FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

CASHTAGS: PROTECTING THE INPUT AND DISPLAY OF SENSITIVE DATA

By

MICHAEL J. MITCHELL

A Dissertation submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Spring Semester, 2015

Michael J. Mitchell defended this dissertation on April 15, 2015.

The members of the supervisory committee were:

An-I Andy Wang
Professor Directing Dissertation.

Linda DeBrunner
University Representative

Gary Tyson
Committee Member

Zhi Wang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

To my wife and family for their unconditional support

ACKNOWLEDGMENTS

My deepest gratitude is owed to my advisor and major professor Dr. Andy Wang. From his vision and research inspiration, to financial support, to patience reviewing and commenting on countless publication revision, I am truly fortunate to have such an amazing advisor. Without his guidance and persistent help, this dissertation would not have been possible. He is also responsible for the development of many technical and analytical skills I will carry with me throughout my professional career.

I also express my gratitude to Dr. Peter Reiher of UCLA. His wisdom, insight, and collaboration have truly been an asset to this research. I would also like to thank the members of my dissertation committee. I thank Dr. Gary Tyson for financial support and research experience in the FSU Mobile Lab, and Dr. Zhi Wang for his guidance and additional research opportunities in mobile security. I would also like to thank my university representative and external committee member Dr. Linda DeBrunner. Her engineering background brought alternate perspective and general context into this research.

I would also like to thank the members of the FSU Operation Systems and Storage research group. Their engaging questions and alternate approaches brought unique perspectives into my research and often were able to identify inadequacies and shortcomings that I could not see myself. I am fortunate to have been part of such an intelligent, dedicated, and talented research group.

I also express my gratitude to the current and former FSU Department of Computer Science support staff. In particular, I thank Daniel Clawson, Eleanor McNealy, and Kristan Mcalpin for their assistance handling many administrative issues and for helping me to navigate the academic bureaucracy of the university.

I also thank Ratnesh Patidar, Manik Saini, and Parteek Singh for their assistance with parts of the human subjects' privacy study, as well as the 600 people who took the time to participate in the research study.

This dissertation contains excerpts from the following publications:

[1] Michael Mitchell, Ratnesh Patidar, Manik Saini, Parteek Singh, An-I Andy Wang, and Peter Reiher. “Mobile Usage Patterns and Privacy Implications.” Proceedings of the IEEE 2015 International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications (PerMoby), 2015.

[2] Michael Mitchell, An-I Wang, and Peter Reiher. “Cashtags: Prevent Leaking Sensitive Information through Screen Display.” Technical Report TR-141209, Department of Computer Science, Florida State University, 2014.

[3] Michael Mitchell, An-I Andy Wang, and Peter Reiher. “Mobile Usage Patterns and Privacy Implications.” Technical Report TR-131104, Department of Computer Science, Florida State University, 2013.

All interaction with human subjects was approved by the Florida State University IRB Human Subjects Committee, approval numbers HSC# 2012.8779 and HSC# 2013.10175. Full approval documentation and supplemental data is contained in Appendices A-C.

This work is sponsored in part by NSF CNS-1065127, “Facets: Exploring Semantic Equivalence of Files to Improve Storage Systems”. Opinions, findings, and conclusions or recommendations expressed in this document do not necessarily reflect the views of the NSF, FSU, UCLA, or the U.S. government.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Mobile computing is the new norm. As people feel increasingly comfortable computing in public places such as coffee shops and transportation hubs, the threat of exposing sensitive information increases. While solutions exist to guard the communication channels used by mobile devices, the visual channel remains, to a significant degree, open. Shoulder surfing is becoming a viable threat in a world where users are frequently surrounded by high-power cameras, and where sensitive information from recorded images can be extracted with modest computing power.

In response, this dissertation presents Cashtags: a system to defend against attacks on mobile devices based on visual observations. The system allows users to access sensitive information in public without the fear of visual leaks. This is accomplished by intercepting sensitive data elements before they are displayed on screen, then replacing them with non-sensitive information. In addition, the system provides a means of computing with sensitive data in a non-observable way. All of this is accomplished while maintaining full functionality and legacy compatibility across applications.

CHAPTER ONE

INTRODUCTION

1.1 Motivation

Shoulder surfing is becoming a concern in the context of mobile computing. As mobile devices become increasingly capable, people are able to access a much richer set of applications in public places such as coffee shops and public transportation hubs. Inadvertently, users risk exposing sensitive information to bystanders via the screen display. Personal information exposure can increase the risk of personal, fiscal, and criminal identity theft. Exposing trade or governmental secrets can lead to business losses, government espionage, and other forms of cyber terrorism [12][13][14].

This problem is exacerbated by the ubiquity of surveillance and high-power cameras on mobile devices such as smartphones and emerging wearable computing devices such as Google Glass [57]. Additionally, the trend towards multicore machines, GPUs, and cloud computing makes computing cycles much more accessible and affordable for criminals or even seasoned hobbyists, seeking to extract sensitive information via off-the-shelf visual analysis tools [58].

This dissertation presents the motivation, design, implementation, and evaluation of Cashtags, a system that defends against shoulder surfing threats. With Cashtags, sensitive information will be masked with user-defined aliases, and a user can use these aliases to compute in public. The system is compatible with legacy features such as auto correct, and the deployment model requires no changes to applications and the underlying firmware, with a performance overhead of less than 3%.

1.2 The Shoulder Surfing Threat

The threat of exposing sensitive information on screen to bystanders is real. In a recent visual data survey of IT professionals, 85% of those surveyed admitted there have been cases when they were able to see unauthorized sensitive on-screen data, 82% admitted that there have been cases where their own sensitive on-screen data could be viewed by unauthorized personnel,

and 82% had little or no confidence that users in their organization would protect their screen from sensitive data exposure to unauthorized personnel [1]. These results are consistent with other surveys indicating that 76% of respondents were concerned about people observing their screens in public [2], while 80% admitted that they have attempted to shoulder surf the screen of a stranger in a public location [3].

The future projection of the shoulder-surfing threat is even worse, as mobile devices are replacing desktop computers. Mobile device sales now account for over 73% of annual technical device purchases [4]. Employees more frequently take their work with them on the go; by 2015, the world's mobile worker population will reach 1.3 billion [5]. This is highest in the U.S., where more than 80% of the workforce continues working when they have left the office [6], and figures suggest that 67% of employees regularly access sensitive data outside at places where they cannot do so safely [2]. While some organizations have implemented specific guidelines and practices to reduce this risk, 44% do not have any defined policy addressing these threats [1]. Advances in screen technology further increase the risk of exposure, with many new tablets claiming near 180-degree screen viewing angles [8].

1.3 The Dangers are Everywhere

Visual exposure of sensitive information in the form of observation-based attacks can come in many forms. Mobile devices with cameras are nearly ubiquitous. There now exist more than 3 billion digital camera phones in circulation [4]. These devices are evolving rapidly, with newer models capable of capturing images at over 40 megapixels of resolution and over 10 times optical zoom for under $100 [7]. Visual exposure can also be captured by one of the billions of security devices in existence. These high-resolution and often insecure cameras are everywhere, especially in major metropolitan areas. For example, figures suggest the average resident of London is captured on CCTV over 300 times every day [9]. Finally, but no less threateningly, sensitive data can be exposed by simple human sight.

Observation-based attacks can also be much more complex. Increasingly sophisticated tools and systems have been developed to capture and exploit sensitive user data. Partial images can be merged, sharpened, and reconstructed, even from reflections. Optical Character Recognition (OCR) is becoming much more capable, with over 40 years of innovation. Offline

and cloud-based OCR solutions are highly accurate with only a small percentage of error in recognition. Embedded OCR solutions are inexpensive and capable even on low-end hardware devices [10].

Personal information exposure can also make other attacks possible. The capture of just a small number of personal information elements can greatly increase the risk of other threats including social engineering attacks, phishing, and other personal identity theft threats.

1.4 The Consequences can be Severe

Observation-based information leaks can lead to significant personal and business losses. Recently, an S&P 500 company's profit forecasts were leaked as a result of visual data exposure. The vice president was working on the figures on a flight while sitting next to a journalist [4]. In a different case, British government documents were leaked when a senior officer fell asleep on a train, thereby permitting another passenger to photograph sensitive data on his screen [11]. In another case, security cameras captured the private details of Bank of America clients through the bank's windows [12]. In yet another case, sensitive personal information relating to the United Kingdom's Prince William was captured and published as a result of on-screen exposure to a bystander [13].

The risk of loss from shoulder surfing is also hurting business productivity. Figures show that 57% of people have stopped working in a public place due to privacy concerns and 70% believe their productivity would increase if they felt that no one would be able to see their screen [2].

1.5 Current Solutions

Several techniques have been developed to limit the visual exposure of sensitive private information. However, the primary focus of these systems has been limited to preventing the visual leakage of password entries [22][23][24][25][33][34][35]. Once the user has been successfully authenticated, all accessed sensitive information is displayed in full view. Clearly, such measures are insufficient for general computing in public when the need to access sensitive information arises. Unfortunately, many techniques used to prevent visual password leaks cannot be readily generalized beyond password protection, a situation that motivates this work.

1.6 Thesis

This dissertation supports the following thesis:

The interception of screen display rendering and the use of sensitive data aliases (cashtags) can provide a usable, convenient, efficient, portable, and legacy compatible solution to protect the input and display of sensitive data elements.

1.7 Scope of the Dissertation

Privacy and security enhancing protection mechanisms cover a broad scope. While many alternative approaches are discussed, the focus of this work is limited to visual information leaks and the protection against this threat.

1.8 Contributions

This dissertation contributes to the field of computing privacy in the following ways:

1. Insight into the elusive concept of privacy as it relates to actual user attitudes though a large-scale human subject study. The results of the ~600 person survey lead to these conclusions:
    - People seem to exercise little caution preserving privacy in mobile computing environments: they perform similar computing tasks in both public and private.
    - Privacy and trust are largely orthogonal: people tend to change their computing behavior around people they know.
    - People underestimate the privacy threats of mobile apps, as people comply with permission requests by apps more than operating systems.
    - Users' understanding of privacy is different than that of the security community. Since users should be provided with the privacy protection they want and will actually use, researchers must ensure that their goals align with users' real privacy desires.

2. The design, implementation, and evaluation of Cashtags, a privacy-enhancing system providing these features:
    - Interception of screen display rendering and the prevention of private sensitive personal identification from being displayed on-screen and thus captured by an observer.
    - A mechanism for users to access and input sensitive private data while being directly or indirectly observed by an attacker.
    - Convenience, efficiency, and practical usability for the end user.
    - Support for legacy apps as well as future releases without the need to modify individual apps.

1.9 Outline of the Dissertation

This chapter has introduced the threat of shoulder surfing for the modern mobile user. Chapter 2 further motivates and quantifies the threats of mobile privacy based on actual user attitudes through a survey-based human subjects study. Chapter 3 details other related works and alternate approaches, and how they cannot be generalized to protect against this shoulder surf threat. Chapter 4 describes the user and threat models, and introduces, Cashtags, a system to protect against this unaddressed visual privacy threat. Chapter 5 discusses the general system design of Cashtags, alternate approaches considered and the path to the current system design. Chapter 6 covers the internal implementation details of the Cashtags system. Chapter 7 evaluates the system based on API and market app coverage, as well as performance and usability overhead. Chapter 8 addresses some current system limitations, and directions for possible future work. Chapter 9 summarizes and concludes the dissertation.

CHAPTER TWO

MOBILE PRIVACY SURVEY

2.1 Survey Motivation

One way to quantify the need and nature of shoulder-surfing defense is to understand how people change their computing behavior in the presence of others. This investigation further explores the deeper, elusive concept of privacy, which varies greatly among users. As a consequence, prior privacy research on automated privacy-protection mechanisms is largely confined to less subjective aspects of privacy such as location tracking [81][85] and sharing [83][84][85]. Human subject studies can help discern privacy situations that cannot be discovered automatically, but since such studies can be tedious, relatively few exist. The few examples include, examining perceived risks of application permission requests [86], studies on specific locations [89], and very limited social groups [90].

2.2 Human Subject-based Solutions

Opinions and perception of privacy require an inherent level of human interaction; they cannot easily be collected or deferred from the system logs or Internet histories. However, a very limited number of such surveys involving human subjects have been conducted in the computer science discipline. An even fewer number of such surveys exist that focus on modern mobile usage and privacy concerns. Existing data sets are largely unavailable to privacy concerns and the possibility of sensitive information leaks from survey participants. In addition, the significant changes in the mobile environment in the past few years would make all but the very most recent surveys irrelevant.

Research involving human subjects is itself quite difficult. Permission for such surveys requires approval from Institutional Review Boards (IRB) and human subject committees. These applications are often complex, with detailed documentation. In addition, human subject research also requires investigators to receive special training, and can take months for the approval

process to be finalized. The addition of privacy-oriented issues can serve to make this process even more challenging to receive approval.

2.3 Mobile Usage and Privacy Implications Survey

Motivated by the need for real people to address privacy concerns, a human subjects study was performed. A questionnaire-based survey was used in order to reach a large population relatively inexpensively. The primary goal of this study was to examine how mobile computing users feel about privacy. What does it mean to be private? Do people change their computing behavior in the presence of other people? What types of people? Do people change such behavior in public? Does perception of privacy differ by gender, age, ethnicity, device ownership, or technical backgrounds? How do we quantify the perception of privacy?

All interaction with human subjects was approved by the Florida State University IRB Human Subjects Committee, approval numbers HSC# 2012.8779 and HSC# 2013.10175. Full approval documentation and supplemental data is contained in Appendices A-C.

2.4 Experimental Methodology

2.4.1 Subject Recruitment

Students of Florida State University (FSU) campus were to be the initial participants in the mobile usage and privacy survey. The general recruitment options considered included flyer-based methods, verbal solicitation, and mass emailing. Participants must also be sufficiently motivated to take part in the survey. The compensation must be sufficient to solicit participation, without excessive motivation to for participants to cheat. This could result in undesired data redundancy, or worse, the inaccuracy of collected data. The two general options for consideration were to offer a small reward for every participant, or to use a raffle-based solution with larger prizes.

A combination of flyers posted on campus and mass emails to university departments was selected for recruiting participants for the survey. The reward would be a raffle-based solution with $1,000 worth of $25 Amazon gift cards was chosen as the appropriate compensation for survey participants.

Over 6 weeks (2/1/2013 – 3/15/2013), these efforts resulted in 292 student responses, nearly all (252) from the mass emailing.

However, college students represent a limited subset of all mobile users; college students at FSU most certainly represent an even smaller subset of users. The behaviors, locations for computing, as well as application usages could potentially be quite different from that of the general population.

Thus, expansion of the survey beyond FSU students was necessary to capture more generalized usage and privacy opinions. Possible expansion options included solicitation for participants on Craigslist, or using crowd-sourced services such as Amazon mechanical Turk, or oDesk. Crowd-sourced solutions involve paying participants a small compensation for taking the time to complete the survey. Instead, it was decided to use the same raffle-based Amazon gift cards reward used in the FSU survey, and recruit participants on Craigslist in the top 10 most populous regions of the country.

Participants were solicited through the volunteer section of in the metro area of the ten most populated U.S. cities. Over 6 weeks (6/1/2013 – 7/15/2013), 303 responses were collected from this part of the survey.

2.4.2 Mobile Usage Questionnaire

Survey participants were asked to answer ~100 questions through a web interface [88], with full contents viewable in Appendix D. The questionnaire started by asking about demographic information such as gender, ethnicity, expertise, device ownership, and background knowledge of privacy-enhancing tools such as encryption.

The questionnaire then asked about the frequency of performing 43 mobile activities in seven categories: entertainment (e.g., listen to music), communication (e.g., access emails), productivity (e.g., calendar), tools (e.g., reviews), financial (e.g., online banking), administration (e.g., configure network), and personal (e.g., health monitoring). The user answered whether an activity is performed hourly, daily, weekly, monthly, or never. To estimate the number of accesses per month during waking hours, a summary of the per-user frequency for a given activity within a month was derived using the following formula:

$$\textit{Accesses per month} = (\#hourly*16*30) + (\#daily*30) + (\#weekly*(30/7)) + (\#monthly)$$

As a sanity check, a prior study showed that users on average access their mobile phone 150 times per day [80], and the survey achieved similar results.

For each activity, users were also asked about the frequency of performing the task either in a public setting (defined as with anyone else present) or a private setting (no one else present). These definitions ensured that participants used the same meanings of public/private settings to estimate the frequency of activities. The complete questionnaire contents can be referenced in Appendix D.

2.5 Survey Results Overview

The survey results are based on ~600 users' reported use of mobile devices, attitudes toward privacy in different activities in various situations, and knowledge of existing privacy tools.

The results suggest directions for building mobile privacy mechanisms and exposing areas where more information from users would help them decide how to tailor these mechanisms. Some other major findings were (1) people seem to exercise little caution preserving privacy in mobile computing environments: they perform similar computing tasks in both public and private; (2) privacy is orthogonal to trust: people tend to change their computing behavior around people they know; (3) people underestimate the privacy threats of mobile apps, as people comply with permission requests by apps more than operating systems; (4) users' understanding of privacy is different than that of the security community, suggesting opportunities for additional privacy studies. These observations, and other results are detailed though the remainder of the chapter.

2.6 Demographics and Market Share

2.6.1 Survey Demographics

The subject pool reflects the general population of where the surveys were conducted. Comparison of survey participant demographics to available data from the US Census Bureau [73], Florida State University [74], Craigslist [75], and Alexa [76] are shown in Figure 1.

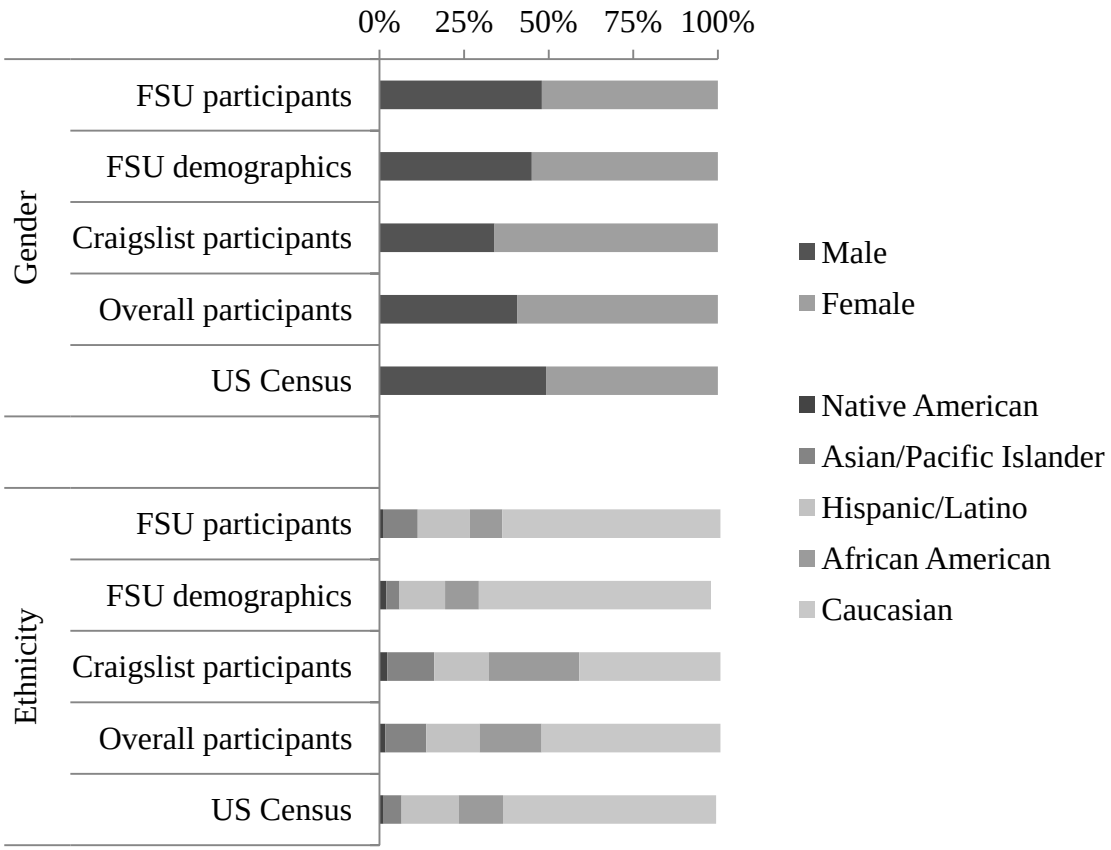

Figure 1: Demographics of survey participants to general population compared with US Census Bureau [73], Florida State University [74], Craigslist [75], and Alexa [76] published data.

FSU survey participants: For the FSU survey, the 292 participants had a median age of 22, with an average of 6 years of computing experience. The gender split of the participants was within 3% of the FSU demographics, with slightly more male participation (Figure 1).

Academic/education background was not quite as characteristic of FSU. The survey had greater participation from computer science (CS)/engineering (by 28%) and undecided/other (by 9%), and correspondingly lower-than- expected participation by literature/language/social science (by 15%) and business-related students (by 8%). This may be due, in part, to the reasonably tech-savvy target audience of the survey, as well as greater access to survey recruitment e-mails and flyers for CS/engineering students.

The ethnicity of the participants largely reflected FSU demographics, except that there were more Asian/Pacific Islander participants (by 7%), possibly due to more participants with a CS/engineering background (Figure 1).

Although these participants may not reflect the findings for the general population, mobile computing nevertheless has the deepest penetration among this age group [91], and this large user base's perception of privacy is worthy of study.

U.S. survey participants: For the U.S. Craigslist survey, the 303 participants had a median age of 27, also with 6 years of computing experience. The gender split was within 8% of U.S. demographics, with more female participation than expected (Figure 1). The ethnicities of survey participants also reflected U.S. demographics. However, the U.S. Craigslist survey had higher minority participation rates (Figure 1).

Although Craigslist has its own bias in terms of user demographics, these users represented a broader age spectrum. Interestingly, the findings for the FSU population are similar to the findings for the Craigslist population. Thus, unless noted, the results reflect the combined 595 responses.

2.6.2 Device Market Share

The smartphone ownership of the survey participants reflected the U.S. market share [77] (3% more iPhone owners and 2% fewer Android phone owners). In addition, tablet ownership of the survey participants reflected the U.S. market share of tablet ownership [78]. Participants in the survey had fewer Android tablets (by 7%) and more iPad/non-Android tablets (by 4%). The demographics of the participants' laptop ownership are less aligned with published data on market share [79]. There were significantly fewer Windows users (by 28%), and significantly more Apple (by 21%) and Linux users (by 7%).

Device ownership: Another research questions was whether device ownership played a role in attitudes toward privacy and mobile usage. To explore this possibility, participants were also split into groups based on the brand of the mobile device they use. The results showed that more often, men, technically-savvy users (defined as participants who work or major in computer science or related fields), and minorities tended to own Android devices (by 10-20%). In addition, men, tech-savvy users, and minorities also more frequently tended to own Windows laptops (by 9-19%). Tech-savvy users owned Android phones, Android tablets, and laptops running Windows or Linux more frequently (by 9-36%) than their less technically savvy peers.

Comparing participants ownerships based on ethnicity, African Americans were found to own iPhones less frequently (by 20%).

Brand homogeneity: Overall, participants tended to own multiple devices from the same manufacturer. iPhone owners more frequently own an Apple laptop or tablet (by 15-28%) compared to non-iPhone owners and Android owners more frequently own an Android tablet (by 15%). The Apple trend was more pronounced in the FSU data set where iPhone owners even more frequently own Apple laptops and tablets (by 15-40%).

2.7 Privacy Results

All results are reported at 95% confidence intervals. Unless otherwise stated, the confidence intervals are within 9% of the mean.

2.7.1 Who makes us change our behavior?

Participants were asked whose presence makes them change their computing behavior. Figure 2 shows that people are most likely to change behavior around their parents, boss, friends, and significant others and least likely to change behavior around subordinates, foreign strangers, roommates, and the technically savvy. A significant number (>10%) do not care who is around and never change their behavior. Women and men behave similarly, but women alter their behavior more often when they are around their parents (by 11%). Tech-savvy users change their behavior more often around their roommates and others who they believe to also be tech-savvy (by 11%).

User hardware preference has the most significant influence over behavioral change around specific people: Apple laptop owners tend to change their behavior around their parents, significant others, friends, and siblings (by 9-16%) more than around other device owners. No significant differences in behavior changes were observed across ethnic groups.

One element of privacy concerns is whether one worries about consequences if certain information is given away. The consequences can be how people perceive you, how people with influence and authority can hold the information against you, etc. Trust of strangers may reflect that the availability of such privacy information to them would be inconsequential. This attitude may also reflect complacency toward the potential privacy threats that strangers can pose. More

technically savvy people seem to be more aware of such threats when around their technically savvy peers.

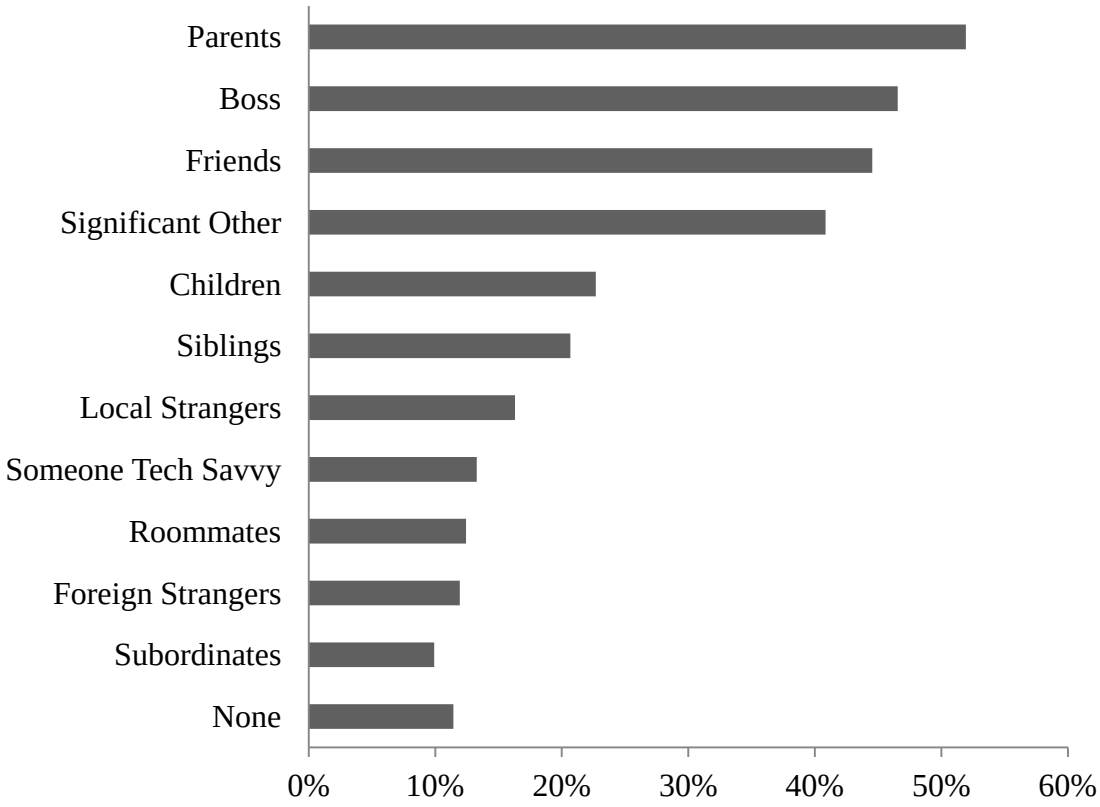

Figure 2: Percentage of participants who change their computing behavior around certain people.

2.7.2 What do we do in public? In private?

Figure 3 shows the top ten most frequently performed activities in public (when anyone is present) and in private (otherwise). Texting, emailing, web browsing, social networking, and listening to music are the top five. Among them, people text equally often in public and in private, while people prefer to email, browse the web, social network, and listen to music in private (by up to 31%). Overall, people largely engage in the same kinds of activities in public and private environments.

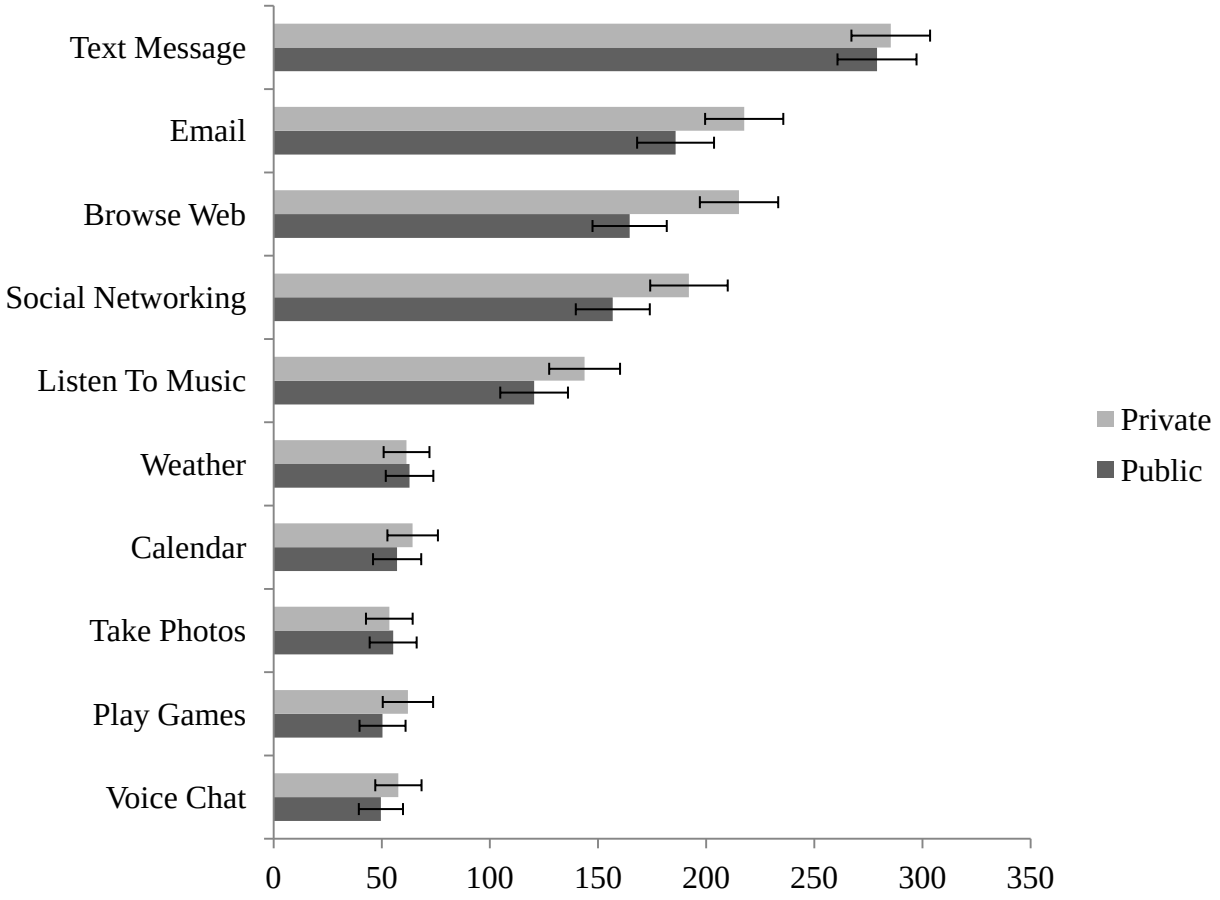

Figure 3: Tasks most frequently performed in public and private in total number of accesses per month.

Figure 4 shows a similar trend for categories of mobile tasks performed. The most commonly performed task categories are entertainment (44%) and communication (28%), with a significant margin over all other mobile tasks. Various task categories are performed more frequently in private (by up to 20%) than those performed in public, but people largely perform the same types of activities in both settings. However, the frequency does not tell the whole story; for example, entertainment tasks may present a low privacy risk, while financial tasks may have significant privacy implications.

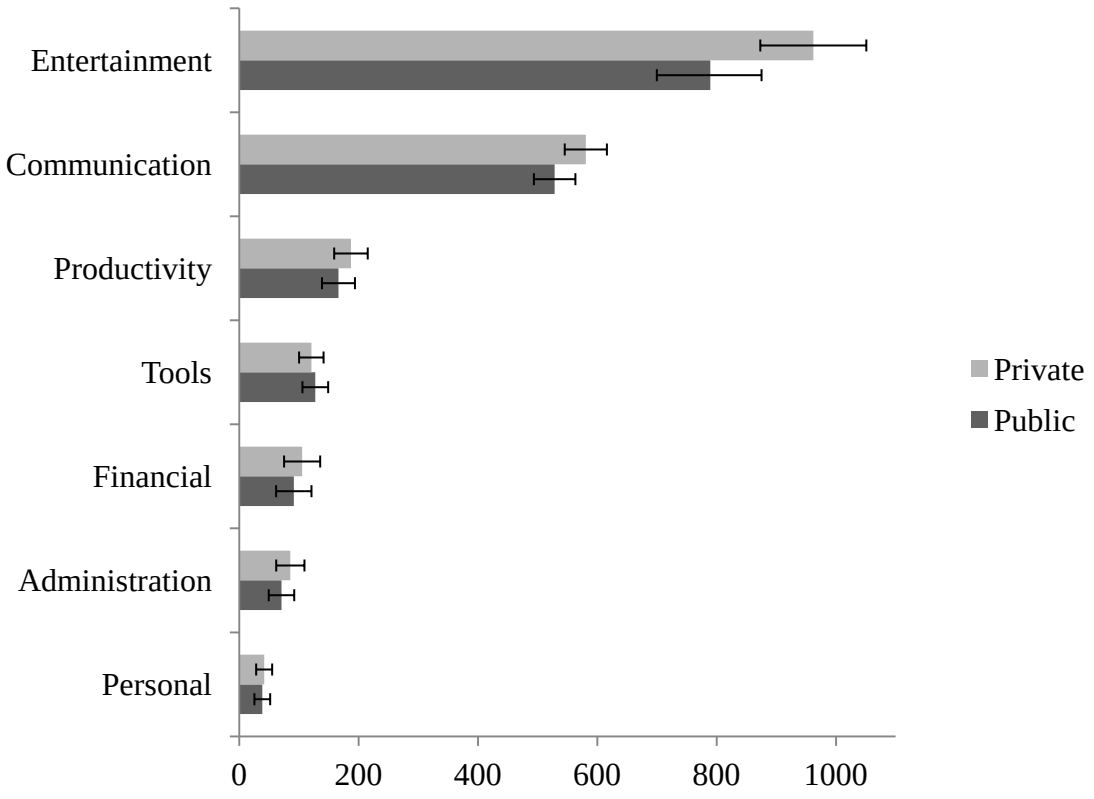

Figure 4: Frequency of public and private mobile tasks, by task category in total number of accesses per month.

Thus, mobile computing tasks were also categorized based on the risk level of the exposed information. High-risk tasks involve information that, if exposed, could be used in identity theft [87], such as financial transactions or online banking. Low-risk tasks, such as playing games or watching videos, have low personal information exposure. The remaining tasks, such as web browsing and social networking, are classified as medium risk, since they can involve risk ranging from low to high. Surprisingly, for high-risk tasks, people behave the same way in public and private (Figure 5), while people perform lower-risk tasks more frequently in private settings. Thus, privacy may also mean not wanting to be disturbed; performing high-risk tasks such as online shopping may benefit from consulting others in public settings.

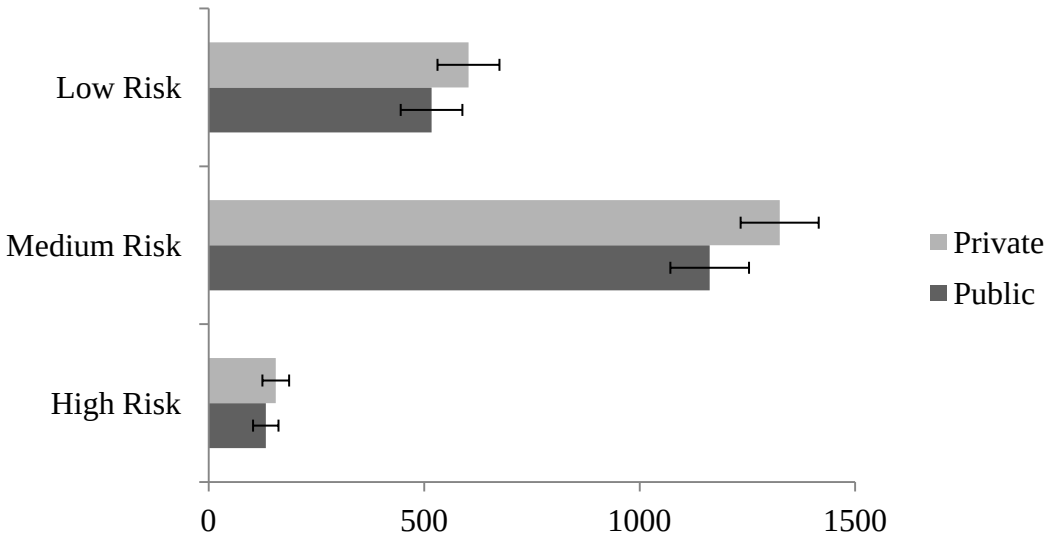

Figure 5: Task frequency organized by risk level of information exposure in total number of accesses per month.

Overall, the behavioral differences in public and private environments across genders, technical backgrounds, and ethnicities are not statistically significant, with a few exceptions. The results showed that women use social networking more frequently than men both in public (by 40%) and private (by 32%), but otherwise behave similarly to men. Tech-savvy users are more likely to access emails than less sophisticated users both in public (by 63%) and private (by 24%), but otherwise behave similarly as well. No significant behavioral differences were found among different ethnic groups.

2.7.3 What do we do when OSs and apps ask for permission?

Users seem to trust and comply with their apps more often than with their operating systems (OSs). Figure 6 shows that 38% of users always comply with OS permission requests, but 61% always comply with such requests from mobile apps. Since apps are written by parties of presumably variable trustworthiness and OSs are written by one well-known party, this result is counterintuitive, and raises interesting questions. Why are people 23% more likely to always agree with a permission request from an app than from their OS? Do the kinds and frequency of permission requests play a role? What makes the mobile app more trustworthy? With the access

most apps have to personal information on a device, this behavior is risky. Perhaps people underestimate the power of apps and the privacy implications. If so, mechanisms to proactively protect user privacy are more important to add to mobile devices. The kinds and frequency of permission requests might also play a role.

No significant statistical differences for operating system and app compliance were observed across gender, ethnicity, or device ownership.

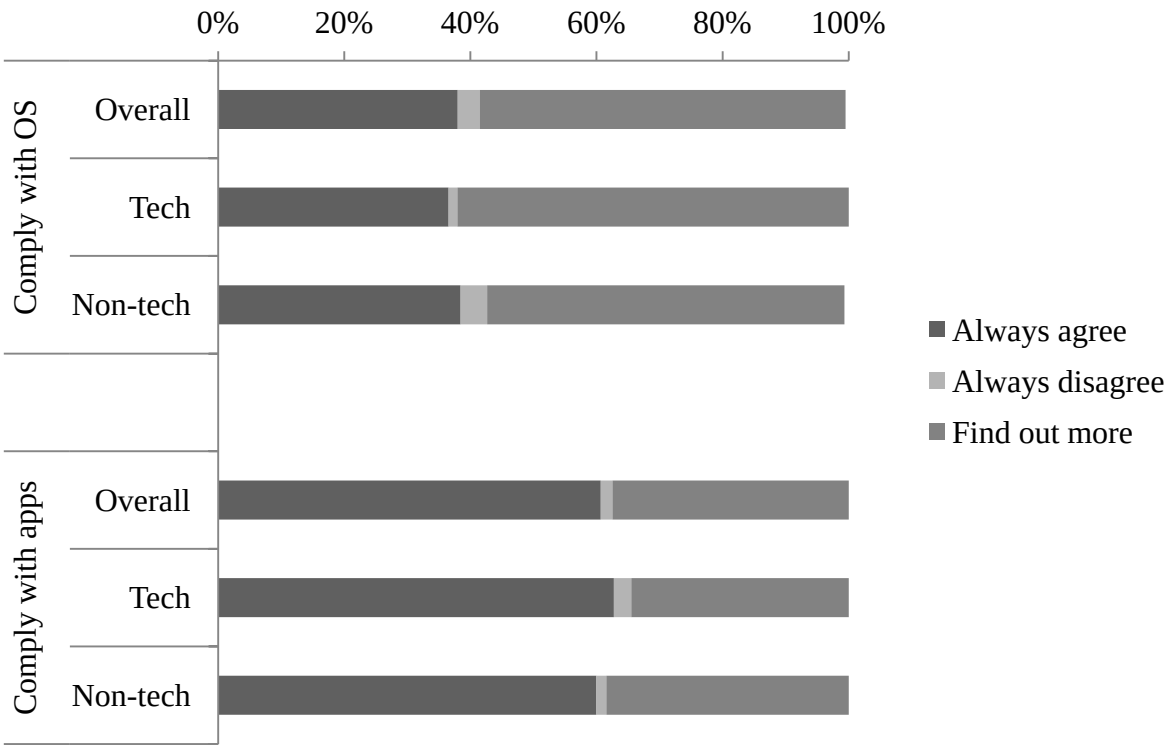

Figure 6: Compliance levels for permission requests from operating systems and mobile apps for subjects with technical and non-technical backgrounds.

2.7.4 Usage of Privacy-enhancing Tools

Subjects were also asked about their use of privacy-enhancing software tools, specifically encryption and password vaults. Figure 7 shows that 44% of participants responded that they use of have used encryption, 31% never encrypt, and 25% were not sure.

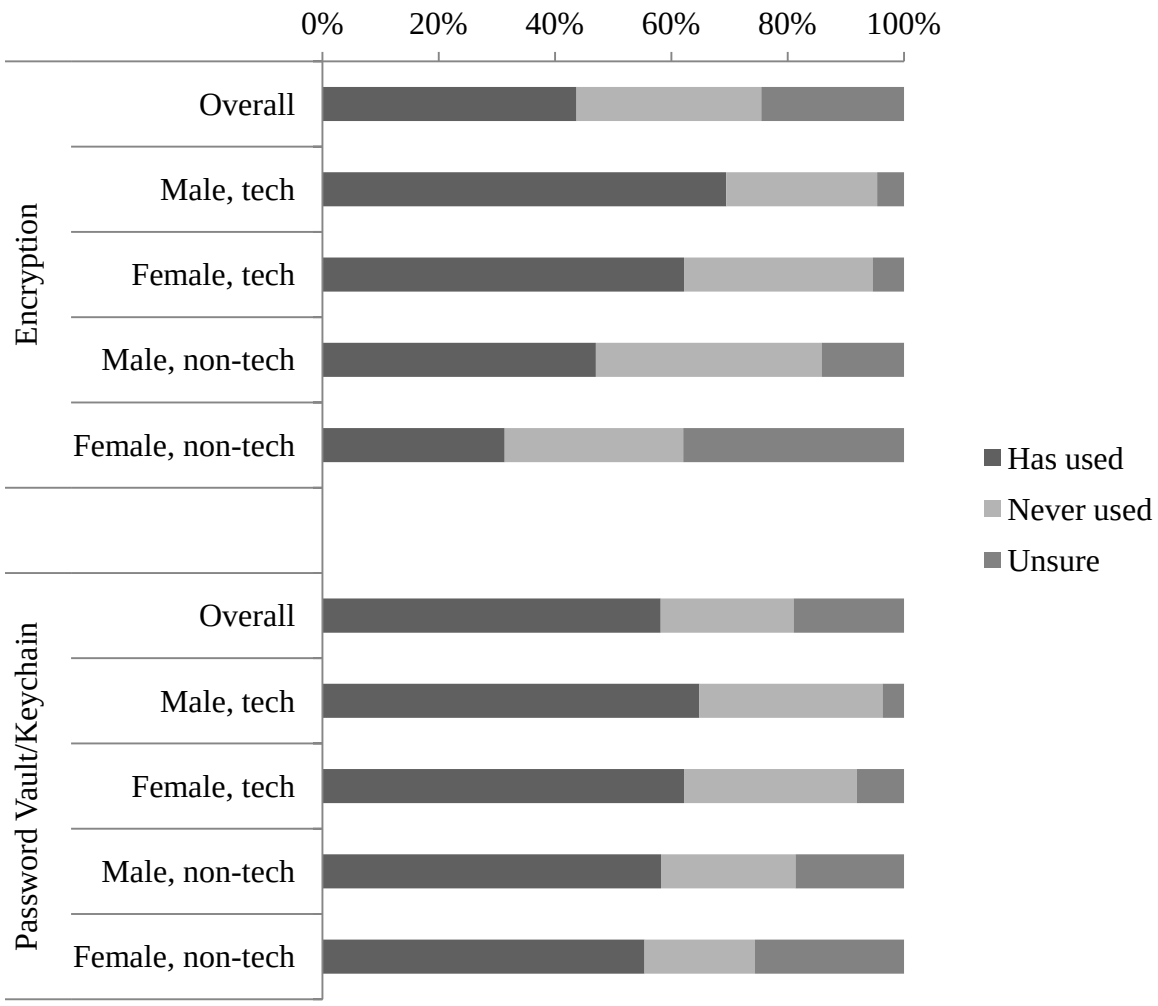

Figure 7: Encryption and password vault/keychain usage patterns by gender and technical background level.

Men were found 21% more likely to use encryption than women. Women were 25% more likely to be unsure about whether they use encryption (35% total). To see if this gender gap is caused by the pool of computer science and engineering students in the FSU sample, comparisons were made against the usage patterns between men and women with and without technical backgrounds. Figure 7 shows that the differences in awareness of encryption are even more pronounced for people without computer science or related backgrounds.

Minorities were found to encrypt more often (by 12%) than Caucasians. In particular, Asians encrypt more often at 12% ±11% above the mean, and Caucasians are the least likely, at

22% below the mean. Tech-savvy people encrypt more often (by 31%) than less sophisticated users. Device ownership preference was also found to play a minor role: Android users were found to encrypt more often (by 7% ± 6%) than non-Android users, and iPhone and Apple laptop owners to encrypt less often (by 7% ± 4%).

Password vaults/keychain usage trends were similar to encryption. Figure 7 shows that 58% of users have used password vaults, and 23% never use them. Password vaults may be slightly less confusing than encryption, but still 19% were unsure about their use. Similar to encryption results, men were 10% more likely to use password vaults than women; women were more likely to be unsure by 12%. Non-technical users were 19% more likely to be unsure than technical users if they use password vaults/key chains. Ethnic group and device ownership did not have a statistically significant influence on password vaults/key chain usage.

2.7.5 How do we connect to WiFi?

WiFi use is nearly ubiquitous, but how privacy-aware are people when they connect? Subjects were asked about the types of networks they connect to using their mobile device. Figure 8 shows that 81% of the participants responded that they use public WiFi without security; only 10% required at least password protection to connect. Eight percent of

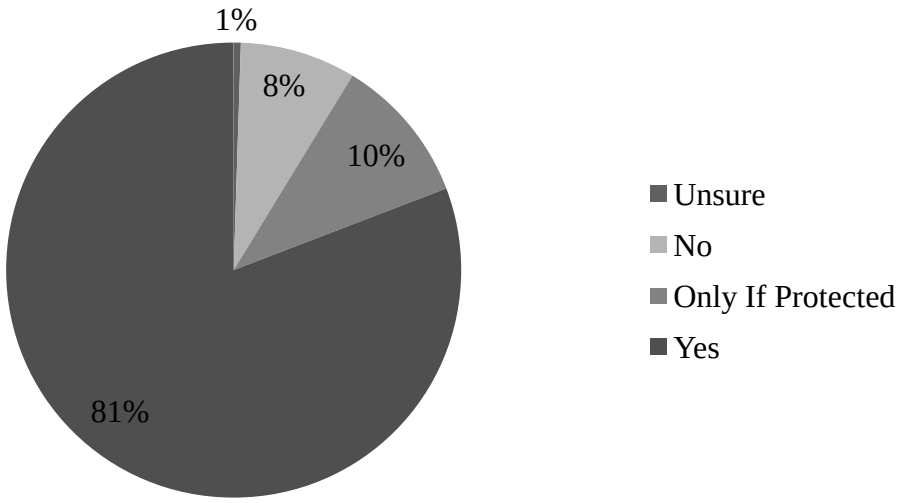

Figure 8: Percentage of users who use protected and unprotected WiFi networks.

respondents never use public WiFi of any kind. Tech-savvy users were 11% more likely to only use protected WiFi than their non-technically savvy peers. Device ownership was also significant in how users approach wireless security, iPhone owners were 10% more likely to use open WiFi networks than were non-owners. Little difference was observed for WiFi usage between genders and ethnic groups. These results suggest that reliance on WiFi encryption alone to protect data in transit is not sufficient, since most users will use an unprotected network.

2.7.6 What kind of apps do we use?

Questionnaire subjects were asked about their usage of different types of mobile apps. As shown in Figure 9, 88% use market apps, 79% use web apps, and 37% use non-market apps. Men and women use market apps and web apps similarly, but men are more likely to use non-market apps (by 12%). For respondents with both technical and non-technical backgrounds, market app use and web app use are similar; however, technical users are more likely to use non-market apps (by 13%) than other users. In terms of ethnicity, minorities are more likely to use non-market apps (by 15%) than the ethnic majority. In particular, African Americans are 21% more likely to use non-market apps.

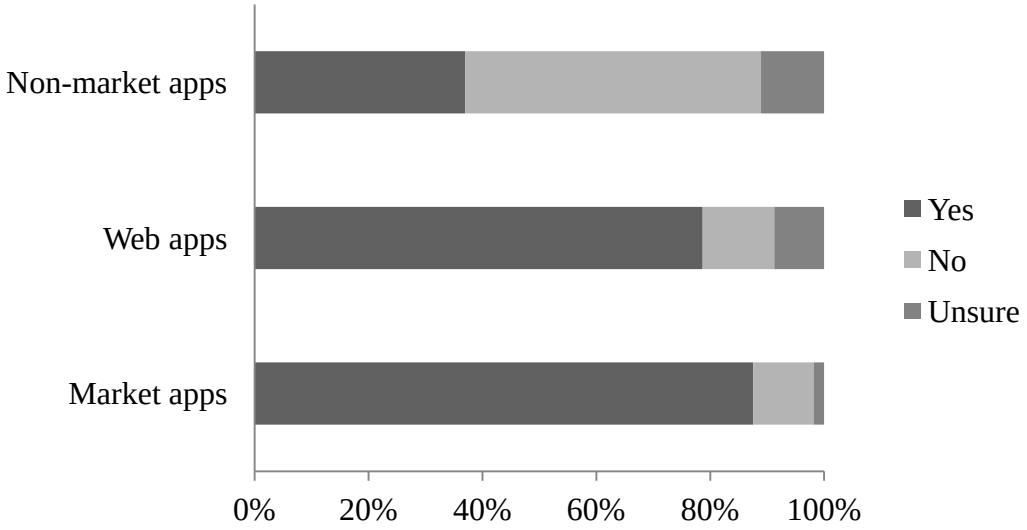

Figure 9: Percentage of subjects who use market, web, and non-market apps.

2.8 Other Findings

2.8.1 Where do we compute?

As shown in Figure 10, the top computing locations are at home, in class, in the library, while waiting in line, and in restaurants. Locations least likely to be used for computing are parks, while exercising, and in the washroom. Men and women behave similarly for most locations, though women are more likely to compute at a park, restaurants, and while waiting in line (by 7-10%). Technical users more often compute in the classroom and at the office (by 14-18%). Non-technical users more often compute in restaurants, while exercising, and while waiting in line (by 9-14%).

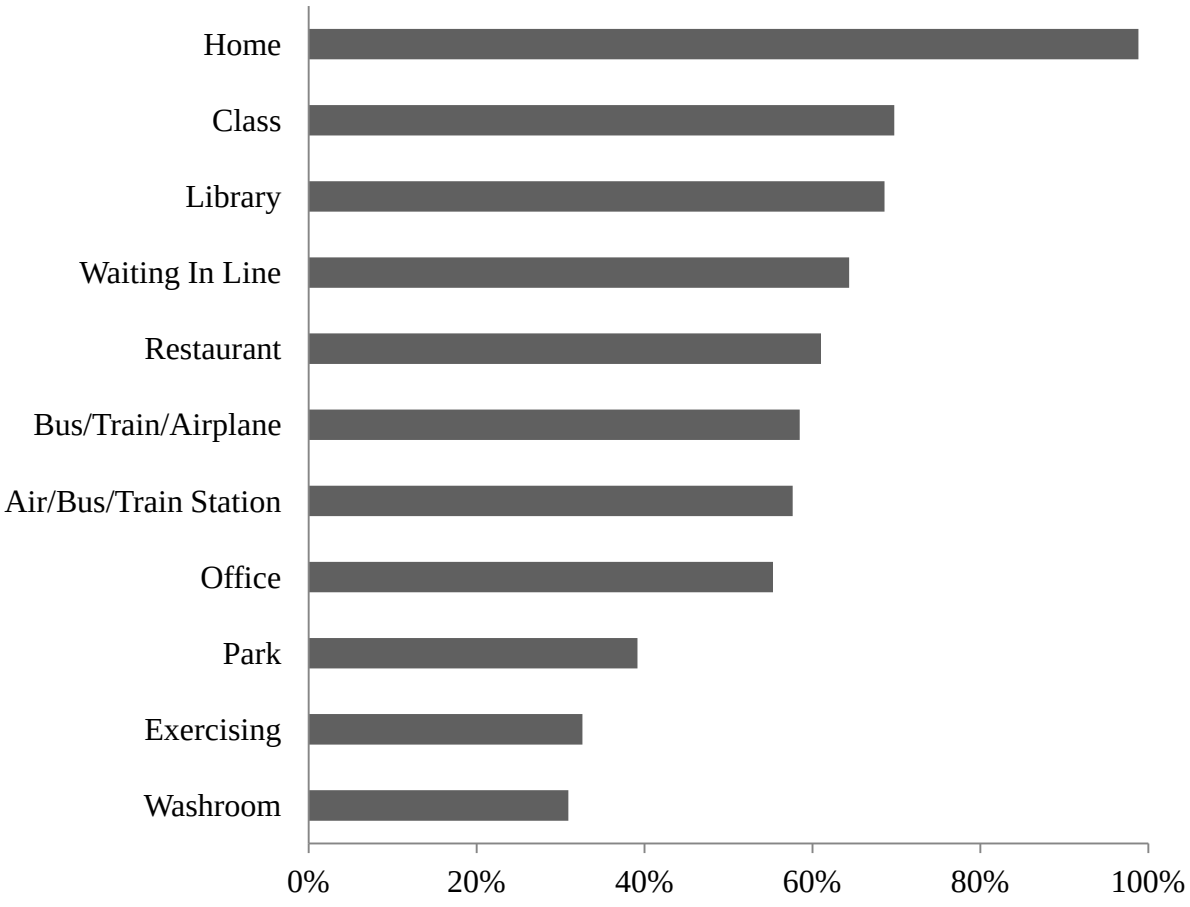

Figure 10: Percentage of subjects who regularly compute in each location.

Device ownership is perhaps the most interesting split. iPhone owners are more likely to compute in restaurants, airports/bus/train stations, public transportation, while exercising, in class, and while waiting in line (by 13-16%). Android device owners are more likely to compute at a park, office, and washroom (by 10-15%). No significant differences in computing locations were found between ethnic groups.

In the FSU survey, the effects of device ownership are even more pronounced. iPhone owners are more likely to compute in restaurants, on a bus/train/airplane, at airports/bus stops/train stations, while exercising, and waiting in line (by 19-26%).

2.8.2 Implications of Apple Ownership

Compared to Android owners, Apple users more frequently use their iPhones, iPads, and Apple laptops in public locations (by 13-16%). One plausible explanation is Apple's greater choices of apps. Another possibility is their use as a status symbol.

Survey participants who own Apple products were found to use their devices for most social mobile computing tasks: texting, e-mailing, and social networking more than owners of other devices. Apple device owners are more likely to e-mail both in public (by 27%) and in private (by 19%), send text messages in public (by 19%) and in private (by 22%), and use social networking in public (by 63%) and in private (by 35%).

As previously discussed, Apple users use their device in more public places. The degree of this increase is shown in Figure 11. Apple device owners also have less regard for WiFi security. Eighty-six percent of iPhone owners use open, public WiFi without security, 6% above average. And Apple device users are less likely to use encryption (by 7%).

2.9 Lessons from the Survey

This survey speaks to user attitudes towards privacy, not necessarily actual behavior. However, experience shows that user attitudes are critical in determining whether a privacy or security measure is widely used; thus, in some senses such attitudes are just as important to a privacy mechanism's success as the technical details of how it works. Certainly designers of mobile computing privacy mechanisms should keep this point in mind when going about their work.

Figure 11: Comparison of computing locations of iPhone and non-iPhone users.

2.9.1 Implications of Privacy on Systems

One finding is that mobile users are far more concerned about protecting their privacy from those who know them well. The presence of parents, for example, seemed to be viewed as a threat to users' privacy more than twice as often as strangers. Users may be more likely to accept privacy preserving mechanisms designed to protect against family and friends than against random eavesdropping by strangers.

In the context of this work on the visual channel protection, family and friends most certainly pose far less of a threat to identify theft than do strangers. Nonetheless, protection of visual information leaks against those most familiar is still worthy of consideration, especially if the mechanism providing the protection is the same for all cases.

2.9.2 Privacy, Trust, and Anonymity

The results suggest that trust and privacy are largely orthogonal. People we trust the most are also the ones whose loss of trust we most fear. This result is consistent with studies of teenagers' behavior on social networking sites [92], but is not necessarily what many academic researchers think about when they address how to achieve privacy in mobile computing systems. Since users should be provided with the privacy they want and will actually use, researchers must ensure that their goals align with users' real privacy desires.

Users' relative indifference to privacy threats posed by strangers might suggest a perception of anonymity, leading to a false sense of security. Frequently, computer users have been behind the curve on what malicious parties can do with information they obtain, and this may be another such case. The perception of anonymity is probably true only if strangers are not interested in us. If they are, and we do not protect ourselves, we may suffer serious consequences. Researchers and developers are clearly interested in protecting users from such threats, but the results suggest that, in today's world, only transparent and simple protection mechanisms against such threats will succeed, since users may be unwilling to take actions that seem inconvenient.

Chapter one has already established that the threats of loss from direct or indirect observation from bystanders is real. If users maintain this false sense of security against the shoulder surf threat, severe consequences could occur. Additional education and awareness of the nature of the threat could be beneficial to decrease this risk. Alternatively, users' mobile devices could automatically protect sensitive data against sensitive data leaks. However, as previously mentioned user acceptance of the protection mechanism is critical, and thus such a system must be as least intrusive as possible.

2.9.3 Mobile apps' privacy implications underestimated?

Why do the respondents trust applications more than OSs? Clearly, the consequences of mistakenly permitting an OS to take action may cause greater harm. However, it appears users

are unaware of how much information a modern mobile app can access and of the resulting potential privacy implications.

App developers often do not take the necessary steps to protect personal data against even the most basic security threats. Even those that do follow the suggested industry protection policies do not in any way address the threat of loss from the visual channel. Further, it is unreasonable to believe that any amount of education, awareness, or quantification of the shoulder surf threat will convince every app developer to take the necessary steps to prevent visual exposure of sensitive data. Thus, a more comprehensive approach to the shoulder surf problem is required.

2.9.4 Survey Overall

Various aspects of the survey, such as the places people feel comfortable computing, the kinds of applications used in public, and the disregard for the presence of other people when using mobile devices, suggest that many users are not concerned about preserving the privacy of their computing in mobile environments. Yet such risks are real. The obvious question is whether the majority of users are unaware of the risks, or are reasonably aware and simply do not care about them. Unfortunately, the study did not include questions that provide insight on this point. However, this point is crucial. If users ultimately care little about preserving their privacy from such risks, only the cheapest, most transparent, least intrusive privacy enhancing mechanisms will succeed.

CHAPTER THREE

EXISTING OBSERVATION-RESISTANT SOLUTIONS

Previous related works include both systems that secure against observation-based attacks and those that provide similar privacy protection over network channels.

3.1 Visual Authentication Protection

Prior work on protection against visual exposure is focused primarily on securing the act of authentication. By far the earliest is the technique of Xing out or simply not printing entered passwords on login screens [72]. Most others can be generalized as augmentation or replacement of password entry mechanisms.

3.1.1 Password Managers

Perhaps the most common method of securing against an observation-based attack is the use of the password manager. These are software tools that allow the user to select a predefined username and password pair from a list for entry into the login fields [14]. Typically a master password is used to unlock the password vault to provide user security against loss. This also allows a user to use different passwords for different applications without the need to each of them individually.

Built in password management is now commonplace in nearly all web browsers including Google Chrome, Firefox, Internet Explorer, and Safari. Other typical features include automatic form filling, cloud-based storage, and synchronization. Commercial applications such as LastPass [15] and 1Password [16] extend this model with cross-platform and cross-browser support.

3.1.2 Hardware-based Authentication

Other related work involves external physical devices to supplement or replace password-based authentication. They modify the problem from being strictly recall based (something you know) to possession based (something you have). Many different specialized pieces of hardware

have been developed for this purpose, but they all conceptually function as a digital key to unlock another device. The simplest solutions do not require any connection or modification to the existing system and rely on the user to read a hardware display and enter the relevant data to authenticate. Other techniques utilize specialized USB dongles [17], audio jacks [18], short range wireless communication using NFC [19], or Bluetooth connections [20] to connect to the authenticating machine (Figure 12).

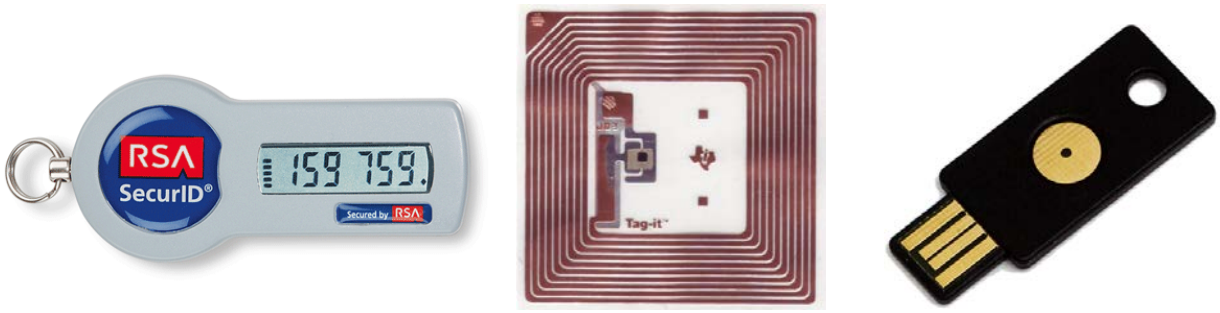

Figure 12: Cryptographic token (left), short range wireless (center), and USB (right) hardware-based authentication devices.

The security credentials transmitted or manually entered by the user from these devices can be static passphrases [17], synchronously or asynchronously generated cryptographic tokens [18], or mechanisms that public key cryptographic challenge response techniques [17, 19] to that do not require the device to directly reveal the security token. Keypads or other input methods can also be used to further enhance the security of these hardware tokens by requiring the user to enter an additional pin or passphrase.

Closely related to hardware-based security token solution is device-based authentication. Commonly known as dual form, two-factor, or multi-factor authentication (MFA), these techniques offer a secondary layer of authentication over strict hardware solutions. In addition to password entry (something you know), MFA supplements this with a hardware device (something you have), typically being a user's smartphone. When the user attempts login to a service, a secondary, time-limited challenge response is sent from the service to a pre-registered device or address owned by the user. This message is sent via an alternate communication channel such as SMS message, e-mail, or through an application installed on the user's device.

The user must enter this time limited, one-time security token in addition to their password to authenticate for the session.

3.1.3 Graphical Passwords

Another technique to help guard against information leaks from visual attacks is the use of graphical passwords or Graphical User Authentication (GUA) [22]. Such techniques remove the alpha-numeric password from the equation and replace it with the use of series of images, shapes, and colors. Common techniques present the user with a series of human faces that must be clicked in sequence [23], object sequences as part of a story [24], or specific regions within a given image that must be clicked in sequence [25] (Figure 13).

While these techniques may help to improve the security of password entries under specific circumstances they may actually serve to increase the risk of shoulder surfing observation. It may take an observer only a few login sessions to capture the image sequences required for authentication.

Figure 13: Image region-based (left), series of human faces (center) and object sequences (right) graphical passwords schemes.

3.1.4 Biometrics

Biometric authentication mechanisms can be generalized as changing or augmenting password entry (something you know), with a feature unique to your personal biology (something you are). There are many inherent physiological characteristics that are sufficiently unique to identify and differentiate one individual from another. The most commonly used of these biometric identifiers include contours of the fingerprints [26], iris and retinal configuration of the eye [27], and geometries the face [28] and hand [29] (Figure 14). Behavioral

characteristics, in contrast to biometric identifiers, including keystroke latency [30], gait [31], and voice [32] can also be used for authentication purposes.

This authentication typically requires additional hardware support in the form of fingerprint scanners, retina scanners, or brainwave detection devices. Notable exceptions are facial feature detection and voice recognition, which can be performed using only the standard digital cameras and microphones present on nearly every mobile device in circulation. Primitive systems using facial geometries or voice recognition were easily tricked by photographs or audio recordings of the user, though they have improved significantly in more recent designs. There are many privacy concerns for biometrics especially for trivial authentication purposes. By design, biometric feature are something that is unique to the user, and not easy to change. In the case of security loss, an attacker permanently gains access to all other services that identify the user by that specific physiological identifier.

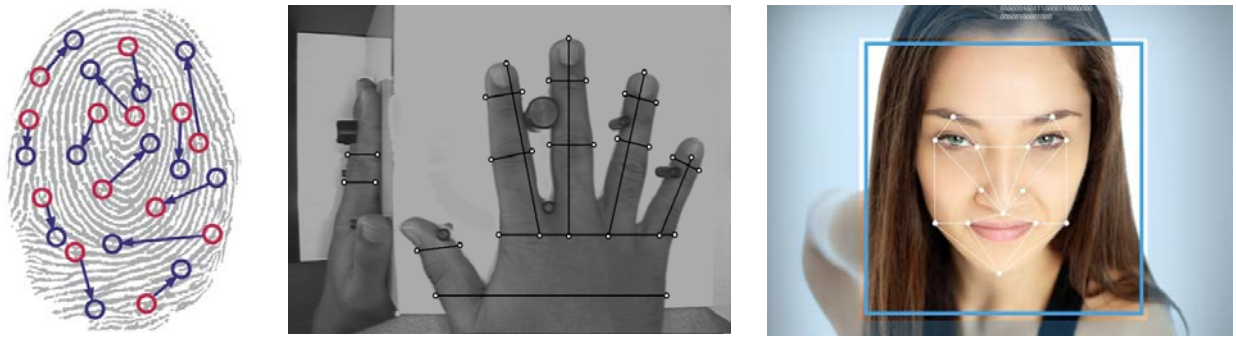

Figure 14: Fingerprint (left), hand geometry (center), and facial recognition (right) biometric authentication techniques.

3.1.5 Gesture-based Authentication

Closely related to both GUA techniques and biometric solutions are gesture based authentication techniques (Figure 15). This method allows the user to perform specific tap [33], multi-finger presses [34], or swipe sequences on-screen [35] to represent a password. The user essentially is able to draw a picture to authenticate, or ‘connect-the-dots’ in a grid in a specific order or pattern. Gesture-based solutions can offer enhanced security over traditional GUA methods due to variations of different users’ finger features and characteristics. It is also less invasive than more traditional biometric methods since these methods still provide a mechanism

to change the login credentials. This method was primarily developed for convenience since password entry using small screened mobile device keyboards can be difficult.

However, screen smudges can also leave evidence of the swipe sequence requiring the user to constantly wipe the screen to remove the residual evidence of the authentication pattern [36]. In addition, similar to the visual inadequacies of graphical passwords, a visual observer may be able to learn the tap, click, or swipe gesture after a single viewing, recordings can be re-played as many times as necessary to deduce and memorize the sequence. Excessive caffeine consumption, certain physical conditions, or side effects of other medications can cause jitters or inadvertent pausing and make such techniques less accurate or even frustrating for practical use for some individuals.

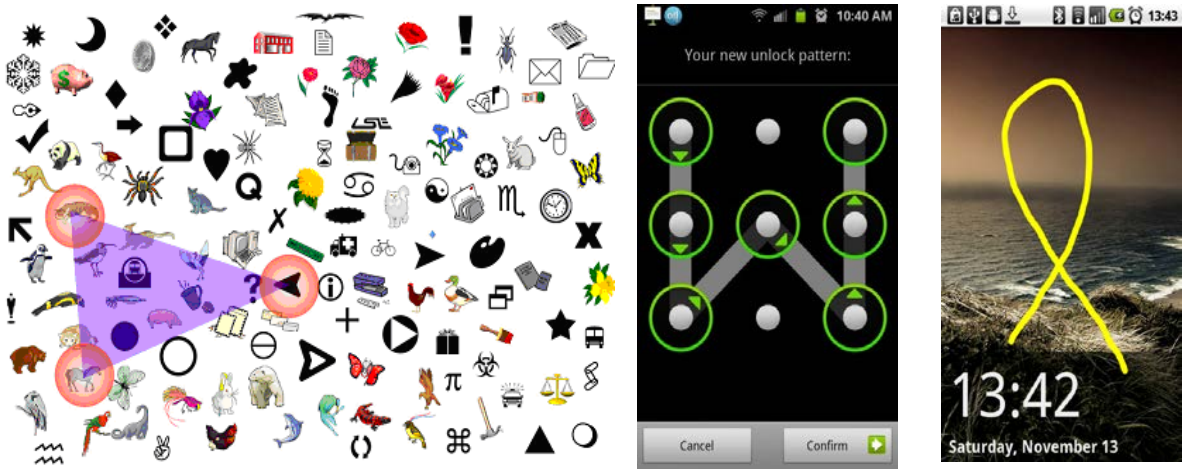

Figure 15: Gesture-based authentication mechanisms.

3.1.6 Cognitive Challenges

Other techniques have attempted to make games of the authentication procedure [37] (Figure 16). Instead of a single password or phrase, these techniques utilize challenge response questions and use of cognitive tasks to increase the difficulty of the login session [38]. In method such as these, an onlooker would likely need to observe multiple login sessions to gather sufficient information about the user to be able to impersonate the user and gain unauthorized access.

The addition of timing constraints between subsequent tasks can further enhance the difficulty for an individual other than the user to trick the system and gain unauthorized access.

These mechanisms can also be used in conjunction with many other authentication techniques discussed here.

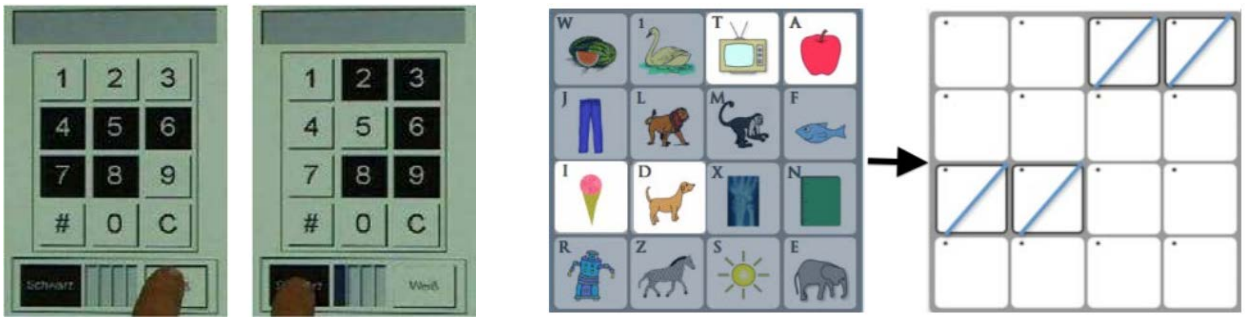

Figure 16: Cognitive challenges as authentication mechanisms.

3.1.7 Obfuscation and Confusion

Other techniques have attempted to remedy the shortcoming of password based authentication through obfuscation and confusion to a visual observer (Figure 17). Such systems are essentially security enhancements to other authentication techniques previously mentioned notably GUA and cognitive challenge based methods. They utilize the hiding of cursors [39], confusion matrices [40], and recognition [41] rather than recall-based methods to trick and confuse onlookers.

Instead of the user directly clicking on a series images within a grid, the challenge can be changed to asking the user a series of questions of whether they can see one or more of their graphical password elements on a given screen. This serves to further complicate the challenge of an attacker to know which elements are the actual components of the user's graphical password. Another method is to use a field of randomly moving cursors during the authentication

Figure 17: Obfuscation and confusion authentication techniques.

process. From their own mouse movements, the user can tell which cursor is genuine and ignore all the others. This can dramatically increase the difficulty on an observer to see which sequence of images was clicked.

3.1.8 Alternate Sensory Inputs

Additional work has been done utilizing other biological sensory inputs to replace or augment password-based authentication. These systems can address two separate parts of authentication process, the cue to the input, or the actual input itself.

In the first case, the additional sensory input serves as a non-observable instruction or hint to the required passphrase entry. These systems utilize audio direction [42] or tactile and haptic feedback from the vibration motors on devices [43] to provide the user the appropriate cue to the necessary response. The user then responds with the phrase corresponding to the cue using traditional input methods.

In the second case, the auxiliary sense serves as the input mechanism itself. These systems extend GUAs by requiring sequential graphical inputs but use mechanics like eye tracking, blinking and gaze-based interaction for the user to input the graphical sequence [44]. Systems have even demonstrated the capability of using brain waves for this task; a user may only need to think a specific thought to authenticate with a system [45]. These methods are also useful alternatives to authenticate for people with visual or audio sensory disabilities [46].

3.1.9 Physical Barriers and Screen Filters

Other related work is in the area of physical barriers to limit visual data leaks (Figure 18). Rather than approaching the problem from the perspective of software-based solutions, these alternatives use physical barriers external to the system to prevent exposure to prevent or limit the amount of screen exposure to bystanders.

Perhaps the simplest and most pervasive of such barriers is the common office cubicle. However due their obvious bulk and lack of portability these are not a particularly general purpose solution, especially for the increasingly mobile modern workforce. Other solutions, such as the 3M Privacy Filter [67] approach the problem by limiting the field of view of the screen. A flexible micro-louvre film overlay is attached to the screen that permits only the user directly in front of the screen to view its contents. This may serve to partially reduce exposure, but larger screens are still visible for a larger area and by unauthorized viewers directly behind the device.

Other solutions such as the Lenovo Sun Vision [68] and Compubody Sock [69] take the concept of screen visibility reduction further by completely blocking out all non-direct visual exposure. However such devices limit situational awareness by forcing the user to exchange their own external visual channel in exchange for preventing the visual exposure to bystanders.

Figure 18: 3M Privacy Filter [67], Lenovo Sun Vison [68], and Compubody Sock [69] screen filters and physical barriers.

3.1.10 Wearable Devices

Wearable headsets such as Google Glass [57] offer a different solution to preventing screen exposure by moving the screen directly in front of the user's eyes (Figure 19). However, current generation devices have limited application support and are not currently able to perform

Figure 19: Google Glass [57] (left) and Oculus Rift [65] (right) wearable devices.

general purpose computing tasks. In addition, much of the user input is currently performed by audio cues, which merely translates the sensitive data leaks from the visual to the audio channel.

To a lesser extent, virtual reality headsets such as the Oculus Rift [65] and Samsung Galaxy Wearable [66] permit similar private screen viewing. However, like other wearables, current generation editions do not permit general-purpose computing. Additionally, such devices also suffer from the shortcoming of other view obstructing privacy partitions, limiting situational awareness of the user due to the blockage of the external visual channel.

3.2 Digital Communication Channel Protection

Many protocols and systems have been developed also handle other aspects of privacy-oriented attacks through the encryption of the digital communication channel. Transport Layer Security and Secure Sockets Layer can enhance security by providing session -ased encryption [47]. Virtual Private Networks can be used to enhance security by offing point-to-point encryption to provide secure resources access across insecure network topologies [48]. Proxy servers [49] and onion routing protocols like Tor [50] can add extra privacy by providing obfuscation of location, and anonymization of IP addresses.

Many other solutions have been developed to enhance security and privacy at the browser level. Do Not Track requests be can included in HTTP headers to request the web server or application to disable its user and cross-site tracking mechanisms [51]. Many browser extensions and plug-ins exist to block advertising [52] as well as analytics, beacons, and other tracking mechanisms [53]. Other systems alert the user when specific privacy elements are leaked [54], prevent the transmission of sensitive data without explicit user permission [55], and cryptography secure access to sensitive data outside of trusted situations [21].

3.3 Existing Solutions Inadequate

Despite the various mechanisms mentioned, the visual channel remains largely open. A limited number of tools are available to obfuscate sensitive data other than during the act of authentication. All existing tools developed for encryption of data are not originally designed for such purposes.

Password-based solutions and biometrics are effective in handling visual leaks during the act of authentication, but cannot be generalized to handle other cases. No existing mechanism is in place to allow arbitrary data to be marked as sensitive. The exposure of personal data on screen remains unaddressed and the potential for loss from shoulder surfing observation remains a real threat.

CHAPTER FOUR

CASHTAGS

In response to the prevalence of the shoulder-surfing threat and the current lack of defense mechanisms against this threat, this work presents Cashtags: a system that defends against observation-based attacks. A Cashtag, an amalgam of the words *cash* and *hashtag*, serves as easy-to-remember aliases for valuable sensitive personal identifiers. A cashtag alias consists of a dollar sign followed by an arbitrary string of printable characters.

The system allows a user to access sensitive information in public without the fear of leaking sensitive information through the screen.

4.1 User Model

Conceptually, Cashtags is configured with a user-defined list of sensitive data items, each with a respective Cashtags alias or a cashtag (e.g., `$visa` to represent a 16-digit credit-card number; see other examples in Table 1). Then, whenever the sensitive term would be displayed on screen, the system displays the pre-defined alias instead (Figure 20).

Table 1: Sample mappings of sensitive private data elements to their corresponding cashtag alias.

SAMPLE MAPPING OF SENSITIVE DATA TO CASHTAG ALIASES		
Type	Actual	Alias
Name	John Smith	`$name`
Email	jsmith@gmail.com	`$email`
Username	Jsmith1	`$user`
Password	p@ssw0rd	`$pass`
Street Address	123 Main St.	`$address`
Phone number	555-111-2222	`$phone`
Birthday	1/1/85	`$bday`
SSN	111-22-3333	`$ssn`
Credit Card	4321 5678 9012 1234	`$visa`
Account number	123456789	`$accts`

At the point at which the sensitive data would be used internally by the device or an app, cashtags will be replaced by the sensitive data items represented by the alias, allowing whatever login, communication, transmission, or upload to proceed normally.

Figure 20: On-screen sensitive data (left) and data protected by masking with cashtag aliases (right).

Also, a user can directly type in a cashtag in place of the sensitive term, permitting more complex data-sensitive tasks such as filling out an application for a credit card or loan without

risk of observation from a bystander. In addition, cashtags are easier to remember than the actual information itself. For example, `$visa` can be used as a shortcut for entering a 16-digit credit card number.

4.2 Threat Model

The threat model for Cashtags is defined as passive, observation-based attacks (e.g., captured video or physical observation by a human). The assumption is made that the attacker can observe both the screen of the user as well as any touch sequences the user may make on the screen, physical buttons, or keyboards. It is further assumed that this threat vector does not present opportunity for an active attack. Through visual observation or digital recording, the attacker cannot directly influence the user in any way.

Although sensitive information can be presented in many forms, the focus of this work is concentrated on textual information to demonstrate the feasibility of the framework. Protecting sensitive information in other forms (e.g., images and bitmaps) will be the subject of future work.

4.3 Compared to Password Managers

The user model of Cashtags is similar to that of a password manager. To add an entry to a password manager, a user is required to key in the username and password pair. Typically, subsequent usage of the stored password involves only selecting the respective account pre-populated with the stored password. Therefore, an observer cannot see the keyed-in sequence for passwords. Similarly, Cashtags requires the user to pre-configure the system by first entering the sensitive term to be protected and the corresponding alias to represent the term. When a sensitive term is displayed, the system replaces the sensitive term with its alias without user intervention. To enter a sensitive term, the user can enter the alias, and the system will translate it into the sensitive term prior to being processed by the underlying apps. Aliases are easier to remember, and the system is compatible with auto completion, which further eases the entry of aliases.

4.4 Design Overview

Although conceptually simple, the design of Cashtags addresses a number of major design points:

- ***The interception of sensitive data elements:*** Cashtags intercepts sensitive data items as they are sent to the display. For apps, this point is located at their common textual rendering library routines. For user input, this point is within the routines that handle software keyboards and physical devices (e.g., USB and wireless input devices).
- ***A convenient and compatible user interface to the service:*** Users can type in cashtags instead of sensitive data items to compute in public. This interface allows cashtags to be compatible with existing tools such as auto completion, spellcheckers, cut and paste, etc. Thus, users can enter the first few characters and auto-complete the full cashtag.
- ***Service-specific internal access to sensitive data:*** User-entered cashtags are converted internally to the sensitive data items before the apps access the data. Cashtags returns either the cashtag or actual text depending upon the service making the request. This way, Cashtags will not break applications due to unanticipated input formats.
- ***Handling of many data formats variants:*** Cashtags can leverage existing libraries to match sensitive data items represented in different formats (e.g., John Smith vs. John Q. Smith for a name; or 123-45-6789 vs. 123456789 for social security number).
- ***An efficient and convenient development and deployment model:*** Cashtags uses a code-injection framework. This approach avoids modifying individual apps and the firmware, while altering the behavior of the overall system to incorporate Cashtags functionality at application runtime.
- ***A centralized Cashtag storage repository:*** The mapping of cashtags to sensitive data items is stored in a centralized, password-protected repository.

- ***Per-application behavior:*** Individual features of the user input and display behaviors of Cashtags can be toggled at the granularity of the individual application.

CHAPTER FIVE

SYSTEM DESIGN

This section will present the design options for each Cashtags design point and explain how the current design was derived.

5.1 Observation-resistant Approaches

The alternate approaches of screen-level-masking and data-entry-tagging system design spaces were considered prior to arrival at the current keyword-oriented design. The general idea is the same for of these methodologies: intercept screen rendering and prevent private data from being displayed on screen. While all of these approaches can prevent sensitive information from being displayed, the main differences are the interception granularity and the portability of the underlying mechanisms.

5.1.1 Screen-level Masking

The simplest and most coarse-grained approach is to mask the full application window or screen regions when encountering sensitive information. Several solutions, such as Kino [61] and Screen Concealer [62] have been developed utilizing this approach with certain regions either completely visible or completely obscured from view. While this approach prevents information leakage, it also prevents the user from computing using the sensitive information. Completely masking all or part of the screen window from view does prevent private information exposure, but also prevents the user from interacting with the screen contents as well. For example, when encountering an online purchase screen, the entire screen would be blurred due to the presence of sensitive information, making it unusable.

5.1.2 Tag-based Approach

A tag-based approach can also be utilized to prevent sensitive on-screen exposure. Unlike screen-level masking, this approach requires the user to predefine which specific data elements as sensitive. These data elements are then tracked as they propagate through the system [16]. If a

tracked data element is to be displayed on the screen, the rendering is intercepted, and the tracked data element is replaced with its corresponding alias.

This approach requires a significant system modification to support this granularity of data tracking, making it a less deployable solution. In addition, the system resources and required accounting to track the data results in significant processing overhead incurred by the system.

5.1.3 Keyword-based Approach

Another approach is to utilize keywords and perform pattern matching on on-screen text elements. Like the tag-based approach, this option works at the individual data element or word granularity. It also has the requirement that the sensitive data element be specified to the system prior to the point of screen rendering and subsequent visual data exposure.

The primary difference, however, is the method in which the sensitive data is identified. Rather than tracking sensitive data as it propagates through the system, this method parses data fields prior to the screen display. If a predefined sensitive element is matched, it is replaced with its corresponding alias before being rendered to the screen. This approach was selected for the Cashtags framework because it achieves word granularity protection without the tag-based overhead and deployment issues.

5.2 Where to Intercept Sensitive Data

To decide where to intercept sensitive data, it is crucial to first understand how sensitive data traverses from apps to the screen through various display data paths. Figure 21 shows the display data paths under the Android application development platform. Although they use differing terminologies, the display data paths of iOS and Windows generally have one-to-one mappings with these components.

5.2.1 Window Manager

A typical app displays information by invoking some user-level display or graphics library routines. Various routines eventually invoke routines in the underlying window management system (e.g., Surface Flinger for Android) before information is processed by the operating system and displayed on the screen.

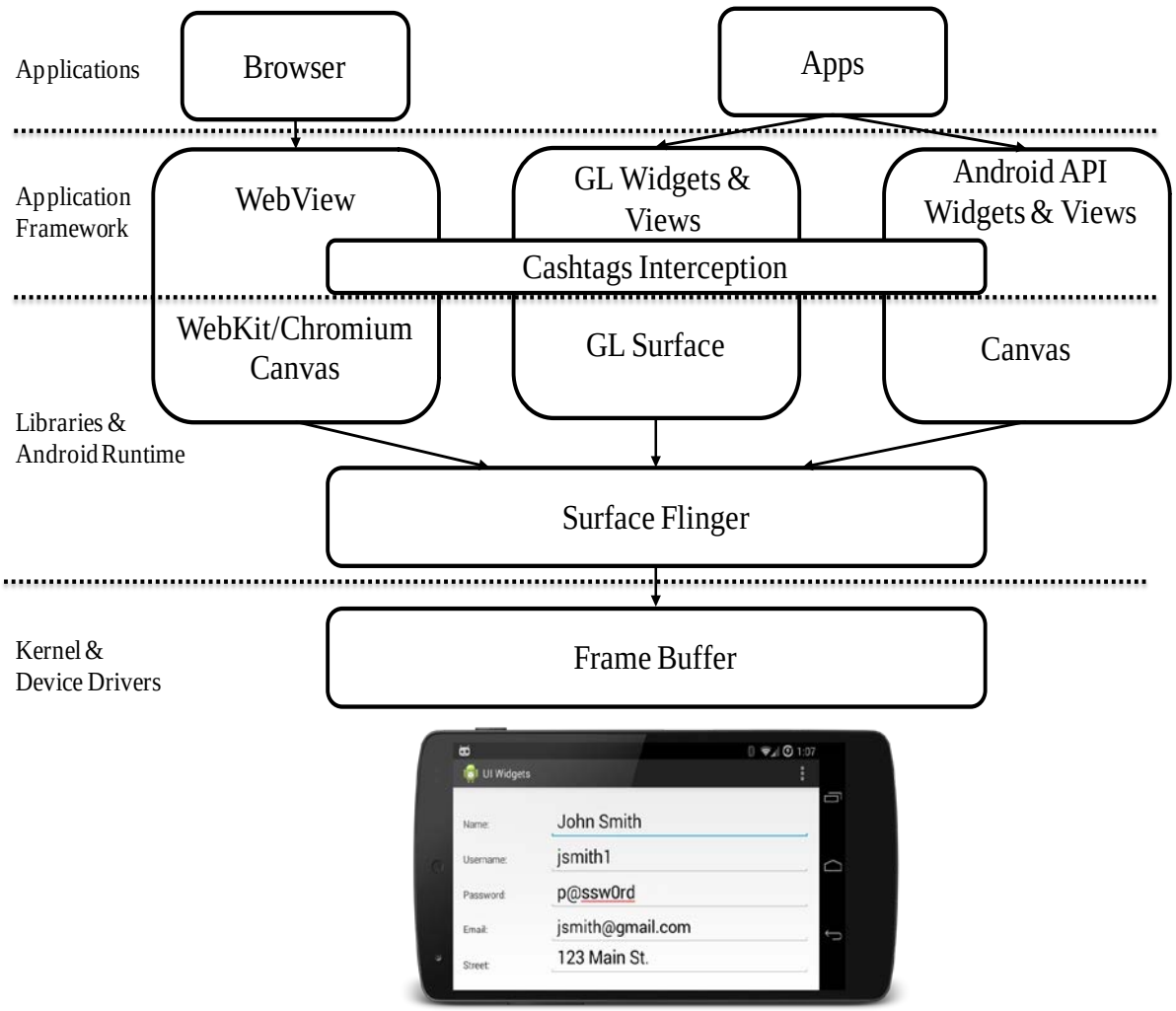

Figure 21: Display data paths for the Android platform.

Arguably, the window management system might seem to be a single point at which all sensitive data can be captured. Unfortunately, by the time sensitive information arrives there, some sensitive information may have been translated into bitmaps. While OCR technologies are computationally cheap enough to be used for launching shoulder surfing attacks, they are still too heavyweight for deployment in the display data path, which is critical for user interactions. In addition, the replacement of sensitive bitmaps with non-sensitive ones poses another other obstacles that should be avoidable if possible.

5.2.2 Applications

Another extreme is to intercept it at the app level, where the sensitive information is introduced. Potentially, the modification of a select few top apps can capture the majority of cases where sensitive information is used. For instance, custom e-mail applications or browsers could offer protection for task-specific usages. However, such solutions may restrict users to using a specific tool for a specific task. In addition, statistics show that specific app usage accounts for 86% of user time, trending away from general-purpose browsers [56]. Thus, the burden of incorporating the required features would spread to a much wider range of app developers, which is undesirable. Further, new apps and updates to old apps would not automatically include the desired protection.

5.2.3 Library Routines

Thus, an intermediary ground is to intercept sensitive data within a few key display and graphics library routines.

5.3 User Interface

5.3.1 Early Design

In the early user-interface design, a user would define English-like aliases in a repository to indicate sensitive data items that they wish not to be shown on screen. For example, a user named John could define an alias for his name as Joe. To discern these aliases when processing, an alternative input channel was used to identify them. This initial design proved to be problematic in a number of ways.

One way to achieve this effect is to add a separate software keyboard that a user would use whenever they want to input sensitive information. Essentially, this keyboard would be an app with elevated privilege to offer input across applications, and it would be easier to port across platforms, deploy, install, and update. However, changing keyboards amidst a stream of input is cumbersome in practice. This method would result in the loss of functionality offered in default keyboards, including swipe-based inputs, emoji support, auto correction, and custom dictionaries.

One step further is to replace the default keyboard with ours, which provides ways (e.g., a screen tapping sequence) to switch modes between normal entries with sensitive entries. Using this method, legacy functionalities such as auto correction and custom dictionaries can be retained. The user learning curve would also be less steep, since no awkward keyboard switching would be involved. On the other hand, the development effort of this approach would be significantly higher, and it would be harder for novice users to install the system, namely, by carrying out the replacement of the default keyboard.

5.3.2 Direct Input of Cashtags

While there are other input interface options than these, the need to perform cumbersome switches of input modes so that the aliases can appear as normal text seems superfluous in many contexts (e.g., using "visa" to represent the 16-digit credit card number).

Thus, the next phase of development explored the use of cashtags, where aliases are prepended with a $ sign, to represent sensitive information. By doing so, a user can directly enter cashtags, and the mode change is explicitly encoded in the cashtag alias (e.g., use `$fname` to represent John and `$gmail` to represent jsmith@gmail.com). This method can leverage the existing custom dictionary for auto completion, which makes it easier for the user to remember and input cashtags. This method can also utilize standard application level development techniques, opening up the range of supported device platforms and decreasing development and installation efforts.

5.3.3 Direct Input of Sensitive Information

Another alternative (albeit with some potential for information leak) is for a user to attempt to enter the initial characters of a sensitive data item. As soon as the auto completion detects that Jo, for example, is likely to mean Joe, it will be automatically masked with the cashtag `$John`. The user then can choose `$John` and proceed.

5.3.4 Additional Cashtags Semantics

Recursion is supported, so a user can define cascading sequences of alias replacements. For example, consider use John Smith, with Cashtags for `$first_name, $last_name $gmail`, with corresponding mappings to John, Smith, and jsmith@gmail.com respectively.

John could further define $signature to represent `$first_name, $last_name $gmail`. Then, John could use $signature with cascading expansion:

`$signature → $first_name $last_name $gmail → John Smith, jsmith@gmail.com`

The system checks for circular recursion mappings that result in infinite recursions. For example, John could not define `$John → $Joe` and `$Joe → $John`, as this would result in infinite recursion:

`$John → $Joe → $John → $Joe → $John ...`

Such situations are checked when the user defines a new alias, and the system does not permit the user to store the alias.

5.4 Accessing Sensitive Information

One design issue involves when to convert cashtags back to the sensitive data for accesses by and from apps. Normally, when an app is accessing sensitive information, it is doing so for the purpose of using the data remotely away from the device, such as a hosting server. For example, in typical online shopping, a credit card number is entered locally to make a remote transaction. Thus, the conversion from alias back to sensitive data must be performed prior to that access, else the app may not be able to cache, store, or transmit the data element as designed. The main concern in this case is that cashtags may not adhere to the type or formatting constraints and break an app inadvertently.

Another important consideration is to ensure that the cashtags are actually entered by the user, not just pre-populated by the app. Otherwise a malicious app could possible extract sensitive information just by displaying a form filled with common aliases.

There are also certain exceptions where it is desirable to directly operate on cashtags instead of the sensitive information. For example, the auto completion task will auto complete cashtags (`$fn` to `$fname`), not the sensitive information it represents. By doing so, the

handling of text span issues is simplified because cashtags usually differ in text lengths when compared to the sensitive information they represent.

5.5 Variants of Data Formats

Sensitive data may be represented in multiple formats. For example, names can be represented as combinations of first, last and middle initials (e.g., John Smith; John Q. Smith; Smith, John Q). Accounts and social security numbers can be represented using different spacing and/or hyphenation schemes (e.g., 123456789; 123-45-6789; 123 45 6789). Fortunately, many existing regular expression libraries (`java.util.regex.*`) can be leveraged to perform such pattern matching.

Another issue involves the type restriction of the input field. For example, a number field (e.g., social security number) may prevent the use of cashtags (`$ssn`). To circumvent these restrictions, support is provided to permit the user to define special aliases (e.g., 000-00-0000) in place of cashtags to represent certain types of sensitive information (e.g., social security numbers).

5.6 Deployment and Development Models

To avoid modifying individual applications, two options were considered to provide system-level changes: (1) custom system firmware images (ROMs) or (2) code-injection frameworks (e.g., Android Xposed). The capabilities and tradeoffs for each method are shown in Table 2.

By utilizing a custom system firmware image, complete control of the operating system is provided. (This approach assumes that the full source is available for modification.) In addition, ROM-based solutions can offer a more unified testing environment. However, the changes would be restricted to device-specific builds; only hardware for which the source is explicitly built would have access to the modified system. This also limits user preference by restricting use only for a specific system image. It would additionally require regular maintenance, and would break vendor over-the-air update functionality.

Table 2: Comparison between custom system firmware and code-injection framework design alternatives.

CUSTOM FIRMWARE VS. CODE-INJECTION FRAMEWORK		
	Custom Firmware Image	Code Injection Framework
Full OS Control	√	√
Portability		√
Ease of Deployment		√
Ease of Development		√
Ease of Testing	√	√

Instead, a code-injection framework was selected. This approach dynamically permits overriding routines as a library, incorporated into execution prior to the starting of apps. Code injection offers more streamlined development, as standard user application development tools can be used. In addition, these modules can be more easily deployed since they can be distributed as applications. Because code injection only relies on the underlying system using the same set library, the deployment is also much more portable and much less coupled with the exact versions and configurations of system firmware.

5.7 Cashtags App and Repository

Cashtags aliases and sensitive data items are maintained in an internal repository. The Cashtags app coordinates the interactions between various apps and the repository. The app also provides password-protected access to add, edit, remove, import, and export sensitive terms and corresponding cashtags.

Cashtags provides per-application blacklisting, excluding specific applications from being code-injected (or activated) with cashtag-replacement code. For example, the cashtag repository itself must be excluded due to circular dependencies. To illustrate, suppose a cashtag entry maps `$first_name` to Joe. If Cashtags is enabled, the screen will show that

`$first_name` is mapped to `$first_name`. When the entry is saved, Joe will be mapped to Joe. Thus, Cashtags is always excluded from servicing itself. Individual application packages can be excluded for lack of relevance to sensitive information exposure or for performance issues (e.g. games, application launchers, home screens).

CHAPTER SIX

IMPLEMENTATION

Cashtags was prototyped on the open-source Android platform. The usage of code-injection framework rather than custom firmware images allows Cashtags to operate on any Android device with the same display and graphics libraries and root access. This section will first detail the display data path in the Android context, then explain the code-injection framework, and finally discuss the details of how various display data paths are intercepted and how cashtags are stored.

6.1 Android Display Elements

Figure 21 has already shown a top-level view of various ways Android apps and browsers display information on the screen. This section provides further background on Android terminologies before beginning the detailed explanation of the system implementation. The corresponding terminologies for iOS and Windows are listed in Table 3.

Table 3: Widget terminologies on Android, iOS, and Windows operating systems.

WIDGET TERMINOLOGY ON OS PLATFORMS			
	Android	iOS	Windows
Text Labels	`TextView`	`UITextView`	`TextBlock`
OpenGL Text	`GLES20Canvas`	`GLKView`	`Direct3D`
Editable Text	`TextView`	`UITextView`	`TextBlock`
Webapp Text	`WebView`	`UIWebView`	`WebView`
Browser/Web Views	`WebView`	`UIWebView`	`WebView`

Android on-screen display is composed of views, layouts, and widgets. View is the base class for all on screen user interface components. All visual elements are descendants of this base class.

6.1.1 Widgets

The term widget is used to describe any graphic on-screen element. Different widgets can be used to display static text labels (e.g., `TextView`), user input boxes (e.g., `EditText`), controls (e.g., Buttons), and other media (e.g., `ImageView`).

Views are organized into ViewGroups, the base class for all screen layouts. Layouts are arrangements of views within vertical or horizontal aligned containers (e.g., `LinearLayout`), or arranged relative to other views. Nesting of ViewGroups and layouts allows more complex custom composites to be defined.

Collectively, this tree of layouts and widgets is called the view hierarchy. When the screen canvas is drawn, the view hierarchy is converted from logical interface components into a raw screen bitmap. Figure 22 shows a simple user input form and its composition of various widgets and layouts.

6.1.2 Text Rendering

Text can be rendered on screen through several mechanisms (Figure 21), the most common being through the `TextView` widget. Fonts, styles, colors, and so forth can be applied to specify how these are displayed. An `EditText` is an extension of the `TextView` that provides an interface for text input. This input can come from the user via the on-screen software keyboard, (integrated, plugged, or wirelessly connected) hardware keypads, voice input, gestures, and so forth. Like `TextView`, these widgets can be pre-filled with text by the app internally. They can also be set through suggestion or auto-correction interfaces.

Text can also be rendered on screen via OpenGL (`GLES20Canvas`) or other graphic rendering libraries. Unlike the `EditText`, this class does not inherit from the base `TextView`, although similar interfaces do exist.

Text can further be rendered on-screen from HTML and Javascript via browser rendering engines such as WebKit or Chromium. This includes mobile web browsing applications as well as many other cross-platform web app APIs such as Phonegap, Apache Cordova, and JQuery Mobile.

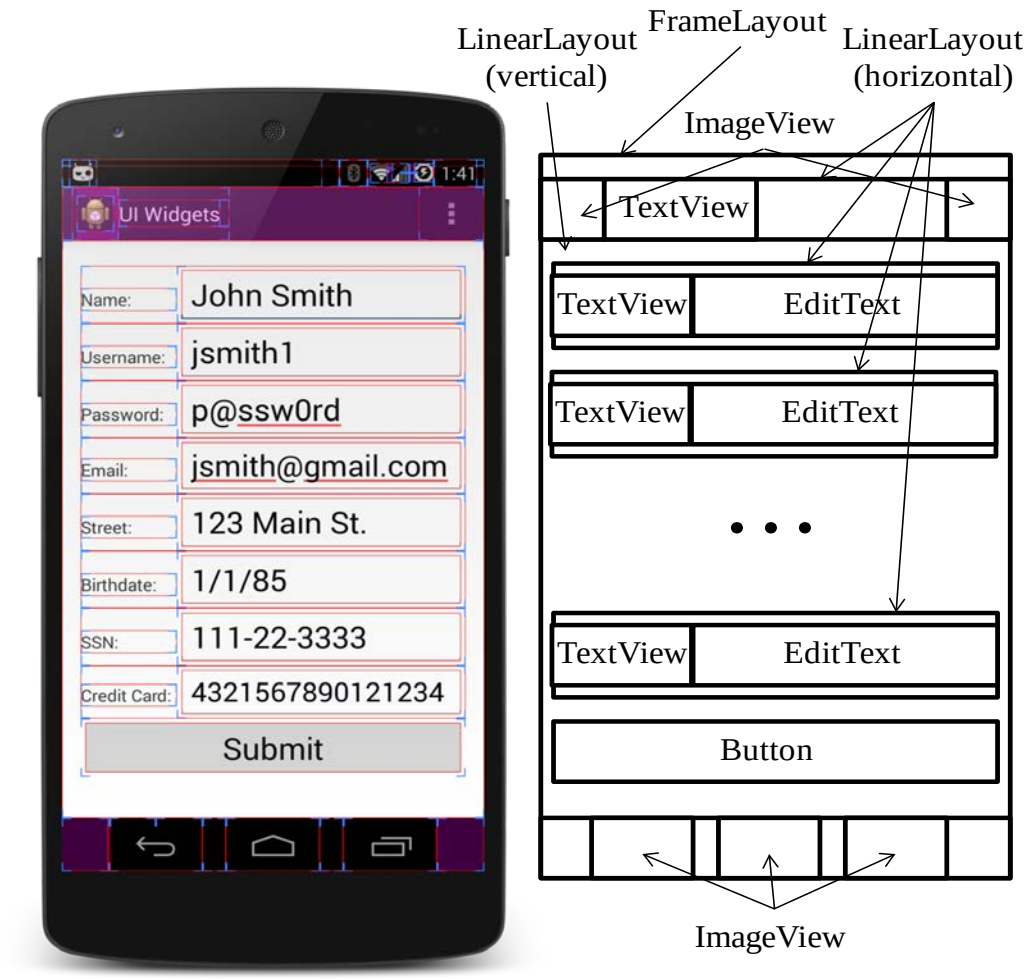

Figure 22: Decomposition of on screen views, layouts, and widgets of a simple app input forms.

6.2 Android Code-injection Framework

Cashtags uses the Android Xposed code-injection framework to intercept and modify the behavior of text widgets at runtime, without being tied to specific system firmware images. The development cycle is also accelerated by short-circuiting the need to perform complete device firmware rebuilds from scratch.

Underneath the hood, whenever an app is started, Android forks off a new virtual machine. The Android Xposed framework allows additional overriding library routines to be

inserted into the Java classpath, prior to the execution of the new virtual machines. Thus, the overall system behavior is altered without modifying either the apps or the underlying firmware.

Individual class methods can be hooked, allowing injected code to be executed prior to a base method calls, following the completion of the base method call, or in place of the base method. Private or protected member fields and functions can also be accessed and modified, and additional fields or functions can be added to the base class or object granularity. Figure 23 shows the API provided by Xposed for injecting method, constructor, and fields.

```
hookAllMethods() / hookAllConstructors()
findMethod() / findConstructor() / findField()
callMethod() / callStaticMethod() / newInstane()
getXXXField() / setXXXField()
getStaticXXXField() /setStaticXXXField()
getAdditionalXXXField() / setAdditionalXXXField()
```

Figure 23: Code injection API provided by XposedBridge. XXX denotes the specified data type, boolean, int, float, etc.

6.3 Sensitive Data Intercepting Points

With the background of Android display data paths (Figure 21) and the code-injection framework, it is possible to determine where and how to intercept sensitive information. As shown in Figure 24, all text-displaying screen widgets are descendants of the base `TextView` class. Thus, injection of `TextView` (`android.widget.TextView`) is sufficient to intercept all static sensitive text for any descendant class. For input, injection of `EditText` (`android.widget.EditText`) is sufficient to capture sensitive data or cashtags entered via on-screen software keyboard, (integrated, plugged, or wirelessly connected) hardware keypads, voice input, and gestures. For display through the OpenGL libraries, the `GLES20Canvas` (`android.view.GLES20Canvas`) is intercepted. For browsers, the point of interception is within the `Webview` (`android.WebKit/WebView`) class.

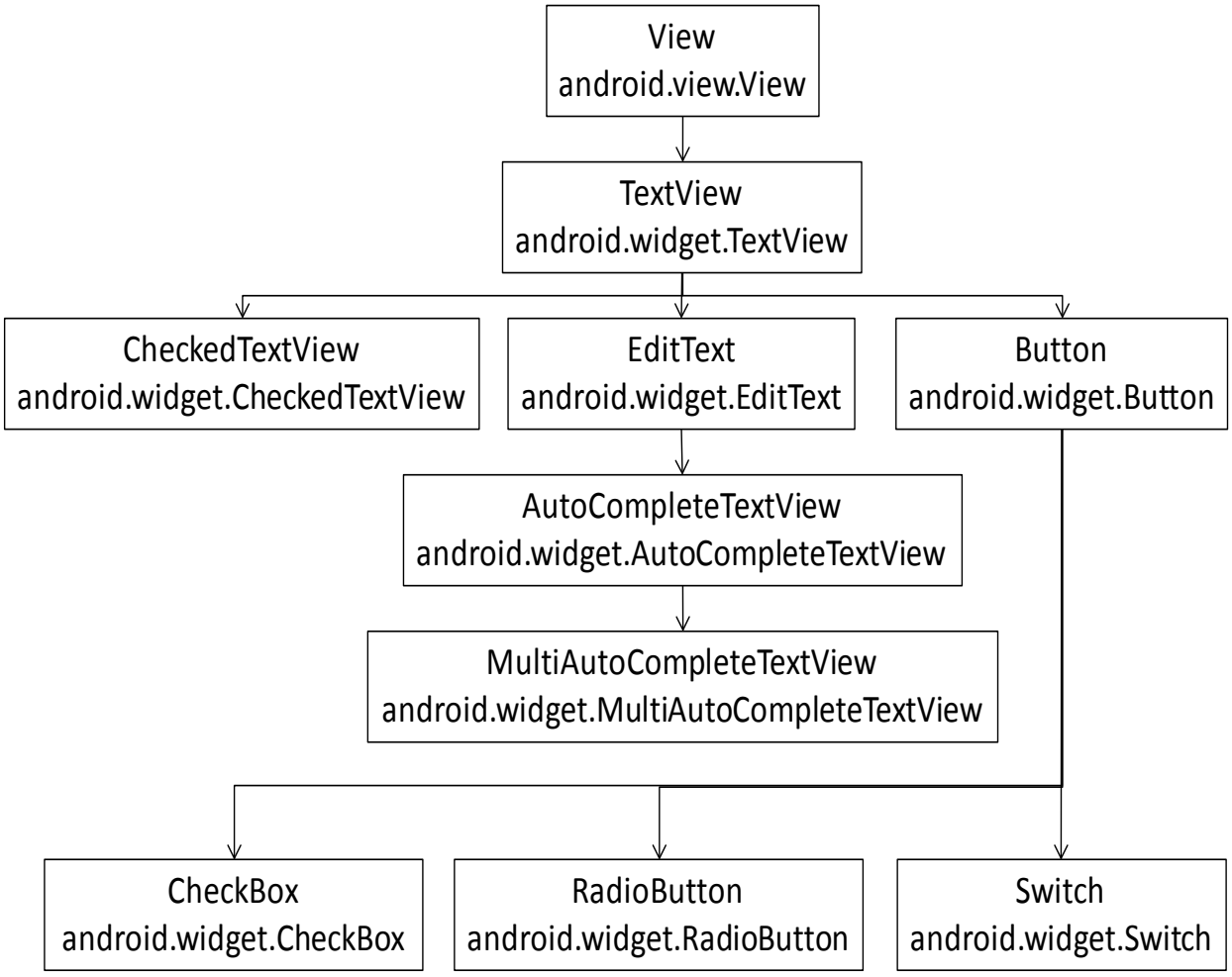

Figure 24: Simplified Android screen widget view hierarchy.

6.3.1 TextView

Figure 25 shows a simplified version of the implementation of the `TextView` widget in the Android API, present since version 1 of the Android SDK. The `getText()` and `setText()` methods of the `TextView` are hooked and modified (the `setText()` method in `TextView` is inherited by `EditText`, to be detailed later). In addition, a private member `mAlias` is also added track the mapping of the sensitive text to the corresponding cashtag.

```
public class TextView extends View implements
 ViewTreeObserver.OnPreDrawListener {
    ...
    private CharSequence mText;
    private CharSequence mAlias:
    ...
    public CharSequence getText() {
        return mText;
    }
    ...
    private void setText(CharSequence text, BufferType
      type, boolean notifyBefore, int oldlen) {
        ...
        mBufferType = type;
        mText = text;
    }
    ...
 }
```

Figure 25: Simplified `TextView` implementation. Bolded functions `getText()` and `setText()` are hooked and modified. An additional private field `mAlias` is added for mapping to a displayed cashtag, if applicable.

Figure 26 and Figure 27 show how Cashtags interacts with `TextView` and `EditText` objects. When these `getText()` and `setText()` methods are called by the app or through system processes like auto correct or to be rendered on screen, Cashtags will determine whether to return the alias or the sensitive data, depending on the caller.

6.3.2 EditText

`EditText` objects are more complex since additional actions can be performed by the user, app, or system to modify on-screen text. For cases where the system or app has pre-populated a text box with input, the `TextView` injection handles the cashtag replacement. Since the `EditText` class extends from the `TextView` base class, this functionality is provided

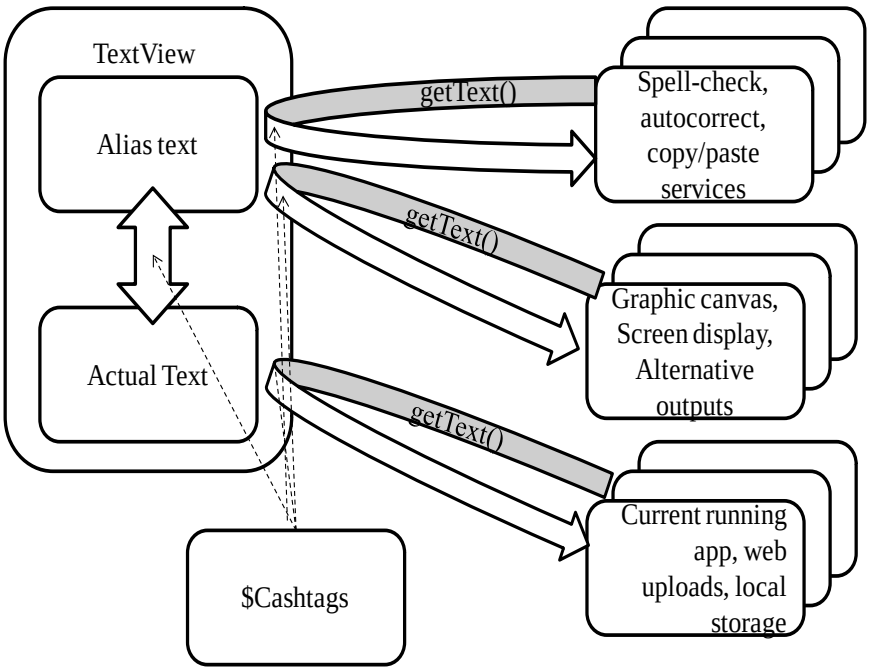

Figure 26: Interactions among Cashtags, `TextView`, and other software components. The `TextView` method `getText()` returns either the cashtag or actual text depending upon the service making the request.

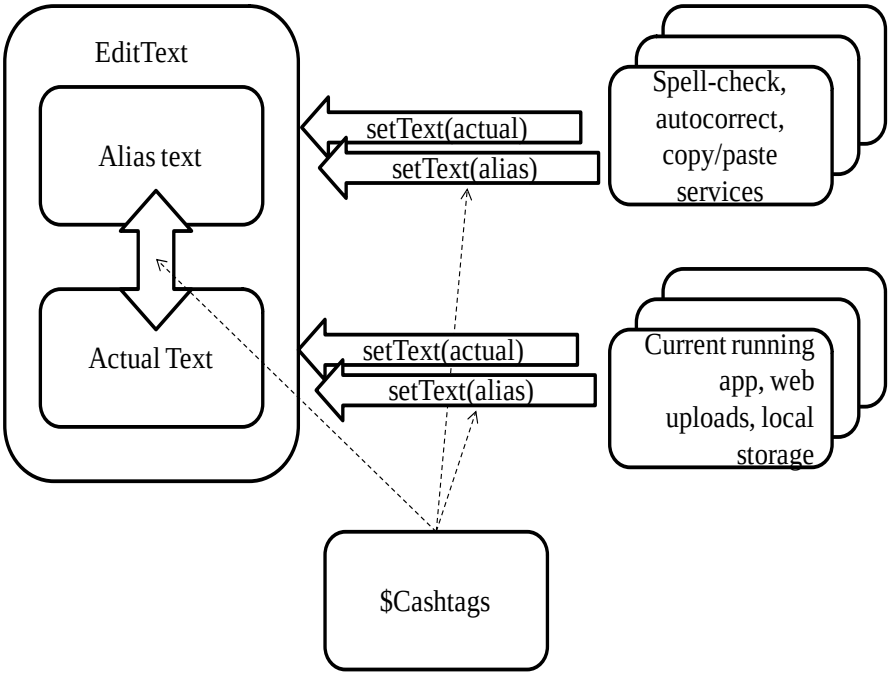

Figure 27: Interactions among Cashtags, `EditText`, and other software components. The `EditText` method `setText()` returns either the cashtag or actual text depending upon the service making the request.

through inheritance. This is also the case for nearly every other on-screen text widget as they are also hierarchically descendant from the base `TextView`.

User input can be entered through software keyboards or through physical devices. In both cases, Cashtags operates similar to, and through the same interface, as the auto-correct service. This `TextWatcher` (`android.text.TextWatcher`) interface handles events when on-screen text has been modified. `EditTexts` internally maintain an array of these `TextWatcher` event handlers. Cashtags, as one of these handlers, is activated after any character is modified within the text field. This per-character granularity of replacement maximizes the protection offered by the Cashtag alias replacement.

This functionality is also achieved through the view `OnFocusChangeListener` (`android.view.View.OnFocusChangeListener`). This event handler works at the granularity of the full text field rather than individual character of the `TextWatcher`. This is more efficient, and incurs much less overhead since the text replacement only occurs once per text field. It does, however, risk additional on-screen exposure of sensitive information, since direct input of actual sensitive terms would remain on-screen as long the cursor remains in that text field. For example, if the user is directly inputting an account number, the number will remain visible until they move to the next text input box, at which point it would be masked with the corresponding alias. Input of cashtag alias does not have this risk and further reduces any partial exposure during term input.

In both the per-character and per-field cases, the constructor of the `EditText` class is dynamically hooked at boot-time using the Xposed `findAndHookConstructor( EditText.class, Context.class, editTextMethodHook)` method. At point, the respective `OnFocusChangeListener` and/or `TextWatcher` is attached. User settings allow activation of either or both options within the Cashtags app settings.

6.3.3 OpenGL Canvas

The implementation solution for OpenGL (`android.view.GLES20Canvas`) is quite similar in simplified form to the base `TextView` only with different parameter types. The only distinction is that no accompanying `getText()` equivalent is present in this object, so no additional manipulation is necessary beyond `drawText()`.

6.3.4 WebView

Distinct from the previous screen widgets, rendering occurs independently of the native UI data path via underlying WebKit or Chromium browser engines. The relevant interception points for screen rendering for these are all below the accessible Android/Java layer and are not able to be code-injected though the same mechanisms used for previous screen widget cases. Using custom compilations of the browser engines with similar widget display interception was explored, but abandoned for portability concerns.

Instead, `WebView` interception is handled similarly to a web browser plug-in. This decision maintains the portability goal of the system design.

Cashtags intercepts web rendering immediately before it is first displayed on-screen. The HTML is pre-processed with JavaScript to extract the DOM. Cashtags iterates over the text nodes and makes the appropriate text replacements of sensitive data to corresponding cashtags.

Another option explored the use specific browser and proxy requests through web servers. However, all apps that use cross-platform frameworks (Phonegap, Apache Cordova, JQuery Mobile, etc.) run locally and could not easily be piped though this service. For this reason, the plug-in approach was selected over other alternatives.

6.4 Cashtags Repository

Sensitive terms are stored as encrypted `SharedPreference` data, which uses AES encryption from the Java Cryptography Architecture (`javax.crypto.*`). This structure is accessed by enabled apps through the `XposedSharedPreference` interface.

6.5 Crowd-sourced Debugging

Cashtags was debugged in part by means of a novel crowd-sourced paradigm. The principle is conceptually simple: The overall project framework is broken down in smaller, independently useful and practical subprojects. Each of these projects can then be published separately and with open source without ever revealing the nature of the project as a whole. Additionally, the open-source nature of the subprojects provides opportunity for the engagement

of the global development community. As issues are reported by users, improvements and bug fixes are then merged back into the main framework.

In the application of this paradigm to the Cashtags framework, these subprojects were all distributed as Xposed modules. Given the requirement of the Xposed framework for functionality, the projects were primarily directed at power users; users who are most willing to beta test new applications and also the most likely to provide useful feedback and submit bug reports.

The most notable of these subprojects was Xposed Macro/Text Expansion. The Xposed module provides functionality to automatically expand text sequences in any text box in most Android apps, as shown in Figure 28.

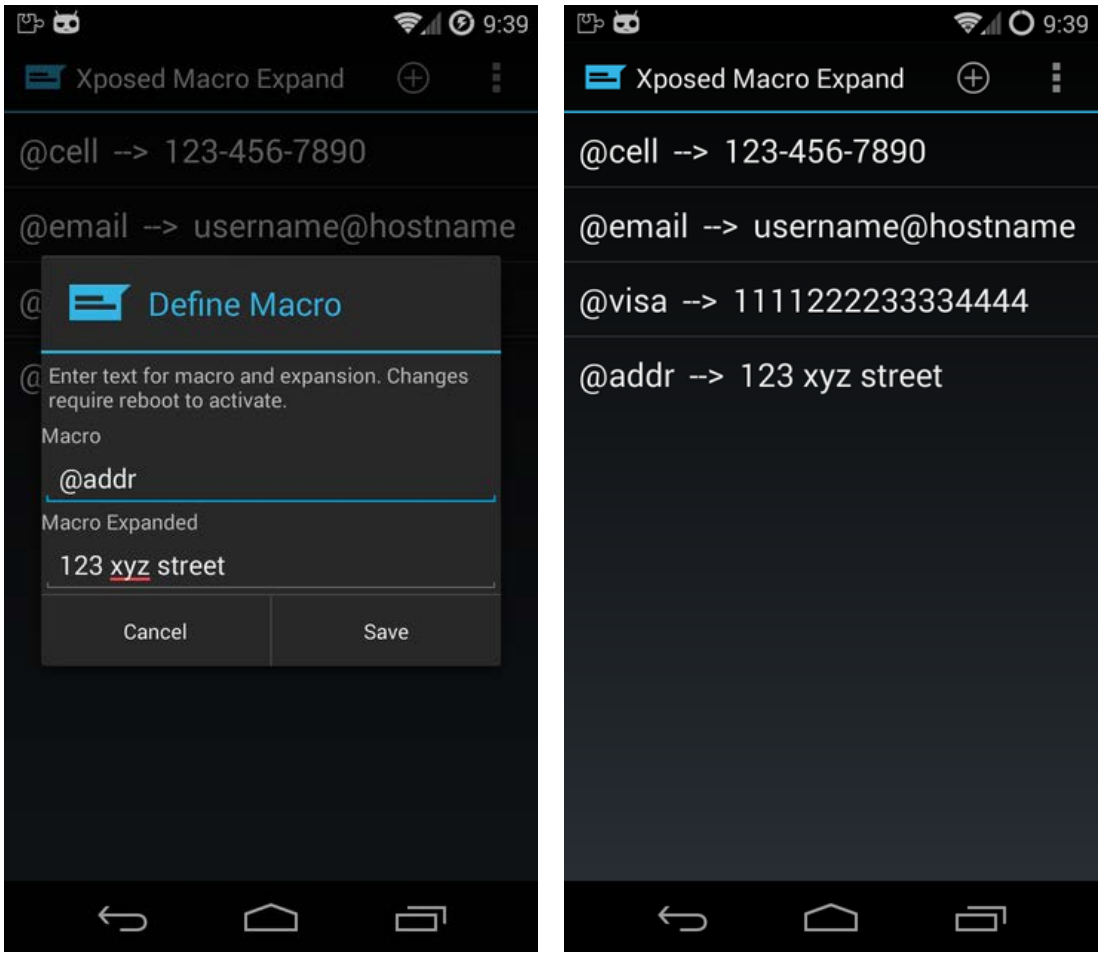

Figure 28: Xposed Macro/Text Expansion subproject of Cashtags, developed using crowd-soured development.

At present, over 10,000 users have downloaded and used this module. The full app description is as follows:

"The module allows short key sequences to be transformed into other, usually more time-consuming, sequences of keystrokes. This means that frequently used or repetitive sequences of keystrokes can easily be automated. Since it uses Xposed and operates at the text widget level, it works with all factory and third party keyboards, and can be added to user dictionaries for even faster use."

The functionality familiarity to Cashtags is not coincidental; the module used the same portion of the Cashtags code base to achieve this functionality. In addition to the volume of app usage, the project also yielded a significant amount of community involvement. Direct bug report and indirect feedback lead to over a dozen fixes and improvements to the module, and thus indirectly, to the Cashtags framework.

CHAPTER SEVEN

EVALUATION

Cashtags was evaluated for how well the intercepted points prevent specified information from being displayed on screen, verified by screen captures and OCR processing. This coverage was measured by enumerating common ways sensitive text can traverse to the screen. The system also evaluated user input through popular apps, making sure that cashtags correctly reverted to the sensitive data items when accessed by apps. Finally, Cashtags is evaluated for performance and usability overhead.

7.1 API Coverage Evaluation

The first test is for Android API coverage. The primary focus is directed at the `TextView` and `EditText` display data paths, which account for more than 86% of usage hours for mobile devices [56]. The selected sensitive information (Table 1) is based on the Personally Identifiable Information (PII) chosen categorically based on US government and NIST standards [59]. All possible combinations of input phrase type is enumerated (e.g., numbers, strings, etc.), and tested for case sensitivity, rendering through common widgets, with alternate layouts, themes, and other configuration options for these data paths. Each combination is used to demonstrate that the PII terms are not displayed on screen from the app internally, as user input of the sensitive data directly, or as user input of cashtag alias. In all three cases, the evaluation demonstrates that the PII term is correctly returned from Cashtags when used internally by the app.

This totals 1,728 tests for static text widgets and inputs, with 526 additional test cases for widgets that permit user input via both software keyboards as well as physical devices (on-board hardware, USB or wireless input devices). The full list of configurations is shown in Table 4.

Table 4: Android API Test Combinations.

ANDROID API TEST COMBINATIONS
Input phrase type (4): Alphabetic phrase, numeric phrase, alphanumeric phrase, Alphanumeric with symbols
Phrase case (2): Case Sensitive Text, Case In-sensitive Text

Widget type (9):
`TextView (android.widget.TextView),`
`CheckedTextView(android.widget.CheckedTextView),`
`Button (android.widget.Button),`
`CheckBox (android.widget.CheckBox),`
`RadioButton (android.widget.RadioButton),`
`Switch (android.widget.Switch),`
`EditText (android.widget.EditText),`
`AutoCompleteTextView (android.widget.AutoCompleteTextView),`
`MultiAutoCompleteTextView`
`(android.widget.MultiAutoCompleteTextView)`

Layout type (2):
`LinearLayout (android.widget.LinearLayout),`
`RelativeLayout (android.widget. RelativeLayout)`

Theme type (3): Default theme, System theme, User-defined theme
Generation method (2): Static XML, Dynamic Java
Lifecycle type (2): Activity-based app lifecycle, Fragment-based app lifecycle

For each test combination in Table 4, the Android Debug Bridge [60] and UIautomator tool [70] is used to capture device layout view hierarchies and screenshots of each case. The contents of the actual and cashtag fields within the view hierarchy XML are compared for conversion correctness. The device screenshot is processed using Tessseract OCR [71] and

confirms if the actual PII term has been properly masked on screen. To confirm the correctness of the OCR detection, this test was also preformed prior to installation of cashtags, which resulted in successful detection of all sensitive data elements with 0% OCR error rate.

For each combination, the evaluation also demonstrates that both text input as an actual sensitive term and cashtag are correctly converted to the actual sensitive term when accessed internally by the app. Since the access of sensitive data within the app normally involves remote actions, this scenario was also emulated with remote verification performed on the offloaded data. Once screen processing is completed, the app accesses the text fields and uploads to Google Sheets/Form. The uploaded actual sensitive items and cashtag submissions are compared for accuracy based on expected values. A flowchart of the overall API testing process is shown in Figure 29.

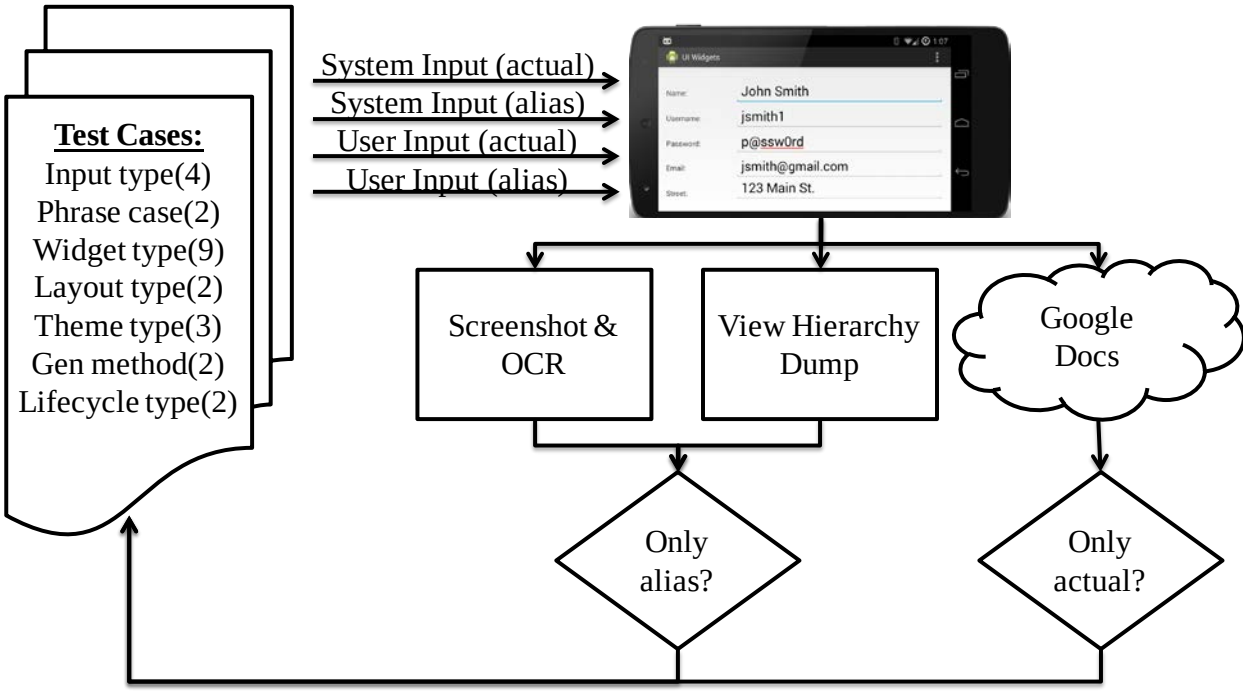

Figure 29: Graphical display of API testing process.

The results of the API evaluation show that Cashtags behaves correctly for all test cases. For each test case, Cashtags identified input containing sensitive data in both actual and cashtag

form, prevented the display on screen of the sensitive term, and determined correctly when to convert back to the sensitive data.

7.2 App Coverage Evaluation

The Google Play market has millions of published applications accessible by thousands of different hardware devices, making the enumeration of all possible users, devices, and application scenarios infeasible. Thus, a representative subset of popular apps must be selected to demonstrate app coverage of Cashtags. Categorically, these application types selected were those that commonly would encounter sensitive PII including email, messaging, social media, cloud and local storage, office, and finance. Table 5 shows the selected apps, arranged according to these categories along with the specific evaluation tasks performed on each app category. These apps were selected using download metrics from the Google Play marketplace (collected 9/2014), excluding games and utility apps for lack of relevance in terms of displaying sensitive data on screen. The presence of a form of external verification was also used in the application selection. The operation performed on each is based on a commonly performed use case or task for each category. Apps typically bundled with mobile devices were also tested for correct operation.

Table 6 shows that Cashtags using test cases from market apps shows correct behavior for 97% of task and app combinations, except the MS Office Mobile tests. The reason these tests does not work is due to the custom View used for the primary user interaction. This View (`id: docRECanvasHost`) is not a descendant of an `EditText` so is not intercepted by Cashtags. All other apps tested have user input through an `EditText`, or a custom class inheriting from an `EditText`. This particular view, as well as any other custom view beyond this scope could also be made to work with Cashtags using case-specific handling for the internal functions and parameters that map to the equivalent `EditText` function.

7.3 Performance Overhead

In terms of overhead, Cashtags was evaluated based on the incremental lag on the system. To perform this test, a modified version of the Android API coverage test (Section 7.1) was run with and without Cashtags enabled. Screenshots, layout hierarchy dumping, and all other non-

Table 5: Evaluation tasks performed for each category apps.

PER-CATEGORY APP TEST TASKS
Email: AOSP Email, Gmail, K9 Mail: A user reads an email containing a sensitive term and its corresponding cashtag. A Cashtags-enabled system should display the email with two instances of the cashtag. A user composes an email with a sensitive term and its cashtag. A remote system not running Cashtags should display the email with two instances of the sensitive term.
Messaging: Messaging, Google Hangouts, Snapchat: A user reads a message containing a sensitive term and its cashtag. A Cashtags-enabled system should display a message containing two instances of the cashtag. A user composes a message with a sensitive term and its cashtag. A remote system not running Cashtags should receive the message containing two instances of the sensitive term.
Social: Facebook, Twitter, Google+: A user reads text containing a sensitive term and its cashtag from tweet/post/update. A Cashtags-enabled system should display the tweet/post/update containing two instances of the cashtag. A user composes a new tweet/post/update with a sensitive term and its cashtag. A remote system not running Cashtags should receive the tweet/post/update with two instances of the sensitive term.
Storage: Dropbox, MS OneDrive, File Manager: A user opens an existing file containing a sensitive term and its cashtag. A Cashtags-enabled system should display the file containing two instances of the cashtag. A user creates a file with a sensitive term and its cashtag. A remote system not running Cashtags should see the file containing two instances of the sensitive term.
Office: GoogleDocs, MS Office Mobile, QuickOffice: A user reads a document containing a sensitive term and its cashtag. A Cashtags-enabled system should display the document with two instances of the cashtag. A user creates a document containing a sensitive term and its cashtag. A remote system not running Cashtags should see two instances of the sensitive term.
Finance: Google Wallet, Paypal, Square: A user reads a document containing a sensitive term and its cashtag. A Cashtag-enabled system should display the document with two instances of the cashtag. A user creates a document containing a sensitive term and its cashtag. A remote system not running Cashtags should see two instances of the sensitive term

Table 6: Market app coverage evaluation results.

MARKET APP COVERAGE EVALUATION				
	User Input Actual	User Input Cashtag	Remote Success Actual	Remote Success Cashtag
Email				
AOSP Email	√	√	√	√
Gmail	√	√	√	√
K9 Mail	√	√	√	√
Messaging				
Messaging	√	√	√	√
Google Hangouts	√	√	√	√
Snapchat	√	√	√	√
Social				
Facebook	√	√	√	√
Twitter	√	√	√	√
Google+	√	√	√	√
Storage				
Dropbox	√	√	√	√
MS OneDrive	√	√	√	√
File Manager	√	√	√	√
Office				
Google Docs	√	√	√	√
MS Office Mobile	√	√	X	X
QuickOffice	√	√	√	√
Finance				
Google Wallet	√	√	√	√
Paypal	√	√	√	√
Square	√	√	√	√

essential automation elements were removed prior to test execution. Test execution durations are compared, and additional incremental lag introduced by the system is calculated. This test is repeated with and without the remote data verification to determine the effects of network lags on system overhead. This testing flow is shown in Figure 30.

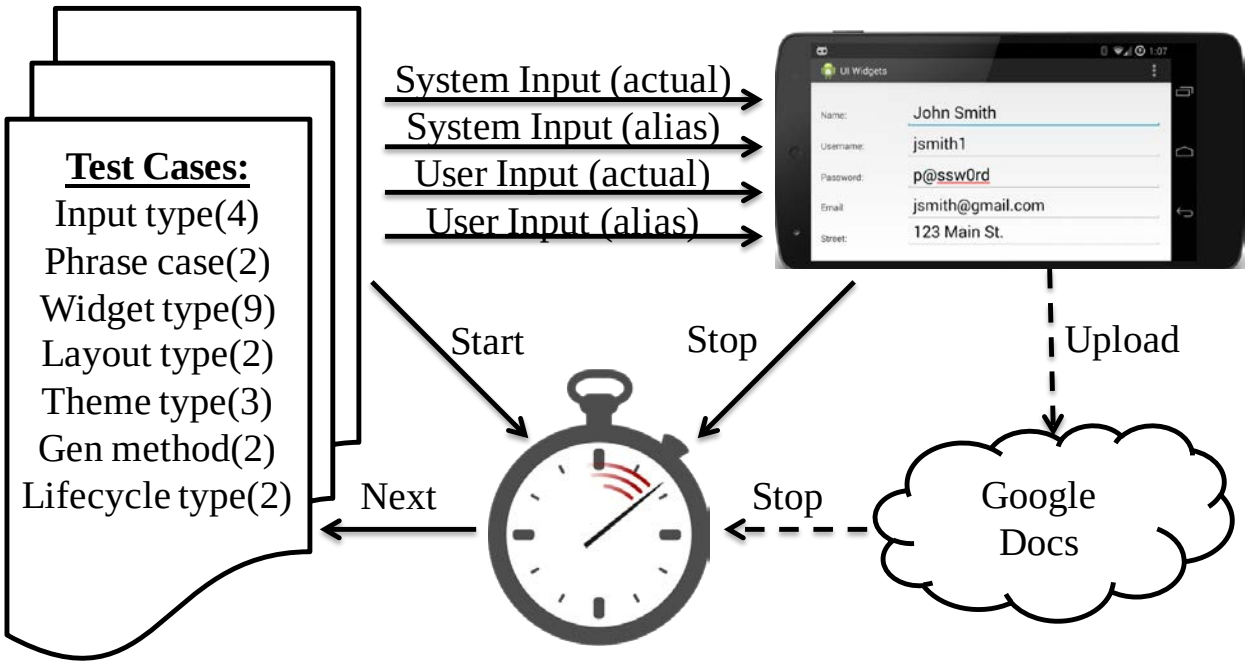

Figure 30: Flow of overhead evaluation test process. The test is repeated with and without data upload to show the impact of network communication on overall test duration. The dashed line shows the optional remote verification test step.

Figure 31 show the Cashtags system incurs an average 1.9% increase in test execution duration. For tests including remote verification, Cashtags incurred an average of a 1.1% increase over baseline tests. For tests excluding the time-consuming remote verification, Figure 32 shows that Cashtags incurred an average of 2.6% over baseline. Therefore, under such conditions, the additional overhead of Cashtags would not be perceivable to the user.

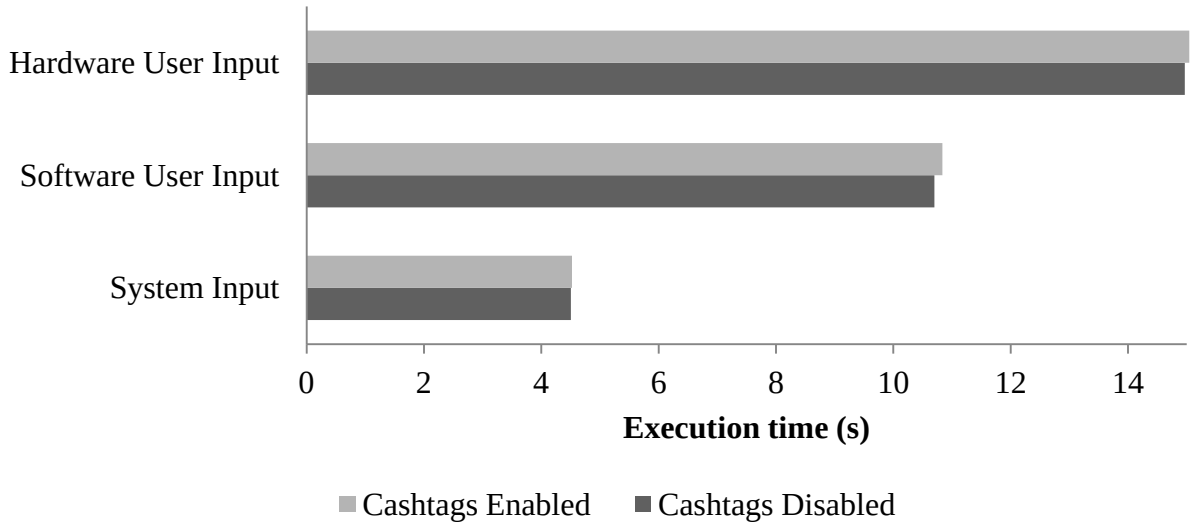

Figure 31: Comparison of mean app task execution time with and without Cashtags enabled, using system, software and hardware text input with web request for tests. Hardware input refers to input from physically or wirelessly connected hardware keyboard and Software Input to input from on screen software keyboard.

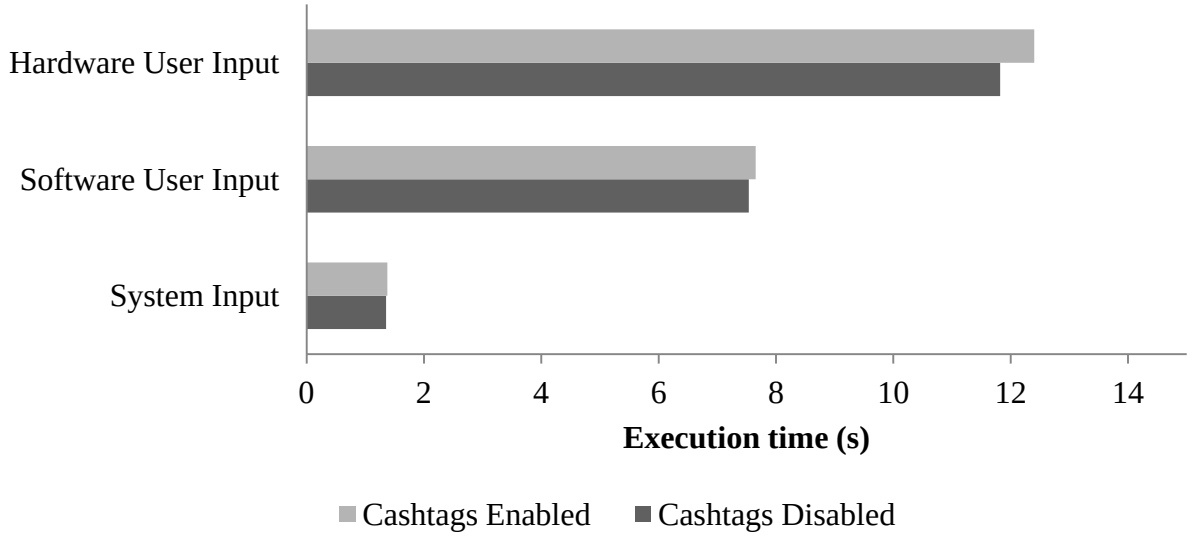

Figure 32: Comparison of mean app task execution time with and without Cashtags enabled, using system, software and hardware text input without web request for tests. Hardware input refers to input from physically or wirelessly connected hardware keyboard and Software Input to input from on screen software keyboard.

Testing was also repeated using more cashtag entries, with 50 and 100 items, which is significantly higher than the list of terms specified by PII. Figure 33 and Figure 34 show the results of these test runs for both system and user input data, using tests with and without the task inclusion of a web request.

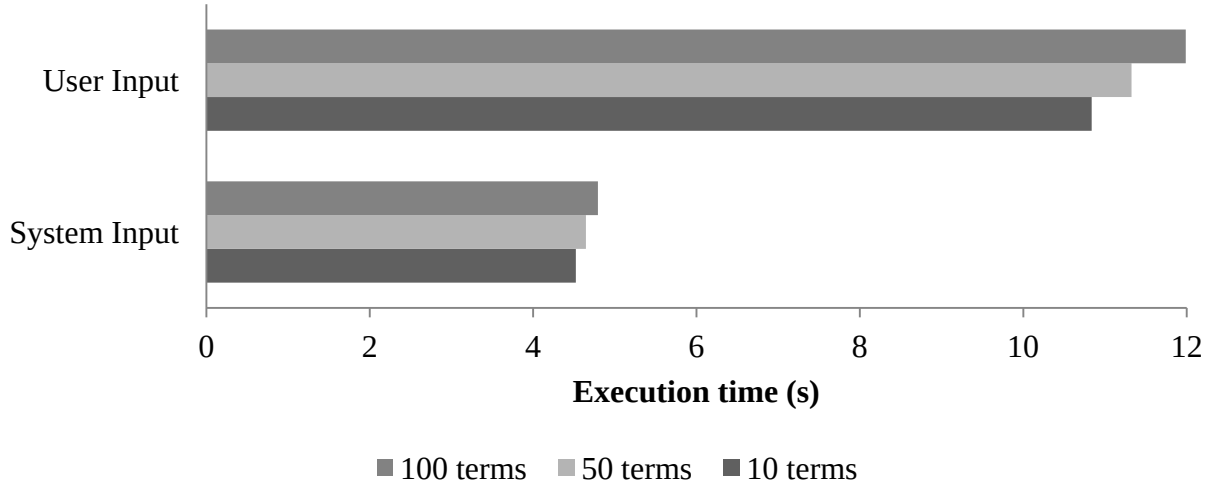

Figure 33: Comparison of mean app task execution time with an increasing number of cashtag entries, using system and user inputs with web request for tests.

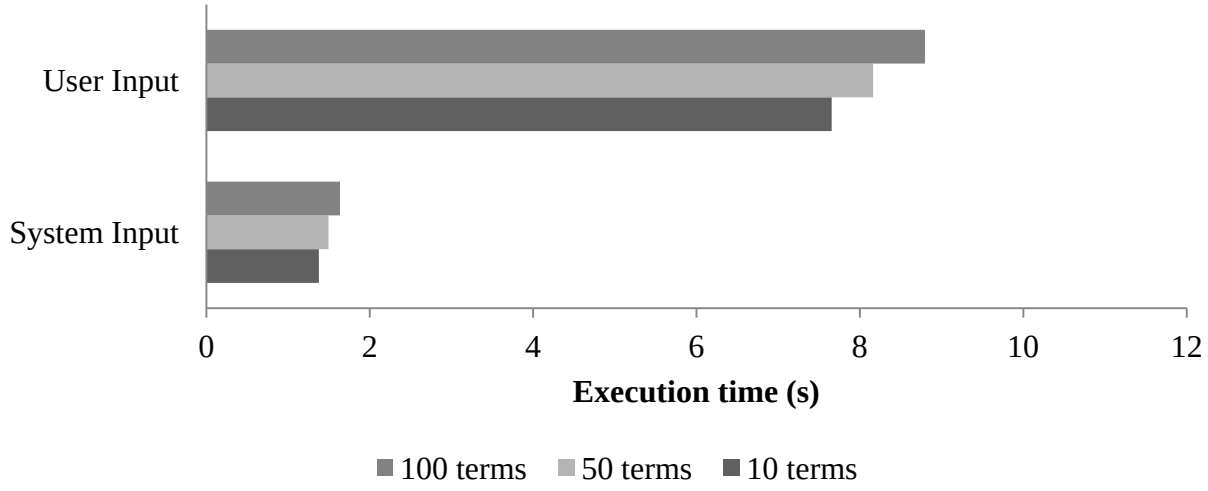

Figure 34: Comparison of mean app task execution time with an increasing number of cashtag entries, using system and user inputs without web request for tests.

Due to the current data structure, the performance degrades linearly as the number of cashtags entries increases. However, the Cashtags internal storage data structure could easily be replaced to make the degradation sublinear.

Cashtags is additionally evaluated for boot time overhead. Changes to the Cashtags repository currently require reboot to take full effect. While this operation is not in the common critical path, the additional overhead for this operation is relevant. The results of the boot lag are shown in Figure 35.

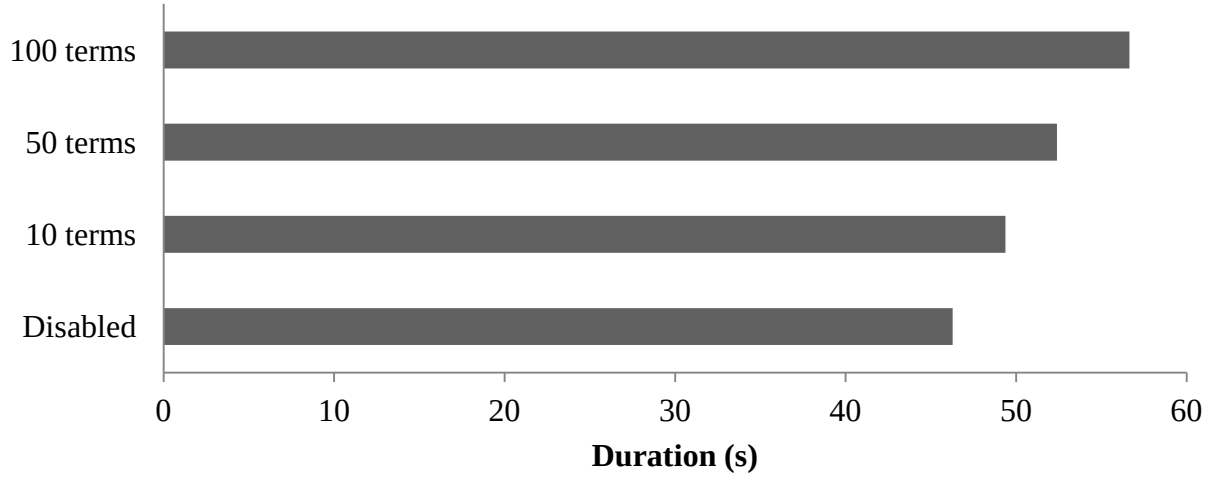

Figure 35: Comparison of device startup times with a varying number of cashtag entries and with system disabled.

7.4 Usability Overhead

To demonstrate the practical usage overhead of Cashtags, the configuration and usage overhead was calculated using data for a typical user. Common private data elements with known character lengths as well as those for which published data on average term length was available were used for the testing.

For other elements without known average length availability, a substituted a typical minimum accepted value was used instead. Table 7 shows the comparison of these fields against the suggested Cashtag alias length.

In nearly every case, the Cashtag alias is significantly shorter than the average length of the sensitive data element. On keystroke count basis, the amortized effort of the initial

configuration can be overcome with only two or three uses of the Cashtag alias. Longer names, emails, etc. require additional keystrokes for initial configuration but yield greater keystroke savings for each time the data element is entered. In addition, the aliases in the table are based on suggested terms for ease of user recall; even shorter terms could be substituted to further reduce additional data entry overhead.

Table 7: Typical keystroke counts for common sensitive private data terms [63][64] and corresponding suggested Cashtag alias.

KEYSTROKE COUNT COMPARISON				
Type	Actual	Alias	Alias	Diff
First Name	6	`$fname`	6	0
Last Name	6	`$lname`	6	0
Full name	13	`$name`	5	8
Email	20	`$email`	6	14
Username	9	`$user`	5	4
Password	9	`$pass`	5	4
Phone number	10	`$cell`	5	5
Birthday	10	`$bday`	5	5
SSN	9	`$ssn`	4	5
Credit Card	16	`$visa`	5	11
Acct. number	12	`$acct`	5	7

7.5 Quantification of the Time Savings for an End User

The real efficiency savings for Cashtags is the ability to represent terms not normally memorized with easy to recall aliases. While many personal identifiable identifiers like name, phone number, and address are commonly memorized, many other sensitive terms like credit card and account numbers are not. With Cashtags, even the longest and most complicated account number or term can be easily used without the need to ever look up. The user simply

assigns the data a memorable alias and can then always refer to that term for the represented data.

This also adds convenience when private data changes. While some data elements like name and SSN are essentially permanent, many others like addresses are variable. When these terms change, the Cashtag data mapping can be updated to reflect the change accordingly. For example, consider the case of stolen credit card. Once the replacement is issued, the user need only update the underlying data element, continuing to use the alias without the need to memorize a new number. In some cases of personal information change, the defunct data could still be considered private and be prevented from screen display. In such cases, the defunct data element could be assigned to a new alias. For example, consider a street address change: the alias $street is assigned to the new data, and the past street address is assigned to the new cashtag $street_old or other easily recalled term selected by the user.

CHAPTER EIGHT

LIMITATIONS AND FUTURE WORK

8.1 System Limitations

8.1.1 Coverage Limitation

Cashtags widget-level text manipulation works for apps that use standard text rendering methods. However, should developers deviate from such standards and create display data paths that do not inherit from the base text widgets, Cashtags would not capture such cases. Still, the additions required to incorporate these custom methods to work within Cashtags would be minimal if knowledge of the custom text display functions and parameters were provided.

8.1.2 Common Name Issue

Commonly occurring names can result in certain side effects. Consider a user John Smith, with Cashtag aliases of his name: John → $fname, and Smith → $lname. Therefore, all on-screen instances of John are masked as $fname. Now, John opens his mobile browser and googles for John Addams, John Travolta, or John Williams. All returned search results would be displayed with on-screen representations as $fname Addams, $fname Travolta, or $fname Williams, respectively. While this may or may not be acceptable to the user, it could also have the unintended consequence of inadvertently visually leaking private data. If an on-looker was able to observe the above search queries in the situation above, and was aware of the operation of Cashtags, they might be able to derive the sensitive data from context; in this case, determining that the user making the searches is named John. This limitation is isolated to common phrases; most instances of numerical phrases would not be relevant to this issue.

There are a number of ways that this shortcoming could be addressed. The common name situation as described above is limited to only a small number of mobile apps, most significantly for mobile web browsing. Since cashtags can operate at per-app granularity, if a user is aware in advance that they would be performing a web search of a term that could contextually leak sensitive data, they could temporarily disable cashtags for that browsing session.

It may also be the case that when private data is displayed on-screen, it is most frequently displayed along with additional private data. A commonly occurring example of this is the use case of inputting or viewing private data through web forms. If this relationship can be quantified, it would be possible to add thresholds to cashtags detection, with additional configurations to dynamically enable and disable text replacement based on the quantity of sensitive data elements displayed on-screen simultaneously. Additional contextual data determined by other text in close proximity could further improve this dynamic behavior.

8.1.3 Data Formatting

Data formatting and types is another issue. Many cases are handled through simple transformations of text fields, including the removal of spaces and symbols, and capitalization mismatches. However, on-screen data that expands between individual `TextViews` is not recognized, e.g., input fields for a credit card split into parts rather than combined into a single field. This could be handled by Cashtags if each part of the credit card number were individually added to the repository.

8.2 Handling Business Use Cases

Cashtags was originally envisioned and developed as a system to protect sensitive personal information leaks for the device owner. Thus, the user model, use cases, and evaluation have been presented to showcase the feasibility of the system and demonstrate the success of this goal.

However, the system is not limited to this application, and can be extended to provide more generalized protection from on-screen data leaks for business use cases. Many professions regularly access data containing sensitive data elements. This use case is becoming more commonplace, as progressively more computing is being performed on mobile devices. The built-in support for recursive processing of data elements permits more complex hierarchal structuring of Cashtags data elements allowing for range-based or categorical schemes.

Only minimal modifications would be required to support wildcards for specific data types. For example, to mask corporate identification numbers beginning with certain prefixes, or to mask all phone numbers on-screen. Further processing of text elements for specific patterns of

text and other data could also be applied to contextually determine which data fields may contain sensitive data. These fields could then be masked accordingly.

Finally, additional user interfaces could be added to connect to the Cashtags system, such as dynamic additions to the Cashtag repository and on-the-fly data masking. Such extensions are interfaces and would not require modification to the core system middleware.

8.3 Future Work

The current implementation of Cashtags is optimized for coverage rather than performance. Thus, one future direction would be to improve the scalability of the sensitive data repository. Disabling of specific classes of widgets unlikely to contain sensitive data is one solution. In addition, more efficient text processing methods and data structures can be considered.

Other future work could include the remote synchronization of Cashtags. Updates to sensitive actual and alias lists could be propagated to other devices automatically. Cashtags could also be modified to provide shared access for multiple users. Permissions could allow a user to share a cashtag for use by another without disclosing the sensitive data. In addition, this method would provide improved redaction of access to shared sensitive resource.

CHAPTER NINE

CONCLUSION

9.1 Summary of the Problem

Shoulder surfing is an important concern for the future of mobile computing. Users are more frequently computing in public locations on increasingly capable mobile devices. Whether unaware or oblivious to the problem, this behavior risks exposing sensitive information to bystanders via the screen display.

The pervasiveness of surveillance cameras, as well as those on smartphones and emerging wearable devices, is increasing the probability of both malicious and inadvertent capture of sensitive data. Text can be extracted from captured images and video accurately and efficiently through increasingly capable and cheap OCR solutions. The capture of just a small number of personal information elements can greatly increase the risk of other threats including social engineering attacks, phishing, and other personal identity theft threats.

In the corporate world, it is becoming increasingly commonplace for sensitive data to be accessed outside of protected workplace environments. Few organizations have taken the necessary steps to implement even the most basic guidelines and practices to reduce the risk of exposure. In addition to the potential for sensitive data loss, business productivity is also suffering as a consequence. Corporate surveys confirm the need for protection against the shoulder surf threat, as many professionals have been forced to stop working in public environments due to privacy concerns.

9.2 The Cashtags Solution

The Cashtags system provides protection against visual data leaks by replacing sensitive on-screen data elements with pre-defined aliases.

For input, a user can enter sensitive data in either actual and alias form. By directly typing a cashtag in place of the sensitive term, a user can perform more complex data-sensitive tasks without risk of observation from a bystander. In addition, cashtags are easier to remember

than the actual information itself. For example, `$visa` can be used as a shortcut for entering a 16-digit credit card number.

Cashtags maintains the alias mapping and returns either the cashtag or actual text depending upon the service making the request. This alias is replaced internally at the point at which the sensitive data would be used internally by the device or an app. This replacement provides legacy compatibility, allowing whatever login, communication, transmission, or upload to proceed normally.

By using Cashtags, a user can both access and input sensitive information in public without the fear of leaking this data through the screen.

9.3 Summary of the Cashtags Design

The summary of the major design points of Cashtags is as follows:

- ***Password-vault-like user model:*** Users pre-define a list of sensitive terms and corresponding aliases. These are stored in a centrally accessible encrypted Cashtags repository.
- ***Screen rendering interception:*** Cashtags intercepts sensitive data items as they are sent to the display. When sensitive data would be displayed on screen, the alias is displayed in place of the sensitive term. For apps, this point is located within their common textual rendering library routines.
- ***User input interception:*** User input is processed by Cashtags at either per-character of field-based granularity. Sensitive data input is masked with the corresponding alias, and alias inputs are identified to later convert to sensitive term at the point the app will use the data.
- ***Context-aware data return:*** Cashtags is aware of the service requesting the data. Cashtags returns either the cashtag or actual text depending upon the service making the request.
- ***Code-injection-based development and deployment model:*** Cashtags uses a code-injection framework. This approach avoids modifying individual apps and the

underlying system firmware, while altering the behavior of the overall system to incorporate Cashtags functionality at application runtime.

9.4 Lessons Learned

The motivation quantification, design, implementation and evaluation of Cashtags provide several major lessons.

Lessons learned from the human subjects' study:

- ***Human subject-based studies are difficult.*** Studies require extensive planning and approval from IRB committees. Human subject recruitment can be time consuming. Analysis of large multi-dimensional data sets can be complex. It comes as little surprise that few related studies have been performed.
- ***Users' understanding of privacy is different than that of the security community***. To the user, privacy concerns may be the desire to be alone, not wanting to be disturbed, or not revealing information to those closest to them. Since users should be provided with the privacy protection they want and will actually use, researchers must ensure that their goals align with users' real privacy desires.

Lessons from the design and implementation:

- ***Using aliases is effective in preventing display of sensitive data on-screen.*** Direct replacement of text does not interfere with any system or app functionality. Aliases are more convenient for users to remember and often result in more efficient input and key-stroke savings. They are also compatible with legacy features like custom user dictionaries and auto-correct.
- ***A keyword-based approach to aliasing is most effective.*** Using screen-level masking can be effective, but at the expense of actual end-user functionality. A tag-based approach is more effective by only aliasing sensitive data elements at the per-term granularity, but at the expense of system efficiency and increased

development efforts. A keyword-based solution can achieve this per-term granularity without this additional overhead.

- ***Screen display interception is most effective within graphic rendering libraries.*** Moving the interception point lower into the window manager is more difficult since text has already been converted to bitmaps at this point. Conversely, moving the interception higher places it within the level of individual applications, moving the burden to the developer of each respective app.
- ***Code-injection frameworks are convenient alternatives to custom system images.*** As long as the point of injection is above the native interface, the ability to perform system modifications is equivalent. They also provide significantly decreased development efforts.

The evaluations of Cashtags also lead to several additional lessons:

- ***Text rendering interception within graphics libraries is nearly comprehensive.*** All possible combinations of screen widgets using the standard programming API are successfully handled by Cashtags. Only one exception was found while testing popular third party applications, which could also be handled on per-app basis.
- ***Code-injection frameworks are efficient.*** Only modest overhead is incurred by using these systems, making them a viable alternative to the more development intensive and time-consuming custom firmware solution.
- ***Most mobile apps adhere to standard API guidelines.*** Of the popular apps tested to evaluate the effectiveness of Cashtags, only one exception was identified. This suggests that code-injection at the API level should also have good coverage for other unrelated system modifications.
- ***Testing automation using OCR is accurate and efficient.*** Standard development tools are effective for unit testing and automation. The addition of screenshot capture, view hierarchy dumping, and OCR provides efficient visual testing automation.

9.5 Contributions

This dissertation contributes to the field of computing privacy in the following ways:

1. Insight into the elusive concept of privacy as it relates to actual user attitudes though a large-scale human subject study. The results of the ~600 person survey lead to these conclusions:
    - By performing similar computing tasks in public and private, users are either unaware of the potential risk or simply do not care about the preservation of privacy. Even technically savvy users do not alter their behavior based on their surroundings.
    - Users are more likely to alter their behavior around people closest to them rather than around strangers. This relative indifference to privacy threats posed by strangers might suggest a perception of anonymity, leading to a false sense of security.
    - Users seem to underestimate the potential threat posed by mobile apps by more frequently complying with permission requests from them than from their operating systems.
    - Users' understanding of privacy is different from that of the security community, suggesting opportunities for additional privacy studies.
2. The design, implementation, and evaluation of Cashtags, a privacy-enhancing system providing these features:
    - Protection against visual data leaks by replacing sensitive on-screen data elements with pre-defined aliases.
    - A mechanism to input sensitive data without the potential for loss, even under direct observation from a bystander.
    - System-wide, legacy compatible protection without custom firmware or the need to modify individual apps.
    - Minimal overhead, practical usability and convenience for the user.

9.6 Final Comments

Cashtags is a first step toward protection against visual leaks of on-screen data. The system demonstrates that it is possible to perform most mobile computing tasks in public locations without exposing sensitive personal information. The evaluation of the system shows that this is accomplished efficiently, with minimal perceived overhead. The app coverage test confirms that the system is general purpose and maintains full functionality with nearly all tested common use cases. These results suggest that Cashtags will likely also work on most other mobile apps, providing unified, device-wide protection against shoulder surfing.

APPENDIX A

IRB APPROVAL MEMORANDUM HSC# 2012.8779

The Florida State University
Office of the Vice President For Research
Human Subjects Committee
Tallahassee, Florida 32306-2742
(850) 644-8673, FAX (850) 644-4392

APPROVAL MEMORANDUM
Date: 12/21/2012
To: Michael Mitchell
Address: 4530
Dept.: COMPUTER SCIENCE
From: Thomas L. Jacobson, Chair

Re: Use of Human Subjects in Research
Mobile Application Usage and Sensitive Data Access (NSF grant: CSR:Medium:Collaborative Research. Facets: Exploring Semantic Equivalence of Files to Improve Storage Systems)

The application that you submitted to this office in regard to the use of human subjects in the proposal referenced above have been reviewed by the Secretary, the Chair, and one member of the Human Subjects Committee. Your project is determined to be Expedited per 45 CFR § 46.110(7) and has been approved by an expedited review process.

The Human Subjects Committee has not evaluated your proposal for scientific merit, except to weigh the risk to the human participants and the aspects of the proposal related to potential risk and benefit. This approval does not replace any departmental or other approvals, which may be required.

If you submitted a proposed consent form with your application, the approved stamped consent form is attached to this approval notice. Only the stamped version of the consent form may be used in recruiting research subjects.

If the project has not been completed by 12/20/2013 you must request a renewal of approval for continuation of the project. As a courtesy, a renewal notice will be sent to you prior to your expiration date; however, it is your responsibility as the Principal Investigator to timely request renewal of your approval from the Committee.

You are advised that any change in protocol for this project must be reviewed and approved by the Committee prior to implementation of the proposed change in the protocol. A protocol change/amendment form is required to be submitted for approval by the Committee. In addition, federal regulations require that the Principal Investigator promptly report, in writing any unanticipated problems or adverse events involving risks to research subjects or others.

By copy of this memorandum, the Chair of your department and/or your major professor is reminded that he/she is responsible for being informed concerning research projects involving human subjects in the department, and should review protocols as often as needed to insure that the project is being conducted in compliance with our institution and with DHHS regulations.

This institution has an Assurance on file with the Office for Human Research Protection. The Assurance Number is IRB00000446.

Cc: Andy Wang, Advisor
HSC No. 2012.8779

APPENDIX B

IRB APPROVAL MEMORANDUM HSC# 2013.10175

The Florida State University
Office of the Vice President For Research
Human Subjects Committee
Tallahassee, Florida 32306-2742
(850) 644-8673, FAX (850) 644-4392

APPROVAL MEMORANDUM (for change in research protocol)
Date: 03/05/2013
To: Michael Mitchell
Address: 4530
Dept.: COMPUTER SCIENCE
From: Thomas L. Jacobson, Chair

Re: Use of Human Subjects in Research
Mobile Application Usage and Sensitive Data Access (NSF grant: CSR:Medium:Collaborative Research. Facets: Exploring Semantic Equivalence of Files to Improve Storage Systems)

The application that you submitted to this office in regard to the requested change/amendment to your research protocol for the above-referenced project has been reviewed and approved.

Please be reminded that if the project has not been completed by 12/20/2013, you must request renewed approval for continuation of the project.

By copy of this memorandum, the chairman of your department and/or your major professor is reminded that he/she is responsible for being informed concerning research projects involving human subjects in the department, and should review protocols as often as needed to insure that the project is being conducted in compliance with our institution and with DHHS regulations.

This institution has an Assurance on file with the Office for Human Research Protection. The Assurance Number is IRB00000446.

Cc: Andy Wang, Advisor
HSC No. 2013.10175

APPENDIX C

HUMAN SUBJECTS INFORMED CONSENT FORM

Sensitive Personal Data Access with Smartphones

Dear Participant:

My name is Michael Mitchell, and I am a graduate student in the Department of Computer Sciences at Florida State University. I am conducting a study about mobile application usage and mobile access to sensitive data in potentially unsecured locations. The goal of the project is to identify how the data is accessed, applications are used to access this data, and in what locations and contexts sensitive data access is most likely to occur. Armed with this data, we hope to provide solutions to mediate the risks associated with data exposure during mobile application access.

The primary goal of the human subject component of the study will be to evaluate how mobile applications are used and how sensitive data is accessed by mobile users. Each participant will be asked to install a research application on their Android phone or tablet. The application will present the user with a series of questions about their mobile usage and sensitive data access habits. While the user is responding to the questions, the application will also collect other data about the user and information about the applications installed on the user's mobile device. This data is limited to device type, product id and manufacturer, operating system and kernel version, hardware features and capabilities, and list of installed mobile applications and vendors.

You have been invited to participate because you have confirmed that you are at least 18 years old. Participation in the study will involve responding to approximately 100 questions. The questions are primarily of single and multiple choice types, with a few short answer types as well. The survey should take at most 30 minutes of your time to complete. The survey itself can be taken as a traditional web form, or via Android mobile app. The contents of the survey are the same in both surveys; the Android app is optimized for smaller device screens. The

associated risk of the survey is minimal; none of the study procedures involve activities that cause risk or discomfort that you would not otherwise be exposed to during standard mobile device usage. If you choose to participate, you have the option to stop participating at any time without penalty or risk.

All questionnaire responses and digital media containing the data collected from the mobile research application will only be retained for the duration of the study. Only my co-investigators and I will have access to the data. To maintain the confidentiality of your records, the data that is retrieved from your mobile device will be assigned an anonymous code. The results of this research study may be published, but your identity will not be revealed. Only group findings will be reported. Confidentiality will be maintained to the extent required by law.

There are several potential benefits to participating in this research. The data that is gathered may be used to help create risk profiles for mobile application usage and for sensitive data access in unsecured environments. Global mobile privacy researchers will also directly benefit from the collected data. While there are a number of research groups in academia and commercial labs working in this area, there are currently no benchmark data sets available to create a standardized approach for data comparison.

Your participation in this study will help us further quantify the problem of secure mobile access to sensitive data and help researchers understand the circumstances surrounding this type of data access. It will also help us understand the feasibility of developing a system to mitigate mobile sensitive data access risks. The study itself will advance the development and shorten the time for the development of such a system.

If you have any questions concerning this research study before or after you give your consent, please email me. You may also contact my advisor and co-investigator An-I Andy Wang, PhD. If you have any questions about your rights as a research participant, or if you feel you have been placed at risk, you can contact the Chair of the Human Subjects Committee on the Institutional Review Board, through the Vice President for the Office of Research at (850) 644-8633.

Sincerely,

Michael Mitchell

Please initial here to indicate that you have read both pages of this letter ____
Please read below for more information.

I wish to participate in the above study.

I understand that my questionnaire responses and all digital media containing data collected from my mobile device will be retained with restricted access and will only be kept for the duration of the study. I understand that only Michael Mitchell and his co-investigators will have access to the data.

I understand that I may withdraw my consent and discontinue my participation at any time without penalty. In signing this consent form, I am not waiving any legal claims, rights, or remedies. A copy of this consent form will be given to me.

Name: _____
Signature _____ Date _____

APPENDIX D

HUMAN SUBJECTS QUESTIONNAIRE

Mobile Computing Device Usage Questionnaire

Objective
The purpose of this questionnaire is to help researchers understand the usage patterns of mobile computing devices in public places.

Personal Background
Gender: __ M __ F Age: _____
Major/Background: _____
How many years have you owned a mobile computing devices
(e.g., smartphone, tablet, laptop)? _____
If you have a smartphone, what type of phone do you have?
__ 3G __ 4G __ not sure

General Usage
Where/when do you compute? (Check all that apply)
__ at home __ at dorm/apartment __ in your office __ in a washroom
__ in class __in your car __ in a library __ waiting in line
__ in a park __ in a store __ on a bus/train/flight
__ in a restaurant __ in a sports arena __ when walking/exercising
__ at bus/train station or airport
__ other (please specify) _____

Do you change your computing behavior in the presence of the following people? (Check all that apply)
__ significant other __ siblings __ parents __ roommates
__ friends __ colleagues __ boss __ subordinates
__ children __ someone technically savvy
__ strangers who do not understand your language
__ strangers who understand your language
__ other (please specify) _____

Do you use WiFi connections in public places?
__ yes __ only if password protected __ no __ not sure

Mobile tasks
Which of the following tasks have you performed on your mobile devices? (check all that apply) For the checked tasks, please indicate the location (check all that apply) and frequency of the tasks as well.

Communication

Access emails
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Text messages
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Voice chat
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Video chat
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Entertainment

Listen to music
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Watch videos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Play games
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Take photos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Record videos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Social networking
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Browse web
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Read e-books
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Download e-books
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Buy e-books

__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Download music
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Download videos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Upload photos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Upload videos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Edit photos
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Productivity

Word processing
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Presentations
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Spreadsheets
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Calendar
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Phone book
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Finance

Check bank balances
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Update balances (e.g., Quicken)

__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Shop with credit cards
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Shop with online accounts (e.g., Paypal)
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Perform investment transactions
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for online shopping/credit card accounts (e.g., Amazon)
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for online merchant accounts (e.g., wordpress for income)
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for student/car/house loans
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for grants/awards/scholarships
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Pay bills
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Access coupons
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Tools

Maps __ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Weather
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Read/write reviews
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Personal

Monitor health
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for jobs
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Apply for housing
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Administrative Tasks

Sync device
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Back up device
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Download apps
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Reset the device
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Configure network
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Replace the OS
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Modify the OS
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

Other (please specify) _____
__ in public hourly/weekly/monthly/yearly/rarely/never
__ in private hourly/weekly/monthly/yearly/rarely/never

What do you generally do when your computer asks for your confirmation to do something? (e.g., installing an app, running a program)
__ always confirm __ always decline __ find out more
What do you generally do when an app asks you to accept a user agreement?
__ always agree __ always decline __ find out more

REFERENCES

[1] Honan, Brian. "Visual Data Security White Paper", July 2012. BH Consulting & European Association for Visual Data Security. http://www.visualdatasecurity.eu/wp-content/uploads/2012/07/Visual-Data-Security-White-Paper.pdf. Retrieved 4/2014

[2] Thomson, Herbert H, PhD. "Visual Data Breach Risk Assessment Study." 2010. People Security Consulting Services, Commissioned by 3M. http://solutions.3m.com/3MContentRetrievalAPI/BlobServlet?assetId=1273672752407&assetType=MMM_Image&blobAttribute=ImageFile. Retrieved 4/2014

[3] Vikuiti Privacy Filters. "Shoulder Surfing Survey". 2007. Commissioned by 3M UK PLC. http://multimedia.3m.com/mws/mediawebserver?6666660Zjcf6lVs6EVs66SlZpCOrrrrQ-. Retrieved 4/2014

[4] European Association for Visual Data Security. "Visual Data Security", March 2013. http://www.visualdatasecurity.eu/wp-content/uploads/2013/03/Secure-Briefing-2013-UK.pdf. Retrieved 4/2014

[5] International Data Corporation. "Worldwide Mobile Worker Population 2011-2015 Forecast." http://cdn.idc.asia/files/5a8911ab-4c6d-47b3-8a04-01147c3ce06d.pdf. Retrieved 4/2014

[6] Good Technology. "Americans are Working More, but on their Own Schedule", July 2012. http://www1.good.com/about/press-releases/161009045.html. Retrieved 4/2014

[7] Nokia, USA. "Nokia Lumia 1020", http://www.nokia.com/us-en/phones/phone/lumia1020/. Retrieved 4/2014

[8] NPD DisplaySearch. "Wide Viewing Angle LCD Technologies Gain Share Due to Tablet PC Demand". January 2012. http://www.displaysearch.com/cps/rde/xchg/displaysearch/hs.xsl/120119_wide_viewing_angle_lcd_technologies_gain_share_due_to_tablet_pc_demand.asp. Retrieved 4/2014

[9] Pillai, Geetha. "Caught on Camera: You are Filmed on CCTV 300 Times a Day in London", International Business Times, March 2012. http://www.ibtimes.co.uk/britain-cctv-camera-surveillance-watch-london-big-312382. Retrieved 4/2014

[10] Loh Zhi Chang and Steven Zhou ZhiYing. "Robust pre-processing techniques for OCR applications on mobile devices", In Proceedings of the 6th International Conference on Mobile Technology, Application & Systems (Mobility '09). ACM, New York, NY, USA, Article 60 , 4 pages. DOI=10.1145/1710035.1710095 http://doi.acm.org/10.1145/1710035.1710095

[11] Owen, Glen. "The zzzzivil servant who fell asleep on the train with laptop secrets in full view", November 2008. http://www.dailymail.co.uk/news/article-1082375/The-zzzzivil-servant-fell-asleep-train-laptop-secrets-view.html. Retrieved 4/2014

[12] Penn, Ivan. "Simple fix to bank security breach: Close the blinds", Tampa Bay Times. December 2010. http://www.tampabay.com/features/consumer/simple-fix-to-bank-security-breach-close-the-blinds/1139356. Retrieved 4/2014

[13] Davies, Caroline. "Prince William photos slip-up forces MoD to change passwords", The Guardian, November 2102. http://www.theguardian.com/uk/2012/nov/20/prince-william-photos-mod-passwords. Retrieved 4/2014

[14] J. Alex Halderman, Brent Waters, and Edward W. Felten. "A convenient method for securely managing passwords", In Proceedings of the 14th international conference on World Wide Web (WWW '05). ACM, New York, NY, USA, 471-479. DOI=10.1145/1060745.1060815 http://doi.acm.org/10.1145/1060745.1060815

[15] LastPass, "About LastPass", 2014. https://lastpass.com/about-lastpass. Retrieved 4/2014

[16] AgileBits, Inc. "1Password", 2014. https://agilebits.com/onepassword. Retrieved 4/2014

[17] Yubico, Inc. "About YubiKey", 2014. http://www.yubico.com/about. Retrieved 4/2014

[18] Square, Inc. "About Square", 2014. https://squareup.com/news. Retrieved 4/2014

[19] Google, Inc. "Google NFC YubiKey Neo", September 2013. http://online.wsj.com/news/articles/SB10001424127887323585604579008620509295960

[20] Wayne Jansen and Vlad Korolev. "A Location-Based Mechanism for Mobile Device Security", In Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering (CSIE '09), Vol. 1. IEEE Computer Society, Washington, DC, USA, 99-104. DOI=10.1109/CSIE.2009.719 http://dx.doi.org/10.1109/CSIE.2009.719

[21] C. Fleming, P. Peterson, E. Kline and P. Reiher, "Data Tethers: Preventing Information Leakage by Enforcing Environmental Data Access Policies," in International Conference on Communications (ICC), 2012.

[22] Blonder, Greg E. "Graphical Passwords". United States patent 5559961, Lucent Technologies, Inc. 1996.

[23] Passfaces Corporation. "The Science Behind Passfaces", June 2004. http://www.realuser.com/published/ScienceBehindPassfaces.pdf

[24] Darren Davis, Fabian Monrose, and Michael K. Reiter. "On user choice in graphical password schemes", In Proceedings of the 13th conference on USENIX Security Symposium - Volume 13 (SSYM'04), Vol. 13. USENIX Association, Berkeley, CA, USA, 11-11.

[25] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. "PassPoints: design and longitudinal evaluation of a graphical password system" International Journal of Human-Computer Studies. 63, 1-2 (July 2005), 102-127. DOI=10.1016/j.ijhcs.2005.04.010 http://dx.doi.org/10.1016/j.ijhcs.2005.04.010

[26] Jain, A.K.; Hong, L.; Pankanti, S.; Bolle, R., "An identity-authentication system using fingerprints," Proceedings of the IEEE, vol.85, no.9, pp.1365, 1388, Sep 1997. doi: 10.1109/5.628674

[27] J. Daugman. "How iris recognition works", IEEE Transactions on Circuits and Systems for Video Technology. 14, 1 (January 2004), 21-30. DOI=10.1109/TCSVT.2003.818350 http://dx.doi.org/10.1109/TCSVT.2003.818350

[28] Anil K. Jain, Arun Ross, Sharath Pankanti. "A Prototype Hand Geometry-based Verifcation System", In Proceedings of 2nd International Conference on Audio- and Video-based Biometric Person Authentication (AVBPA), Washington D.C., pp.166-171, March 22-24, 1999.

[29] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. "Face recognition: A literature survey". ACM Computing Surveys. 35, 4 (December 2003), 399-458. DOI=10.1145/954339.954342 http://doi.acm.org/10.1145/954339.954342

[30] Rick Joyce and Gopal Gupta. "Identity authentication based on keystroke latencies", Communications of the ACM ,33, 2 (February 1990), 168-176. DOI=10.1145/75577.75582 http://doi.acm.org/10.1145/75577.75582

[31] Davrondzhon Gafurov, Kirsi Helkala, Torkjel Søndrol. "Biometric Gait Authentication Using Accelerometer Sensor", Journal of Computers, Vol. 1, No. 7, October 2006.

[32] Roberto Brunelli and Daniele Falavigna. "Person Identification Using Multiple Cues", IEEE Transactions on Pattern Analysis and Machine Intelligence. 17, 10 (October 1995), 955-966. DOI=10.1109/34.464560 http://dx.doi.org/10.1109/34.464560

[33] 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 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 987-996. DOI=10.1145/2207676.2208544 http://doi.acm.org/10.1145/2207676.2208544

[34] Ioannis Leftheriotis. "User authentication in a multi-touch surface: a chord password system" In CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13). ACM, New York, NY, USA, 1725-1730. DOI=10.1145/2468356.2468665 http://doi.acm.org/10.1145/2468356.2468665

[35] Ming Ki Chong, Gary Marsden, and Hans Gellersen. "GesturePIN: using discrete gestures for associating mobile devices", In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 261-264. DOI=10.1145/1851600.1851644 http://doi.acm.org/10.1145/1851600.1851644

[36] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. "Smudge attacks on smartphone touch screens", In Proceedings of the 4th USENIX conference on Offensive technologies (WOOT'10). USENIX Association, Berkeley, CA, USA, 1-7.

[37] Volker Roth, Kai Richter, and Rene Freidinger. "A PIN-entry method resilient against shoulder surfing", In Proceedings of the 11th ACM conference on Computer and communications security (CCS '04). ACM, New York, NY, USA, 236-245. DOI=10.1145/1030083.1030116 http://doi.acm.org/10.1145/1030083.1030116

[38] T. Perkovic, M. Cagalj, and N. Rakic. "SSSL: shoulder surfing safe login", In Proceedings of the 17th international conference on Software, Telecommunications and Computer Networks (SoftCOM'09). IEEE Press, Piscataway, NJ, USA, 270-275.

[39] Alice Boit, Thomas Geimer, and Jorn Loviscach. "A random cursor matrix to hide graphical password input", In SIGGRAPH '09: Posters (SIGGRAPH '09). ACM, New York, NY, USA, Article 41, 1 pages. DOI=10.1145/1599301.1599342 http://doi.acm.org/10.1145/1599301.1599342

[40] Rohit Ashok Khot, Ponnurangam Kumaraguru, and Kannan Srinathan. "WYSWYE: shoulder surfing defense for recognition based graphical passwords", In Proceedings of the 24th Australian Computer-Human Interaction Conference (OzCHI '12), ACM, New York, NY, USA, 285-294. DOI=10.1145/2414536.2414584 http://doi.acm.org/10.1145/2414536.2414584

[41] Rachna Dhamija and Adrian Perrig. "Deja; Vu: a user study using images for authentication", In Proceedings of the 9th conference on USENIX Security Symposium - Volume 9 (SSYM'00), Vol. 9. USENIX Association, Berkeley, CA, USA, 4-4.

[42] Mary Brown and Felicia R. Doswell. "Using passtones instead of passwords", In Proceedings of the 48th Annual Southeast Regional Conference (ACM SE '10). ACM, New York, NY, USA, Article 82, 5 pages. DOI=10.1145/1900008.1900119 http://doi.acm.org/10.1145/1900008.1900119

[43] Andrea Bianchi, Ian Oakley, and Dong Soo Kwon. "The secure haptic keypad: a tactile password system", In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 1089-1092. DOI=10.1145/1753326.1753488 http://doi.acm.org/10.1145/1753326.1753488

[44] Alain Forget, Sonia Chiasson, and Robert Biddle. "Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords", In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 1107-1110. DOI=10.1145/1753326.1753491 http://doi.acm.org/10.1145/1753326.1753491

[45] Julie Thorpe, P. C. van Oorschot, and Anil Somayaji. "Pass-thoughts: authenticating with our minds", In Proceedings of the 2005 workshop on New security paradigms (NSPW '05). ACM, New York, NY, USA, 45-56. DOI=10.1145/1146269.1146282 http://doi.acm.org/10.1145/1146269.1146282

[46] Nitesh Saxena and James H. Watt. "Authentication technologies for the blind or visually impaired", In Proceedings of the 4th USENIX conference on Hot topics in security (HotSec'09). USENIX Association, Berkeley, CA, USA, 7-7.

[47] T. Dierks, E. Rescorla. "The Transport Layer Security (TLS) Protocol, Version 1.2", August 2008.

[48] Mason, Andrew G. "Cisco Secure Virtual Private Network". Cisco Press, 2002, p. 7.

[49] Marc Shapiro. "Structure and Encapsulation in Distributed Systems: the Proxy Principle", In Proceedings of the 6th IEEE International Conference on Distributed Computing Systems (ICDCS), Cambridge MA (USA), May 1986.

[50] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: the second-generation onion router", In Proceedings of the 13th conference on USENIX Security Symposium (SSYM'04), Vol. 13. 2004 USENIX Association, Berkeley, CA, USA, 21-21.

[51] Do Not Track. "Do Not Track - Universal Web Tracking Opt Out", http://donottrack.us. Retrieved 4/2014

[52] Adblock Plus. "Adblock Plus : About", https://adblockplus.org/en/about. Retrieved 4/2014

[53] Evidon, Inc. "About Ghostery", https://www.ghostery.com/en/about. Retrieved 4/2014

[54] Braden Kowitz and Lorrie Cranor. "Peripheral privacy notifications for wireless networks", In Proceedings of the 2005 ACM workshop on Privacy in the electronic society (WPES '05). ACM, New York, NY, USA, 90-96. DOI=10.1145/1102199.1102217 http://doi.acm.org/10.1145/1102199.1102217

[55] Sunny Consolvo, Jaeyeon Jung, Ben Greenstein, Pauline Powledge, Gabriel Maganis, and Daniel Avrahami. "The Wi-Fi privacy ticker: improving awareness & control of personal information exposure on Wi-Fi", In Proceedings of the 12th ACM international conference on Ubiquitous computing (Ubicomp '10). ACM, New York, NY, USA, 321-330. DOI=10.1145/1864349.1864398 http://doi.acm.org/10.1145/1864349.1864398.

[56] Simon Khalaf. "Apps Solidify Leadership Six Years into Mobile Revolution," Flurry, http://www.flurry.com/bid/109749/Apps-Solidify-Leadership-Six-Years-into-the-Mobile-Revolution, 2014.

[57] Google, Inc. Google Glass. http://www.google.com/glass/start/

[58] Rahul Raguram, Andrew M. White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. 2011. iSpy: automatic reconstruction of typed input from compromising reflections. In Proceedings of the 18th ACM conference on Computer and communications security (CCS '11). ACM, New York, NY, USA, 527-536. DOI=10.1145/2046707.2046769 http://doi.acm.org/10.1145/2046707.2046769

[59] Erika McCallister, Tim Grance, Karen Scarfone. Guide to Protecting the Confidentiality of Personally Identifiable Information (SP 800-122). National Institute of Standards and Technology, http://csrc.nist.gov/publications/nistpubs/800-122/sp800-122.pdf

[60] Android Developers, Android Debug Bridge. https://developer.android.com/tools/help/adb.html

[61] Kino. The Kino Project. http://sourceforge.net/projects/pskino/

[62] Screen Concealer. http://screenconcealer.com/

[63] U.S. Census Bureau, Population Division, Population Analysis & Evaluation Staff. http://factfinder.census.gov/

[64] Jonathan Lampe. “Beyond Password Length and Complexity”, The Infosec Institute. January, 2014. http://resources.infosecinstitute.com/beyond-password-length-complexity/

[65] Oculus Rift - Virtual Reality Headset for 3D Gaming. Oculus VR, LLC. https://www.oculus.com/

[66] Samsung Gear VR. Samsung Electronics Co., Ltd. https://www.samsung.com/global/microsite/gearvr/

[67] 3M Privacy and Screen Protectors. The 3M Company. http://solutions.3m.com/wps/portal/3M/en_EU/3MScreens_EU/Home/PrivacyFilters/

[68] Lenovo Sun Visor. Lenovo Group Ltd. http://blog.lenovo.com/en/blog/see-in-the-light

[69] Becky Stern. Compubody Socks or Knitted Body-Technology Interfaces. Sternlab. http://sternlab.org/?p=90

[70] Android Developers, Uiautomator. https://developer.android.com/tools/help/uiautomator/index.html

[71] Google, Inc. Tesseract-OCR. https://code.google.com/p/tesseract-ocr/

[72] CTSS Programmers Guide, 2nd Ed., MIT Press, 1965

[73] U.S. Census Bureau. “State & county quickfacts 2007,” http://quickfacts.census.gov, 2014.

[74] Florida State University. “About Florida State University: student body,” http://www.fsu.edu/about/students.html, 2014.

[75] Craigslist.org. “Craigslist: classifieds for jobs, apartments, personals, for sale, services, community, and events,” http://craigslist.org, 2014.

[76] Alexa, Inc. “Craigslist.org site info,” http://www.alexa.com/siteinfo/craigslist.org, 2014.

[77] Smith A. “Pew Internet Research,” Smartphone Ownership – 2013 Update. http://www.pewinternet.org/files/old-media//Files/Reports/2013/PIP_Smartphone_adoption_2013_PDF.pdf, 2014.

[78] King P. Strategy Analytics. Android Dominates the Tablet Market in 2013 Q2. http://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5403, 2014.

[79] NetMarketShare. Desktop Operating System Market Share. http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0, 2014.

[80] Sullivan B. How the Smartphone Killed the Three-day Weekend. http://www.cnbc.com/id/100765600, 2014.

[81] Lederer S, Mankoff J, Dey AK. “Who wants to know what when? privacy preference determinants in ubiquitous computing,” In CHI, 2003.

[82] Wiese J, Kelley PG, Cranor LF, Dabbish L, Hong JI, Zimmerman J. “Are you close with me? are you nearby? Investigating social groups, closeness, and willingness to share,” In UbiComp, 2011.

[83] Kelley PG, Benisch M, Cranor LF, Sadeh N. “When are users comfortable sharing locations with advertisers?” In CHI, 2011.

[84] Barkhuus L, Dey A. “Location-based services for mobile telephony: a study of users' privacy concerns,” In INTERACT, 2003.

[85] Barkhuus L. “Privacy in location-based services, concern vs. coolness. In Workshop on Location System Privacy and Control, 2004.

[86] Felt AP, Egelman S, Wagner D. “I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns,” In SPSM, 2012.

[87] Consolvo S, Jung J, Greenstein B, Powledge P, Maganis G, Avrahami D, “The wifi privacy ticker: improving awareness & control of personal information exposure on wifi,” In Ubicomp, 2010.

[88] Google, Inc. “Google docs—online documents, spreadsheets, presentations, surveys, file storage and more,” http://docs.google.com, 2014.

[89] Kientz JA, Choe EK, Truong KN. “Texting from the toilet: mobile computing use and acceptance in private and public restrooms. Knowledge Media Design Institute,” University of Toronto. Technical Report KMD-13-1, 2013.

[90] Walton M, Donner J. “Public access, private mobile: the interplay of shared access and the mobile Internet for teenagers in Cape Town.” Global Impact Study Research Report Series, 2012.

[91] Neilson.com. “Survey new U.S. smartphone growth by age and income,” http://www.nielsen.com/us/en/newswire/2012/survey-new-u-s-smartphone-growth-by-age-and-income.html, 2014.

[92] Boyd D. “Why youth (heart) social network sites: The role of networked publics in teenage social life,” MacArthur Foundation Series on Digital Learning–Youth, Identity, and Digital Media Volume, 2007.

BIOGRAPHICAL SKETCH

Michael was born in Louisville, KY in 1984. He graduated from Pella Community High School in 2002. Michael studied Computer Science and Management Information Systems at Iowa State University before later earning a BS in Software Engineering at the University of Phoenix in 2009, and a MS in Computer Science from Florida State University in 2011. His graduate research has covered a wide range of topics including mobile operating systems, context-aware computing, privacy, security, and semantic filesystems. The publication of this document marks the completion of Michael's PhD in Computer Science in 2015.

For the latest information about Michael and his work, visit his personal website: http://michaeljmitchell.com