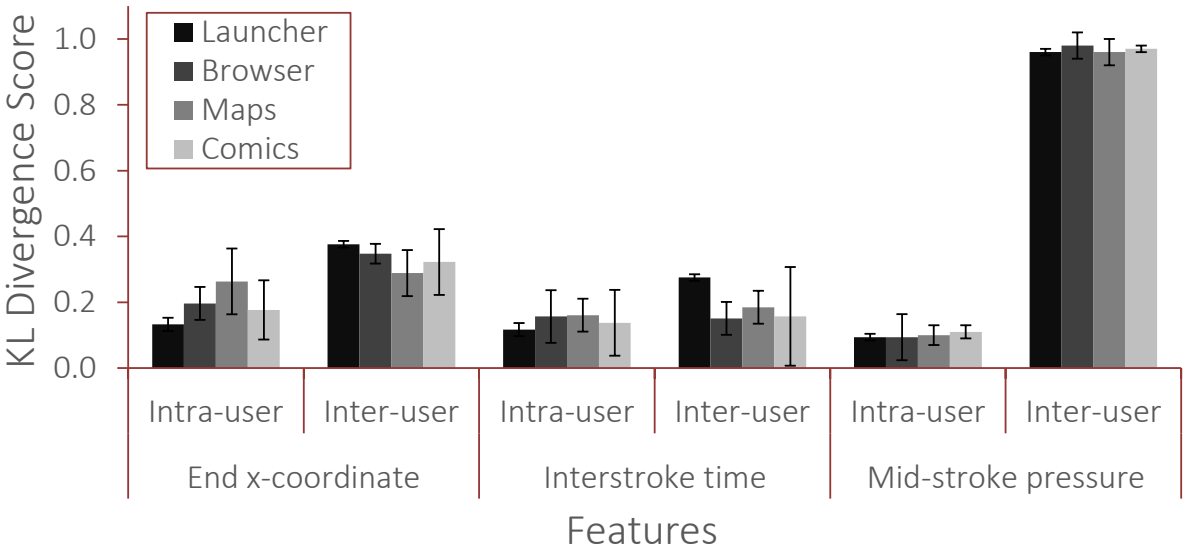


Figure 6.1: KL divergence score of features across four apps used in this study

users' start and end points are more random since they are locating some POI on the map unlike the launcher app, where a user generally swipes at a specific location on the screen. Similarly, the browser app fails to provide a high inter-user KL divergence score for inter-stroke time feature. We suspect that this is due to the dependence of inter-stroke time on the content that a user is browsing (reading an article might result in lower inter-stroke times as compared to skimming it). While the majority of the features are not good discriminating features for all the apps, mid-stroke pressure and mid-stroke area covered were the only features that had consistently low intra-user and high inter-user KL divergence scores. Since only two features cannot provide a good separation boundary between different users' behaviour (due to more chances of collisions), for efficient classification all the good discriminating features for an app should be used.

Our analysis of the device-centric approach shows its failure to capture the variance in apps' nature. A simple solution to this limitation would be to create a separate behavioural model of each app at the device level. This would allow the device-centric approach to compare the runtime behaviour of an app with its behavioural model to get a more accurate authentication score. While this approach would fix the accuracy degradation to some extent, it neither provides fine grained authentication control nor mitigates the limitations due to the device-centric approach's obliviousness of an app's nature.

Table 6.1: Accuracy evaluation of device- and app-centric approaches (95% CI)

	Device-centric approach		**App-centric approach**		**Feature tuned app-centric**	
App	**FAR**	**FRR**	**FAR**	**FRR**	**FAR**	**FRR**
Launcher	16% (5)	6% (3)	11% (4)	4% (4)	7% (3)	3% (3)
Browser	19% (4)	8% (3)	12% (4)	6% (3)	6% (3)	4% (2)
Maps	16% (4)	12% (5)	13% (4)	7% (4)	5% (4)	4% (3)
Comics	17% (3)	7% (4)	8% (5)	6% (4)	6% (4)	4% (3)

6.2.2 App-centric Touchalytics

For effective and pragmatic IA, we propose app-centric IA in which an app decides when and how to authenticate a user. While designing an app, an app developer first identifies activities that may lead to potential misuse. For example, for the browser application, the app developer may consider access to all websites that ask for a user's credentials as potential sources of data leakage. After identifying the activities, the app developer only implicitly authenticates a user when these activities are performed. The app developer also decides *how* to authenticate by choosing a suitable classifier for his app (as per the app's nature). For example, looking at the user's interaction with his browser app, an app developer may choose a classifier based on touch input behaviour (since more swipe data is readily available than keystrokes data). Finally, the app developer tunes the features of the classifier as per their app's nature. We note that these activities increase the app developer's development overhead; however, this overhead can be mitigated by providing a library to the app developer that provides a generic implementation of behaviour-based classifiers that can be extended and reused. We discuss this further in the following sections.

We now evaluate the sample apps on our dataset in an app-centric approach. To simplify our evaluations, we assume that these apps require IA for all activities and a classifier based on touch IA is the right choice for them. We now show how we tuned the features for these apps and we then present accuracy results.

Feature tuning

For feature tuning, an app developer collects sample data from some test users. They then determines the features that are good discriminating features for their app and trim the rest of the feature from the classifier. The app developer can optionally add new features for robust classification. We now summarize our experience of these feature tuning aspects for our sample apps.

Feature deletion: For each app, we deleted those features that had high intra-user KL divergence score and low inter-user KL divergence score. This trimming left 27, 25, 22, and 23 features for the launcher, browser, maps and comics apps (out of 31 total features of Touchalytics), respectively. Feature trimming ensures that the accuracy of IA scheme will not suffer due to irrelevant features.

Feature addition: We plotted swipes from users' interaction with each app to determine app-specific features. For example, by looking at the users' swipes for the browser and launcher apps, we found out that the distance between two consecutive swipes was a good discriminating feature since users had unique swipe cluster patterns. Similarly, we observed that the orientation of a phone along its x-axis (pitch) also affected a user's input behaviour and we used it as a feature for all the apps.

Other app-specific improvements: We also identified behavioural anomalies that were specific to an app. For example, looking at the browser's touchscreen input data, we found out that when a user had to scroll down on a large document, they did it by rapid vertical swipes on the touchscreen. This uncommon behaviour was observed for some participants and Touchalytics was unable to correctly classify it. In order to deal with such behavioural anomalies, we made appropriate changes to the IA scheme.

Accuracy evaluation

For the accuracy evaluation of the app-centric approach, we tune the app features as discussed in the last section. We then invoke the classifier based on touchscreen input behaviour on each app's data and perform five-fold cross-validation on the training data. To quantify the impact of app-specific training and classification, and feature tuning separately, we report accuracy results for app-centric approach with and without feature tuning. The accuracy results of our evaluation on app using device- and app-centric approaches are provided in Table 6.1.

The results show that by using an app-centric approach we are able to reduce FAR by 5%, 6%, 3% and 9% for the launcher, browser, maps and comics apps, respectively. These accuracy gains are observed since a separate behavioural model is created for every app and the variation in application nature does not affect the behavioural model. Another 4%, 6%, 8% and 1% reduction in FAR is recorded when we tune features according to the app nature. In addition to reduction in FAR, a significant and consistent reduction in FRR is also observed when the app-centric approach is used. These results show that by using an app-centric approach, we achieved accuracy improvements in addition to low authentication overhead and fine grained authentication control.

Discussion

Empirical evaluations of the app-centric approach show that it provides significant accuracy improvements as compared to the device-centric counterpart. This strengthens our argument that the decisions of when to authenticate, which IA schemes to authenticate with and feature tuning of the IA scheme should be delegated to the app. This delegation increases development overhead on the app provider. We now present Itus, a framework that enables app developers to effortlessly provide IA support in their apps.

6.3 Motivation for IA Framework

In the previous sections we established that the app-centric approach is superior in terms of usability and extensibility. We now identify the need and expectations from a framework for the app-centric IA deployment. To this end, we consider our audience to be two-fold: the app developers who wish to incorporate IA into their apps, and the IA developers who wish to improve existing IA schemes. In this section, we first discuss some sample apps and the motivations their developers might have for including IA support.

We consider the following apps as potential candidates for using IA:

An Enterprise Email Client: Provides employees access to corporate emails from their mobile devices. The email client app gives access to large amounts of sensitive corporate data. Employees generally want to have access to their corporate email account from a variety of places, including at home and while traveling to remote sites or client premises. The bring-your-own-device phenomenon makes it easy and natural for employees to have access to their corporate email from their personal smartphones, but raises a host of security issues for the employer. Now that devices are being carried to non-work locales where unaffiliated parties may gain access to them, the employer will need to add some form of authentication to the client app. Adding IA at the app level allows the employer to ensure only authorized users are accessing sensitive emails without disincentivizing users from installing the app on their own devices due to inconvenient security mechanisms.

A Web Browser: Provides all the standard web browser features including a password manager. The web browser is an app that allows users to access potentially privacy-sensitive data from a variety of different sources. The developers of the browser might be interested in providing IA support for their users in order to allow them to benefit from the convenience of a password manager without the decreased security that comes with storing and automatically filling in a user's passwords. The wide variety of distinct

web services now available means the web browser becomes a portal for many different categories of user interaction patterns, and so a flexible framework would seem to be of paramount importance in this case.

6.3.1 App Developers

Next, we refine our model of the spectrum of involvement by the app developers adding support for IA. In particular, we consider two levels of developer interest in IA frameworks:

Cursory interest: app developers who only want to add support for IA to their apps without tuning its accuracy or providing any application-specific behaviour. There are several types of app developers that may fall into this category. In the most trivial case, an app developer may simply be interested in experimenting with IA and wants to add it to their app without spending significant time on configuration and re-engineering of their app. Or, the app developer may be developing an app that contains such generic tasks that the default behaviour is "good enough" out-of-the-box. For example, consider the email client described above—user typing cadence tends to be unique enough [HCP09] that the default keystroke classifier can easily deal with most usage scenarios of this app.

Significant interest: app developers who wish to fine-tune an IA scheme for accuracy and performance. Such developers might do one or more of the following: restricting calculation of behavioural features to some subset to minimize computational overhead; configuring parameters of machine learning algorithms; experimenting with beta users to determine which configurations work best and retaining the data gathered during this phase for later training. It is important that the framework be able not only to support these developers in their efforts but to facilitate them by providing tools to assist and automate in these scenarios.

6.3.2 IA Developers

We now discuss the other class of audience — the developers who wish to contribute to the framework's IA schemes. To this end, we consider two representative scenarios.

Consider first those developers who wish to improve existing IA schemes (for example, investigating a novel set of behavioural features or simply tuning parameters). To demonstrate the efficacy of their proposal, they need to develop a prototype and evaluate its performance. Ideally, they should have access to an existing framework, giving them the

necessary components to perform tasks common to all IA schemes (e.g., data storage, training, classification) without re-engineering. The framework should allow these developers to rapidly prototype their ideas without having their contributions moderated.

Secondly, consider developers who want to add support for behavioural classification modules that the core framework does not anticipate. Specifically, the framework should allow for new machine learning classification algorithms and new sensor-derived behavioural features to be included alongside the core framework.

The framework should support these stakeholders and provide them with interfaces so that they can use or contribute to the framework as smoothly as possible.

6.4 Design Goals

In the previous section we highlighted the two-fold audience we consider while outlining the need for an app-centric IA framework: app developers and IA developers. Both groups of developers are important for a successful IA framework to be adopted and remain relevant over time. In this section, we outline our design goals for Itus — a framework for app-centric IA; and how it supports both groups of developers in their efforts to provide usable IA to end users.

Separation of roles: Performing end-user authentication in a secure, usable manner is a challenging task that involves collaboration between IA developers and app developers. Our IA framework should synergize the efforts of different stakeholders to protect user data on mobile devices. In this spirit, we aim to incorporate a clear separation of roles that distinguishes between app developers, who may not be domain experts in the mechanisms underlying the system, from the IA developers or researchers working to improve the system. We further recognize that security, systems and classification researchers all have unique contributions to make and strive to make deployment of this individual components in a full-scale system as painless a task as possible.

Ease of Use: Our framework should not simply permit app developers to provide IA support in their apps, but also provide the app developer with an API to do so with minimal effort. App developers should be rewarded for their extra attention to user security, not punished with a burden of extended development time. Our goal is for Itus to be deployable in a reasonable default configuration with the absolute minimum number of lines of code changed as possible. The API should also allow configuration directives to be provided in as straightforward a manner as possible.

Flexibility: While ease of use is critical for adoption of IA by the broader audience of app developers, it is also important to make considerations for developers who have unique app needs or who wish to fine-tune the accuracy or performance of the IA scheme to their app. For example, for the web browser example discussed in Section 6.3, the developer might be interested in making authentication decisions in a heavily context-aware manner. To this end, he might decide that after users start playing an embedded video, they are likely to hand off the device to another person and thus it would be undesirable for the IA mechanism to interrupt viewing with an explicit authentication prompt. In the case of the enterprise email client, the company's IT department may wish to invest heavily to provide high accuracy to ensure data security. In this case, the app developers should be able to configure Itus with pre-recorded training data or parametrize the classifier to improve its accuracy. We consider design features that allow for such examples of app-specific functionality when providing flexibility to app developers.

Extensibility: Extensibility is a prerequisite design goal of any system that is to be adopted by the community. Individual IA developers should be able to create and evaluate prototypes for various subcomponents of the Itus framework without any dependence on a centralized authority, and they should be able to distribute successful IA schemes independently from the core Itus framework. Ideally, this leads to a situation in which IA developers are able to deploy iterative improvements to the IA schemes while simultaneously allowing app developers to adopt these improvements at their own, perhaps asymmetric, rate. Our framework should allow each of these groups to make contributions without becoming embroiled in the specifics of framework subcomponents unrelated to their particular tasks.

Performance: Finally, performance in terms of computational overhead and power consumption is important to both the app developers and the researchers working on IA. App developers are especially concerned with any performance penalty that is going to be imposed on their apps, as this has a direct effect on their end-users' experience. Therefore, it is important that the framework itself consumes minimal resources so as to allow the behavioural classification subcomponents as much time as possible to perform their tasks before the end user is able to discern any differences.

To a lesser extent, IA researchers also have a stake in the performance of any IA implementation. If they are to commit to implementing their prototypes in our framework, then any computational cost added by the framework that comes in addition to the prototype's cost must be minimal or it may reflect badly on the prototype. To this end, we consider performance at each stage of the design and implementation of Itus, and conduct a performance measurement using real-world apps to make sure our goals have been met.

6.5 Architecture

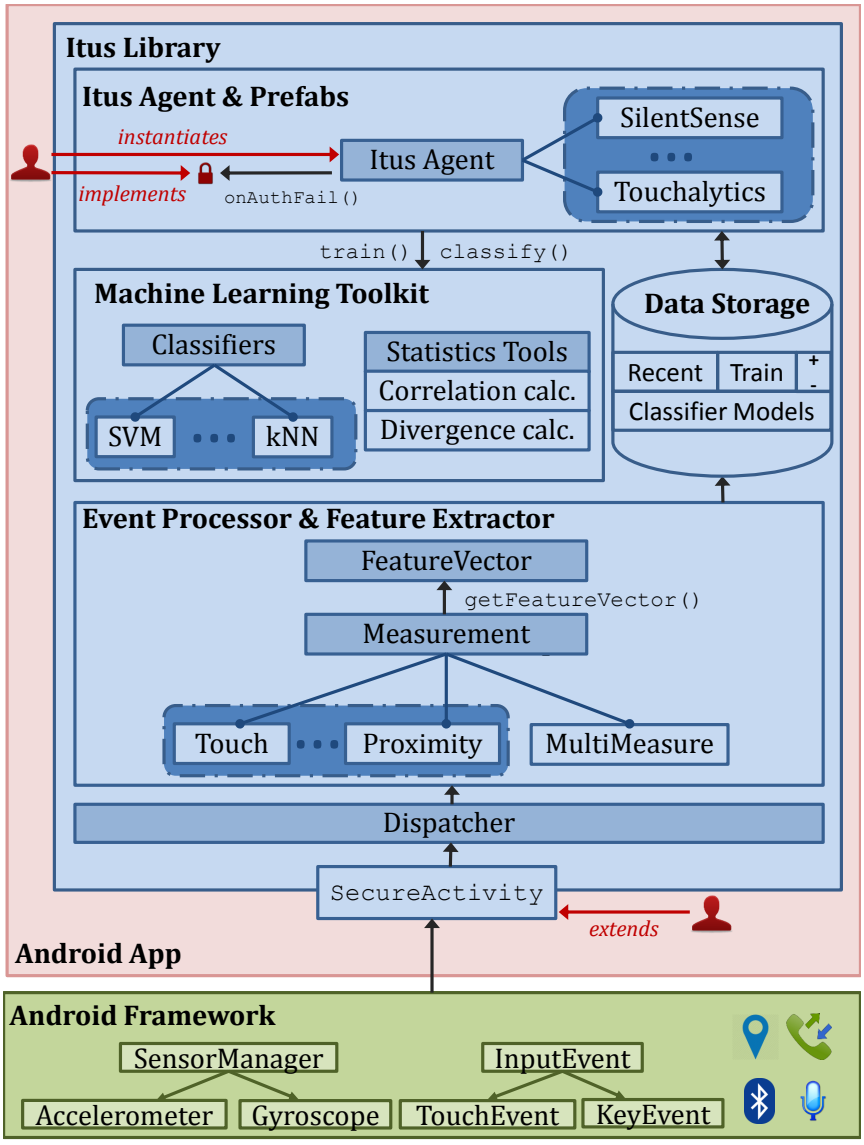


Figure 6.2: Itus framework architecture. Subclasses that are intended to be contributed by IA developers are in dark blue dotted boxes. The interfaces exposed to app developers are in red font.

This section gives an overview of the Itus system architecture. The core architecture of

Itus, illustrated in Figure 6.2, is as follows: app developers extend a customized Android activity called `SecureActivity`, provided by the Itus library. Itus is then able to intercept user interaction events, which are passed through feature extractors called `Measurement`s to obtain feature vectors. The Itus Agent gathers these feature vectors and hands them to the classification algorithms for training and classification. In the event a classifier returns a negative result (failed authentication), the Itus Agent then either notifies the parent app of the failure or independently switches the app view to a lock screen. In what follows, we will elaborate on how each of these links in the chain fit into the Itus framework.

6.5.1 SecureActivity

Interactive Android apps are composed of one or more "activities", which are app components that present the device user with a user interface to the app's functionality. When a developer creates a new Android app, one of the first steps is to create a subclass of Android's `Activity` class. Itus provides its own subclass of `Activity` called `SecureActivity`. To add IA to an app using Itus, a developer simply changes any classes that extend `Activity` to extend `SecureActivity` instead. This provides an incredibly simple way for developers to add Itus to their existing apps, supporting our ease-of-use goal discussed in Section 6.4. It also makes it trivial for developers to partition apps composed of multiple activities into those that require authentication and those that do not (for example, a banking app may not want to authenticate a person when he is locating a nearby ATM); this supports our flexibility goal. This implementation puts our framework in an ideal place to intercept user interaction events with the app in order to then use them as behavioural features for classification.

6.5.2 Measurement and subclasses

Behavioural biometrics research examines a broad array of sensor values that may be useful for distinguishing between different device users (see Chapter 2). In the Itus framework, we generalize these widely varying channels to say that behavioural features are generally calculated using some form of *measurements* taken from some on-board sensor. Thus we provide an abstract class in Itus called `Measurement`, subclasses of which are intended to extract measurements from any source accessible within the Android API (or extensions). To further our examples of sensor values, Itus might contain subclasses of `Measurement` called `Touch`, `Keystroke`, and `Movement`, respectively.

`Measurement` objects can register to receive events allowing them to process input data, which are subsequently passed to them via a callback method called `procEvent()`. There are two types of sensor readings: event-driven inputs and continuous readings. Event-driven inputs are intuitive to handle; registered handlers are called to process an event as soon as it occurs. The touch and keystroke examples are cases of event-driven inputs. Continuous readings, in which a sensor provides measurements any time it is polled (e.g., the accelerometer), are more subtle to handle. To deal with these, we create `Periodic` events, which function similarly to clock timer interrupts. `Measurement` objects analyzing continuous feedback sensors are then invoked periodically at parameterizable intervals. Some `Measurement`s may provide feature values calculated from an aggregate of sensor readings and not at every regular time interval; for these objects, we allow the event-processing function to simply consume data and indicate that it is not yet ready to provide any feature values.

It is worth noting at this point that Itus provides the ability for app developers to specify exactly *which* subclasses of `Measurement` they wish to use in their apps. If a particular configuration of Itus does not use any `Measurement`s that require special app permissions within Android, then adding the Itus framework in that configuration to the app means the app itself will also require no new permissions. Since apps are able to interact with the touchscreen and keyboard automatically when they are running in the foreground, developers will always be able to add Itus to their app using `Touch` and `Keystroke` measurements without modifying their app's permissions list.

6.5.3 Dispatcher

In the last subsection, we said that events are given to various `Measurement` subclasses for feature extraction. In keeping with our performance goals outlined in Section 6.4, it would not be efficient to invoke the `procEvent` method of every single `Measurement` for every single event. Instead, we create a `Dispatcher` class, which is responsible for delegating event processing to any configured `Measurement` objects. When Itus is configured at runtime to use a given `Measurement` subclass, an initialisation method is run from within that class. The primary purpose of initialisation method is to register with the `Dispatcher` and specify which events the `Measurement` would like to receive. Any events handled by Itus—either intercepted from user input by `SecureActivity` or generated periodically—are thus given to the `Dispatcher`, which in turn looks up the event type in a table to see which `Measurement`s have registered for it, optionally adds context information, and passes them the event data via `procEvent()`.

6.5.4 FeatureVector

As discussed previously, `Measurement` subclasses are responsible for taking data from events and turning them into useful feature values. When `procEvent()` is called to deliver event data to a `Measurement` object, the `Measurement` object will signal the `Dispatcher` if it has a new set of feature values ready to be exported. After receiving the signal, the `Dispatcher` will invoke the `getFeatureVector()` method of the `Measurement` subclass to retrieve these values. Specifically, the feature values are stored in a class called `FeatureVector`. This class is a 2-tuple made up of an array of `double` values and a boolean representing the class of the feature vector (valid/invalid user).

6.5.5 DataStorage

A `FeatureVector`, as obtained above, is processed as follows: if training has already been performed, then we simply classify this most recent sample. If training has yet to be performed, the `FeatureVector` should be stored until enough data is collected to perform training. In both cases, the `Dispatcher` will hand the `FeatureVector` over to a `DataStorage` object, which will store the `FeatureVector` in a list (hereafter called a "bin") depending upon the classifier state. In the former case, when training is complete, the `FeatureVector` will be placed into a bin labeled **bin_recent**, which can have an upper limit imposed on its size; this results in a sliding window, or FIFO queue, in which older data is discarded after consumption by the classification module or upon arrival of newer data. In the latter case, when training is pending, the `FeatureVector` is placed in the longer-term **bin_training** to be used for training once the bin reaches sufficient size.

The bins themselves are stored in a hashmap pointing to lists, and their names are specified using arbitrary strings. This is to provide further support for our flexibility goal: developers are able to use the `DataStorage` object's functionality for novel purposes other than the two outlined here. The `DataStorage` object can write the contents of its bins to disk when requested, and will load them back each time the app is launched. It reads and writes only to the app's internal storage, so no special permissions need be added to the app.

The `DataStorage` object is also used to store classifier models once they have been trained. It has `get()` and `put()` methods allowing arbitrary data to be stored, allowing IA developers to use it as a convenient storage mechanism via the app's internal storage.

6.5.6 Itus Agent

Now that we have discussed the basic modules of our framework, we provide details of how the actual invocation of training and classification are coordinated. These are done by a class called `Itus`, which we refer to as the 'Itus Agent'. The Itus Agent runs in a separate thread from the main app. It is the main object that an app developer will interact with when they are adding the Itus framework to their app and configuring it. An instance of the Itus Agent class performs the configuration for all subcomponents of the Itus framework, including which `Measurement` subclasses are used and which classifiers are employed (see Section 6.5.8). The Itus Agent drives the periodic events described in Section 6.5.2, enforces the authentication policy with training/classification and, where necessary, locks the app in the event of a failure to implicitly authenticate.

In more detail, once training has been completed and a classifier has been obtained, the Itus Agent then simply runs it periodically against the recent `FeatureVector`s stored in **bin_recent**. If a classifier returns a negative result representing an unauthorized user, the Itus Agent reacts by performing its configured (or default) lock-screen action. If training has yet to be completed, the Agent explicitly authenticates the user in order to establish the ground truth for training data.

6.5.7 Itus Oracle

While we separate the domain knowledge of IA from its deployment by providing app developers with a high-level API to provide IA support in their apps, we understand the importance of correct selection of behavioural classifiers and suitable parametrization of the underlying machine learning algorithms. To bridge the gap between app developers' inexperience with IA schemes and the need for advanced configurations, we provide the Itus Oracle to automatically determine the right classifier and optimal configuration parameters.

To use the Itus Oracle, the app developer deploys Itus in 'configuration mode'. In this mode, Itus collects and logs all feature vectors and gathers other measurements concerning the data collection and performance of the app, such as timestamps, data sources, and processing times. After multiple sessions of beta use in the configuration mode, the developer then connects the device to a development computer and runs the Itus Oracle. The Itus Oracle downloads the logged data from the device and analyzes it. It experiments with various machine learning tools, including any classification algorithms compatible with Itus, and provides suggestions based on its analysis to the app developer. More information about the implementation of the Itus Oracle is provided in Section 6.7.

6.5.8 Machine Learning Toolkit

The number of viable candidates for machine learning algorithms for classifying behavioural feature sets is continually expanding. This is due not only to the ongoing research in the field but also due to the fact that a classifier may work well in combination with a certain feature set, while other classifiers may perform much better in cases where it is weak. Therefore, it is highly desirable that the Itus framework be able to add support for many classification algorithms without the development overhead of coupling them tightly with the rest of the IA framework.

Itus does this by defining a `Classifier` interface with two self-documenting methods: `train(List<FeatureVector>)` and `classify(FeatureVector)`. The `classify()` method should return `boolean` value `true` for the positive class (authorized user) and `false` for the negative class (unauthorized user). More fine-grained control, such as interacting with confidence values, can be obtained by accessing `Classifier` objects directly. The learned classifier is transparently loaded to and from disk between app launches via the `DataStorage` object. We initially provide Itus with an implementation of the k-nearest-neighbours (kNN) classifier, and a wrapper that allows the Java version of libSVM (Support Vector Machines) [CL11] to be used as `Classifier`s.

An IA scheme may require use of only a subset of `FeatureVector`s from the training set. For instance, the LXG scheme discussed in Chapter 5 only compares the unknown (to be classified) swipe with positive and negative samples in the same swipe direction. Such conditions are catered by providing an overloaded version of the `train` function, which takes an additional argument of `pivotIndex`. When the `pivotIndex` argument is provided, Itus employs only those `FeatureVector`s for classification that have matching `pivotIndex` value.

Finally, the machine learning toolkit also contains statistics tools to measure correlation and divergence between features to determine effectiveness of features of behavioural classifiers. The Itus Oracle uses these tools to generate recommendations on features which should be used for an app.

6.6 Workflow

In this section we outline the workflow that app developers must follow to provide IA support in their apps using Itus. We also discuss how IA developers can benefit from Itus and in return how they can contribute.

6.6.1 App Developers

As discussed in Section 6.3, we broadly consider two types of app developers: those who want to use IA out-of-the-box, and those who want to tune the accuracy or performance of Itus. For the former type, Itus can be effortlessly imported and used in their apps. The app developer simply (i) identifies activities that should be protected, (ii) extends the activities from Itus' `SecureActivity` class, and (iii) starts the main thread of an Itus object. Training and classification are performed automatically at this point (see Section 6.7 for details).

On the other hand, if the developer wants to tune the performance or the accuracy of a classifier, he uses the Itus Oracle to determine the best configurations for his app. To this end, the app developer (i) identifies activities that require implicit authentication and extends them from `SecureActivity`, (ii) starts Itus in configuration mode and gives device(s) to beta users for data collection, (iii) connects the device to the Itus Oracle (on the desktop) so that it can analyze data and generate recommendations, (iv) chooses from the recommended configurations so that the Itus Oracle can repackage the library, and (v) adds the repackaged library to the app and disables configuration mode.

Most classifiers require negative (non-owner) training data in order to provide high accuracy. For this, an app can either use default negative instances included with Itus, or it may package a subset of data collected during configuration mode as negative training data. Since data collected during the configuration mode is labeled against each user, the negative training set of a user is created by excluding his data. We anticipate that after Itus is released in the public domain, it will also benefit from IA datasets that have been made publicly available by the research community [FBM+13, SPW13].

6.6.2 IA Developers

Section 6.3 presented two types of IA developers we envision contributing to Itus. Itus provides deliberate separation for contributions to the IA mechanism. In Section 6.5.2 we described how new sensor features are added to Itus by extending the `Measurement` class. Similarly, Section 6.5.8 described how new classification algorithms are added to Itus be extending the `Classifier` class. These subclasses can be contributed to Itus independently by the second group of IA developers discussed in Section 6.3.2.

Section 6.5.6 explained how app-specific configurations are stored in an `Itus` object. Itus can provide for a variety of default configurations by simply distributing pre-built `Itus` objects. In the Itus framework, we call such objects Prefabs and implement them by

deriving subclasses of `Itus`. These subclasses inherit all the functionality of the default Itus Agent, and simply need to add a constructor that performs any configuration directives that would normally be added by the app developer. These Prefabs can be compiled and distributed separately from the Itus framework, allowing the first group of developers discussed in Section 6.3.2 to propose new configurations or even modified behaviour of the Itus Agent without needing their contribution to be accepted upstream by the core Itus framework.

6.7 Implementation

We implemented the Itus framework in Android Java and the Itus oracle in Java. Itus is distributed as a standalone library of and it can be imported and used by any Android project that supports Android version 2.23 and above. This allows Itus to support the majority of the Android devices in use today (99.9% Android devices in-the-wild as of April, 2016 [Goo16b]). In this section, we discuss some of the significant implementation decisions of Itus.

Event Interception: The API of Itus has been designed to abstract away the underlying low-level event collection from the app developer. While there exist mobile sensor collection frameworks such as SoundSense [LPL+09] and Jigsaw [LYL+10], unfortunately these frameworks exist for Nokia and Apple iOS and not for the Android OS. For event handling in Android Activities, Android provides `EventListener` and `EventHandler` to the app developers. Every user activity in an app is derived from the Android `Activity` class and these events are first delivered to `Activity.dispatchTouchEvent()`. We provide the `SecureActivity` class, which extends the `Activity` class and additionally provides constructs to optionally intercept and copy events before delivering them to the users. App developers who want to provide IA support in their apps are expected to extend `SecureActivity` for transparent event interception.

Data Storage: For permanent storage, Itus uses an app's internal storage. We chose internal storage since: an app's internal storage can only be accessed by the app itself and a malicious app cannot gain access to the training model; and accessing an app's internal storage space does not require any explicit permissions. We require the training model classes be `Serializeable` so they can be written and subsequently read from the permanent storage automatically.

Training Set Size and Retraining: To perform training when the user first interacts with a new app, a sufficient amount of data must be collected. The definition of

'sufficient' here has some room for interpretation, and so we provide two distinct ways for developers to specify the training data threshold: absolutely and empirically. In the absolute case, the developer sets the minimum threshold to some integer value (N), and when **bin_training** receives N samples of `FeatureVector`s, training is triggered. In the empirical case, the developer specifies some minimum accuracy level to be attained during training before considering training to be complete. Here, the Itus Agent runs training periodically and evaluates accuracy using x-fold cross-validation (where x is another parameter) and stops when it achieves the desired target accuracy. This second method is obviously significantly more performance-heavy than simply training at N instances, so the oracle tool in Section 6.5.7 helps developers determine appropriate values of N.

In order to improve the user experience by reducing false rejects, Itus provides app developers with the option to automatically retrain the classifier to improve its accuracy. To this end, Itus first temporarily stores the `FeatureVector`s that are classified as non-owner's by the behavioural classifier. It then triggers the lockout activity to explicitly authenticate the user. If the user successfully authenticates, Itus uses the misclassified feature vectors to retrain the behaviour-based classifier. While retraining improves the user experience, in addition to training overhead, it requires additional storage space to save misclassified `FeatureVector`s.

Lockout Action: In case of authentication failure, the app developer may launch the device's default authentication mechanism (PINs/pass-locks). However, a large number of device owners do not configure pass-locks on their devices and furthermore, if the attacker has gained access to the device, he has already compromised the primary authentication mechanism. Therefore, to support our off-the-shelf design goal, Itus provides app developers with a default `PasswordConfigure` activity, which is displayed when the app is launched for the first time to configure a password that should be used for explicit authentication in case the IA scheme detects misuse. Itus also provides a `LockoutActivity` to lock the app when misuse is detected. The `LockoutActivity` is also used during the training phase to establish the ground truth during data collection. The `LockoutActivity` overrides the `onBackPressed` method of `Activity` to ignore in-app navigation attempts.

In addition to this, we provide the app developer with a more flexible option to deal with authentication failures. The app developer registers a callback object, implementing our `AuthFailedListenter` interface. This interface specifies a single `onAuthFailed` method, which will be called whenever a classifier fails to implicitly authenticate the user. The app developer is then able to handle authentication failure in any manner desired, supporting our flexibility goal from Section 6.4. For example, a browser might choose to delete the session cookies for any websites the user is currently logged in to.

Managing Timeouts: Itus provides app developers with the ability to pause and resume IA to reduce performance overheads and to save battery life. Itus also provides app developers with the functions to configure timeout intervals so that once a user is successfully authenticated, Itus pauses feature collection and training for the specified interval and resumes its operations when interval times out. App developer can also specify whether they want to reset timeouts and resume IA in case of a screen-off event.

Multi-measurements and Multimodal Schemes: For advanced IA scenarios, we enable the app developers to use multi-measurements by employing feature samples from different measurement modules. For example, the SilentSense [BZL+13] uses events from touch input and data from the motion sensor. Itus provides constructs that can be used by the app developer to define relationships between measurement data from different sources (`MultiMeasurement` in Figure 6.2). The app developer simply defines the causal relationship between two events and the resultant `FeatureVector` is automatically generated. Similarly, Itus provides high-level constructs to the app developer to use multiple behavioural classifiers. These constructs can be used by the developers to combine the authentication score from different behavioural classifiers in a pre-condition or majority score setting. For example, the enterprise email client might use a location-based classifier as a pre-condition to trigger a touch-based classifier.

Itus Oracle: The Itus Oracle identifies suitable classifiers, optimal feature sets, and operating threshold recommendations for the app developer. It determines an appropriate classifier by evaluating the accuracy of Itus Prefabs on the data collected during the configuration mode. This decision also takes into account the availability of data for different Itus Prefabs. The optimal feature set for a particular app is determined by calculating information gain for each feature on collected data. For example, for a navigation app, Itus would automatically detect that the direction of a swipe is not a good feature (due to its high variance on sampled data). Finally, the Itus Oracle provides the app developer with the optimum threshold values by determining the operating point with the highest accuracy.

Prefabs Selection: In the future, we envision using the Oracle as a method of curating the implementations of IA presented to app developers. This will allow us to prevent malicious schemes from entering the ecosystem, such as an IA developer creating a keystroke `Measurement` that also functions as a keylogger. However, as running the Oracle or accepting its recommendations are not necessary steps in order to use Itus, this approach allows for researchers to freely experiment with IA, and for IA developers to distribute standalone extensions to Itus. For now, we want to provide a diverse set of prefabs with Itus to demonstrate its extensibility. To this end, we choose a keystroke classifier [FZCS13], a classifier based on touch behaviour (Touchalytics) [FBM+13] and a classifier based on the micro-

movements of the mobile device caused by touch (SilentSense) [BZL+13]. The keystroke classifier and Touchalytics only use user generated events (`KeyEvent` and `TouchEvent`, respectively) while SilentSense uses data that merges user generated events (`TouchEvent`) with periodic events (accelerometer data). Furthermore, the keystroke classifier and Touchalytics use kNN for classification while SilentSense employs SVM for classification.

6.8 Performance Evaluation

To provide a seamless user experience, it is critical for Itus to have minimum performance overhead. Furthermore, since smartphones are power constrained devices, high battery consumption of Itus may reduce its utility. In this section, we first discuss the experimental setup that we use for performance evaluation and we then provide the results of our evaluation.

6.8.1 Experimental Setup

Device Selection: For performance evaluations, we use an HTC Nexus 1 and an LG Nexus 4. The HTC Nexus 1 has Android OS v2.23 (`Gingerbread`) on a 1 GHz processor with 512 MB of RAM. The LG Nexus 4 has Android OS v4.4 (`KitKat`) on a quad-core 1.5 GHz processor with 2 GB of RAM. Our selection of these diverse devices supplies an overview of Itus performance on both old and recent hardware.

Performance Metrics For empirical evaluations, we are interested in the performance of Itus against the IA schemes employed. For performance evaluations, we measure the performance overhead in terms of elapsed CPU time and heap size of the app. We also evaluate the battery consumption overhead of Itus. Finally, we evaluate the impact of Itus on user experience by measuring the relative performance overhead of Itus with our demo apps.

Demo Apps For demo apps to evaluate for performance purposes, we choose: (i) Zirco Browser[2]: an open-source browser with between 50,000 and 100,000 installs at the Google Play Store; and (ii) TextSecure[3]: a popular open-source encrypted communication app with between 100,000 and 500,000 installs at the Google Play Store. These apps were selected as our demo apps for two reasons: (i) both apps manage private data of the

[2] https://play.google.com/store/apps/details?id=org.zirco
[3] https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms

user, and (ii) usage of these apps results in different event types (`TouchEvent` for Zirco and `KeyEvent` for TextSecure), which allows us to test the different classifiers discussed in Section 6.7.

6.8.2 Evaluation Results

Development Overhead: For a developer who is employing Itus Prefabs, the development overhead is minimal. While quantification of development overhead is a non-trivial task, we herein provide our experience of adding Itus to the demo apps. To avoid any bias due to the absence of a learning curve, we only report development overhead in terms of the number of lines of code added/modified. To provide default IA support in Zirco Browser and TextSecure, in addition to importing the Prefab class, we only modified 2 lines of code. As discussed in Section 6.6, the app developer extends the `SecureActivity` class and launches a suitable Prefab (both operations are highlighted in Figure 6.3(a)).

However, if the app developer wants to optimize the classifiers, there would be more development overhead (depending on the type of optimizations). The app developer can use the Itus Oracle to perform optimizations automatically or choose to manually perform optimizations in order to control certain aspects of IA scheme. In Figure 6.3(b), we show one of the possible workflows an app developer might follow to manually employ touch behaviour features for IA using the SVM classifier. The 10 lines of code (Lines 132–144) in Figure 6.3(b) show how the developer: (i) instantiates Itus (Line 132); (ii) defines a list of feature that should be used by the `Measurement` module (Lines 133, 139, 140); (iii) configures parameters of the SVM classifier (Lines 135–137); and (iv) configures Itus to use the `Measurement` and `Classifier` objects, and starts the Itus Agent (Lines 141–144).

Performance Evaluation of Itus: We instrument Itus to measure its performance overhead in terms of elapsed CPU time and size of heap memory for different runtime configurations. Since the operations of Itus depend on events that cannot be accurately controlled manually, for repeatable experiments we use Monkey scripts [Goo16a] to simulate event generation. We repeat each experiment 15 times for three different runtime configurations and report averages.

Table 6.2 and Table 6.3 show that both the keystroke classifier and Touchalytics require low amount of CPU time and heap memory, respectively. More specifically, the feature extraction and classification operations that are triggered for every input event, take under 1 and under 2 ms for the keystroke and Touchalytics classifiers, respectively on the Nexus 4 device. On the other hand, SilentSense — which uses SVM — takes close to 1 and 6 seconds for initialization and training, respectively. The CPU and memory overhead in

```
122 public class MainActivity extends SecureActivity
123     implements IToolbarsContainer, OnTouchListener,
124     IDownloadEventsListener {
125
126     @Override
127     public void onCreate(Bundle savedInstanceState) {
128         super.onCreate(savedInstanceState);
129         INSTANCE = this;
130         /*create an Instance of Touchalytics classifier
131          * & start the Itus thread*/
132         (new Touchalytics()).start();
```

(a) Using a Prefab to provide IA support

```
126     @Override
127     public void onCreate(Bundle savedInstanceState) {
128         super.onCreate(savedInstanceState);
129         INSTANCE = this;
130
131         /*Configure & launch Itus*/
132         Itus itus = new Itus(this);
133         String [] featureList = {"Start X", "Start Y",
134                 "Average Velocity", "Interstroke Time"};
135         Classifier svm = new SVMClassifier(
136             featureList.length, "negative_instances");
137         svm.setFeatureScaling(true);
138
139         Measurement touch = new Touch();
140         touch.setFeatureList(featureList);
141         itus.useMeasurement(touch);
142         itus.setTrainingSize(100);
143         itus.useClassifier(svm);
144         itus.start();
```

(b) Using low-level constructs to provide IA support

Figure 6.3: Itus' development overhead for Zirco Browser

Table 6.2: CPU time in milli-seconds for different configurations of Itus. 95% confidence intervals are provided in parenthesis.

		Initialization	**Feature Extraction**	**Training**	**Classification**
Nexus 1	Keystroke	21 (2)	<1 (≃0)	<1 (≃0)	<1 (≃0)
	Touchalytics	5 (≃0)	<1 (≃0)	65 (2)	2 (≃0)
	SilentSense	1162 (81)	<1 (≃0)	10384 (91)	<1 (≃0)
Nexus 4	Keystroke	12 (≃0)	<1 (≃0)	<1 (≃0)	<1 (≃0)
	Touchalytics	3 (≃0)	<1 (≃0)	15 (≃0)	1 (≃0)
	SilentSense	972 (67)	<1 (≃0)	5937 (329)	<1 (≃0)

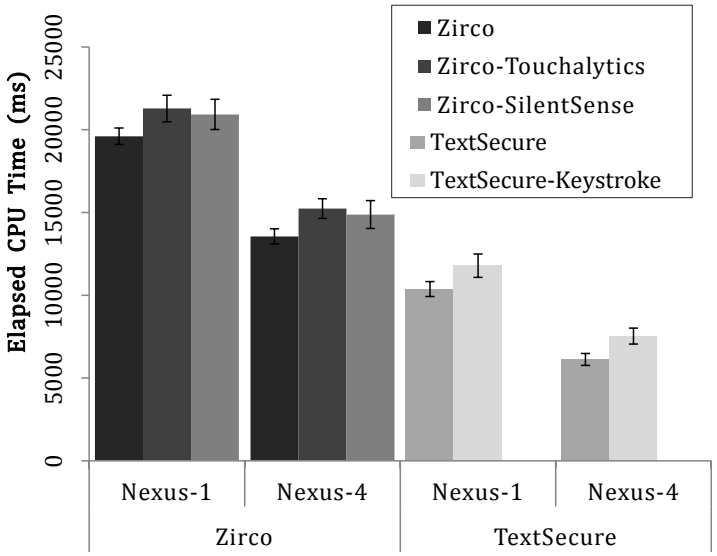
Table 6.3: Heap memory in kBytes for different configurations of Itus. 95% confidence intervals are provided in parenthesis.

		Initialization	**Feature Extraction**	**Training**	**Classification**
Nexus 1	Keystroke	185 (6)	<1 (≃0)	<1 (≃0)	3 (≃0)
	Touchalytics	17 (3)	8 (≃0)	49 (1)	51 (1)
	SilentSense	1236 (15)	15 (1)	2472 (18)	4 (≃0)
Nexus 4	Keystroke	186 (2)	<1 (≃0)	<1 (≃0)	3 (≃0)
	Touchalytics	16 (≃0)	11 (≃0)	53 (≃0)	56 (5)
	SilentSense	776 (34)	17 (≃0)	2453 (39)	4 (≃0)

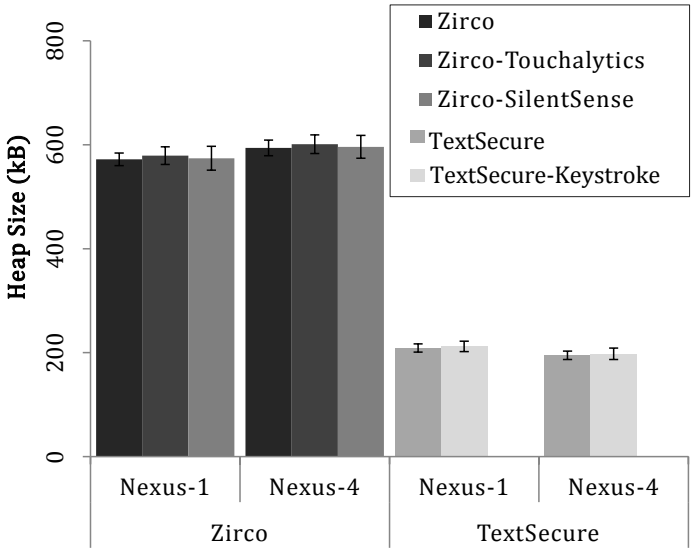
the initialization process is due to the loading of negative instances from an app's internal storage. However, this overhead is incurred only once; after the creation of the training model, feature extraction and classification takes less than a millisecond for a swipe. The execution results show that both devices are able to extract features, and classify in a reasonable amount of time.

Battery Consumption Overhead of Itus: We use PowerTutor [ZTQ+10] to measure battery consumption overhead by Itus. Micro-level overheads are recorded only during individual user input events and are computed relative to the individual demo apps, while macro-level overheads are recorded across a longer period of usage and computed relative to the device. For reproducible experiments, we do not perform any network operations (i.e., swipes are made on pre-downloaded pages in Zirco and for TextSecure the typed message is not transmitted). Finally, battery consumption overhead results do not include the one-time training cost of Itus prefabs.

For overhead at the micro-level, we measure the power consumption of the Keystroke prefab by generating 160 keystroke events for classification. Similarly, we generate 40 swipe events for classification by the Touchalytics and SilentSense prefabs. Our empirical



(a) CPU Overhead



(b) Memory Overhead

Figure 6.4: Itus' CPU and memory overhead for demo apps. Error bars represent 95% confidence intervals.

Table 6.4: Battery consumption overhead of Itus Prefabs on demo apps with a Nexus 1 device. Standard deviations are calculated across every hour of testing for each of 12 hours, and are shown in parentheses.

	Overhead on the app	**Overhead on the device**
Keystroke	39% (3)	1% (≃0)
Touchalytics	9% (≃0)	4% (≃0)
SilentSense	14% (1)	6% (1)

evaluation results, provided in the "overhead on the app" column of Table 6.4. The battery overhead for TextSecure is significantly higher because the battery consumption for normal operation is negligible (unlike Zirco which requires relatively expensive graphic rendering when swiped).

Computation of the macro-level overhead is a non-trivial task due to the large number of variables involved, such as the number of running apps, network connection (3G/WiFi) and the usage of other apps by the device user. To mitigate these variations, we create a simple setup in which Nexus 1 devices are only executing the demo app, the Google Mail app, the Google Talk app and the Launcher app. These devices are also executing the default supporting systems Network Location, User Dictionary, Media Server and the Radio and WiFi subsystems. Monkey scripts [Goo16a] were used to generate 40 swipe and 160 keystroke events after every 10 minute interval for 12 hours. Table 6.4 provides the results for the macro level overhead in the "overhead on the device" column. It can be observed from these results that all Itus Prefabs incur reasonably low overhead (1%, 4% and 6% for Keystroke, Touchalytics and SilentSense prefabs, respectively) even on Android devices with a minimal set of apps running.

Performance Overhead on Demo Apps: In addition to the performance evaluation of standalone Itus, we also measure the performance overhead imposed by Itus on two demo apps. The objective of this evaluation is to show that the relative performance overhead of Itus is negligible enough to not compromise user experience. We instrument and measure elapsed CPU time and heap memory size for the demo apps with and without Itus. The experiment with Zirco was conducted by accessing the BBC and CNN homepages using Zirco and swiping 20 times on each website without any delays. The experiment with the TextSecure app was conducted by composing a text message of 160 characters (network transmission was not included). Figures 6.4(a), 6.4(b) show the CPU overhead and memory overhead for demo apps, respectively. It can be observed from Figure 6.4 that the CPU and memory overhead for Itus is acceptable and Itus can be used without compromising user experience.

6.9 Limitations

There are a few limitations to deployment the deployment of Itus on old and low-end Android devices. Processing intensive IA schemes on Android devices with low-end processors might affect user experience. Similarly, for IA schemes that rely on sensor data from different on-board sensors, the accuracy of Itus will be negatively affected on some of the old devices with low sampling rates. However, these are trivial limitations given the high penetration of modern Android devices.

Another limitation of our approach is that each instance of Itus executes in isolation from other instances on the same device. Therefore, every app that uses Itus requires a separate training model and an instance of the Itus library in memory. This requires additional storage and increases apps' memory footprints. Finally, while independent execution of multiple Itus instances precludes any information sharing across these instances, this is by design to prevent any potential security issues arising from malicious apps.

Itus deployment is of course subject to all the limitations of existing IA mechanisms. For example, if an intruder comes across a device resting on a stable surface and protected by the SilentSense prefab, there would be no movement data available to classify him. Similarly, it may be possible for attackers to use mimicry attacks to defeat IA (similar to the ones uncovered in Chapter 5). One of the benefits of providing an extensible IA framework is that it can easily incorporate defences against mimicry attacks as soon as they are proposed by security researchers. Furthermore, the low-level constructs provided by Itus can be leveraged to identify mimicry resilient features to further research on building secure IA schemes that are not vulnerable to such attacks.

6.10 Conclusion

We have proposed Itus, a framework for providing IA support on smartphones, and provided an open-source implementation for Android. Itus separates the domain knowledge of IA from its deployment to bridge the gap between IA research and practice. The architecture of Itus is designed with flexibility in mind for app developers, allowing them choice between modular subcomponents implementing different mechanisms for behavioural feature classification. The Itus framework is easily extensible to allow IA developers to easily provide such subcomponents. Itus also provides app developers with the ability to optionally configure the behaviour of the IA mechanism to their application's needs by using an Oracle program to determine suitable classifiers and configuration parameters. The API of

Itus has been designed to effortlessly provide IA in Android apps. Empirical evaluations of Itus in real-world demo apps show that it has acceptably low overhead. We have made Itus publicly available in open-source for Android. We hope that Itus will enable the research community to collaborate better to further research in the IA domain and also to enable the adoption of IA.

Chapter 7

Conclusion and Future Work

To conclude this thesis, we summarize our research contributions, discuss research directions based on this work and offer concluding remarks.

7.1 Research Contributions

Our major objective was to perform a realistic evaluation of the usability, security, and deployability related issues of IA. To this end, we reviewed existing IA literature to uncover serious limitations including: (1) IA proposals are evaluated on limited evaluation criteria on synthetic datasets and weak adversarial models; (2) the existing IA literature on usability evaluations reports results only on lab-based experiments and without briefing the participants on the limitations of IA; (3) shoulder surfing attacks on IA have been dismissed without empirical evaluations; and (4) the existing IA framework proposals have failed to provide any working implementation for IA deployments. Some of these limitations have led to misleading results in terms of the efficacy of IA.

To further gain insight into these limitations, we evaluated six diverse IA schemes on real-world datasets using eight evaluation metrics including: (1) accuracy; (2) data availability; (3) training delay; (4) detection delay; (5) CPU and memory overhead; (6) uniqueness of behavioural features; (7) vulnerability to mimicry attacks; and (8) deployment issues on mobile platforms. Our results showed that while the majority of IA schemes provided reasonable accuracy with low detection delay, touch IA outperformed others by providing near real-time misuse detection with high accuracy. We also found that except touch IA schemes, IA schemes frequently did not have enough data available for classification and incurred significant training and detection delays. These findings provide a holistic

picture of the performance of IA and confirm the significance of the extended evaluation criteria. Similarly, we noted lower accuracy than the accuracy reported in the original papers, which indicates the importance of using real-world datasets for evaluations.

We performed lab- and field-based evaluations to establish the usability and security perceptions of IA. To this end, we found no significant difference between IA and explicit authentication using the system usability scale, although more users agreed that IA was more convenient. We also found that the majority of our participants were interested in adopting or trying IA with possibility of adoption. On the other hand, annoyance was a potential issue with IA in terms of usability. Our debriefing of IA along with its limitations and the field experience of the participants also uncovered that the detection delay and false accepts were a security concern.

We scrutinized the weak adversary model by evaluating two realistic attack scenarios from malicious insiders on touch IA. Our evaluation showed that it is surprisingly easy to bypass these schemes using shoulder surfing and offline training attacks. We showed the fallacy of the accepted assumption that shoulder surfing attacks on touch IA are infeasible due to the hidden nature of some features. We also make available the necessary apparatus for researchers to evaluate these attacks on future IA proposals.

Finally, we identified some of the deployment challenges to IA and showed there is a need for a framework that supports: (1) different behavioural classifiers, given that different apps have different requirements; (2) app developers using IA without becoming domain experts; and (3) real-time classification on resource-constrained mobile devices. We developed Itus, an IA framework for Android that allows the research community to further research in the IA domain without excessive re-engineering while allowing app developers to adopt these improvements with minimum effort.

7.2 Future Research

We have described several possible avenues for future work in the previous chapters. We now consolidate these to outline key challenges that we believe are detrimental to IA adoption.

In Chapter 3, we noted the significance of using real-world traces for the evaluation of IA schemes. We also noted the unavailability of publicly available, real-world datasets of user behaviour on smartphones. A major factor that prohibits data sharing for a majority of data collection campaigns is privacy of the users. A system that gathers, curates and

provides access to unbiased behavioural data while preserving privacy of the users can prove to be beneficial to the IA research community.

Chapter 4 provided empirical evidence for the human side of IA. However, our focus was only on the accuracy and detection delay metrics. Other metrics identified in Chapter 3 (such as training delay or battery consumption due to CPU overhead) may also contribute to the users' willingness to (not) adopt IA. Furthermore, we observed that the detection delay was a concern for several users. The emphasis of multi-modal IA proposals is to improve the accuracy. Instead, multi-modal IA proposals should intelligently combine data from various sources to reduce the detection delay. It is important that the detection delay is used as a critical evaluation metric for multi-modal IA proposals to alleviate users' concern regarding the security of IA.

Our usability evaluations of IA were based on a pseudo-IA scheme. This enabled us to control the frequency and timings of false rejects during the three day field study. While we chose realistic false reject rates, the behaviour of the pseudo-IA scheme was different from a real IA scheme. A longer evaluation study that employs a fully operational IA scheme to further demonstrate true rejects needs to be performed to establish more accurate security perceptions and willingness to adopt IA.

In Chapter 5, we established the susceptibility of touch IA to shoulder surfing and offline training attacks. Similar attacks were successfully mounted on gait IA (see Section 2.4.8). However, it is unclear whether keystroke IA and multi-modal IA schemes are subject to similar attacks. Furthermore, our investigations of touch IA showed that some features were easier to mimic than others. It needs to be established whether a multi-modal fusion of behavioural features or filtering of easier to mimic features will reduce the severity of these attacks. Establishing the mimicry resilience of IA, in our opinion, is the most pressing avenue of future work in the IA domain.

Chapter 6 introduced Itus, which can be used by the IA research community to further research in IA and by app developers to provide IA support on off-the-shelf Android devices. In terms of IA deployment in the enterprise environment, there are scenarios where it is beneficial to share the authentication score or behavioural data across apps. For instance the enterprise calendar and email apps can share some of Itus' state information (such as the most recent authentication score) to better authenticate the users. It would be useful to provide interfaces in Itus to enable state sharing across apps from the same app providers.

We also introduced the Itus Oracle, which performs automated evaluations to determine a suitable classifier based on the data availability and accuracy. The Itus Oracle can be extended to support other evaluation metrics that we identified in Chapter 3. By

supporting other evaluation metrics, Itus will provide a holistic view of IA schemes' performance. Finally, Itus requires ongoing maintenance from the research community in terms of novel measurements and corresponding features.

7.3 Conclusion

Our goal in this thesis was to perform a realistic evaluation of IA to determine whether state of the art IA proposals can mitigate the usability and security limitations of primary authentication mechanisms. However, our evaluations indicated that several critical challenges in terms of security properties and usability of IA need to be resolved before IA is ready for wide scale deployments. To help resolve these challenges, we provide design guidelines and an open source framework. We emphasize the need for more comprehensive and rigorous evaluations of IA with active involvement from human subjects (for usability and stronger adversarial model perspective) than done by the current state of the art.

References

[AGM+10] Adam J Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. In *4th Usenix Conference on Offensive Technologies*. Usenix Association, 2010.

[AGSN15] Asadullah Al Galib and Reihaneh Safavi-Naini. User authentication using human cognitive abilities. In *18th International Conference on Financial Cryptography and Data Security*. Springer, 2015.

[AKH16] Lalit Agarwal, Hassan Khan, and Urs Hengartner. Ask me again but don't annoy me: Evaluating re-authentication strategies for smartphones. In *12th Symposium on Usable Privacy and Security*. USENIX Association, 2016.

[AMN+98] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[And16] Android Authority. Android Jelly Bean Face Unlock liveness check easily hacked with photo editing. http://www.androidauthority.com/android-jelly-bean-face-unlock-blink-hacking-105556/, March 2016.

[App16] AppLock. Third party app to password protect apps. https://play.google.com/store/apps/details?id=com.sp.protector.free, March 2016.

[ARKR13] Kumar Abhishek, Sahana Roshan, Prabhat Kumar, and Rajeev Ranjan. A comprehensive study on multifactor authentication schemes. In *Advances in Computing and Information Technology*. Springer, 2013.

[ATOY13] Panagiotis Andriotis, Theo Tryfonas, George Oikonomou, and Can Yildiz. A pilot study on the security of pattern screen-lock methods and soft side

channel attacks. In *6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2013.

[BAKS+11] Noam Ben-Asher, Niklas Kirschnick, Hanul Sieger, Joachim Meyer, Asaf Ben-Oved, and Sebastian Möller. On the need for different security methods on mobile phones. In *International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2011.

[BC94] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, 1994.

[BCVO12] Robert Biddle, Sonia Chiasson, and Paul C Van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys*, 44(4):19, 2012.

[BDG14] Senaka Buthpitiya, Anind K Dey, and Martin Griss. Soft authentication with low-cost signatures. In *International Conference on Pervasive Computing and Communications*. IEEE, 2014.

[BDLA15] Daniel Buschek, Alexander De Luca, and Florian Alt. Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices. In *33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015.

[BDLA16] Daniel Buschek, Alexander De Luca, and Florian Alt. Evaluating the influence of targets and hand postures on touch-based behavioural biometrics. In *34th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2016.

[BH14] Ulrich Burgbacher and Klaus Hinrichs. An implicit author verification system for text messages based on gesture typing biometrics. In *32nd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2014.

[BHVOS12] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*. IEEE, 2012.

[BHvOS15] Joseph Bonneau, Cormac Herley, Paul C van Oorschot, and Frank Stajano. Passwords and the evolution of imperfect authentication. *Communications of the ACM*, 58(7):78–87, 2015.

[BK06] Brian P Bailey and Joseph A Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, 22(4), 2006.

[BLM07] Lucas Ballard, Daniel Lopresti, and Fabian Monrose. Forgery quality and its implications for behavioral biometric security. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(5):1107–1118, 2007.

[Bro96] John Brooke. SUS – a quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194), 1996.

[BUI+15] Chandrasekhar Bhagavatula, Blase Ur, Kevin Iacovino, Su Mon Kywe, Lorrie Faith Cranor, and Marios Savvides. Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption. In *NDSS Workshop on Usable Security*, 2015.

[BW12] Salil P Banerjee and Damon L Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[BZL+13] Cheng Bo, Lan Zhang, Xiang-Yang Li, Qiuyuan Huang, and Yu Wang. Silentsense: silent user identification via touch and movement behavioral biometrics. In *19th Annual International Conference on Mobile Computing & Networking*. ACM, 2013.

[CC11] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Usenix HotSec*. Usenix Association, 2011.

[CC13] Hsin-Yi Chiang and Sonia Chiasson. Improving user authentication on mobile devices: a touchscreen graphical password. In *15th International Conference on Human-computer Interaction with Mobile Devices and Services*. ACM, 2013.

[CF07] Nathan Clarke and Steven Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1), 2007.

[CG96] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996.

[CH67] Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[CK13] Tao Chen and Min-Yen Kan. Creating a live, public short message service corpus: The NUS SMS corpus. *Language Resources and Evaluation*, 47(2):299–335, 2013.

[CKF09] Nathan Clarke, Sevasti Karatzouni, and Steven Furnell. Flexible and transparent user authentication for mobile devices. In *Emerging Challenges for Security, Privacy and Trust*. Springer, 2009.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[CPH11] Prima Chairunnanda, Nam Pham, and Urs Hengartner. Privacy: Gone with the typing! identifying web users by their typing patterns. In *IEEE 3rd International Conference on Social Computing Privacy, Security, Risk and Trust*. IEEE, 2011.

[CR14] Heather Crawford and Karen Renaud. Understanding user perceptions of transparent authentication on a mobile device. *Journal of Trust Management*, 1(7), 2014.

[CRS13] Heather Crawford, Karen Renaud, and Tim Storer. A framework for continuous, transparent mobile device authentication. *Computers & Security*, 39:127–136, 2013.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CvO13] Jeremy Clark and Paul C van Oorschot. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *IEEE Symposium on Security and Privacy*. IEEE, 2013.

[DHA10] Paul Dunphy, Andreas P Heiner, and N Asokan. A closer look at recognition-based graphical passwords on mobile devices. In *6th Symposium on Usable Privacy and Security*. ACM, 2010.

[DHS12] Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification*. John Wiley & Sons, 2012.

[DKP14] Dimitrios Damopoulos, Georgios Kambourakis, and Georgios Portokalidis. The best of both worlds: a framework for the synergistic operation of host and cloud anomaly-based IDS for smartphones. In *7th European Workshop on System Security*. ACM, 2014.

[DLHB+12] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and I know it's you!: implicit authentication based on touch screen patterns. In *30th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2012.

[DLHVZH15] Alexander De Luca, Alina Hang, Emanuel Von Zezschwitz, and Heinrich Hussmann. I feel like I'm taking selfies all day!: Towards understanding biometric authentication on smartphones. In *33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015.

[Dro16] DroidLife. Smart lock. http://www.droid-life.com/tag/smart-lock/, March 2016.

[DZZ14] Benjamin Draffin, Jiang Zhu, and Joy Zhang. Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction. In *Mobile Computing, Applications, and Services*. Springer, 2014.

[EJP+14] Serge Egelman, Sakshi Jain, Rebecca S Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. Are you ready to lock? In *ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2014.

[ERLM15] Simon Eberz, Kasper B Rasmussen, Vincent Lenders, and Ivan Martinovic. Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics. In *22nd Annual Network & Distributed System Security Symposium*, 2015.

[FBM+13] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transactions on Information Forensics and Security*, 8(1):136–148, 2013.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3), 1997.

[FKK10] Denis Foo Kune and Yongdae Kim. Timing attacks on pin input devices. In *ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2010.

[FLK+12a] Tao Feng, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang, and Ngac Ky Nguyen. Continuous mobile authentication using touchscreen gestures. In *Symposium on Technologies for Homeland Security*. IEEE, 2012.

[FLK+12b] Tao Feng, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang, and Nhung Nguyen. Continuous mobile authentication using touchscreen gestures. In *Symposium on Technologies for Homeland Security*. IEEE, 2012.

[FMP10] Jordan Frank, Shie Mannor, and Doina Precup. Activity and gait recognition with time-delay embeddings. In *AAAI Conference on Artificial Intelligence*, 2010.

[FWGK15] Lex Fridman, Steven Weber, Rachel Greenstadt, and Moshe Kam. Active authentication on mobile devices via stylometry, application usage, web browsing, and gps location. *arXiv preprint arXiv:1503.08479*, 2015.

[FYY+14] Tao Feng, Jun Yang, Zhixian Yan, Emmanuel Munguia Tapia, and Weidong Shi. Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In *15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014.

[FZCS13] Tao Feng, Xi Zhao, Bogdan Carbunar, and Weidong Shi. Continuous mobile authentication using virtual key typing biometrics. In *12th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013.

[FZD+15] Tao Feng, Xi Zhao, Nick DeSalvo, Tzu-Hua Liu, Zhimin Gao, Xi Wang, and Weidong Shi. An investigation on touch biometrics: Behavioral factors on screen size, physical context and application context. In *International Symposium on Technologies for Homeland Security*. IEEE, 2015.

[Gar16] Gartner, Inc. Predicts 2014: Mobile and wireless. https://www.gartner.com/doc/2620815/predicts--mobile-wireless, March 2016.

[GHS06] Davrondzhon Gafurov, Kirsi Helkala, and Torkjel Søndrol. Biometric gait authentication using accelerometer sensor. *Journal of Computers*, 1(7):51–59, 2006.

[GMCB14] Cristiano Giuffrida, Kamil Majdanik, Mauro Conti, and Herbert Bos. I sensed it was you: Authenticating mobile users with sensor-enhanced keystroke dynamics. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014.

[Goo16a] Google Play. Android UI/Application Exerciser Monkey. https://developer.android.com/tools/help/monkey.html, March 2016.

[Goo16b] Google Play. Google play install stats. http://developer.android.com/about/dashboards/index.html, March 2016.

[GS09] Barney G Glaser and Anselm L Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Transaction Publishers, 2009.

[GSB07] Davrondzhon Gafurov, Einar Snekkenes, and Patrick Bours. Spoof attacks on gait authentication system. *IEEE Transactions on Information Forensics and Security*, 2(3):491–502, 2007.

[HCP09] Seong-seob Hwang, Sungzoon Cho, and Sunghoon Park. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1):85–93, 2009.

[HDBDJ96] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. PWS Publishing Company, 1996.

[HDLE16] Marian Harbach, Alexander De Luca, and Serge Egelman. The anatomy of smartphone unlocking. In *34th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2016.

[HDLME16] Marian Harbach, Alexander De Luca, Nathan Malkin, and Serge Egelman. Keep on lockin' in the free world: A multi-national comparison of smartphone locking. In *34th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2016.

[Hol79] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[HRS+12] Eiji Hayashi, Oriana Riva, Karin Strauss, AJ Brush, and Stuart Schechter. Goldilocks and the two mobile devices: going beyond all-or-nothing access to a device's applications. In *8th Symposium on Usable Privacy and Security*. ACM, 2012.

[HVZF+14] Marian Harbach, Emanuel Von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. It's a hard lock life: A field study of smartphone (un) locking behavior and risk perception. In *10th Symposium on Usable Privacy and Security*, 2014.

[HWY+15] Feng Hong, Meiyu Wei, Shujuan You, Yuan Feng, and Zhongwen Guo. Waving authentication: your smartphone authenticate you on motion gesture. In *33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2015.

[Jol05] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[JRP04] Anil K Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1), 2004.

[JSGC09] Markus Jakobsson, Elaine Shi, Philippe Golle, and Richard Chow. Implicit authentication for mobile devices. In *4th Usenix Conference on Hot Topics in Security*. Usenix Association, 2009.

[JXBJ+12] Felix Juefei-Xu, Chandrasekhar Bhagavatula, Aaron Jaech, Unni Prasad, and Marios Savvides. Gait-id on the move: pace independent human identification using cell phone accelerometer dynamics. In *5th International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2012.

[KAH14a] Hassan Khan, Aaron Atwater, and Urs Hengartner. A comparative evaluation of implicit authentication schemes. In *Recent Advances in Intrusion Detection*. Springer, 2014.

[KAH14b] Hassan Khan, Aaron Atwater, and Urs Hengartner. Itus: an implicit authentication framework for android. In *20th Annual International Conference on Mobile Computing & Networking*. ACM, 2014.

[KH14] Hassan Khan and Urs Hengartner. Towards application-centric implicit authentication on smartphones. In *15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014.

[KJB+14] Hilmi Günes Kayacık, Mike Just, Lynne Baillie, David Aspinall, and Nicholas Micallef. Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors. In *Mobile Security Technologies*, 2014.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[KPJ15] Rajesh Kumar, Vir V Phoha, and Anshumali Jain. Treadmill attack on gait-based authentication systems. In *8th International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2015.

[KWM10] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Cell phone-based biometric identification. In *International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2010.

[KY04] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, 2004.

[LCL13] Chien-Cheng Lin, Chin-Chun Chang, and Deron Liang. A novel non-intrusive user authentication method based on touchscreen of smartphones. In *International Symposium on Biometrics and Security Technologies*. IEEE, 2013.

[LCPD10] Fudong Li, Nathan Clarke, Maria Papadaki, and Paul Dowland. Behaviour profiling on mobile devices. In *International Conference on Emerging Security Technologies*. IEEE, 2010.

[LL15] Li Lu and Yongshuai Liu. Safeguard: User reauthentication on smartphones via behavioral biometrics. *IEEE Transactions on Computational Social Systems*, 2(3):53–64, 2015.

[Loo16] Lookout Blog. Sprint and lookout consumer mobile behavior survey. http://blog.lookout.com/blog/2013/10/21/sprint-and-lookout-survey/, March 2016.

[LPL+09] Hong Lu, Wei Pan, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *7th International Conference on Mobile systems, Applications, and Services*. ACM, 2009.

[LYL+10] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010.

[LZX13] Lingjun Li, Xinxin Zhao, and Guoliang Xue. Unobservable reauthentication for smart phones. In *20th Network and Distributed System Security Symposium*, 2013.

[MBK+13] Ildar Muslukhov, Yazan Boshmaf, Cynthia Kuo, Jonathan Lester, and Konstantin Beznosov. Know your enemy: the risk of unauthorized access in smartphones by insiders. In *International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2013.

[MCGCN] Emanuele Maiorana, Patrizio Campisi, Noelia González-Carballo, and Alessandro Neri. Keystroke dynamics authentication for mobile phones. In *Symposium on Applied Computing*. ACM.

[MJB+15] Nicholas Micallef, Mike Just, Lynne Baillie, Martin Halvey, and Hilmi Güneş Kayacik. Why arent users using protection? investigating the usability of smartphone locking. In *17th International Conference on Human-computer Interaction with Mobile Devices and Services*, 2015.

[MKV+13] Michelle L Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. Measuring password guessability for an entire university. In *ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013.

[MLV+05] Jani Mantyjarvi, Mikko Lindholm, Elena Vildjiounaite, S-M Makela, and HA Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2. IEEE, 2005.

[MM13] Muhammad Muaaz and René Mayrhofer. An analysis of different approaches to gait recognition using cell phone based accelerometers. In *International Conference on Advances in Mobile Computing & Multimedia*. ACM, 2013.

[MM14] Muhammad Muaaz and Rene Mayrhofer. Orientation independent cell phone based gait authentication. In *12th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2014.

[MMMC+14] Shrirang Mare, Andres Molina-Markham, Cory Cornelius, Ronald Peterson, and David Kotz. Zebra: Zero-effort bilateral recurring authentication. In *IEEE Symposium on Security and Privacy*. IEEE, 2014.

[MSBF15] Rahul Murmuria, Angelos Stavrou, Daniel Barbará, and Dan Fleck. Continuous authentication on mobile devices using power consumption, touch gestures and physical movement of users. In *Research in Attacks, Intrusions, and Defenses*. Springer, 2015.

[MVBC12] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. In *10th International Conference on Mobile Systems, Applications, and Services*. ACM, 2012.

[New16] New Scientist. Touchscreen phones know it's you from taps and swipes. http://www.newscientist.com/article/dn24193-touchscreen-phones-know-its-you-from-taps-and-swipes.html, March 2016.

[NWS15] Tempestt J Neal, Damon L Woodard, and Aaron D Striegel. Mobile device application, bluetooth, and wi-fi usage data as behavioral biometric traits. In *8th International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2015.

[OHD+12] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *12th Workshop on Mobile Computing Systems and Applications*. ACM, 2012.

[PP14] Saurabh Panjwani and Achintya Prakash. Crowdsourcing attacks on biometric systems. In *10th Symposium On Usable Privacy and Security*. Usenix Association, 2014.

[RHM15] Aditi Roy, Tzipora Halevi, and Nasir Memon. An hmm-based multi-sensor approach for continuous mobile authentication. In *IEEE Military Communications Conference*. IEEE, 2015.

[RNI95] Stuart Russell, Peter Norvig, and Artificial Intelligence. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[RQSL12] Oriana Riva, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. Progressive authentication: deciding when to authenticate on mobile phones. In *21st USENIX Security Symposium*, 2012.

[SBAIM12] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, and Nasir Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *31st Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2012.

[Sch16] Schneier on Security. Using the iwatch for authentication. https://www.schneier.com/blog/archives/2013/02/using_the_iwatc.html, March 2016.

[SCY+14] Michael Sherman, Gradeigh Clark, Yulong Yang, Shridatt Sugrim, Arttu Modig, Janne Lindqvist, Antti Oulasvirta, and Teemu Roos. User-generated free-form gestures for authentication: Security and memorability. In *12th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2014.

[SDW12] Florian Schaub, Ruben Deyhle, and Michael Weber. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 2012.

[SLM+13] Aaron Striegel, Shu Liu, Lei Meng, Christian Poellabauer, David Hachen, and Omar Lizardo. Lessons learned from the netsense smartphone study. In *5th ACM workshop on HotPlanet*. ACM, 2013.

[SLS13] Muhammad Shahzad, Alex X Liu, and Arjmand Samuel. Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. In *19th Annual International Conference on Mobile Computing & Networking*. ACM, 2013.

[SNJC10] Elaine Shi, Yuan Niu, Markus Jakobsson, and Richard Chow. Implicit authentication through learning user behavior. In *Information Security*. Springer, 2010.

[SP13a] Abdul Serwadda and Vir V Phoha. Examining a large keystroke biometrics dataset for statistical-attack openings. *ACM Transactions on Information and System Security*, 16(2):8, 2013.

[SP13b] Abdul Serwadda and Vir V Phoha. When kids' toys breach mobile phone security. In *ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013.

[Spi83] Charles D Spielberger. Manual for the State-Trait Anxiety Inventory STAI (Form Y). 1983.

[SPW13] Abdul Serwadda, Vir V Phoha, and Zibo Wang. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *6th International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2013.

[SSTA14] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N Asokan. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In *18th International Conference on Financial Cryptography and Data Security*, 2014.

[SSY+16] Zdenka Sitova, Jaroslav Sedenka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran Balagani. Hmog: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security*, 11(5):877–892, 2016.

[Ste14] Paul Steiner. Going beyond mobile device management. *Computer Fraud & Security*, 2014(4):19–20, 2014.

[SWKW13] Florian Schaub, Marcel Walch, Bastian Könings, and Michael Weber. Exploring the design space of graphical passwords on smartphones. In *9th Symposium on Usable Privacy and Security*. ACM, 2013.

[SYJ+11] Weidong Shi, Jun Yang, Yifei Jiang, Feng Yang, and Yingen Xiong. Senguard: Passive user identification on smartphones using multiple sensors. In *7th International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE, 2011.

[TGG13] Chee Meng Tey, Payas Gupta, and Debin GAO. I can be you: Questioning the use of keystroke dynamics as biometrics. 20th Annual Network & Distributed System Security Symposium, 2013.

[Tho12] Gordon Thomson. Byod: enabling the chaos. *Network Security*, 2012(2):5–8, 2012.

[Thr16] Threatpost. Lock Screen Bypass Flaw Found in Samsung Androids. http://threatpost.com/lock-screen-bypass-flaw-found-samsung-androids-030413/77580, March 2016.

[TSK+12] Shari Trewin, Cal Swart, Larry Koved, Jacquelyn Martino, Kapil Singh, and Shay Ben-David. Biometric authentication on a mobile device: a study of user effort, error and task disruption. In *28th Annual Computer Security Applications Conference*. ACM, 2012.

[Vap98] Vladimir Vapnik. *Statistical Learning Theory*. Wiley New York, 1998.

[VGSPJ16] Tom Van Goethem, Wout Scheepers, Davy Preuveneers, and Wouter Joosen. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In *International Symposium on Engineering Secure Software and Systems*. Springer, 2016.

[VZDDL13] Emanuel Von Zezschwitz, Paul Dunphy, and Alexander De Luca. Patterns in the wild: a field study of the usability of pattern and pin-based authentication on mobile devices. In *15th International Conference on Human-computer Interaction with Mobile Devices and Services*. ACM, 2013.

[VZDLH14] Emanuel Von Zezschwitz, Alexander De Luca, and Heinrich Hussmann. Honey, i shrunk the keys: influences of mobile devices on password composition and authentication performance. In *8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*. ACM, 2014.

[Wri12] Scott Wright. The Symantec Smartphone Honey Stick Project. *Symantec Corporation, Mar*, 2012.

[WTNH03] Liang Wang, Tieniu Tan, Huazhong Ning, and Weiming Hu. Silhouette analysis-based gait recognition for human identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1505–1518, 2003.

[XZL14] Hui Xu, Yangfan Zhou, and Michael R Lyu. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *10th Symposium On Usable Privacy and Security*. USENIX Association, 2014.

[YCLO16] Yulong Yangt, Gradeigh D Clarkt, Janne Lindqvistt, and Antti Oulasvirta. Free-form gesture authentication in the wild. In *34th Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2016.

[YGD+15] Lei Yang, Yi Guo, Xuan Ding, Jinsong Han, Yunhao Liu, Cheng Wang, and Changwei Hu. Unlocking smart phone through handwaving biometrics. *IEEE Transactions on Mobile Computing*, 14(5):1044–1055, 2015.

[Zdn16] Zdnet. Apple iPhone fingerprint reader confirmed as easy to hack. http://www.zdnet.com/apple-iphone-fingerprint-reader-confirmed-as-easy-to-hack-7000021065/, March 2016.

[ZFS13] Xi Zhao, Tao Feng, and Weidong Shi. Continuous mobile authentication using a novel graphic touch gesture feature. In *6th International Conference on Biometrics: Theory, Applications and Systems*. IEEE, 2013.

[ZSKF09] Saira Zahid, Muhammad Shahzad, Syed Ali Khayam, and Muddassar Farooq. Keystroke-based user identification on smart phones. In *Recent Advances in Intrusion Detection*. Springer, 2009.

[ZTQ+10] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2010.

[ZWWZ13] Jiang Zhu, Pang Wu, Xiao Wang, and Joy Zhang. Sensec: Mobile security through passive sensing. In *International Conference on Computing, Networking and Communications*. IEEE, 2013.

[ZXL+12] Yang Zhang, Peng Xia, Junzhou Luo, Zhen Ling, Benyuan Liu, and Xinwen Fu. Fingerprint attack against touch-enabled devices. In *2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM, 2012.

APPENDICES

Appendix A

Pre-study Survey for Usability Evaluations

For the pre-study screening, in addition to collecting their name, email address, gender, age group, device that they owned (such as Nexus 4, Samsung SIII, LG G2), profession, domain (such as technology, medicine) and how long they have used an Android device, we asked the participants:

1. Which protection mechanism do you use on your smartphone:

   (a) None; (b) PIN (4 digit or more); (c) Password (characters and numbers); (d) Pattern Lock; (e) Face recognition; (f) Fingerprint recognition; (g) Other (please specify)

2. **IF NO AUTHENTICATION** Why do you not use any protection mechanism (choose all that apply):

   (a) It's too much of a hassle / takes time; (b) There is nothing on my phone that I need to hide; (c) No one would care about what's on my phone; (d) In an emergency, others can use my phone; (e) I've never thought about it; (f) Other (please specify)

3. **IF SOME AUTHENTICATION** Which of the following scenarios do you want to protect against (choose all that apply):

   (a) Phone protected if stolen; (b) Phone protected if lost (c) Phone protected if misplaced; (d) Phone protected if left unattended (e) Someone casually picking up the phone; (f) Unwanted disclosure, pranks; (g) Other (please specify)

4. **IF SOME AUTHENTICATION** Which of the following describes you (choose all that apply):

   (a) Unlocking my phone is annoying sometimes; (b) I like the idea that my phone is protected from unauthorized access; (c) It takes too much time; (d) Unlocking my phone is easy

5. **IF SOME AUTHENTICATION** Which of the following attackers do you want to protect from (choose all that apply):

   (a) Coworker; (b) Spouse; (c) Roommate; (d) Own children; (e) Other unwanted individual/stranger; (f) Friends; (g) Other (please specify)

6. Do you sometimes take additional measures to protect your smartphone (choose all that apply):

   (a) None; (b) I leave my phone in a safe place before going somewhere; (c) I conceal my smartphone in my clothes or in a bag (d) I have changed security settings on my device to improve security (such as reduced automatic lock time) (e) I have enabled device encryption on my smartphone (f) Other (please specify)

7. Do you share your smartphone with your friends or family members:

   (a) Never; (b) Rarely (once a month); (c) Occasionally (once a week); (d) Daily

Appendix B

Post-study Survey for Usability Evaluations

The following questions were asked in the post-survey conducted after each in-lab session in which a participant tested an authentication scheme (IA or EA). The questions that were only asked after the IA session are marked [**IA only**].

1. How satisfied are you with the level of protection that is provided against: (*5-point Likert scale "Very satisfied"–"Very dissatisfied"*)

   (a) Coworkers; (b) Spouse; (c) Roommate; (d) Own children; (e) Friends; (f) Strangers

2. How satisfied are you with the level of protection that is provided against the following phone states: (*5-point Likert scale "Very satisfied"–"Very dissatisfied"*)

   (a) Lost at public location; (b) Lost at work; (c) Unattended at work; (d) Unattended at home

3. How satisfied are you with the level of protection that is provided when you are performing following tasks on your Smartphone: (*5-point Likert scale "Very satisfied"–"Very dissatisfied"*)

   (a) Online banking; (b) Online shopping; (c) Checking emails; (d) Checking texts; (e) Social networking (Facebook); (f) Checking photo gallery

4. How satisfied are you with the overall level of protection that is provided? (*5-point Likert scale "Very satisfied"–"Very dissatisfied"*)

5. Do you agree with the statement "I think this method takes a lot of time"? (*5-point Likert scale "Strongly agree"–"Strongly disagree"*)

6. Do you agree with the statement "I think this method is annoying"? (*5-point Likert scale "Strongly agree"–"Strongly disagree"*)

7. Do you agree with the statement "I think this method is tiring"? (*5-point Likert scale "Strongly agree"–"Strongly disagree"*)

8. How annoying were the interruptions for authentication? (*5-point Likert scale "Very annoying"–"Not annoying at all"*)

9. [**IA only**] How annoying were the interruptions for authentication as compared to your current authentication method? (*5-point Likert scale "Very annoying"–"Not annoying at all"*)

10. [**IA only**] How secure this method is as compared to no authentication? (*5-point Likert scale "A lot more secure"–"A lot less secure"*)

11. [**IA only**] How secure this method is as compared to your current authentication method? (*5-point Likert scale "A lot more secure"–"A lot less secure"*)

12. [**IA only**] Would you use this authentication method?

    - Yes, I would replace my current scheme with IA
    - Yes, I would use it in addition to my current authentication scheme
    - I may use it
    - No, I will not use it.

Appendix C

SUS Survey for Usability Evaluations

The modified SUS form [Bro96] that was completed by participants after each in-lab session in which they tested an authentication scheme (IA or EA). For each question, participants responded on a 5-point Likert scale ("Strongly agree"–"Strongly disagree").

1. I think that I would like to use this method frequently
2. I found this method unnecessarily complex
3. I thought this method was easy to use
4. I think that I would need the support of a technical person to be able to use this method
5. I thought there was too much inconsistency in this method
6. I would imagine that most people would learn to use this method very quickly
7. I found this method very cumbersome to use
8. I felt very confident using the system
9. I needed to learn a lot of things before I could get going with this system

Appendix D

STAI Survey

The STAI form that was completed by participants before and after each in-lab session in which they tested an authentication scheme (IA or EA). For each question, participants responded on a 4-point Likert scale ("Not at all"–"Very much so").

Directions: A number of statements which people have used to describe themselves are given below. Read each statement and then select the appropriate options to the right of the statement to indicate how you feel right now, that is, at this moment. There are no right or wrong answers. Do not spend too much time on any one statement but give the answer which seems to describe your present feelings best.

1. I feel calm
2. I feel secure
3. I am tense
4. I feel strained
5. I feel at ease
6. I feel upset
7. I am presently worrying over possible misfortunes
8. I feel satisfied

9. I feel frightened
10. I feel comfortable
11. I feel self-confident
12. I feel nervous
13. I feel jittery
14. I feel indecisive
15. I am relaxed
16. I feel content
17. I am worried
18. I feel confused
19. I feel steady
20. I feel pleasant

Appendix E

Semi-structured Interviews for Usability Evaluations

Participants were asked the following open-ended questions during the semi-structured interviews:

E.1 Lab-based experiment

- What did you like about IA?
- What did you dislike about IA?
- Why did you perceive a specific protection level for IA?
- Why do you think IA will provide more/less/same protection as compared to your current scheme?
- Why (or why not) would you use IA?
- **IF NOT SATISFIED WITH IA PROTECTION LEVEL:** Why would you still use IA?
- **IF IA IS ANNOYING:** Why would you still use IA?
- Any particular scenarios where you think IA will be particularly useful/useless?

- **IF INTERRUPT-AUTHENTICATES ARE ANNOYING:** How do you think we can mitigate the annoyance?

E.2 Field study

- How was your longer term experience of IA?
- Have you changed your opinion about IA? If yes, why?
- How annoying were the interruptions for authentication?
- Which apps were you using on your device when the interruptions were particularly annoying?
- Which apps were you using on your device when then interruptions were not annoying?
- Any particular scenarios where you think IA will be particularly useful/useless?
- Did you find the threshold adjustment bar useful? Why or why not?