

# THE USABILITY OF CAPTCHAS ON MOBILE DEVICES

by  
Gerardo Reynaga

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

DOCTOR OF PHILOSOPHY

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario

September, 2015

© Copyright by Gerardo Reynaga, 2015

## Abstract

Completely Automated Public Turing tests to tell Computers and Humans Apart (captcha) are challenge-response tests used on the web to distinguish human users from automated bots. Mobile devices such as smartphones and tablets have become a primary means of accessing online resources for many users, however most existing captchas do not properly fit mobile devices and may lead users to abandon tasks.

Captchas have become sufficiently hard for users to solve that some web sites refrain from deploying them and others are actively looking at alternatives. For users of smartphones, the reduced screen size can lead to typing mistakes and loss of position. In addition, environmental context and device orientation also have an impact on the user experience.

In this thesis, our research revolves around three primary, inter-related questions: *How can we effectively assess usability issues of captchas accessed on smartphones?* *What are the most prevalent usability issues of captchas accessed on smartphones?* *How can we improve captchas for smartphone usage?* We conducted lab and heuristic evaluations on existing and prototype captcha schemes, and identified areas for improvement. We developed, refined and tested a set of domain specific heuristics to evaluate captcha schemes on smartphones. We designed and tested four captcha prototypes to assess the viability of different input methods. From the empirical work, we identified design strategies for the development of new captcha schemes for smartphones.

## Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors Prof. Sonia Chiasson and Prof. Paul Van Oorschot for their excellent insights and continuous guidance. I cannot possibly quantify the importance of Prof. Sonia Chiasson's guidance, valuable advice and limitless patience during the dissertation work. Thanks to Prof. Paul Van Oorschot for his incredible eye for detail and his faith in me.

I would also like to thank my committee: Prof. Nur Zincir-Heywood for her insightful questions, Prof. Guy-Vincent Jourdan for his clarifying remarks and questions, Prof. Gabriel Wainer for his accurate observations, and Dr. Andrew Patrick for his expertise and inquiring comments on statistics. Their rapid feedback, expertise, and different perspectives have helped improve this dissertation. I am also grateful to them for agreeing to holding my defence particularly during the summer time.

Many thanks to Prof. Robert Biddle for unveiling the fascinating world of usable security, his stimulating lectures and unique ideas. Thanks to my colleagues in the Carleton's Human Oriented Research in Usable Security and Carleton's Computer Security Lab research groups who helped with experiments, listened to presentations, and offered valuable feedback and insight throughout the process.

To Kamilla and the kids for their tireless support, understanding and endless patience throughout this long journey. To my parents for their perpetual enthusiasm and support. Muchas gracias de todo corazón por su continuo apoyo. To the rest of my family and friends (Reykjavík and Guadalajara) for their support and celebration of various milestones. I am grateful to my friends in Ottawa who gave me a hand every time I needed it.

# Table of Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Tables</b>	<b>5</b>
<b>List of Figures</b>	<b>6</b>
<b>Chapter 1 Introduction</b>	<b>8</b>
1.1 Motivation . . . . .	8
1.2 Thesis Statement . . . . .	10
1.3 Thesis Overview and Organization . . . . .	11
1.4 Main Contributions . . . . .	12
1.5 Related Publications . . . . .	13
<b>Chapter 2 Background</b>	<b>15</b>
2.1 Introduction . . . . .	15
2.2 Captcha Taxonomy . . . . .	16
2.2.1 Character-Recognition Captchas . . . . .	17
2.2.2 Audio Captchas . . . . .	18
2.2.3 Image-Recognition Captchas . . . . .	19
2.2.4 Moving-Image Object Recognition Captchas . . . . .	20
2.2.5 Cognitive Captchas . . . . .	23
2.2.6 Summary . . . . .	24
2.3 Security Analysis . . . . .	25
2.4 Security Threats . . . . .	27
2.4.1 Human-in-the-loop attacks . . . . .	27
2.4.2 Automated attacks . . . . .	28
2.4.3 Replay attack . . . . .	30

	2
2.4.4 Naïve attacks . . . . .	31
2.4.5 Summary of Attacks on Captchas . . . . .	31
2.5 Captcha Applications . . . . .	31
2.5.1 Defence mechanisms . . . . .	33
2.5.2 Collateral purposes . . . . .	35
2.6 Alternative Proposals to Captchas . . . . .	36
2.7 Captchas and Smartphones . . . . .	37
2.7.1 Captcha proposals for mobile devices . . . . .	37
2.7.2 Captcha user studies for mobile usage . . . . .	39
2.8 Usability Analysis . . . . .	40
2.8.1 User Studies . . . . .	40
2.8.2 Heuristic evaluation . . . . .	41
2.8.3 Thematic Analysis . . . . .	43
2.9 Summary . . . . .	44
<b>Chapter 3 Captcha User Studies</b>	<b>45</b>
3.1 Introduction . . . . .	45
3.2 User Study 1: Captchas on Desktops . . . . .	46
3.2.1 Captcha Schemes . . . . .	47
3.2.2 Methodology . . . . .	50
3.2.3 Data Collection . . . . .	51
3.2.4 Analysis . . . . .	51
3.2.5 Outcomes . . . . .	52
3.3 User Study 2: Captchas on Smartphones . . . . .	59
3.3.1 Captcha Schemes . . . . .	59
3.3.2 Methodology . . . . .	60
3.3.3 Data Collection . . . . .	61
3.3.4 Analysis . . . . .	62
3.3.5 Outcomes . . . . .	63
3.4 Discussion . . . . .	67

<b>Chapter 4</b>	<b>Heuristics for the Usability Evaluation of Captchas on Smartphones</b>	<b>68</b>
4.1	Introduction . . . . .	68
4.2	Heuristic Evaluations . . . . .	69
4.3	Developing the MC Heuristics . . . . .	70
4.4	Proposed Heuristics: Mobile Captcha Heuristics (MC) . . . . .	72
4.5	Assessing the MC Heuristics . . . . .	78
4.5.1	Target Schemes . . . . .	80
4.6	Evaluation 1: Heuristic Evaluation - the MC Heuristics . . . . .	80
4.6.1	Data Analysis . . . . .	81
4.6.2	Results: Raw Problems . . . . .	82
4.6.3	Results: Unique Problems. . . . .	84
4.6.4	Results: Evaluator Performance . . . . .	85
4.6.5	Results: Overall Rating per Heuristic . . . . .	86
4.7	Evaluation 2: Heuristic Evaluation - Nielsen and Molich Heuristics . . . . .	88
4.7.1	Data Analysis . . . . .	89
4.7.2	Results: Raw Problems . . . . .	89
4.7.3	Results: Unique Problems. . . . .	89
4.7.4	Results: Evaluator Performance . . . . .	90
4.8	Effectiveness of the MC Heuristics . . . . .	90
4.8.1	Comparison between MC and Nielsen . . . . .	91
4.8.2	Comparison between MC and User Study . . . . .	94
4.9	Discussion . . . . .	95
<b>Chapter 5</b>	<b>Smartphone Captcha Prototypes</b>	<b>98</b>
5.1	Introduction . . . . .	98
5.2	Existing Schemes . . . . .	100
5.3	Proposed New Input Mechanisms . . . . .	101
5.3.1	Mobile Prototype Design . . . . .	102
5.3.2	Proposal 1 (P1) - Gesture reCaptcha . . . . .	102
5.3.3	Proposal 2 (P2) - Gesture Emerging . . . . .	105
5.3.4	Proposal 3 (P3) - Asirra Slide . . . . .	107

	4
5.3.5	Proposal 4 (P4) - reCaptcha Buttons . . . . . 110
5.3.6	Recognizers' Accuracy . . . . . 112
5.3.7	Design Challenges . . . . . 113
5.4	User Study . . . . . 115
5.4.1	Methodology . . . . . 115
5.4.2	Data Collection . . . . . 119
5.4.3	Data Analysis . . . . . 119
5.5	Outcomes . . . . . 120
5.5.1	Performance . . . . . 120
5.5.2	Post-hoc Statistical Analysis of Correct Answers . . . . . 122
5.5.3	User Perception . . . . . 124
5.5.4	Experimenter Observations . . . . . 126
5.5.5	Summary of Results . . . . . 130
5.6	Discussion . . . . . 131
5.7	Conclusion . . . . . 132
<b>Chapter 6</b>	<b>Design Recommendations and Conclusion 133</b>
6.1	Introduction . . . . . 133
6.2	Design Recommendations . . . . . 134
6.2.1	Recommendations for design of challenges . . . . . 134
6.2.2	Recommendations for screen layout . . . . . 136
6.2.3	Recommendations for environment of use and deployment . . 138
6.3	Research Objectives . . . . . 140
6.4	Research Contributions . . . . . 141
6.5	Directions for Further Research . . . . . 142
6.6	Conclusion . . . . . 144
<b>Bibliography</b>	<b>145</b>
<b>Appendix A</b>	<b>Nielsen and Molich's Heuristics 159</b>
<b>Appendix B</b>	<b>Bertini <i>et al.</i> Heuristics 161</b>

## List of Tables

2.1	Character Recognition-based scheme features . . . . .	18
2.2	Audio-based scheme features . . . . .	20
2.3	IR-based scheme features . . . . .	21
2.4	MIOR-based scheme features . . . . .	22
2.5	COG-based scheme features . . . . .	24
2.6	Different types of attacks on the captcha classes. . . . .	32
3.1	MIOR study. Mean and standard deviation . . . . .	53
3.2	MIOR study. Sample participant comments for each variant. . . . .	58
3.3	Captchas on smartphones study. Mean and standard deviation . . . . .	62
3.4	Captchas on smartphones study. Sample participant comments. . . . .	66
4.1	Methods for developing domain specific evaluation heuristics. . . . .	70
4.2	Methods of validating domain specific heuristics. . . . .	78
4.3	MC Heuristics. Unique problems found by heuristic per scheme. . . . .	84
4.4	Heuristic Evaluations. Problem classification. . . . .	91
4.5	Unique problems for both HEs and user study. . . . .	92
5.1	Overview of experimental setup. . . . .	118
5.2	Summary of performance results. . . . .	120
5.3	Tukey comparisons for each scheme in group (B). . . . .	123
5.4	Sample comments from participants. . . . .	127



## List of Figures

2.1	Examples of Captchas . . . . .	17
2.2	Captcha taxonomy . . . . .	17
2.3	Examples of CR captcha schemes . . . . .	19
2.4	Examples of IR captcha schemes . . . . .	21
2.5	Examples of MIOR captcha schemes . . . . .	22
2.6	Examples of COG captcha schemes. . . . .	23
2.7	Example of captcha scheme proposals for mobile devices. . . . .	39
3.1	Standard replicates NuCaptcha’s original scheme . . . . .	47
3.2	Extended variant. . . . .	48
3.3	Overlapping characters . . . . .	48
3.4	Three backgrounds used for the challenges, shown for the <i>Semi-Transparent</i> variant . . . . .	48
3.5	Emerging captcha. . . . .	49
3.6	MIOR study. Mean number of success, error, and skipped outcomes for <i>Standard, Extended, Overlapping, Semi-Transparent and Emerging</i> variants, respectively. . . . .	52
3.7	MIOR study. Time taken to solve the MIOR captchas. . . . .	54
3.8	MIOR study. Location of errors within the codewords. . . . .	55
3.9	MIOR study. Likert-scale responses . . . . .	56
3.10	Captchas on smartphones study. Target Schemes. . . . .	60
3.11	Captchas on smartphones study. Mean number of success, error and skipped outcomes. . . . .	63
3.12	Captchas on smartphones study. Likert-scale responses . . . . .	65
4.1	MC Heuristics. Target Schemes. . . . .	80
4.2	MC Heuristics. Mean severity ratings for all raw problem reports. . . . .	83
4.3	MC Heuristics. Unique problems per captcha scheme. . . . .	87

4.4	MC Heuristics. Mean Likert scale responses for overall performance ratings. . . . .	88
4.5	MC Heuristics. Nielsen & Molich's Heuristics. . . . .	90
4.6	Unique problems from the user study and the MC HE. . . . .	95
5.1	Screenshots of the five pre-existing evaluated captcha schemes.	101
5.2	P1. Gesture reCaptcha. . . . .	103
5.3	P1. Gesture reCaptcha's message and activity flow. . . . .	103
5.4	P2. Gesture Emerging. . . . .	106
5.5	P2. Gesture Emerging's message and activity flow. . . . .	106
5.6	P3. Asirra Slide. . . . .	108
5.7	P3. Asirra Slide's message and activity flow. . . . .	108
5.8	P4. reCaptcha Buttons. . . . .	110
5.9	P4. reCaptcha Buttons' message and activity flow. . . . .	111
5.10	Percentages of Successes, Errors and Skips per scheme. . . . .	121
5.11	Mean scores for the Likert-scale responses. . . . .	124
5.12	Prototypes. Wrong canvas rendering. . . . .	128

# Chapter 1

## Introduction

Completely Automated Public Turing tests to tell Computers and Humans Apart (captcha<sup>1</sup>) are challenge-response tests used on the web to distinguish human users from automated bots [144]. Captchas typically display distorted characters which users must correctly identify and type in order to proceed with a web-based task intended only for humans, such as creating an account, making an internet purchase, or posting to a forum.

Mobile devices such as smartphones and tablets have become a primary means of accessing online resources for many users. Mobile usage has grown substantially in the last few years and the trend continues [18, 106]. In 2014, over 58% of the US population owned a smartphone [18] and saturation is even higher in other places such as Hong Kong, the UK, and Australia [105]. In fact, 84% of mobile users in the US had used their devices for shopping within the first quarter of 2013 [107].

Facilitating mobile web interactions is clearly an important focus area, as is enforcing web security. In particular, we are looking at one aspect of secure web interactions: the usability of captchas on smartphones. Captchas have become sufficiently hard for users to solve that some web sites are actively looking at alternatives. Finding an alternative captcha that addresses the usability issues while maintaining security has potential uses for any mobile-enabled website concerned with spam and bots.

### 1.1 Motivation

We use web services for banking, bidding to purchase articles, purchasing goods and services, interacting with government services, accessing health services, taking courses, sharing data and socializing, to name a few. The importance of web services and having an Internet presence is by now well established.

---

<sup>1</sup>To provide better legibility we write the acronym CAPTCHA in lowercase, *i.e.*, captcha.

These web services require defence from a myriad of possible attacks. The protection mechanisms take many forms: input validation, protection against denial of service, defence against automatic form completion by web bots<sup>2</sup> (or bots for short). Typically the latter defences are deployed as challenge-response tests which present a sequence of alphanumeric characters in a distorted image and request the user to type the sequence into an input field. This security mechanism is based on ideas related to an Automated Turing Test<sup>3</sup> [100], and is commonly known as a captcha. Major commercial companies utilize captchas against malicious Internet bots. For example, Google, Yahoo!, Microsoft, and Facebook have deployed captchas to reduce the harvesting of free accounts by spammers and fraudsters [52, 55, 93, 152].

In addition, the widespread adoption of smartphones and tablets has led web developers, companies, and content providers to adapt their services to new interaction paradigms, new environments, and reduced display real estate in order to provide mobile-enabled websites and services.

Captchas are often challenging for users to complete on regular computer interfaces [24] and even though virtual mobile keyboards have improved and several types exist, our empirical data (Chapter 3) shows that current keyboards continue to be a cause of errors and frustrations for users. Swapping between numeric and alphabetic keyboards is a source of mistakes while solving challenges, for example. The small screen makes it difficult to distinguish the distorted characters so users must zoom and pan, the virtual keyboard obstructs the challenge, and auto-correct modifies the typed characters. Existing captchas do not properly fit mobile devices and may lead users to abandon tasks [5]. Captchas are not only deployed on web sites; applications such as Snapchat are including captchas as part of their interface [134].

---

<sup>2</sup>A web robot “can be defined as a computer program that executes a sequence of operations repeatedly, carrying out tasks for other programs or users without the need of human interaction” [11].

<sup>3</sup>In the original Turing Test, a human judge was allowed to ask a series of questions to two players. One player was a computer and the other a human. The computer pretended to be a human and the judge had to distinguish between them [144].

In this thesis, we consider smartphones those devices which provide computational power to run applications and are equipped with a SIM card able to make calls using a cellular network. Smartphones are able to run an advanced mobile operating system which includes features of a personal computer. Within the scope of this thesis, we use the term mobile device interchangeably with term smartphone.

The area of captcha research has seen continuous interest from a variety of fields such as security, machine learning, computer vision, and speech recognition who seek to either break existing schemes through automated means or present new types of challenges that are more resistant to attack. Captchas are an excellent example of usable security. They have to strike the right balance between usability and security. Captchas described as secure are too complex for humans to solve; or vice-versa, simplistic challenges are easily defeated by automated programs.

## 1.2 Thesis Statement

The overall research topic addressed in this thesis is to facilitate the selection of viable new captcha schemes by assessing and improving the usability of captchas on smartphones. In particular, our research revolves around three primary, inter-related questions:

1. How can we effectively assess usability issues of captchas accessed on smartphones?
2. What are the most prevalent usability issues of captchas accessed on smartphones?
3. How can we improve captchas for smartphone usage?

Based on these questions, we define four main objectives for the thesis.

**Objective 1:** Develop a set of evaluation heuristics specific to captchas, with a focus on smartphone interaction.

**Objective 2:** Empirically validate the effectiveness<sup>4</sup> of the proposed set of heuristics.

**Objective 3:** Identify promising interaction methods for captchas using touch screens, then create and evaluate an alternative captcha prototype employing these new interaction methods.

**Objective 4:** Identify underlying design characteristics of successful smartphones captcha schemes and generalize to develop design guidelines for captcha on smartphones.

### 1.3 Thesis Overview and Organization

We summarize the literature on interaction paradigms, input methods, mobile device interaction design, captcha usability concerns, and general captcha security considerations (Chapter 2). Our captcha user studies from Chapter 3 provided a foundation on the usability of captchas for the remainder of the work found in this thesis.

To address *Objectives 1 & 2*, Chapter 4 introduces our set of domain specific heuristics for evaluating captchas on smartphones. The chapter synthesizes captcha properties and surveys methodologies employed to develop domain specific heuristics. We identify the gaps and need for a new set of heuristics. We review the academic literature and identify the most relevant papers in the area of heuristic evaluations. Then, we define a set of heuristics, provide their rationales, and review published heuristic validation methodologies.

In response to *Objective 2*, we conducted three separate evaluations: two heuristic evaluations and a user study to assess the effectiveness of the proposed heuristics. We recruited expert evaluators and conducted the heuristic evaluations for four captcha schemes. The heuristic evaluations are presented in Chapter 4 and the user study in

---

<sup>4</sup>We refer to effectiveness as the evaluators' ability to uncover problems [68].

Chapter 3. Using the results of these evaluations we validated the effectiveness of our proposed heuristics.

For *Objective 3*, we researched, designed, and implemented three novel input mechanisms for captchas, and conducted an empirical evaluation to determine whether our designs were effective. Chapter 5 contributes to *Objective 3*.

Chapter 6 offers a series of design strategies and recommendations for adapting captcha schemes suitable for smartphone usage. This chapter contributes to meeting *Objective 4*. In addition, Chapter 6 describes further research directions beyond the scope of this thesis. Concluding remarks are made in this chapter.

The scope of this thesis includes the usability aspects of captcha schemes rather than their security. Nonetheless, security is central to any scheme. The approach to “breaking” a scheme varies according to the captcha class. For example, assessing the security of character recognition captchas requires a different set of approaches than an image recognition captcha. No usability technique can evaluate the *security* of a captcha scheme. Even if security is not our focus, there is an implicit assumption that to be viable, captchas must meet at least some minimal threshold of security. For this reason, we rely on schemes currently available for deployment or which have been assessed for security elsewhere, modifying the user interaction but maintaining the underlying challenge intact.

## 1.4 Main Contributions

This research contributes original ideas and knowledge to the fields of web services security, touch screens and gestures, expert evaluations, and usable security. The main contributions of this research are enumerated below.

1. We conducted an empirical study of captchas on desktops [150, 151]. This evaluation helped us develop a set of seven domain-specific heuristics for evaluating captchas on smartphones. The effectiveness of these heuristics was tested on four existing captcha schemes and iterated with experts in HCI and security. In addition, we conducted a heuristic evaluation utilizing Nielsen’s general heuristics [104, 109]. We also completed a complementary user study of these same captcha schemes on smartphones with end-users [120].

2. We designed and implemented four prototypes of novel captcha input mechanisms [121]
3. We extracted and generalized the main design characteristics of our prototype captcha proposals that led to design strategies and recommendations for adapting captcha schemes more suitable for smartphone usage [122].

## 1.5 Related Publications

This thesis includes research that has been peer-reviewed and published in academic venues. I am primary author on the following full-length paper publications except for the last two publications where I was the primary researcher responsible for the user study and usability analysis sections of the paper. Significant portions of the text in this thesis is taken from these papers.

1. **Gerardo Reynaga** and Sonia Chiasson. *The Usability of CAPTCHAs on Smartphones*. International Conference on Security and Cryptography, (SECRYPT), pages 427-434, August 2013. SCITEPRESS.
2. **Gerardo Reynaga** and Sonia Chiasson and Paul C. van Oorschot. *Exploring the Usability of CAPTCHAs on Smartphones: Comparisons and Recommendations*. Workshop on Usable Security (USEC 2015), 10 pages, February 2015.  
Co-located with Internet Society's Network and Distributed System Security Symposium (NDSS).
3. **Gerardo Reynaga** and Sonia Chiasson and Paul C. van Oorschot. *Heuristics for the Evaluation of Captchas on Smartphones*. British HCI Conference, 10 pages, July 2015. ACM.



4. Yi Xu and **Gerardo Reynaga** and Sonia Chiasson and Jan-Michael Frahm and Fabian Monrose and Paul C. van Oorschot. *Security and Usability Challenges of Moving-Object CAPTCHAs: Decoding Codewords in Motion*. USENIX Security Symposium, 16 pages, August 2012. USENIX Association.
  
5. Yi Xu and **Gerardo Reynaga** and Sonia Chiasson and Jan-Michael Frahm and Fabian Monrose and Paul C. van Oorschot. *Security Analysis and Related Usability of Motion-based CAPTCHAs: Decoding Codewords in Motion*. IEEE Transactions on Dependable and Secure Computing (TDSC), 11(5):480-493, September 2014.

## Chapter 2

### Background

#### 2.1 Introduction

A captcha is a program that generates and grades challenges that are human solvable and should be unsolvable by current computer programs [144, 158]. A *challenge* refers to a single captcha puzzle to be solved by the user. To date, a large number of text, audio, image, motion and cognitive captchas have been suggested. Nevertheless, most of the deployed implementations are distorted alphanumeric characters presented as a challenge to the user on an image. Moreover, many of these schemes have succumbed to a variety of attacks [22, 25, 54, 96, 155, 157, 161], later described in Section 2.3.

In 1996, Naor [100] proposed for the first time a Turing Test to verify whether the service requester over the web was human. His proposal is based on the adaptation of secure protocols. To identify the requester, the service sends a “human-in-the-loop” challenge along with the web form. The challenge is in the form of a question. Before the service request is processed, the correctness of the answer is verified. Naor proposed the following suggestions as sources for Turing Tests: recognizing gender out of multiple images, identifying facial expression (sad vs. happy), clicking body parts on an image, deciding nudity, labelling drawings from a list of choices, reading handwriting, filling words or arranging the permutation of sentences, and disambiguating sentences (determining what “it” means in a sentence). Naor’s main motivation was combating junk mail.

In the literature, the terms HIP (Human Interactive/Interaction Proof), Reverse Turing Tests (RTTs) and captcha are often used interchangeably. Chew and Hall argue that a captcha is a specific type of HIP [34, 62] because a captcha performs an automatic evaluation. They note that a HIP is used as a general term for tests involving a human-in-the-loop. On the other hand, von Ahn *et al.* [144] make a distinction between the terms captcha and RTT citing that the use of RTT can be

a misleading term since it “has been used to refer to a form of the Turing Test in which both players pretend to be a computer” [144]. In this thesis we use the term captcha as suggested by von Ahn *et al.*, and Yan and El Ahmad [144, 158]: a captcha is “a program that generates and grades tests that are human solvable, but beyond the capabilities of current computer programs”.

We review and present a captcha taxonomy in Section 2.2. Security analysis and types of attacks on captchas are described in Section 2.3 and Section 2.4, respectively. Section 2.5 lists applications for which captchas are used. An overview of alternative proposals to captchas is presented in Section 2.6. An overview of captchas and smartphones is presented in Section 2.7. To provide the needed context for future chapters, we present usability background information in Section 2.8. The chapter’s summary is given in Section 2.9.

## 2.2 Captcha Taxonomy

We find many alternative taxonomies of captchas in the literature [10, 64, 158]. In this thesis we classify captchas as follows. Character-recognition (CR) captchas involve still images of distorted characters (Figure 2.1 (a)). Audio captchas (AUD) use words or spoken characters as the challenge (Figure 2.1 (b)). Image-recognition captchas (IR) involve classification or recognition of images or objects other than characters (Figure 2.1 (c)). Cognitive-based captchas (COG) include image based, text based, audio based puzzles, questions, and other challenges related to the semantics of images or language constructs. For example, we include in this category content-based video-labelling of YouTube videos [74], or Yamamoto *et al.*’s [153] proposal to use the human ability to recognize strangeness in sentences (Strangeness in Sentences captcha, SS-CAPTCHA). For both CR and IR, we further identify dynamic subclasses. That is, the CR-dynamic class encompasses dynamic movement of text as the challenge and the IR-dynamic class uses moving objects as the challenge. These two can be grouped as a cross-class category: moving-image object recognition captchas (MIOR) [150] involving objects in motion through animations, emergent-image schemes, and video [38, 39, 40, 80, 94, 110, 129]. Figure 2.2 depicts the described taxonomy.



(a) Baidu search engine [7]      (b) Microsoft's audio [93]      (c) ARTiFACIAL[124]

Figure 2.1: Examples of Captchas

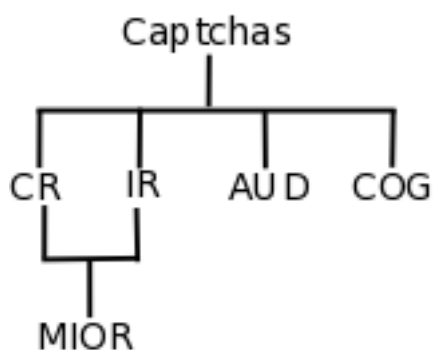


Figure 2.2: Captcha taxonomy

### 2.2.1 Character-Recognition Captchas

The premise with *character-recognition (CR) captchas* is that some text pattern recognition tasks are extremely difficult for automated programs but easy for humans. CR schemes are the most commonly deployed captcha schemes. Table 2.1 describes possible design features for this captcha class. The column *Content* refers to the text displayed as challenge, and this content is conveyed with certain *Appearance*.

Figure 2.3(b) depicts Yahoo's older captcha [97]. Using this scheme as example we describe the features from Table 2.1. This captcha had a random **length**<sup>1</sup> using **alphabetic** characters and **words** as challenge, the **language** was English. The captcha *appearance* was **black text** and **coloured-textured background**, there was **distortion** applied to the text, and finally the **font size** and **font family** appear to be constant for all characters. Many schemes rely on the background as their main

<sup>1</sup>The typewriter font indicates a feature from the table.

security feature, nevertheless there is strong evidence that indicates it often does not provide the anticipated security [25, 155]. Moreover, recent schemes (*e.g.*, reCaptcha, Yahoo!, eBay, Google) do not have background distortions or these are reduced to a minimum. Figure 2.3 depicts additional examples of CR captcha schemes.

Content		Appearance	
Length	constant, random length	Colour	text, background
Char set	[a-zA-Z0-9], !,@,#,\$,%,&,*+, non-Roman	Background	noise: textured backgrounds, clutter: arcs, lines, geometric shapes, no noise, no clutter
Challenge	words, pseudo-random, random	Distortion	translation, rotation, scaling, warping, overlap, elasticity, squeeze, blurring
Language	natural language	Presentation	font family, font size, resulting image size

Table 2.1: Character Recognition-based scheme features (extended from [33, 158]).

### 2.2.2 Audio Captchas

*Audio (AUD) captchas* are typically complementary to CR captchas. These are usually offered as an alternative for blind, visually impaired, or disabled people. However, in their survey Bursztein *et al.* [24] highlight the poor human success rate of these schemes. They evaluated eight audio captchas and found that all evaluators agreed on only 31.2% of answers (out of 1,100 people<sup>2</sup>).

Table 2.2 groups design features and attributes of AUD schemes. The column *Content* refers to the audio played as challenge on the captcha. The column *Sound (distractor)* includes attributes employed to add difficulty for a bot to solve the challenge. Using Solve Media as an example (Figure 2.3(c)), we describe the features of Table 2.2. Solve Media’s audio challenge is 14 letters in length. The spoken letters

<sup>2</sup>This is our best interpretation of the result; the original paper is unclear on this point.

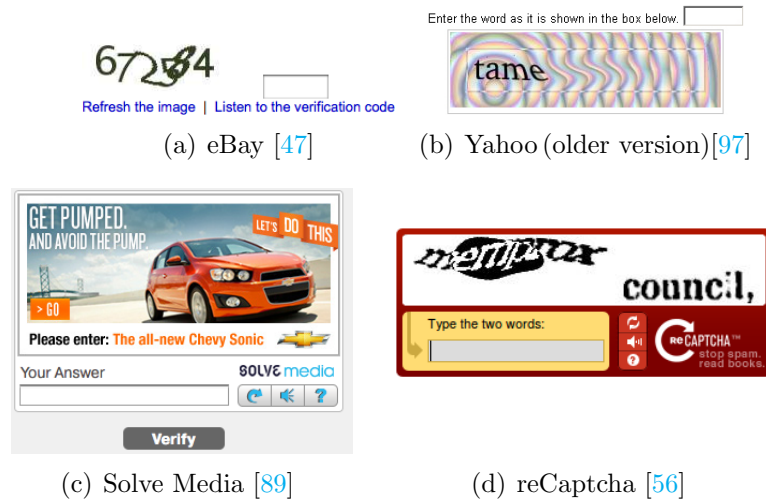


Figure 2.3: Examples of CR captcha schemes

use English language, combining **female** and **male** voices. The preamble “Please enter the following” is immediately followed by the challenge. The **time** between each letter is reduced to the time it takes to pronounce the letter. The background **noise** includes a low volume white background noise. In the case of reCaptcha [56], the challenge consists of a set of ten numbers with no background distractor. The voice is **synthetic** and **distortion** is applied to each unit. The challenge is repeated twice.

### 2.2.3 Image-Recognition Captchas

We classify as *image-recognition (IR) captchas* those schemes that require the user to identify an image on the screen. Compared to CR schemes, in which the answer is typically typed, IR schemes have a broader range of response formats (Table 2.3). This type of scheme benefits of the limitation of software to recognize concepts expressed by images [10]. Tasks involving image-based captchas vary from proposal to proposal; some require the user to **click on parts of the image** [8], find characteristics that **set apart two** sets of images [15, 144], perform actions with the mouse [43], or identify a subset from a set of images [15, 51]. Scene tagging is a proposal by Matthews and Zou [87]; it tests the ability to recognize a relationship between multiple objects in an image (Figure 2.4 (b)). SQUIGL-PIX [15] is one of various proposals

Content		Sound (distractor)	
Length	number of units	Recording	seconds
Unit	words, letters, numbers, unique sound,	Background	noise, echo, conversations, decoy words, music
Language	dependent of localization	Language	independent of localization
Time	required for each unit	Voice	female, male, children, synthetic, mix
Preamble	instructions, beep	Distortion	applied to unit, background noise, entire duration, between units, tone change

Table 2.2: Audio-based scheme features

from Luis von Ahn and his team at The CAPTCHA project site [15] (Figure 2.4 (c)). In the SQUIGL-PIX proposal, the user has to identify an object and trace it with the mouse.

Asirra<sup>3</sup> a well-known example of this class, was proposed [51, 92] and later broken [54]. The Asirra captcha is depicted in Figure 2.4 (a) and involved **object classification** (*i.e.*, distinguishing cats from other animals). In this scheme, dogs (**non-class images**) are used as **distractors** and the **number** of cat images to identify varied between 4 to 7 cats.

#### 2.2.4 Moving-Image Object Recognition Captchas

Common *Moving-image object recognition (MIOR) captcha* schemes include little or no distortion as part of the appearance since they rely on the difficulty of detecting movement.

---

<sup>3</sup>The captcha service was permanently closed in October 2014.

Content		Appearance	
Length	number of objects to classify/select	Presentation	multiple areas: 1 or 2 challenge areas and a solution area,
Artifact	objects, animals, people characters, non-existing concepts, single image		single challenge and a solution area
Challenge	classify objects, rotate image, mouse area selection, draw, disambiguation, drag and drop, clicking on image or objects, spatial location, object quantity	Distractor	non-class objects or images image rotation, image overlap, image distortion, lines, non-related objects, object rotation, image background

Table 2.3: IR-based scheme features

For instance, NuCaptcha [110] (see Figure 2.5(a)) is a commercial implementation of a MIOR; it was recently broken [150]. NuCaptcha offers several variants. All include movement as a distractor. In some variants a sentence is mingled with the artifact. In NuCaptcha, the artifacts are characters in the form of a sentence that act as an additional distractor.

Table 2.4 depicts the common attributes of MIOR captcha schemes. Other examples of MIOR captchas are Animated captcha and “Are you a human?”. Animated

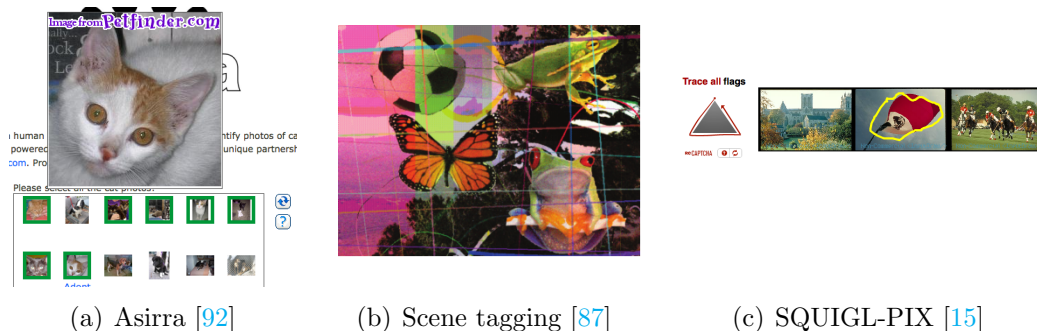


Figure 2.4: Examples of IR captcha schemes



captcha [142] is an experimental scheme. The challenge typically consists of six alphanumeric characters arranged in a patterned cylinder that rotates in the centre of the captcha screen. The similarly patterned background portrays what could be the floor (or base) where the cylinder sits; this floor swivels up and down (Figure 2.5 (b)). “Are you a human?” [2] is another commercial scheme in which the user is required to drag and drop moving objects to a predetermined area of the challenge (Figure 2.5 (c)). The distractors are **challenge movement** and **non-related artifacts** included in the captcha.

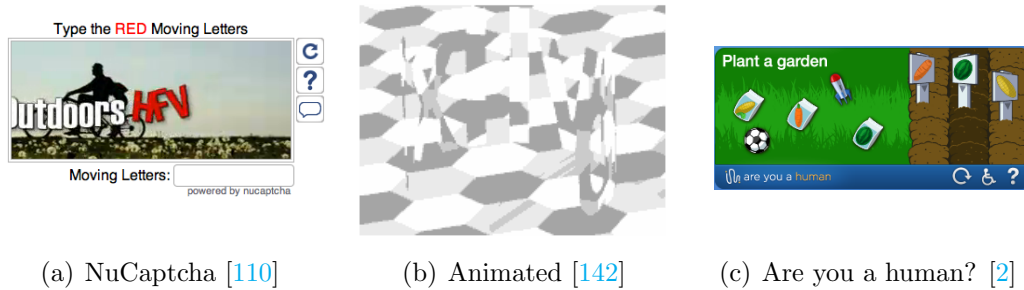


Figure 2.5: Examples of MIOR captcha schemes




Content		Appearance	
Length	number of objects or characters	Presentation	little or no distortion, animation
Artifact	objects, images, words, characters	Distractors	relies mainly on movement: background movement, challenge movement, non-related artifacts included
Challenge	type characters, move objects, classify objects		

Table 2.4: MIOR-based scheme features

### 2.2.5 Cognitive Captchas

More than classification of objects and images, *Cognitive (COG) captcha* schemes base their challenges on the semantics of the images or language constructs.

Table 2.5 describes common features found in this class. For example, video labelling [74] of YouTube videos (Figure 2.6 (b)) does not employ distortion but rather leverages the image semantics of pre-tagged videos. Objects or images are pre-tagged by users uploading videos and the task of solvers is to describe the contents of the video. If a predetermined number of describing words match the tags then the test is passed. In general, this class of captcha does not utilize the common distortions or background noises other schemes employ, but rather relies on the ability to make semantic interpretations of the challenges. These include solving puzzles by recognizing certain objects, answering an arithmetic question [128] (Figure 2.6 (a)), solving graphical puzzles [53] (Figure 2.6 (c)), or answering quizzes [35].

There are 5 , 3  and 6  on a table. How many fruits are there on the table in total?

(a) Question-based [128]



Type 3 words that best describe this video:

Submit

(b) Video tagging [74]



(c) Jigsaw puzzle [53]

Figure 2.6: Examples of COG captcha schemes.

<b>Content</b>		<b>Appearance</b>	
Length	number of objects, arithmetic operations, words	Presentation	little or no distortion
Artifact	objects, labels, characters, images	Distractors	lexical semantics, image semantics, language semantics
Challenge	type labels, rearrange images, perform arithmetic operations		

Table 2.5: COG-based scheme features

### 2.2.6 Summary

In this background section, we described a captcha taxonomy (Figure 2.2). We explained the attributes of each of the categories as well showed examples of these classes.

**CR** Character-recognition captchas involve still images of distorted characters.

**CR dynamic** captchas are a CR subcategory. The CR-dynamic class encompasses dynamic movement of text as the challenge.

**IR** Image-recognition captchas involve classification or recognition of images or objects other than characters.

**IR dynamic** captchas are a IR subcategory. The IR-dynamic class uses moving objects as the challenge.

**AUD** Audio captchas are based on words or spoken characters as the challenge.

**COG** Cognitive-based captchas include puzzles, questions, and other challenges related to the semantics of images or language constructs.

**MIOR** Moving image object recognition captchas rely on the difficulty of detecting movement.

### 2.3 Security Analysis

Captchas are security mechanisms designed to protect web services against bots and unwanted automated access. Therefore, the security of a captcha scheme relies on the cumulative effect of the design, implementation and deployment choices. Factors such as the underlying AI problem on which some schemes are based, the implementation and deployment of the scheme, the challenge generation (*i.e.*, the size of the database employed and randomness), and evaluation of tests affect the strength of a given scheme. Often the *attack threshold* is the only metric used to evaluate the strength of a given scheme. We also consider the *challenge space* as part of the implementation and deployment considerations of a scheme. Below we expand more on these two ideas.

*Attack Threshold:* Assessing the strength of any captcha scheme is quite complex and often an approximation. The most straightforward evaluation metric is the proportion of times a bot can successfully solve challenges. There is presently no consensus on the required security threshold in terms of bounding the success rate for adversaries. A measurement that is often referenced in the literature is the robustness threshold from Chellapilla *et al.* [32]: “automatic scripts should not be more successful than 1 in 10,000 (0.01%)”. Other authors consider this threshold “a very challenging number” and set less stringent thresholds. For example, Zhu *et al.* [161] propose that the bot success rate should not exceed 0.6%. Bursztein *et al.* [25] suggest a more realistic security goal of 1% before reaching the threshold for the scheme to be considered insecure. Others argue that “if it is at least as expensive for an attacker to break the challenge by machine than it would be to pay a human to take the captcha, the test can be considered secure” [64].

However, one of the most challenging phases of designing new captcha schemes is to ensure that the scheme withstands state-of-the-art captcha attacks. Even if the captcha designers develop tests with an acceptable threshold, there might be future automated programs that will break the proposal.

*Challenge Space:* In addition to scrutinizing automated programs for the security analysis, we consider that the size of the captcha challenge space is essential for the captcha security analysis. We define the *captcha challenge space* as the “alphabet”

employed to generate challenges. The captcha challenge space is instrumental in the security analysis and the adoption of captcha schemes. To illustrate, CR schemes typically use words or a combination of alphanumeric characters (a-z, A-Z, 0-9) as their alphabet. For example, consider the set of all possible 6-character alphabetic challenges. Excluding symbols and numbers, there are 26 characters in the English language giving a captcha challenge space a cardinality of  $26^6$ . However this challenge space is sometimes reduced to a subset of the original challenge space, resulting in an *effective challenge space*. For example, some schemes reduce the challenge space by eliminating confusing characters (*e.g.*, 0/0, 6/G/b, 5/S/s, 2/Z/z, 1/l, vv/w, cl/d, nn/m, rn/m) or mapping tuples of confusing characters to a same answer. Moreover, Bustztein *et al.* argue that reducing the set of character does not considerably impact the security of CR schemes [22]<sup>4</sup>. Thus the effective challenge space is determined by the number of functionally unique responses. The effective number of responses is calculable but the number of challenges is harder to calculate since some of distortions are randomized for some captcha schemes.

The challenge space of some captcha schemes (*e.g.*, IR, MIOR or COG), other than CR, require a source database for the generation of challenges. Hence, populating and testing the database have to be considered as part of the overall security assessment of a captcha scheme. An approach to populate the source database is proposed by Chew and Tygar [35]. The selection and filtering of the database follow these phases:

- a) *Training*, the set is labeled or rated.
- b) *Testing*, a subset is used for human testing to gauge estimates and compare to actual ratings.

Ideally the third phase should also be included:

- c) *Reseeding*, new elements are added to the database as users pass and, in some cases, label challenges.

---

<sup>4</sup>Bursztein *et al.* do not specify the limit to how many characters can be eliminated without impacting the security of the scheme.

## 2.4 Security Threats

Captcha designers have to consider known and existing threats. In the following sections we describe common attacks and threats on captchas. We categorize the security threats in four large categories: *human-in-the-loop*, *replay*, *naïve*, and *automated* attacks. Since the most widely deployed captcha class is character-recognition (CR) schemes, it is also the one having received the most extensive security analysis. As we discuss below, many of the known attacks and threats are tailored for CR captchas.

### 2.4.1 Human-in-the-loop attacks

A captcha aims to differentiate machines and humans. Thus, they become vulnerable to attacks from humans themselves because the captcha cannot distinguish legitimate human users from malicious or misguided human users. *Human-in-the-loop* attacks include:

- Captcha redirection, relay or pornography attack [15, 141, 153]: Attacks can shift the solving of captchas to other sites (*e.g.*, pornographic sites) where unsuspecting users respond to the challenges with the promise of free services or free content. For a relay attack to be profitable to the attacker, it should be simple to relay, it should not require special software, and it should have low overhead [98].
- Laundry or outsourcing attack [6, 43, 98]: This is also referred to as crowd sourcing of captcha solutions. This attack involves services like Mechanical Turk or sweatshops in which humans are paid to solve challenges. The motivations of the attacker are likely to be simplicity, low economical cost and practicality. Captcha-solving services offer individuals to pay for solving captchas. Solving services and offers to recruit individuals can be easily found. For example, searching in G+<sup>5</sup> for the tag #captcha<sup>6</sup> results in both captcha-solving services

---

<sup>5</sup><https://plus.google.com>

<sup>6</sup><https://plus.google.com/u/0/explore/captcha>

and paid offers to recruit individuals for solving services. Often the human-solving alternative is less costly than developing an automated tool to solve captchas [98]. For example, a site called `2captcha.com` offers to pay \$0.5 per 1,000 solved challenges[3].

- Prompted solver<sup>7</sup> [154]: In the context of MMORPGs<sup>8</sup>, it is possible for a bot to differentiate a captcha challenge from other game events. A human could be alerted by the bot to answer the challenge. The paper’s author suggests that this could become an attractive income alternative for some people by “attending to their bots occasionally”.

### 2.4.2 Automated attacks

From the attacker’s point of view, developing a tool that would generate *automatic* correct answers to challenges is an effective form of *attack*<sup>9</sup>. Researchers have developed a number of automated tools to test and validate the security of captchas schemes. Typical automated tools to break CR schemes follow these stages [25, 50, 64]:

1. Pre-process: remove colour (transform to black-and-white) and background (noise, arcs, lines, clutter)
2. Segment or isolate: cutout and separate captcha letters. The most common segmentation technique is Colour Filling Segmentation (CFS) [25, 50]. Bursztein *et al.* [25] add an intermediate stage: post-segmentation in which segments are processed individually, and normalized, to make the recognition easier.
3. Recognize or identify: teach the classifier what each letter looks like, in training mode. In testing mode, the classifier predicts each character.

---

<sup>7</sup>Referred to as “Housewife attack” by the author.

<sup>8</sup>Massive Multiplayer Online Role-Playing Games

<sup>9</sup>As long as it is profitable for the attacker. That is, developing the tool is cheaper than paying a human to solve challenges

4. Post-process: improve recognition. For example, if it is known that the captcha scheme uses actual words, spell checking or dictionaries are used to make an educated guess of the string.

*Automated* attacks include:

- Optical Character Recognition (OCR) attack [34, 156]: This type of attack is often the first method attempted by attackers. Many captcha implementations are in-house applications that are poorly designed and fall prey to simple OCR techniques.
- Pixel count attack [155]: Exploiting several design flaws, Yan and El Ahmad [155] were able to break several CR captchas having a distinct pixel count per letter. Thus, they used pixel counts to identify letters.
- Segmentation attacks [50, 155, 157]: Many schemes rely on the principle of segmentation-resistance for security robustness. Segmentation-resistance features include the use of random arcs, connected random lines or crowding characters together intending to prevent separation of characters. El Ahmad *et al.* [50] propose various segmentation attacks against CR schemes, showing that segmentation-resistance features.
- Classification (machine learning) attacks [23, 33, 54, 138, 161]: Recent approaches to breaking captchas utilize machine learning classifiers to analyze, learn and recognize challenges. Machine learning techniques have been used for breaking audio, image and CR captchas.
- Motion trajectory attack [150]: Xu *et al.* [150] show that simple MIOR (*e.g.*, Nu-Captcha [110]) schemes can be broken. Trajectories, machine vision and learning algorithms are used to decode moving-character challenges (MIOR). The authors first perform object tracking techniques, then the foreground is extracted. Afterwards, foreground segmentation is done and finally character classification is performed using a neural network classifier.



- Automated attack on a game captcha [95]: Mohamed *et al.* [95] present an analysis and an automated attack on a particular game-captcha, similar to “Are you human?” [2]. The attack is based on image processing attacks and unsupervised learning. The attack consists of: a) learning and identifying the background and foreground objects, b) identify the target area and the target area center(s), c) identifying and learning the correct answer objects, d) building a dictionary of answer objects and corresponding targets, e) continuously learning from new challenges.
- Image distortion estimation attack [99]: Moy *et al.* [99] introduce distortion estimation techniques. Although used to break two older captcha schemes (EZ-Gimpy and Gimpy-r), the approach can be optimized and generalized as a tool for object recognition.
- Known lexicon attacks or dictionary attacks [8, 155]: Several schemes use dictionary words as their challenges. If partial results are returned by the basic algorithm (segmentation and pixel count), these results are used as a string pattern to identify candidate words in the dictionary that match the pattern.
- Speech recognition [90]: Meutzer *et al.* [90] use speech recognition to defeat a commonly deployed captcha. Their results show higher success rate than previously reported classification methods. The authors suggest leveraging state-of-the-art speech recognition techniques for evaluating the security of AUD captchas.
- Mori and Malik attack [96]: The authors use sophisticated object-recognition algorithms to recognize objects (*e.g.*, letters) in clutter.

### 2.4.3 Replay attack

In a replay attack, the attacker bypasses the challenge by reusing a known (*e.g.*, previously seen or pre-recorded) challenge session ID or solution. Challenges should thus be unique; if challenges are reused, the vulnerability of a replay attack arises.

- Captcha smuggling attack [48]: Egle *et al.* [48] introduce this attack. These attacks use malware on a compromised host that intercepts the http request to a web server. The attacker injects captcha challenges into the web browsing session of unsuspecting users and delays the execution of the user’s original request until the challenged is solved.

#### 2.4.4 Naïve attacks

We refer to naïve attacks as the random guessing or brute force attacks.

- Naïve attack (no-effort) [124]: Rui describes these attacks as ones that can solve a captcha “without solving the hard AI problem” [124]. The simplest example is an attack against the Bongo captcha [15] since the answer has only two possible values: left or right. Thus an attacker always achieves 50% success by random guess.

#### 2.4.5 Summary of Attacks on Captchas

Table 2.4.5 organizes the attacks to which each class of captcha is vulnerable. Each row of the Table identifies the specific attacks for a captcha class, organized by type of attack. As previously described in Sections 2.4.1 to 2.4.4 the Human-in-the-loop and Automated attack categories encompass several different attacks so specific attacks to which the captcha class is vulnerable are enumerated in those columns. The Replay and Naïve attack columns each represent a single type of attack, therefore a checkmark or X is used to indicate whether the captcha class is vulnerable to this attack. A checkmark (✓) indicates that the class is vulnerable to the attack and an ✗ indicates that the class is resilient to the attack.

### 2.5 Captcha Applications

There are a number of applications for captchas in the literature and in actual deployment. In the following sections, we briefly describe some of these areas where captchas are useful. We group the applications in two large categories: *defence mechanisms* and *collateral purposes*. The main purpose of a captcha is as defence mechanism.

Class	Human-in-the-loop attacks	Automated attacks	Replay attacks	Naive attacks
CR	Captcha redirection, Outsourcing	OCR, Pixel count, Captcha smuggling, Segmentation, Known lexicon, Classification methods	✓	✓
AUD	Captcha redirection, Outsourcing	Speech recognition, Classification methods	✓	✓
MIOR	Captcha redirection Outsourcing, Offers some level of resistance	Motion trajectory, Game attack	✗	✓
IR	Captcha redirection, Outsourcing	Classification methods, Object recognition,	✓	✓
COG	Captcha redirection, Outsourcing	Classification methods, OCR, Speech recognition	✓	✓

Table 2.6: Different types of attacks on the captcha classes. Vulnerable: ✓, not vulnerable: ✗.

However, this security comes at expense of time for human users to solve them. Some schemes incorporate a secondary benefit to solving captcha challenges, and making use of that time towards a positive collateral purpose.

### 2.5.1 Defence mechanisms

- *Prevent abuse* by obstructing spammers from *registering for free e-mail accounts*. When Naor [100] first proposed the Turing Test to verify whether the service requester over the web was human, his main motivation was to combat e-mail spam. This remains one of the main reasons for captcha deployments.
- *Deter automated password guessing* (brute-force and dictionary-based attacks) [42, 116, 141]. If there is no mechanism to prevent an automated password guessing attack, an automated computer program could abuse a web login service by submitting large numbers of username/password tuples sequentially. After a predefined number of attempts a captcha is presented before allowing more login attempts.
- *Reduce anti-spam comments on blogs* and discussion boards [10, 98]. Flooding web blogs, forums or wikis with unsolicited messages can influence result placement on major search engines, can divert traffic to malware-distributing web sites, or can be used as a form of harassment or vandalism. Captchas help reduce the risk by slowing automated posting.
- *Prevent fraud against web auction systems* [10, 23]. Bursztein and Bethard [23] mention that eBay uses captchas to avoid a flood of scams and the posting of illegal items automatically submitted by bots.
- *Verify e-banking transactions* [78]. Once the customer has initiated an e-banking session, they could be subject to a Man-in-the-Middle (MitM) attack. Many financial institutions have deployed captchas for transaction verification. This transaction verification consists of embedding part of the transaction data in a challenge-response verification. Hence, the main goal of this type of captchas is to prevent bots from manipulating transactions.

- *Stop spyware attacks* [147]. Wang *et al.* propose an authentication scheme which combines graphical passwords with character-recognition (CR) captchas to provide password protection against spyware attacks. To be authenticated, users select their pass-images and solve a CR captcha. Each image has a corresponding captcha, adding resistance against replay attacks since the captcha has to be solved before submitting the password.
- *Defend against click-fraud* [36]. Chow *et al.* propose a clickable captcha scheme that could be used against click-fraud. Some models of online advertisement use cost-per-click as means of revenue to the advertising company. Each click costs the advertiser, thus automated clicks are never desired and are possibly fraudulent. A captcha deployment could help reducing the number of automated clicks.
- *Reduce denial of service (DoS) attacks* [32, 43]. DoS attacks are an attempt to make a computer resource unavailable to its intended users. One way of doing so is by generating a large number of automated requests for services or resources which cripple the response capability of the server. By introducing a captcha prior to responding to the service request, administrators prevent the flooding of requests.
- *Prevent anti-bulk tickets purchase* [64, 98, 99]. A bot can buy large number of tickets for an event at a Web site within a few seconds and later resell them at a much higher price. Using captchas as part of the form submission prevents repetitive purchasing without human involvement.
- *Detect phishing attacks* [44]. Dhamija and Tygar's proposal is a variation of traditional captchas. Instead of the server issuing a challenge to the user, the user issues a challenge to the server. This class of HIPs allows a human to distinguish one computer from another. The implementation consists of a trusted password window in which a random image is assigned to each user. The server uses a hash value to generate an abstract image. Both the user's browser and the server can independently compute the same image. The browser presents the user with the image that it expects from the server. This image serves as

the background of the password window and always appears in that window. If the website image matches the image displayed in the password window, the user can easily verify that the information request originates from an expected party.

- *Deter influencing online polls* or bot abuse. An often cited case of poll manipulation is the online poll by slashdot.com in November 1999 asking which was the best graduate school in computer science. Students from both MIT and Carnegie Mellon developed bots to alter the poll [10, 144]. After this occurrence, many sites implemented captchas to deter bots from altering surveys and polls.
- *Reduce farming of game resources* or tools [64, 154]. In massive multiplayer online role-playing games (MMORPGs), collecting in-game resources is highly valued. Some bots are designed to collect these resources that would normally take considerable effort or time to obtain manually. Captchas prevent this automatic harvesting of resources by distinguishing “honest players” from bots.

### 2.5.2 Collateral purposes

- *Facilitate pet adoption* (Asirra) [51, 92]. Asirra<sup>10</sup> (Figure 2.4 (a)) captchas presented the user with images of dogs and cats. The user had to identify and select which of them were cats. The images are taken from a pet adoption web site (petfinder.com). The captcha scheme presents an “Adopt me” link under each image. The idea behind the partnership is to encourage pet adoptions.
- *Aid with book digitization* (reCaptcha) [56]. reCaptcha is currently one of the most deployed captcha schemes. The versions prior to 2015 included a corpus of words used as challenges which came from newspapers, books and old radio shows that were being digitized but were rejected by optical and speech recognition software. There are two words on each challenge, one for which the answer is already known and one that the basic OCR did not recognize. When humans solve a challenge, the unknown word gets scored and eventually counted and

---

<sup>10</sup>The Asirra captcha service was closed permanently in October 2014.

added as digitalized word once sufficient agreement is reached, while the second word acts as the verifiable challenge.

- *Provide advertisement* (NuCaptcha, Solve Media) [89, 110]. NuCaptcha offers several variants of their MIOR scheme (Figure 2.5(a)). One variant consists of including advertisers' slogans in a sentence: "NuCaptcha gives advertisers exclusive real estate where users are the most engaged...Advertisers have the ability to customize their message to the user experience." Solve Media also uses their captcha scheme for advertisements: "Solve Media's proprietary TYPE-IN<sup>TM</sup> Ads guarantee that you connect with your audience. TYPE-IN<sup>TM</sup> Ad campaigns are performance-based, for agencies and advertisers seeking superior engagement, awareness and recall."

## 2.6 Alternative Proposals to Captchas

Alternatives to captchas exist; these may be used as alternatives or as complementary tasks.

Markus Jakobsson [69] proposes a throttling mechanism to control access to web resources. The fundamentals of this proposal is to ensure that "the adversary's cost to create multiple valid requests during a specified time interval will be significant" and that this cost will increase as the number of requests increases. The proposal involves: users, sites, and one or more trusted third parties. The user visits the site; the site wishes to throttle access to resources; the trusted third parties keep track of accesses on a per-user basis and identify potential abuses.

A client-side (browser) solution focusing on GUI components was introduced by Strilechi and Vaida [136]. Their approach includes event handling functions that make use of JavaScript `OnFocus` and `OnChange` events and replaces `OnSubmit` events with custom submit code. Their proposal is based on the assumption that a bot can inject data to any form field but a human is required to shift focus on each of the fields in the form to enter data.

The commercial service Akismet [4] offers a service to filter comments and help combat the problem of spam in blogs. Their website describes the service as "each

time a new comment, trackback, or pingback is added to your site it's submitted to the Akismet web service which runs hundreds of tests on the comment".

However, among the concerns found in these alternatives are deployment issues, the proposal has not been tested for production environments, or the proposal may not scale adequately.

## 2.7 Captchas and Smartphones

Captchas have become sufficiently hard for users to solve that some web sites refrain deploying them and others are actively looking at alternatives. For example, Ticket-Master decided to stop using traditional CR captchas and move to a cognitive-based captcha [12]. For users of smartphones, the problem is compounded by various factors: a reduced screen size can lead to typing mistakes [71], and loss of position [13]. Environmental conditions and device handling orientation also have an impact on the user experience [85]. While captchas have existed for some time, only limited work has been carried out on captchas suitable for mobile devices. Some sites even disable captchas on the mobile version of their web site. Below, we group captcha proposals for mobile devices and usability studies of captchas on mobile devices.

### 2.7.1 Captcha proposals for mobile devices

Chow *et al.* [36] introduce clickable CR captchas (Figure 2.7(a)). The scheme consists of a 3-by-4 grid of CR captchas. The data entry is done using the feature phone's<sup>11</sup> keypad, and each of the grid's elements is mapped to one of the keys. The answer is input by clicking on the key corresponding to the grid element which satisfies the challenge. For example, the user may have to identify a subset of captchas with embedded words, as opposed to random strings. Since the answer consists of selection by clicking, this scheme can be used on mobile devices. Their user tests include desktop computers and a feature phone (Nokia 5200). They conclude that their proposal can be solved faster in smartphones than traditional CR captchas: the

---

<sup>11</sup>We consider a feature phone a mobile phone which allows internet access, simple applications, store and play music but it has limited capabilities compared to those of modern smartphones which typically include touch screens.



average time with a feature phone screen and keypad was 11.1 seconds, compared to 15.9 seconds for a regular Google captcha. They speculate that input using touch sensitive phones may be “somewhat” faster than their feature phone tests.

Gossweiler *et al.* [59] present an IR captcha scheme that, although not designed for mobile devices, can be adapted for mobile usage. Their scheme consists of rotating an image to its upright / natural position with a slider. The mobile version would allow direct image rotation with finger gestures (Figure 2.7 (b)).

Lin *et al.* [81] introduce two captcha schemes for mobile devices. The first is “captcha zoo”, an IR captcha scheme that works similarly to Asirra’s principle of object classification: *i.e.*, discriminating certain target animals from a set of animals (Figure 2.7 (c)). There are only two sets of animals per challenge, *e.g.*, dogs and horses, and the user must click on all animals of one set. The images are 3D models. Their second proposal, a CR scheme, presents a four-character challenge with distorted letters. The only difference between this scheme and the standard CR is that it provides a set of eight buttons with characters that include the answer. The user clicks one of these buttons (Figure 2.7 (d)). In both proposals, the interaction for inputting the solution is by tapping objects on the screen.

A captcha cloud-based service is proposed by Saxena *et al.* [126]. The proposal focuses on the deployment and framework of their distributed architecture rather than the usability of their scheme. This scheme also follows Asirra’s principle of object classification. The captcha consists of a grid with coloured ovals which the user selects by clicking based on challenges such as “Select the two ...XYZ<sup>12</sup>... Colored points”, “Select all the ...XYZ... colored points”, “Select all the colored dots which occur the maximum number of times”. The interaction is done by tapping objects on the screen.

Commercial companies such as NuCaptcha [110], Solve Media [89], and Confident Technologies (Confident captcha) [37] offer captcha challenges for mobile devices. For NuCaptcha and Solve Media, the user has to type the answer using the device’s on-demand keyboard. Confident captcha offers multimodal interaction: the user can tap on images or type the letter assigned to each image to answer the challenge.

---

<sup>12</sup>As denoted by the authors.



(a) Clickable captchas [36] (b) What's UP captcha [59]



(c) Captcha zoo [81] (d) Lin's *et al.* captcha for mobile [81]

Figure 2.7: Example of captcha scheme proposals for mobile devices.

### 2.7.2 Captcha user studies for mobile usage

Wisner *et al.* [148] evaluated captchas on a tablet (iPad). The schemes tested were: Google captcha [15], ESP-PIX [15], and Confident (see previous paragraph). Google captcha was tested for touch only while the last two schemes were tested for voice and touch. Their findings suggest that “with current voice recognition technology, captchas using touch input perform better and are more preferred by users than captchas using voice input” [148].

Shirali-Shahreza *et al.* [130] propose two schemes designed for mobile devices: SeeSay and HearSay. The two proposals use speech as an input mechanism to answer the challenge. The system presents either a CR challenge or an AUD challenge. Similarly to Wismer *et al.* [148], their findings show that speech recognition limitations and errors are a concern for participants.

## 2.8 Usability Analysis

This section briefly introduces the reader to the methodological approaches used in future chapters. We provide an overview of usability evaluation through user studies, heuristic evaluation, and thematic analysis.

### 2.8.1 User Studies

User studies are typically conducted to collect quantitative data on usability concerns, on the performance of software or devices, or on the mental and/or physical demands of software or devices. There exist a collection of tools, methods and techniques to conduct usability evaluations.

*Usability testing* involves measuring users' performance on typical tasks. Performance is measured in terms of effectiveness and efficiency and satisfaction in terms of subjective assessment [125]. Effectiveness is generally recorded by noting the number of errors and kind of errors occurring while performing the task. Efficiency measures the time to complete the task. User satisfaction questionnaires, diaries, and interviews measure users' opinions and perceptions. Usability testing is typically conducted under a controlled environment where both the format of testing and the work environment are controlled. It tends to occur in the later stages of development for ensuring consistency in navigation structure, use of terms, and how the system responds to the user. User studies can be useful for evaluation of specific interface features. Sharp *et al.* suggest the following uses for usability testing [127]: a) help identify opportunities for new technology; b) establish design requirements; c) facilitate the introduction of technology; d) evaluate technology.

Typically, users studies involve the following activities: set evaluation goals and questions; choose the evaluation approach and methods; design, implement and test a prototype or prepare the target software for evaluation; prepare questionnaires and forms (develop demographics questionnaires, develop or choose perception questionnaires, and prepare inform consent forms); develop the user tasks; write down the test procedure and script so all participants are given the same instructions; recruit representative participants; conduct the evaluation with the participants; to collect data through observation, data recording by software, written notes, audio, and/or video; and analyze the data.

Regarding usability evaluation of mobile devices, the Mobile HCI research community is actively looking at new methodologies and techniques that utilize field studies for the evaluation of mobile technologies [46, 61, 72, 73]. *Field studies* also involve user participation, however in this approach the study is done in natural settings. The aim is to understand the user interaction and experience as it occurs in real usage.

### 2.8.2 Heuristic evaluation

A heuristic is an abstraction of a guideline or principle that can provide guidance at early stages of design; or be used to evaluate existing elements of a user interface [127].

According to Nielsen, “heuristic evaluation is the most popular of the usability inspection methods” [104]. This type of evaluation, also known as expert evaluation, is considered a discount usability technique. Heuristic evaluation (HE) was first developed by Nielsen and Molich [104, 109]. The strength of this technique is that it is cheap and easy to carry out. Cheap, since it does not require a user study with a large number of users. Sometimes users are not easily accessible, a user study maybe expensive to run, or the evaluation may take too long. Easy to carry out, because people knowledgeable in both the domain area and interaction design are recruited to conduct the evaluation. Expert evaluation involves having a number of evaluators judge and focus their attention on particular issues.

A natural question that follows is how many experts should be recruited? The number should be large enough to find a variety of problems but not too large that it becomes too expensive. Nielsen [102, 108] highlights the importance of having multiple evaluators. He points out that many usability problems are very easy to find by all evaluators, but there are some problems that are found only by a few evaluators. Nielsen's recommendation of between three and five experts suggests that collectively, evaluators can typically find around 75% of the total usability problems. Jaferian *et al.* [67] point out that more complex systems may require a larger base of evaluators. In their heuristic evaluation of security management tools, between eight and ten evaluators found 75% of the usability problems, collectively. The quality of the heuristic evaluation is highly dependent on various factors, such as the evaluator's experience, knowledge of the application domain, task coverage, and skill.

Similarly to a user study, the heuristic evaluation involves creating a prototype or setting up the target software for evaluation. Currently, there is no standard tool to collect heuristic evaluation data, thus the collection instrument needs to be set in place. This tool sometimes takes the form of electronic documents or online surveys. The heuristic evaluation does not require the researcher to be present while the evaluation is on-going. The evaluation stages can be grouped as follows:

1. Briefing session: the experts are told what to do. By using same instructions for all with a prepared script.
2. Evaluation period: the experts individually go over the system a few times using the heuristics as guide to evaluate, note the usability problems found, and rate their severity.
3. Debriefing session: when possible, experts get together and discuss their findings, reassign severity ratings if needed, and suggest solutions.

### 2.8.3 Thematic Analysis

Thematic Analysis is an analytical approach for qualitative data analysis that aims at identifying, organizing, and offering insight into patterns of meaning (themes) across a data set. This method helps to identify commonalities on a topic and make sense of them [17]. This approach focuses on the content of the data set, and on the supposed meaning that it holds for the author or the intended audience.

The approach focuses mainly on summarizing and obtaining themes. Moreover, the iterative process does not necessarily aggregate data but it is used to repeatedly summarize and group the data to obtain appropriate themes for the analysis at hand, *e.g.*, qualitative analysis of usability problems.

Typically, thematic analysis consists of the following phases:

- Familiarization with the data. This phase involves the researcher immersing herself in the data by parsing and reading the textual data (*e.g.*, transcripts of interviews, responses to qualitative surveys), listening to audio recordings or watching video recordings. Notes are taken during this process to highlight items of potential interest.
- Problem synthesis (*open coding*), or initial coding. Open coding is the process through which conceptual themes and subthemes are extracted from the data. The size of the unit of data to code is determined in this phase: word-by-word, line-by-line, incident-to-incident, etc.
- Consolidate problems (*axial coding*) and *selective coding*. Axial coding relates themes to subthemes, specifies the properties and dimensions of a theme, and reassembles the data previously decomposed during the open coding to give coherence to the emerging analysis. A theme captures something relevant about the data set in relation to the research question or objective. It represents some level of patterned response or meaning within the data set.
- Reviewing potential themes. This involves iteration over the data set where the developing themes and subthemes are reviewed in relation to the coded data

and entire data set. This phase is about questioning the previous phase and coding, reviewing themes and codes.

Although usually referred to as Grounded Theory [30], this approach has been used for many HCI and security studies. Several studies report using Grounded Theory, but adjust the approach to fit the research area. The adjustments often result in a process similar to Thematic Analysis in that it categorizes and makes sense of the data without reaching the last step of forming the theory. Petrie and Power [113] used such an approach to categorize more than 900 usability problems obtained from a heuristic evaluation. Similarly, Jaferian *et al.* [67] categorize 164 guidelines for building IT security management tools. Preston *et al.* [118] specifically employ thematic analysis for their pedestrian navigation data analysis. Their data set includes semi-structured interviews with participants which were audio recorded and later transcribed.

## 2.9 Summary

In this chapter, we presented a captcha taxonomy. We listed the applications for which captchas have been utilized. We discussed security elements and considerations of captcha schemes and classify the attacks on captchas. Alternative solutions to captchas are presented, and we introduced the reader to usability concepts utilized in later chapters.

Our survey of captchas on smartphones revealed that the published work includes only a small number of proposals for smartphones. Their respective authors consider some of these proposals good candidates for mobile environments but not all of them are fully tested on the mobile environment. Without both usability and security evaluations, it is challenging to ascertain the suitability of any scheme for real world usage.

Moreover, the alternative captchas for smartphones have not yet fully explored the technological richness that mobile devices offer. For example, sensors, multi-touch gestures, gestures using phones, or cameras can be leveraged as part of both the challenge and the input response.

## Chapter 3

### Captcha User Studies

#### 3.1 Introduction

In this chapter, we present two user studies assessing the usability of captcha schemes. The first user study was the antecedent to the rest of the work found in this thesis. The user study of captchas schemes on desktops provided us with a general overview of the usability of captchas. After our initial user study, we focused on the usability of captchas on smartphones. The second user study uncovered an additional set of issues related to standard captchas on smartphones.

The first evaluation, herein referred to as the MIOR study, consisted of five moving-image object recognition (MIOR) captcha schemes, one of which was a commercial captcha scheme and the rest were academic proposals addressing security flaws found in NuCaptcha [110]. The user study was our contribution to a larger security analysis and usability study of MIOR captchas by Xu *et al.* [150] presented at USENIX Security. The larger study included an attack that defeats challenges of NuCaptcha. Several design modifications (*e.g.*, extending the length of the code word, randomly changing the font of the challenges, overlapping characters more closely) were tested as mitigation strategies to the attack. The user study showed that the modified challenges fail to offer viable usability on desktop web browsers, even when the captcha strength is reduced below acceptable thresholds.

The second evaluation was an exploratory analysis of a user study of captchas on smartphones; we aimed at identifying opportunities and guiding improvements for captchas on smartphones. Results showed that existing desktop captcha schemes face effectiveness and user satisfaction problems when deployed on websites intended for smartphone usage. Among the more severe problems found were the need to often zoom and pan, and too small control buttons. These results were presented at SECURE 2013 [120].



The remainder of this chapter is organized as follows. Section 3.2 describes the methodology, data collection, analysis, and outcomes of the user study of five captcha schemes on desktops. The second user study of existing captcha schemes tested on smartphones is presented in Section 3.3.

### 3.2 User Study 1: Captchas on Desktops

In this section we report on a research ethics board-approved user study with 25 participants that we conducted to assess the usability of mitigation strategies proposed to address the weaknesses found in NuCaptcha [150, 151]. In collaboration with our colleagues we published the *Security Analysis and Related Usability of Motion-based CAPTCHAs* [150, 151]. This work describes a design and implementation of an automated attack based on computer vision techniques. The approach is suitable for captchas in the moving-image object recognition class.

If the challenges produced by the countermeasures proved too difficult for both computers and humans to solve, then they are not viable as captcha challenges. We chose a controlled lab study because besides collecting quantitative performance data, it gave us the opportunity to collect participants' impromptu reactions and comments, and allowed us to interview participants about their experience. This type of information is invaluable in learning *why* certain mitigation strategies are unacceptable or difficult for users and learning which strategies are deemed most acceptable.

Additionally, while web-based or Mechanical Turk studies may have allowed us to collect data from more participants, such approaches lack the richness of data available when the experimenter has the opportunity to interact with the participants one-on-one. Mechanical Turk studies have previously been used in captcha research [24] when the goal of the studies are entirely performance-based. However, since we are studying new usability mitigation strategies, we felt that it was important to gather both qualitative and quantitative data for a more holistic perspective.

### 3.2.1 Captcha Schemes

NuCaptcha is a commercial captcha scheme which offers multiple variants of character-moving challenges [110]. The challenge consisted of red characters (*i.e.*, codewords). Users must accurately enter the sequence of characters from the red codeword, ignoring the surrounding distraction text. This applies to all schemes derived from the Standard scheme. The countermeasures to our attack and proposed in our work by Xu *et al.* [150, 151] are as follows:

- *Standard*: this variant mimics NuCaptcha’s design. In this captcha scheme the video contains scrolling text with 2-3 words in white font, followed by 3 random red characters (codeword) that move along the same trajectory as the white words. NuCaptcha’s original captcha scheme is depicted in Figure 3.1.
- *Extended*: the codeword consists of  $m > 3$  random characters moving across a dynamic scene, Figure 3.2.
- *Overlapping*: same as the *Standard* case (*i.e.*,  $m = 3$ ), except characters are more closely overlapped, Figure 3.3.
- *Semi-Transparent*: identical to the *Standard* case, except that the characters are semi-transparent, Figure 3.4.
- *Emerging objects*: a different MIOR captcha where the codewords are 3 characters but created using an “Emerging Images” [94] concept, Figure 3.5.

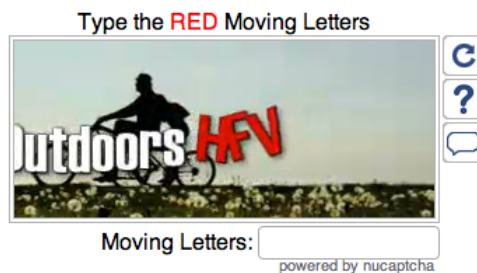


Figure 3.1: Standard replicates NuCaptcha’s original scheme [110].

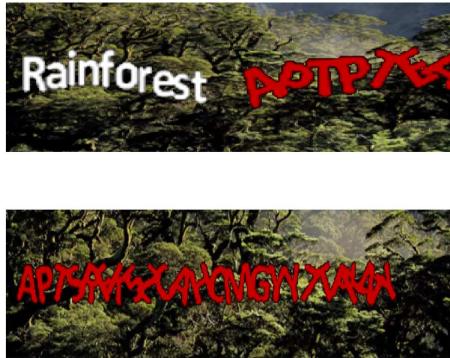


Figure 3.2: Extended variant. Top: scrolling; bottom: in-place [150, 151].



Figure 3.3: Overlapping characters (with ratio = 0.49) [150, 151].

The *Standard* variant is similar to NuCaptcha, the red characters (i.e., the code-words) also independently rotate as they move. For the *Extended* strategy, we set  $m = 23$ . All 23 characters are continuously visible on the screen. During pilot testing, we also tried a scrolling 23-character variation of the *Extended* scheme, Figure 3.2. However, this proved extremely difficult for users to solve and they voiced strong dislike (and outrage) for the variation.

For the *Overlapping* strategy, we set the ratio to be 0.49. At this ratio, the middle character is overlapped 100% of the time, and the others are 51% overlapped, Figure 3.3. The ratio refers to the overlap between characters. We estimate the degree that two characters overlap by the ratio of the horizontal distance of their centers and



(a) Forest background

(b) Beach background

(c) Sky background

Figure 3.4: Three backgrounds used for the challenges, shown for the *Semi-Transparent* variant [150, 151]. Note that the three images have characters in red.

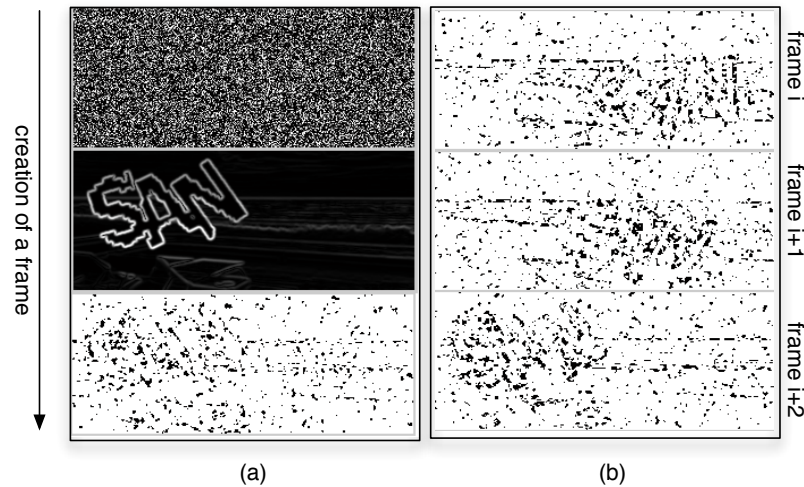


Figure 3.5: Emerging captcha. (a) Top: noisy background frame. Middle: derivative of foreground image. Bottom: single frame for an Emerging captcha. (b) Successive frames [150, 151].

their average width. That is, suppose that one character is 20 pixels wide, and the other is 30 pixels wide. If the horizontal distance of their centers is 20, then their overlap ratio is computed as  $20/\frac{20+30}{2} = 0.8$ . The smaller this overlap ratio, the more the characters overlap. A ratio of 0.5 means that the middle character is completely overlapped in the horizontal direction. In both the original captchas from NuCaptcha and our *Standard* case, the overlap ratio is 0.95 for any two adjacent characters.

For the *Semi-Transparent* strategy, we set the ratio to be 80% background and 20% foreground pixel ratio. For all experiments, we use the same alphabet (of 20 characters) as in NuCaptcha’s original videos, Figure 3.4.

A *challenge* refers to a single captcha puzzle to be solved by the user. Each challenge was displayed on a 6-second video clip that used a canvas of size  $300px \times 126px$  and looped continuously. This is the same specification used in NuCaptcha’s videos. Three different HD video backgrounds (of a forest, a beach, and a sky) were used. Some examples are shown in Figure 3.4. Sixty challenges were generated for each variation (20 for each background, as applicable).

We also tested the *Emerging* strategy. This countermeasure is based on the notion of Emerging Images proposed by Mitra *et al.* [94]. Emergence refers to “the unique human ability to aggregate information from seemingly meaningless pieces,

and to perceive a whole that is meaningful” [94]. The three-character codeword was represented by black and white pixel-based noise. The codeword moves on a wave across a canvas in an endless loop. The movement renders visible the characters, but observing any one frame does not reveal the characters from among the noise.

### 3.2.2 Methodology

**Study Design.** The user study was done in a controlled environment. Each participant completed their session one-on-one with the experimenter. A simple web-based user interface was designed where users could enter their response in the textbox and press submit, could request a new challenge, or could access the help file. Indication of correctness was provided when users submitted their responses, and users were randomly shown the next challenge in the set. Sixty challenges were generated using the same video parameters as the other conditions. A within-subjects experimental design was used, where each participant had a chance to complete a set of 10 captchas for each strategy. The order of presentation for the variations was counter-balanced according to a  $5 \times 5$  Latin Square to eliminate biases from learning effects; Latin Squares are preferred over random ordering of conditions because randomization could lead to a situation where one condition is favored (e.g., appearing in the last position more frequently than other conditions, giving participants more chance to practice). Within each variation, challenges were randomly selected. A session lasted at most 45 minutes and users were compensated \$15 for their time.

**Participants.** The twenty-five participants were undergraduate, graduate students, staff and faculty (15 males, 10 females, mean age 26) from a variety of disciplines. None had participated in any prior captcha studies.

**Procedure.** The study protocol consisted of the following steps:

1. Briefing session. We explained the goals of the study, detailing the study steps, and asking the participants to read and sign the consent form.
2. Demographics questionnaire. Before solving challenges, participants responded to a demographic questionnaire.
3. Captcha solving. Participants completed 10 challenges for each variation.

4. Satisfaction questionnaire. Immediately after completing the 10 challenges, users were asked to complete a paper-based questionnaire collecting their perception and opinion of that variation.
5. Brief interview. At the end of the session, a brief interview was conducted to gather any overall comments.

***Equipment and software.*** Participants used the lab’s desktop computer and the experiment was setup for them when they came into the lab.

***Ethics approval.*** This research was approved by our institution’s Research Ethics Board.

### 3.2.3 Data Collection

The user interface was instrumented to log each user’s interactions with the system. For each challenge, the user’s textual response, the timing information, and the outcome was recorded. A challenge could result in three possible outcomes: success, error, or skipped. Questionnaire and interview data was also collected.

### 3.2.4 Analysis

Our analysis focused on the effects of five different captcha variants on outcomes and solving times. We also analyzed and reviewed questionnaire data representing participant perceptions of the five variants. We used several statistical tests and the within-subjects design of our study impacted our choice of statistical tests; in each case the chosen test accounted for the fact that we had multiple data points from each participant. In all of our tests, we chose  $p < 0.05$  as the threshold for determining statistical significance.

One-way repeated-measures ANOVAs [76] were used to evaluate aggregate differences between the means for success rates and times. When the ANOVA revealed a significant difference, we used post-hoc Tukey HSD tests [84] to determine between which pairs the differences occurred. Here, we were interested only in whether the four proposed mitigation strategies differed from the *Standard* variant, so we report only on these four cases.

Our questionnaires used Likert-scale responses to assess agreement with particular statements (1 - Strongly Disagree, 10 - Strongly Agree). To compare this ordinal data, we used the non-parametric Friedman’s Test [84]. When overall significant differences were found, we used post-hoc Pairwise Wilcoxon tests with Bonferroni correction to see which of the four proposed variants differed from the *Standard* variant.

### 3.2.5 Outcomes

Participants were presented with 10 challenges of each variant. Figure 3.6 shows a stacked bar graph representing the mean number of success, error, and skipped outcomes. Table 3.1 indicates the means and standard deviations from the schemes for the number of success, error, and skipped outcomes. To be identified as a *Success*, the user’s response had to be entirely correct. An *Error* occurred when the user’s response did not match the challenge’s solution. A *Skipped* outcome occurred when the participant pressed the “Get A New Challenge” button and was presented with a different challenge. We observed differences in the outcomes, with the *Standard* variant being most successful and the *Semi-Transparent* variant resulting in the most skipped outcomes.

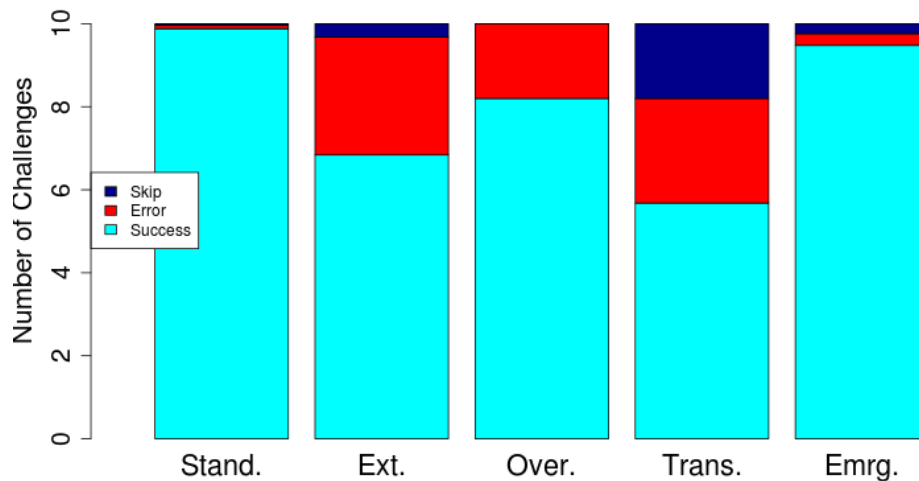


Figure 3.6: MIOR study. Mean number of success, error, and skipped outcomes for *Standard*, *Extended*, *Overlapping*, *Semi-Transparent* and *Emerging* variants, respectively.

For the purposes of our statistical tests, errors and skipped outcomes were grouped into one category since in both cases the user was unable to solve the challenge. Each participant was given a score comprising the number of successful outcomes for each variant (out of 10 challenges). One participant opted to view only six challenges in each of the *Extended* and *Emerging* variants. We count the remaining four as skips.

A one-way repeated-measure ANOVA showed significant differences between the five variants ( $F(4, 120) = 29.12, p < 0.001$ ). We used post-hoc Tukey HSD tests to see whether any of the differences occurred between the *Standard* variant and any of the other four variants. The tests showed a statistically significant difference between all pairs except for the *Standard* $\leftrightarrow$ *Emerging* pair. This means that the *Extended*, *Overlapping*, and *Semi-Transparent* variants had significantly fewer successes than the *Standard* variant, while *Emerging* variant showed no difference.

Scheme	Success Mean (SD)	Error Mean (SD)	Skips Mean (SD)
Standard	9.80 (0.3)	0.08 (0.2)	0.04 (0.2)
Extended	6.80 (2.3)	2.80 (2.1)	0.30 (0.3)
Overlapping	8.20 (1.6)	1.80 (1.6)	0.00 (0.0)
Transparent	5.60 (2.0)	2.50 (1.7)	1.80 (1.80)
Emerging	9.40 (0.9)	0.20 (0.5)	0.2 (0.1)

Table 3.1: MIOR study. Mean and standard deviation for number of success, error, and skipped outcomes.

***Time to Solve:*** The time to solve was measured as the time between when the challenge was displayed to when the response was received. This included the time to type the answer (correctly or incorrectly), as well as the time it took the system to receive the reply (since the challenges were served from our local server, transmission time was negligible). Times for skipped challenges were not included since users made “skip” decisions very quickly and this may unfairly skew the results towards shorter mean times. We include challenges that resulted in errors because in these cases participants actively tried to solve the challenge. The time distributions are depicted in Figure 3.7 using boxplots. Notice that the *Extended* variant took considerably longer to solve than the others.



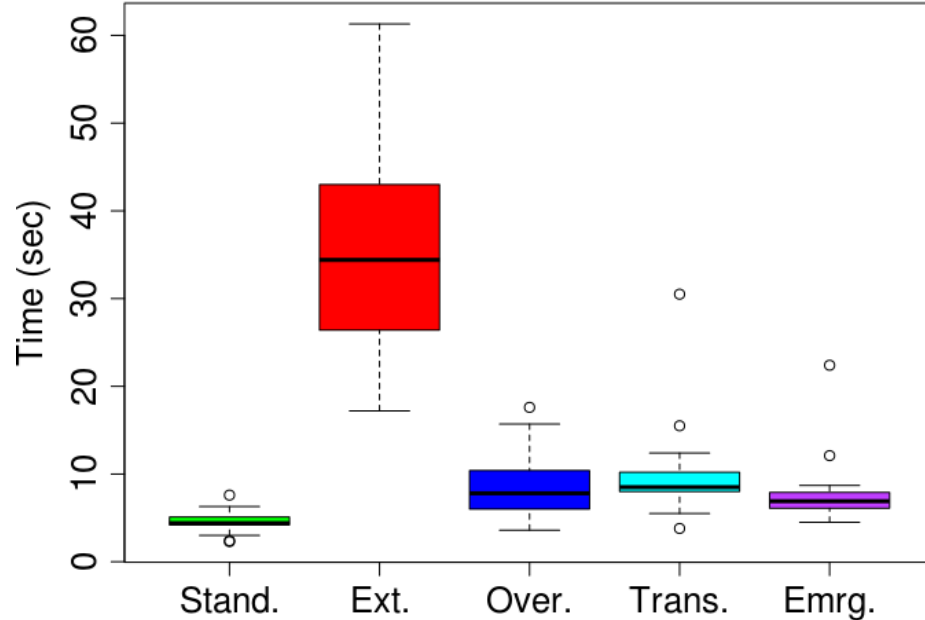


Figure 3.7: MIOR study. Time taken to solve the MIOR captchas.

We examined the differences in mean times using a one-way repeated-measure ANOVA. The ANOVA showed overall significant differences between the five variants ( $F(4, 120) = 112.95, p < 0.001$ ). Once again, we compared the *Standard* variant to the others in our post-hoc tests. Tukey HSD tests showed no statistically significant differences between the *Standard*  $\Leftrightarrow$  *Emerging* or *Standard*  $\Leftrightarrow$  *Overlapping* pairs. However, significant differences were found for the *Standard*  $\Leftrightarrow$  *Semi-Transparent* and *Standard*  $\Leftrightarrow$  *Extended* pairs. This means that the *Semi-Transparent* and *Extended* variants took significantly longer to solve than the *Standard* variant, but the others showed no differences. To verify our statistics, since the time data was skewed, we confirmed that our significant results were also significant with the equivalent non-parametric tests (Friedman / post hoc Wilcoxon with Bonferroni adjustment).

**Skipped outcomes:** The choice of background appears to have especially impacted the usability of the *Semi-Transparent* variant. Participants most frequently skipped challenges for the *Semi-Transparent* variant and found the Forest background especially difficult to use. Many users would immediately skip any challenge that appeared with the Forest background because the transparent letters were simply too difficult to see. For the *Semi-Transparent* variant, 35% of challenges presented on the

Forest background were skipped, compared 17-18% of challenges using the other two backgrounds. Participants' verbal and written comments confirm that they found the Forest background very difficult, with some users mentioning that they could not even find the letters as they scrolled over some parts of the image.

**Errors:** We remind the reader, an *Error* occurred when the user's response did not match the challenge's solution. Figure 3.8 shows the distribution of errors. It shows that the majority of errors were made on the middle characters of the challenge. We also examined the types of errors, and found that most were mistakes between characters that have similar appearances. The most commonly confused pairs were: S/5, P/R, E/F, V/N, C/G, and 7/T. About half of the errors for the *Extended* variant were due to confusing pairs of characters, while the other half involved either missing letters or including extra ones. For the other variants, nearly all errors were due to confusing pairs of characters.

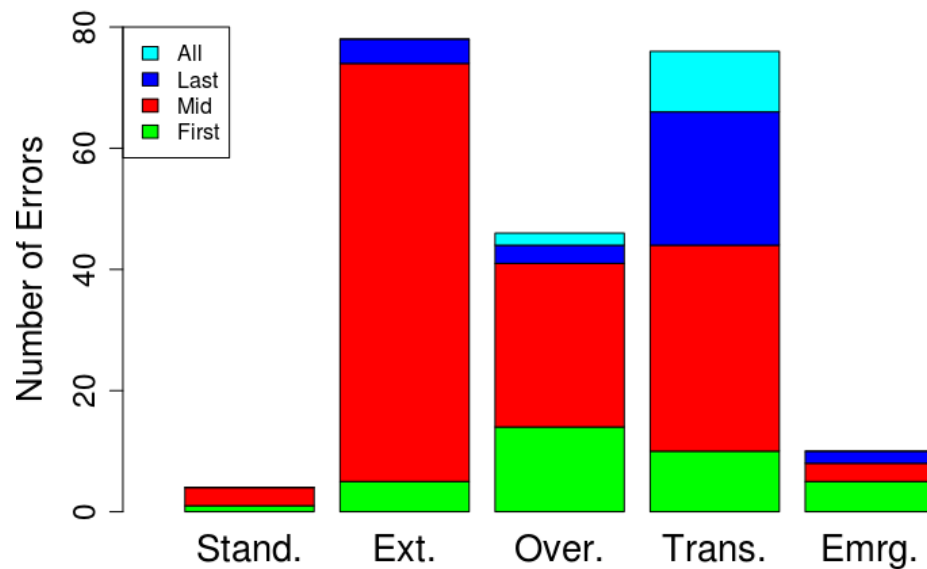


Figure 3.8: MIOR study. Location of errors within the codewords.

**User perception:** Immediately after completing the set of challenges for each variant, participants completed a Likert-scale questionnaire to collect their opinion and perception of that variant. For each variant, participants were asked to rate their agreement with the following statements:

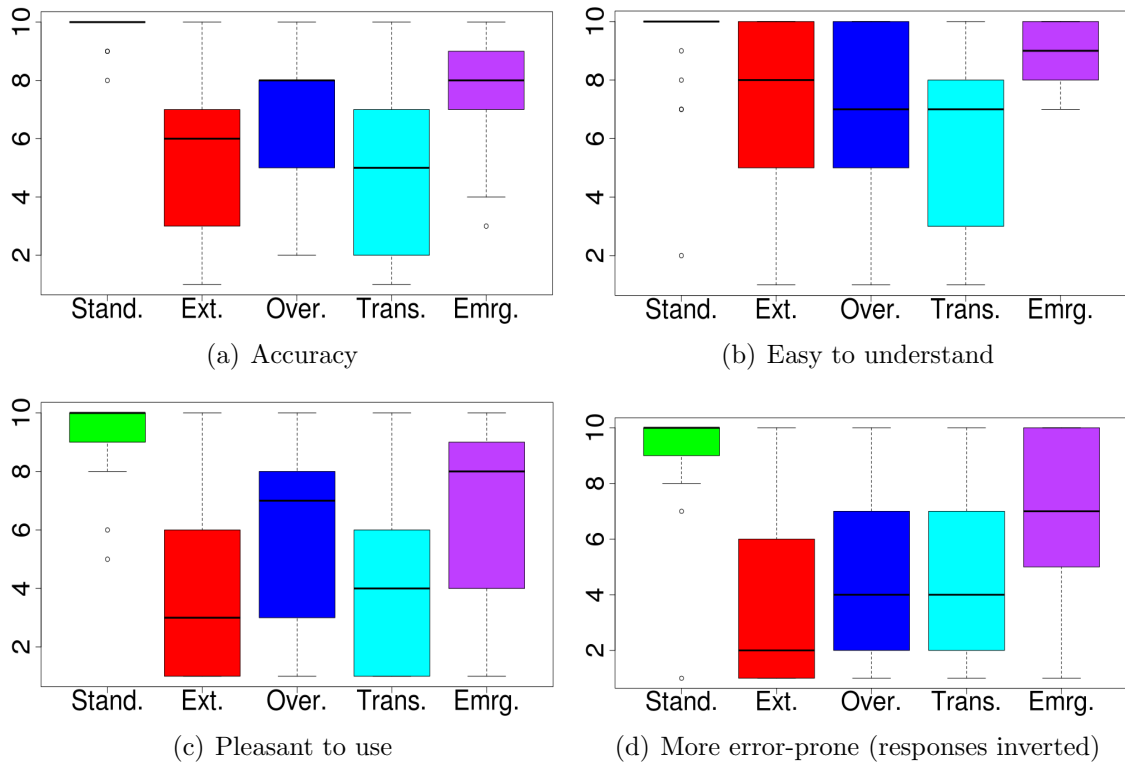


Figure 3.9: MIOR study. Likert-scale responses: 1 is most negative, 10 is most positive.

1. It was easy to accurately solve the challenge
2. The challenges were easy to understand
3. This captcha mechanism was pleasant to use
4. This captcha mechanism is more prone to mistakes than traditional text-based captchas

Figure 3.9 shows boxplots representing users' responses. Since Q.4 was negatively worded, responses were inverted for easier comparisons. In all cases, higher values on the y-axis indicate a more favorable response.

The results show that users clearly preferred the *Standard* variant and rated the others considerably lower on all subjective measures. Friedman’s Tests showed overall significant differences for each question ( $p < 0.001$ ). Pairwise Wilcoxon Tests with Bonferroni correction were used to assess differences between the *Standard* variant and each of the other variants. Significant differences were found between each pair compared. The only exceptions are that users felt that the *Extended* and *Emerging* variants were no more difficult to understand (Question 2) than the *Standard* variant. This result appears to contradict the results observed in Figure 3.9 and we believe that this is because the Wilcoxon test compares ranks rather than means or medians.

**Comments:** Participants had the opportunity to provide free-form comments about each variant and offer verbal comments to the experimenter. Samples are included in Table 3.2. Participants clearly preferred the *Standard* variant, and most disliked the *Extended* variant. Of the remaining schemes, the *Emerging* variant seemed most acceptable although it also had its share of negative reactions (e.g., one subject found it to be hideous).

Standard (NuCaptcha) and Emerging schemes had the lowest *time to solve* rates and showed no significant difference between them. The choice of background appeared to have an impact on the usability, in particular for *Semi-Transparent*. Participants most frequently skipped challenges for *Semi-Transparent* variant and found the forest background especially difficult to use. The longest variant, *Extended*, showed the most errors. The most commonly confused pairs were S/5, P/R, E/F, V/N, G/C, 7/T. The users preferred the *Standard* variant over the others. Although *Emerging* had a wide range of scores for *Pleasant to use*, its accuracy rating as measured by actual performance was similar to the *Standard* variant. Observing the participants allowed us to gather qualitative data. For example, several participants were not aware that (character recognition) CR answers are case insensitive. Some participants were typing each character of the answer in capital letters.

In addition to the explicit results obtained by the study described in this section, it established part of the ground work for our future studies. The section below explores captcha schemes designed for desktops but accessed on smartphones. At the time of the writing of *Security Analysis and Related Usability of Motion-based*

Variant	Comments
<i>Standard</i>	<ul style="list-style-type: none"> <li>- “User friendly”</li> <li>- “It was too easy”</li> <li>- “Much easier than traditional captchas”</li> </ul>
<i>Extended</i>	<ul style="list-style-type: none"> <li>- “My mother would not be able to solve these”</li> <li>- “Giant Pain in the Butt! Sheer mass of text was overwhelming and I got lost many times”</li> <li>- “Too long! I would prefer a shorter text”</li> <li>- “It was very time consuming, and is very prone to mistakes”</li> </ul>
<i>Overlapping</i>	<ul style="list-style-type: none"> <li>- “Letters too bunched – several loops needed to decipher”</li> <li>- “Takes longer because I had to wait for the letter to move a bit so I can see more of it”</li> <li>- “Still had a dizzying affect. Not pleasant”</li> <li>- “Some characters were only partially revealed, ‘Y’ looked like a ‘V’ ”</li> </ul>
<i>Semi-Transparent</i>	<ul style="list-style-type: none"> <li>- “Tree background is unreadable, any non-solid background creates too much interference”</li> <li>- “With some backgrounds I almost didn’t realize there were red letters”</li> <li>- “It was almost faded and very time consuming. I think I made more mistakes in this mechanism”</li> </ul>
<i>Emerging</i>	<ul style="list-style-type: none"> <li>- “Not that complicated”</li> <li>- “I’d feel dizzy after staring at it for more than 1 min”</li> <li>- “It was hideous! Like an early 2000s website. But it did do the job. It made my eyes feel ‘fuzzy’ after a while”</li> <li>- “It was good, better than the challenges with line through letters”</li> </ul>

Table 3.2: MIOR study. Sample participant comments for each variant.

*CAPTCHAs* [150, 151], the use of captcha schemes conceived for desktops on web pages, and accessed by mobile devices was a common web design solution due to a lack of mobile and smartphone-specific captchas.

### 3.3 User Study 2: Captchas on Smartphones

Now we present a user study conducted of captcha schemes conceived for desktops but accessed on smartphones. The goals of the study were to assess the following aspects: 1) the effectiveness of captcha schemes on smartphones, and 2) the user's experience of captchas on smartphones. The four evaluated captcha schemes are described below, and illustrated in Figure 3.10.

#### 3.3.1 Captcha Schemes

The schemes selected were intended to serve as good representative of each of the main captcha categories: character-recognition (CR), image-recognition (IR), and moving-image object recognition (MIOR).

- *reCaptcha* [56] is a free service that is widely deployed on the Internet. The CR challenge consists of recognizing and typing two words.
- *Asirra* [92] is a research IR captcha that was available<sup>1</sup> for deployment. The challenge consisted of selecting images of cats from a grid of 12 images containing dogs and cats. The image database came from a pet adoption service called Petfinder [112]. These images are pre-labeled by the person uploading each pet's image.
- *NuCaptcha* [110] is a commercial MIOR scheme. The challenge consists of either reading alphanumeric characters that overlap as they swing independently left to right (statically pinned at the centre of each letter), or reading a code word in a phrase that loops endlessly in the captcha window.
- *Animated* captcha, (Vappic 4D) [142], is an experimental captcha. The MIOR challenge typically consists of six alphanumeric characters arranged in a patterned cylinder that rotates in the centre of the captcha screen. The similarly patterned background portrays what could be the floor (or base) where the cylinder sits; this floor swivels up and down.

---

<sup>1</sup>The service was closed permanently in October 2014.



Figure 3.10: Captchas on smartphones study. Target Schemes. reCaptcha (CR), Asirra (IR), NuCaptcha (MIOR), and Animated (MIOR).

### 3.3.2 Methodology

The user study was done in a controlled environment. Each participant completed a one-on-one session with the experimenter and the session was video and audio taped. A within-subjects experimental design was used, where each participant attempted ten challenges for each scheme. Participants received random challenges from the respective demo sites. The content of each challenge was determined by the scheme. The solving order for the each of the schemes was determined by a  $4 \times 4$  Latin Square.

**Participants.** Ten participants were asked to complete challenges on either a provided smartphone or their own smartphone. The participants (5 females, 5 males) were graduate and undergraduate students, university staff, a private company IT employee, faculty members and a freelance employee. They ranged in age from 18 to 44, mean age of 32 years old. None had participated in any prior captcha studies. The average self-reported expertise using smartphones was 6.33 out of 10. The average phone ownership was 3.3 years. All except two had encountered captchas before the study. Participants were paid \$15 honorarium for their cooperation.

**Procedure.** The study protocol consisted of the following steps:

1. Briefing session. We explained the goals of the study, detailing the study steps, and asking the participants to read and sign the consent form.
2. Demographics questionnaire. Before solving the challenges, participants answered a demographic questionnaire.
3. Captcha testing. Participants visited a host page with links to the four schemes located on third party demo sites from the smartphone.
4. Satisfaction questionnaire. After each scheme, participants completed an on-line satisfaction questionnaire collecting their satisfaction and opinion of that scheme.

***Equipment and software.*** Seven participants used an Android OS (ver. 2.3.6) smartphone and three used iOS (iOS 4.0). The demographics and satisfaction questionnaires were implemented using Limesurvey<sup>2</sup>. We chose not to implement our own version of the schemes due to two main reasons: first, visiting the original demo sites allowed testing of the latest version of the schemes; second, we did not have access to implementation and deployment details which could impact the behaviour of the schemes.

***Ethics approval.*** This research was approved by our institution’s Research Ethics Board.

***Audio Captcha Pilot Test.*** We pilot tested audio schemes from several major websites. We realized that audio schemes are currently unusable on smartphones due to their high operational complexity and strong need for recall, so discontinued them from our tests. We found that the audio would open in different window or tab, the audio would open on a different application, or the audio decoder was not supported.

### 3.3.3 Data Collection

The participants responded to a demographics questionnaire and a satisfaction survey. Their performance measurements were limited to noting the typed answers, number of successes, skips/refresh, and errors while answering the challenges.

---

<sup>2</sup>LimeSurvey <http://www.limesurvey.org/>



### 3.3.4 Analysis

As in the MIOR study, we counted the number of successes, skips/refresh, and errors while answering the challenges (Figure 3.11). Table 3.3 indicates the means and standard deviations from the schemes for the number of success, error, and skipped outcomes. We counted a *success* when a user’s answer to the challenge is deemed correct by the demo site. An *error* was counted when the user’s response did not match the challenge’s solution and was indicated as incorrect by the demo site. A *skipped* outcome was counted when the participant pressed the “Request new images”, “Get A New Challenge” or “Skip” button and was presented with a different challenge.

NuCaptcha shows the most successful outcomes compared to the other schemes, followed by reCaptcha. A possible explanation is that challenges for NuCaptcha consisted of only three characters with no distortion, while reCaptcha uses distortion on only one of the two words. However, we noted some participants were flipping the phone from landscape to portrait mode a few times, attempting to find the best fit to see and answer challenges without panning. Asirra requires selecting images which expand and obscure other images, forcing users to pan across the challenge; we observed that this was the cause of a large number of errors. Animated demands considerable attention from users. We noticed participants often shifting their sitting position and handling of the phone while solving this scheme and verbally indicating their discomfort. We observed that Animated’s movement exacerbates the known issue of confusable characters and thus participants were prone to typing errors and requests for new challenges.

Scheme	Success	Error	Skips
	Mean (SD)	Mean (SD)	Mean (SD)
reCaptcha	7.8 (1.7)	1.5 (1.2)	0.7 (1.4)
Asirra	7.2 (2.2)	2.6 (2.2)	0.1 (0.3)
NuCaptcha	9.6 (1.2)	0.4 (1.2)	0.0 (0.0)
Animated	6.1 (3.6)	2.6 (2.6)	1.2 (1.3)

Table 3.3: Captchas on smartphones study. Mean and standard deviation for number of success, error, and skipped outcomes.

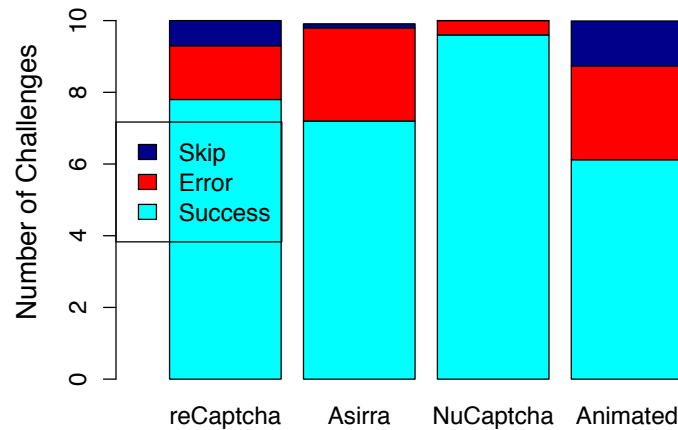


Figure 3.11: Captchas on smartphones study. Mean number of success, error and skipped outcomes.

### 3.3.5 Outcomes

We now present the outcomes from our usability study. From this study we collected performance data, usability problems and perceived qualitative indicators. We do not report statistical analysis because our goal was to formatively identify strengths and weaknesses, not to compare the schemes against each other.

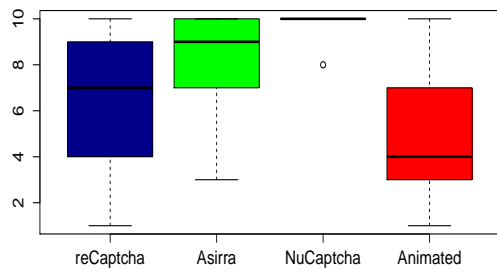
**Usability Problems.** Two researchers watched and coded the videos of the testing sessions. Usability problems were identified and summarized through an iterative process where the researchers reached mutual consensus of the main categories of problems and identified the most serious issues. We group the usability problems uncovered by the user study in six groups:

1. Small buttons: Participants found control buttons (skip, audio, help) too small and sometimes they pressed these by mistake.
2. Interface interaction: Input interaction can interfere with answering challenges. Some IR schemes require tapping on images. While solving challenges in Asirra, participants found the scheme’s zooming mechanism obscured other thumbnails. We believe that most of the usability problems with this scheme, when this scheme is used on a smartphone, are due to the scheme’s autozoom feature that blocks other images and forces unnecessary panning and zooming.

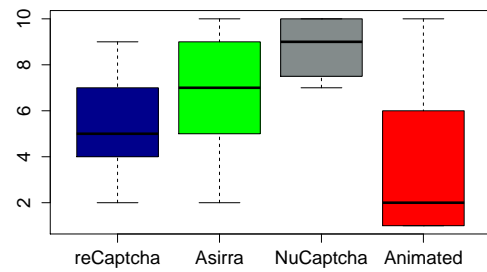
3. Confusing characters/images: Captchas are by nature somewhat confusing to solve, but the problem is compounded on small screens. We observed participants confusing characters (*e.g.*, 1/i/l) and confusing images of dogs with those of cats primarily because the small image made it difficult to identify details.
4. Inefficient schemes: Several participants pointed out that the challenges were so small that they needed to zoom and pan across the screen to locate and reply to them. Some tasks were tedious, time consuming, and frustrating to solve on a smartphone. CR schemes sometimes mixed alphanumeric characters, forcing users to swap between input keyboards.
5. Data plans: Several participants were concerned about data transfer due to costly data plans. Although our sample of participants is not large enough, any significant subset of users concerned with data transmission costs could be important to some sites. If these sites would not want to lose site traffic, schemes that are image or video intensive are probably not good options for mobile devices.
6. Lack of instructions: We observed, and heard from, participants not knowing if CR challenges were case sensitive or not, or being unsure if spaces were required for challenges with two words, being unsure how to clear previous image selections, and being confused about where the challenge started. Deselecting images in Asirra required double tap on the image under iOS, where as Android required a single touch for selecting or deselecting. Instructions were not immediately apparent to users as they struggled with the interface problems.

In summary, the most severe problems were found due to the small buttons, the interface interaction (input mechanisms) and confusing characters/images.

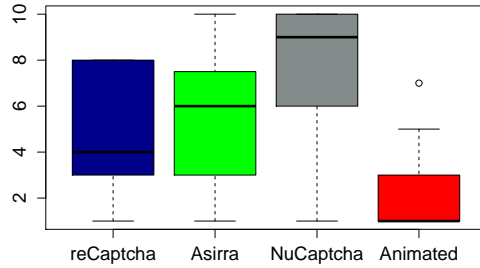
***User Perception.*** Users answered a number of Likert scale questions about each scheme. The collected satisfaction results are graphed in Figure 3.12. The results show that users clearly favoured the NuCaptcha scheme and rated the others lower on all subjective measures. We speculate that NuCaptcha is favoured over the rest due to its lack of distortion and short challenges which were considerably easier



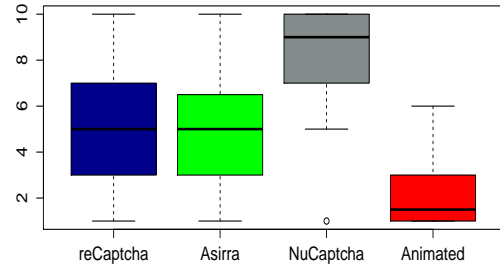
(a) Easy to understand



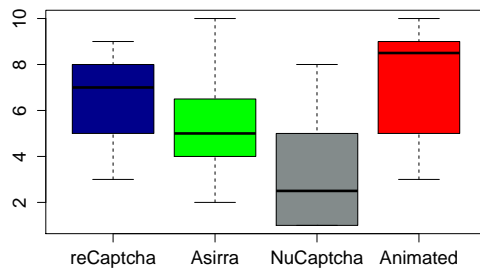
(b) Pleasant to use



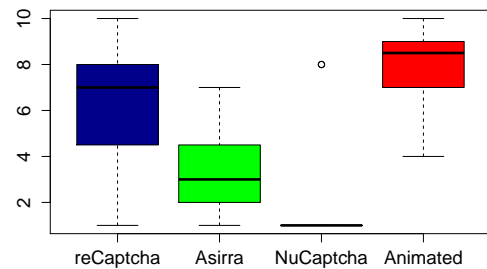
(c) Preference over other schemes



(d) Good candidate for smartphones



(e) More error-prone



(f) Harder to solve challenges

Figure 3.12: Captchas on smartphones study. Likert-scale responses: 1 is Most Negative, 10 is Most Positive.

than the other schemes (although also least secure). Animated was clearly the most disliked scheme.

**Comments.** Participants had the opportunity to provide free-form comments about each scheme and offer verbal comments to the experimenter. We highlight a few comments about each scheme. Overall participants preferred schemes that involve quick, simple challenges and little or no distortion. Participants disliked ambiguity in the challenge (*e.g.*, S/5, 1/I, 0/0).

Variant	Comments
<i>reCaptcha:</i>	- “Text entry on smart phones needs to be mastered better” - “Challenges are long to type for a mobile device keyboard”
<i>Asirra:</i>	- “The number of images presented became crowded on my phone” - “It was to big! I want to see things on one screen, don’t like to move so much”
<i>NuCaptcha:</i>	- “...the letters didn’t move at all so it was very ease for me and the attackers!” - “When you enter the text I can press the keypad enter or the captcha button, didn’t know which one to press at first”
<i>Animated:</i>	- “It hurts my head - it requires too much thought...” - “The captcha controls and smartphone input mechanism were overlapping.”

Table 3.4: Captchas on smartphones study. Sample participant comments.

We observed differences in the participants’ outcomes (number of corrects, errors or skips), NuCaptcha scheme being most successful, and Animated resulting in the least successful outcomes (*i.e.*, more errors or skips). The most skipped outcomes were observed for the Animated scheme. NuCaptcha was found the most pleasurable, while Animated was rated the least. We remind the reader the *Standard* variant in user study 1 is the equivalent to NuCaptcha [in the present study]. We note that the satisfaction results for this user study are only a reflection of users’ comparison among these four schemes; positive scores are not necessarily an indication that schemes do not have usability problems on smartphones.

### 3.4 Discussion

The MIOR user study provided (Section 3.2) a wealth of information regarding the usability of captchas in general. The second study uncovered an additional set of issues related to standard captchas on smartphones.

Although NuCaptcha's outcomes for both studies were favourable, we note the following outstanding issues. Regarding its security, NuCaptcha has been broken [150, 151], along with several potential improvements to the scheme. It is not advisable to use it as a security mechanism at this time [150, 151]. NuCaptcha provides a clear example of a security mechanism that meets usability criteria but does not provide adequate security, therefore failing to meet its intended purpose. When designing security mechanisms that involve users, both usability and security must be given equal attention. In some cases usability problems lead to decreased security as users find ways to circumvent the security system. In other instances, such as with captchas, usability problems lead users to abandon the related primary task which is equally problematic for websites who lose business as a result.

From both studies, qualitative and quantitative data enabled us to start extracting common usability issues faced by users. This information was fundamental for developing the heuristics described in Chapter 4.

The similarities between the problems uncovered by the two studies highlighted the fact that captchas have become sufficiently hard for most users, and these are seen as a nuisance piece of software. Generally speaking many users are unaware that character-recognition challenges are case-insensitive.

While the MIOR study uncovered overall usability problems, the second study uncovered problems that would be recurrent in further evaluations. The captchas on smartphones evaluation included more captcha classes than the MIOR study; the differences between the two studies underlined the perceptual and usability problems users faced while solving captchas on smartphones. From unplayable audio challenges, excessive panning and zooming to small buttons the problems hindering the correct solving of challenges served as indicators for developing guidelines preliminary to the heuristics.

## Chapter 4

# Heuristics for the Usability Evaluation of Captchas on Smartphones

### 4.1 Introduction

Our previous two studies of character-recognition captchas on desktop computers, (Section 3.2) and the multi-scheme usability study of captchas designed for desktop on smartphones, (Section 3.3) provided a basis and initial guidelines towards evaluating the usability of captchas on smartphones. We apply the insight acquired from these two studies to develop domain specific heuristics for captchas on smartphones.

While extensive research is available on captchas for traditional desktop computing [24, 49, 143, 158], work on captchas for mobile devices is limited despite the popularity of mobile web browsing. The goal of this chapter is to develop and evaluate a new set of heuristics for evaluating the usability and deployability of captchas on mobile devices, in particular smartphones. Heuristic evaluation [104] is a good fit as an evaluation technique to assess the usability of captcha schemes for several reasons. It provides quick evaluations compared to a user study that spans several days or weeks, it avoids costs related to user studies, and yields results which can be used either during development to improve a design or to assess an existing design. Existing heuristics in the mobile domain are not well-adapted to effectively assess captchas on mobile devices. Domain specific heuristics are often developed due to the lack of specificity of existing heuristics [109, 127]. In this chapter, we explore how domain specific heuristics can help determine if captcha schemes are suited for mobile usage. We focus on the heuristics and validating them as tools rather than on the specific usability of the captcha schemes themselves. Any proposed set of heuristics should be independent of the evaluated software, so we demonstrate their use on four different schemes.

Although many sites offer tailored mobile versions of their content, they use regular desktop captchas which can affect the overall success of the website. Thus evaluating the usability of existing and new captcha schemes should be a requirement before deployment by captcha designers, IT practitioners or Webmasters. Recent captcha schemes propose to address the usability issues of captchas on mobile devices [79, 130] and our new heuristics could help independently evaluate their usability.

The outline of this chapter is as follows. Related work on heuristics is described in Section 4.2. In Section 4.3, we summarize prevalent methods for developing domain specific heuristics. Section 4.4 presents our new Mobile Captcha (MC) heuristics. We compile methodologies to evaluate the effectiveness of heuristics in Section 4.5. To evaluate the effectiveness of our heuristics, two heuristic evaluations were conducted. For further analysis, we compared results of the user study in Section 3.3 to the results of the MC heuristics. The first evaluation using the MC heuristics is described in Section 4.6, and the second evaluation using Nielsen's heuristics is explained in Section 4.7. Our results are shown in Section 4.8. Finally, we give concluding remarks in Section 4.9.

The heuristics and the heuristic evaluation presented in this Chapter were published as full paper at British HCI 2015 [122].

## 4.2 Heuristic Evaluations

Expert-based evaluation techniques, such as Heuristic Evaluation (HE) [104], are well known methods that allow relatively quick and easy usability assessments. Given that a captcha is a relatively simple piece of software, a heuristic evaluation gives evaluators the freedom to explore the scheme from several perspectives.

Research in mobile computing and usability has sparked interest in expert evaluations [14, 117, 160]. Since being originally proposed as a methodology by Nielsen, several authors have adapted the original set of heuristics [109] to better fit the requirements of specific application domains<sup>1</sup>. We describe a few examples of such domain specific heuristics. Most relevant to our work, Bertini *et al.* [14] developed heuristics to evaluate usability issues in general applications for *mobile computing*,

---

<sup>1</sup>Refer to Appendix A for Nielsen's set of heuristics.



Approach to developing domain specific heuristics	Domain	Number of heuristics
1. Adapt or extend Nielsen's	Ambient Displays [86]	12
	Security [160]	6
	VE <sup>3</sup> [137]	12
	Mobile [14]	8
2. Base on theory from target domain	Security [68]	7
	CSCW <sup>4</sup> [60]	5
3. Transform design guidelines	Mobile [14]	8
	Web services [140]	7
	General advice [123]	(advice)
4. Use domain experience & analysis of usability problems	General [109]	9
	General [103]	10

Table 4.1: Methods for developing domain specific evaluation heuristics.

with emphasis on contextual usage<sup>2</sup>. Jaferian *et al.* [68] developed a set of heuristics for IT *security management* tools evaluation. Their heuristics focuses on the collaborative nature of security tools (*e.g.*, tools for threat and vulnerability management). Other examples include the widely cited work on *ambient displays* [86] which adapts Nielsen's heuristics and Sutcliffe and Gault's [137] proposed a set of heuristics for *virtual environments* considering natural engagement, natural expression of action, and realistic feedback.

Our literature review revealed four common and prevalent methods for developing special purpose usability heuristics. Table 4.1 lists these methods for domain specific heuristics with examples. The methods are not mutually exclusive; for example, the first and third methods are often combined.

### 4.3 Developing the MC Heuristics

Our approach to developing the MC heuristics was to extend and adapt Nielsen's heuristics [103]. The nature of captchas on smartphones motivate a set of problems different from desktop captchas; in particular, the diverse, and sometimes complex, captchas adopted by websites typically become too cumbersome for mobile users

<sup>2</sup>Refer to Appendix B for Bertini's *et al.* set of heuristics.

<sup>3</sup>Virtual Environments.

<sup>4</sup>Computer Supported Cooperative/Collaborative Work.

[79, 120, 130] and these issues are not easily captured by existing heuristics.

The focus of our work is providing: a) heuristics that find problems hindering correct and efficient human challenge-solving of different captcha schemes on smartphones; and b) a tool for IT practitioners in charge of deployment of captchas on their web sites to quickly and inexpensively assess the usability of available captcha schemes.

No usability technique can evaluate the *security* of a captcha scheme. There is a wide range of methods to evaluate security [23, 25, 54, 150]. Even if security is not a main topic of our work on heuristics, there is an implicit assumption that in order to be viable, captchas must meet at least some minimal threshold of security.

We used the following steps for developing the MC heuristics.

1. We reviewed the academic literature and identified approximately 50 of the most relevant papers in human-computer interaction, interaction with mobile devices, and security. The publication venues included (but were not limited to): CHI, TOCHI, MobileHCI, EUROSEC, ACM CCS, Advances in Computers, USENIX Security, SOUPS, ACSAC, HotMobile, and EUROCRYPT.
2. We summarized documentation on interaction paradigms, input methods, mobile device interaction design, captcha usability concerns, general captcha security considerations, mobile internationalization and cultural concerns, and contextual usage.
3. We developed an initial set of heuristics and corresponding guiding questions. The two user studies presented in Chapter 3 provided us a preamble to developing this initial set of heuristics. We invited expert evaluators with experience on HCI and Security to use the heuristics to evaluate four captcha schemes<sup>5</sup> on smartphones.
4. Once we received evaluations from this first group of evaluators, we refined, combined, and discarded heuristics that were not capturing or properly reflecting issues and problems. For example, the security heuristic was eliminated since

---

<sup>5</sup>Details regarding the evaluation are given in Section 4.5.

the evaluation provided by the evaluators was purely based on their perception and personal assessment of the schemes, rather than systematic study.

5. More HCI evaluators were invited to carry out a second heuristic evaluation.
6. The heuristics went through a second refinement iteration. On this iteration, no heuristics were eliminated or added but some wording and guiding questions were improved based on feedback.

#### 4.4 Proposed Heuristics: Mobile Captcha Heuristics (MC)

Our proposed heuristics, related heuristics, guiding questions to help evaluators identify problems, their *raison d'être*, the mobile perspective, and examples of the issues each heuristic evaluates are described below. Heuristics MC1 - MC6 address usability while MC7 focuses on deployability.

**MC 1.** *User control and freedom.* Input mechanisms required to answer the captcha should not lead users to make mistakes. The scheme should provide controls to correct, retype, select, or clear the input before submitting. Does the input mechanism obstruct the challenge? Can the user easily obtain a new challenge? Are appropriate controls provided (e.g., zoom, audio pause)?

**Rationale.** This heuristic is similar to Nielsen's *User control and freedom* heuristic; while Nielsen's heuristic focuses on providing a way out from an unwanted state, our heuristic considers input mechanisms. These artifacts often interfere with the rest of the displayed page. We also generalize control mechanisms beyond typed input, e.g., voice input. A related heuristic from the gaming mobile domain is *Device UI and game UI are used for their own purposes*, by Korhonen *et al.* [75]. Based on direct manipulation principles, an object should remain visible while the user performs physical actions on it, and the result of these actions should be visible as well. The direct manipulation principles [131] are: a) continuous representation of the objects and actions; b) rapid, reversible, incremental actions with immediate feedback about the object of interest; c) physical actions and button pressing instead of issuing commands with complex syntax. From the perspective of mobile devices, visibility of objects and their affordances is challenging due to constrained screen sizes. Lent [70]

remarks that on-screen keyboard typing is a disruptive operation. He recommends providing all navigation inside the web application rather than using the browser's navigation controls and giving the user adequately-sized targets. Others have recognized the small screen as a usability problem leading to typing mistakes [19, 71].

Additional sources of input mistakes are the “fat finger problem” (accidentally touching the wrong field or button), the size and position of controls and external artifacts, and voice input. Designs must also consider that mobile devices could be handled with one or two hands, while standing, sitting, or walking although problematic, typing remains the most popular captcha input mechanism. Captchas not falling into the traditional character-recognition class may instead require the user to select by pressing on objects, to trace contours, or to use speech input. Schemes should also provide means to deselect, correct or erase user actions.

**MC 2. *Learnability.*** The scheme should provide, and not require more than, brief instructions. The captcha scheme should be intuitive, without excessive cognitive load. Can the user figure out how to use the captcha quickly/easily? Is there any guidance? Can a small set of instructions explain it sufficiently? Are such instructions provided? Is the scheme understandable?

**Rationale.** This heuristic derives from Nielsen's *Help and documentation*. In addition, *Learnability* is a usability goal [101, 123]. Ideally, a scheme could be used with minimal or no documentation, especially since the captcha challenge is an obstacle to users' overall objective. Korhonen *et al.* [75] suggest these related heuristics for games: *the player does not have to memorize things unnecessarily*, *the game contains help*, and *the game provides clear goals or supports player created goals*. Similarly, Carroll's and van der Meij [27] minimalist approach to designing instruction and documentation relies on task orientation. They suggest presenting only material that is essential to performing the task and structuring the material to fit an action-oriented approach. These principles are a good fit for mobile devices. Mobile best practices and design guidelines advocate for clear and simple language, labelling all form controls, properly positioning the labels in relation to their form controls, and making it simple for users to access help [28, 146]. The growing number of captcha schemes and their increased adoption may lead to user confusion. From character-recognition

(CR) to moving-image object recognition (MIOR) the task at hand is not always clear to users. A common learnability issue with text captchas is that users often do not know if their answer must respect the case of the challenge. The basic captcha task should not be complex (for humans to answer), hard to understand, hard to remember, or unintuitive but this is rarely the case. For instance, many audio schemes require users to recall long strings or multiple words; this could be excessive cognitive load for some users.

**MC 3.** *Efficiency of use.* The scheme should be quick and efficient to use. The scheme should avoid controls or non-captcha related artifacts that may lead to errors or inefficiencies, and minimize unnecessary complexity. Do any barriers hinder solvability (by humans) of the challenge? Is the captcha solvable within a reasonable time?

**Rationale.** Nielsen’s original *Flexibility and efficiency of use* heuristic focuses on allowing customizations for frequent and advanced users. In contrast, our heuristic addresses issues that may create stumbling blocks that hinder solvability and decrease the efficiency of the captcha scheme. Efficiency pertains to the way a system or product assists users in carrying out their tasks [123]. Thompson and Kemp [139] include *flexibility and efficiency of use* as a heuristic. Bertini [14] refers to efficiency of use as allowing mobile users to dynamically configure the system based on contextual needs. In the MC heuristics, *efficiency* refers to the way a captcha scheme supports users in completing the challenge: if needed, provide control buttons, allow users to obtain new challenges, and if security is not negatively impacted, include play controls (*e.g.*, pause and play controls for audio schemes). For example, a challenge that requires swapping alpha and numeric keyboards further decreases efficiency. Although the captcha task is not necessarily time consuming, solving captchas on smartphones may become tedious. In related mobile research, Shrestha [132] found that participants performed too much scrolling and often lost visibility within a web page; these are especially problematic when solving captchas. Further, the screen size may also lead to loss of position [13]. Zooming has the drawback of making users lose overview of the information space while panning [20]; if the challenge becomes hidden then it negatively impacts the ability of users to answer the challenge.

**MC 4. *Input mechanisms.*** The scheme should support input mechanisms allowing easy completion of the challenge on different devices. Does the input mechanism cause any additional problems for completing the challenge? Does the scheme have appropriate input methods for the target device? Does the scheme allow device appropriate input alternatives? Does it allow for user input preference?

**Rationale.** This heuristic derives from Bertini’s *Ease of input, screen readability and glanceability* which has similar reasoning to ours: “Mobile systems should provide easy ways to input data, possibly reducing or avoiding the need for the user to use both hands” [14]. Some interface types are primarily concerned with a function, while others focus on the interaction style used, the input/output device used, or the target platform/device [123]. Ten years ago there were only two competing paradigms<sup>6</sup> for mobile input: pen-based input and keyboard-based input [85]. Now, we also find touch-sensitive screens, speech input, gyroscopes, accelerometers, gestures, scroll rings and other features allowing for a multitude of input mechanisms. From the perspective of mobile devices, there are a number of mobile text input modalities, including the unistroke letter set, optimized on-screen keyboard layouts (optimized for input fields such as email addresses or URLs), and soft keyboard error correction [159]. Furthermore, users have to switch their attention between the keyboard and the input area due to the lack of tactile feedback, making this activity error-prone [85], [111]. Consider the case in which the device allows for voice input, but the captcha challenge is in a language other than the user’s settings: the speech recognition mechanism may not recognize the answer, or the user is burdened with changing language settings. Similarly, CR schemes in non-roman alphabets could require special text entry solutions [83].

**MC 5. *Solvability.*** Captcha challenges should be simple for users to solve while preventing automated computer solutions. Challenges should minimize user confusion and be suitable for mobile devices (with small screens, and restricted input methods). Does the challenge contain ambiguous elements that could lead to confusion for users?

---

<sup>6</sup>Paradigms are overarching approaches that comprise a set of accepted practices and framing of questions and phenomena to observe[123].

Do distractors<sup>7</sup> make it too difficult to solve the challenge?

**Rationale.** This heuristic does not relate to other existing heuristics. It represents a delicate balance between the secure and the usable. A purely usability-based theoretical background for this heuristic could lead to overlooking security attributes. Captcha schemes employ various distortion or distraction techniques to prevent adversarial attacks. However, excessive or poorly designed distractions can lead to challenges with unacceptably low solution rates. Challenges, regardless of the class, should be easy for humans, hard for machines [22, 124, 158]. Design guidelines for CR captchas suggest using multiple fonts, sans-serif fonts (non-confusable character set), and varying the font size [22], [158] to improve the scheme’s security. If not carefully applied, these design guidelines can have a negative impact on solvability by humans. From the mobile perspective, security features often contradict usability design guidelines or may not fit the constraints of small screens. For instance, CR challenges are often rendered as images. If the image is too large, it may not be rendered by the device [146]. If the image is too small, distorted, and complex, the user may need to zoom and lose overview [20]. Some challenges in the audio category are implemented as spoken characters, mainly English, and some characters are prone to confusion such as a/8, p/b, g/j [124]. Another consideration for audio captchas is that audio distortion techniques are commonly used to prevent attacks; however, a noisy environment may make the challenge unsolvable. Content-confusion is not unique to audio challenges; character-based challenges may also include confusable character sets (*e.g.*, 1/1, 0/0, 6/G/b, 5/S/s, 2/Z/z, nn/m, rn/m).

**MC 6.** *User perception.* Captchas should be pleasant to solve and should cause no discomfort to users. Does the scheme have potential to cause discomfort (*e.g.*, eye strain or nausea) Is the scheme pleasant to use? Is the scheme usable and acceptable?

**Rationale.** Nielsen’s closest heuristic is *Aesthetic and minimalist design*, which pertains more to dialogs. Korhonen *et al.* ’s [75] mobile gaming heuristic: *the players are rewarded and rewards are meaningful* is linked to gamers’ satisfaction and engagement. *User perception* pertains to the user’s satisfaction while using a system. This aspect is a subjective quality of the scheme. While users typically cannot choose the

---

<sup>7</sup>We remind the reader, distractors are techniques, elements or distortions to obfuscate bot-solving.

type of captcha, they can choose, in some cases, whether to continue using the web site serving it. A mobile user’s expectations and experience is impacted more easily due to the variable context of use [29]. Hiltunen *et al.* [65] and Botha *et al.* [16] identify the following factors influencing the mobile experience: user (unique attributes of mobile users, cultural context, and skills), interaction techniques, task context (multitasking, interruptible and mobile), physical context, social context, technological context, privacy and security, device, and connection (network and bandwidth concerns). Solving challenges on smartphones can become a frustrating task if the scheme is too challenging, or if the captcha requires more cognitive load than the user is willing to dedicate to the task. Yan and El Ahmad [158] include *satisfaction* as one of their usability criteria for captchas: “are users subjectively satisfied and would they be willing to use such a scheme?”. For example, users may not appreciate CR challenges based on Roman characters when their devices have a different character set [88].

**MC 7.** *Consistency with user’s localization and environment.* The scheme should be universal<sup>8</sup> and should be suitable for the range of situations and environments in which users may access the scheme. Language and culture should not impose additional barriers to solving challenges. Are the task demands appropriate for the environments in which users will be accessing them? Is the challenge independent of culture, language, location? Does the scheme provide ample time to solve the challenge (*i.e.*, time-out does not impede solvability)?

**Rationale.** Nielsen’s heuristics do not explicitly address localization and environment, but the *Consistency and standards* heuristic relates to following platform conventions. Other heuristics related to contextual usage are *the game accommodates to the surroundings* and *interruptions are handled reasonably* [75]. Localization “refers to the adaptation of a product, application or document content to meet the language, cultural and other requirements of a specific target market (a locale).” [145]. From the perspective of mobile devices, the usage environment and context of use are of special interest. Cui and Roto [41] identify four contextual factors: spatial (mobile

---

<sup>8</sup>Universal design is the term used to reflect a particular perspective upon the design of interactive products and services that respects and values the dimensions of diversity intrinsic in human capabilities, technological environments and contexts of use [135].



or stationary), temporal (duration of breaks), social (alone or in group) and access factors (WLAN or cellular network). Lee *et al.* [77] classified mobile Internet use into personal and environmental contexts. Personal context pertains to the state of the users themselves in terms of emotion, time and movement. Environmental context include all factors and circumstances external to the user: location, lighting conditions, distraction, and crowding (*e.g.*, other people, social interactions) in the immediate environment. In terms of captchas and their deployability, a captcha scheme is often chosen based solely on the reasoning that it is a popular solution. Although most captcha design suggestions include universality, not all deployed schemes achieve this goal. Challenges should be independent of users’ physical location, language, or culture [124]. Furthermore, whether the challenge is friendly to non-native speakers is another usability concern [158]. As part of captcha deployment, IT practitioners must also consider that captcha schemes will be accessed from a variety of devices. Thus, captcha schemes should be portable to various devices, screen sizes, and input mechanisms.

Approach to validate heuristics	Authors
1. Comparative study (Nielsen’s vs Proposed set)	[68], [160], [86]
2. Nielsen’s approach	[109], [9]
3. Ad hoc methods	[137], [115]
4. Comparative user study vs Proposed set	No references found

Table 4.2: Methods of validating domain specific heuristics.

#### 4.5 Assessing the MC Heuristics

The most common approach to validate heuristics is to evaluate an application with both the proposed heuristics and Nielsen & Molich’s heuristics [103]. This approach varies from proposal to proposal in the thoroughness of the analysis. For instance, Jaferian *et al.* [68] apply the two sets of heuristics to an identity management system. Baker *et al.* [9] assess heuristics to evaluate groupware. In both cases, two main factors are analyzed and compared against Nielsen’s: a) problems found (false positives,

raw, consolidated), and b) inspector performance (average performance, consistency of individual performance, proportion of inspectors who found each problem, ranking of individual performance, number of inspectors required to uncover a good number of problems).

Other authors use more ad hoc methods. These methods include comparing the problem-finding consistency of the proposed heuristics for two applications and asking evaluators to rate the heuristics. Table 4.2 summarizes evaluation methods for domain-specific heuristics. Method 4 compares the results of a user study to the results of a heuristic evaluation using the proposed set. This method is uncommon since the idea of HE is to save time, money, and recruiting participants. However, user testing should provide a more comprehensive and well-rounded evaluation. The ultimate criterion for assessing the effectiveness of usability evaluation methods is finding real usability problems [63]. Hartson *et al.* [63] define realness as follows: “a usability problem is real if it is a predictor of a problem that users will encounter in real work-context usage and that will have an impact on usability (user performance, productivity and/or satisfaction)”.

To evaluate our proposed heuristics, we carried out two heuristic evaluations. The first evaluation consisted of using the proposed heuristics in a standard heuristic evaluation process assessing four captcha schemes (Section 4.6). We repeated the process with Nielsen & Molich’s heuristics for comparison for the second evaluation. For further analysis, we compared results of our laboratory-based user test in which users solved challenges on these same captcha schemes (Section 3.3) to the results obtained from the MC heuristic evaluation. The latter evaluation gave us the chance to investigate what types of usability problems are encountered by real users that are missed by evaluators and vice versa.

### 4.5.1 Target Schemes

The schemes used in the evaluation were the same schemes as in Section 3.3.1. We include a brief description of the schemes, for details refer to that section. As reminder, the target schemes are depicted in Figure 4.1.

- *reCaptcha* [56] is a widely deployed CR scheme.
- *Asirra* [92] is a research IR captcha<sup>9</sup>.
- *NuCaptcha* [110] is a commercial MIOR scheme, moving CR challenge.
- *Animated* captcha, (Vappic 4D) [142], is an experimental MIOR captcha scheme.



Figure 4.1: MC Heuristics. Target Schemes.

## 4.6 Evaluation 1: Heuristic Evaluation - the MC Heuristics

A heuristic evaluation (HE) was done by experts using the MC heuristics (Section 4.4) to examine the four captcha schemes. Our evaluation method followed Nielsen’s recommendations [102, 103] with a few modifications, described below. Since captchas are not feature rich programs, our HE design allowed evaluators to create their own task scenarios using the questions listed in Section 4.4 as a guide. The general task was to solve captcha challenges and explore the overall interface of each of the target schemes.

*Evaluators.* To recruit evaluators, we e-mailed experts we knew through previous interactions in HCI and Security. All the experts we contacted agreed to participate

<sup>9</sup>The service as research project was discontinued in late 2014

in the evaluation. Once the evaluators agreed, we e-mailed details on how to conduct the heuristic evaluation. Nine evaluators (4 females, 5 males) completed an evaluation of at least one scheme. Evaluators rated their knowledge of security and HCI, their means were 4.1 and 4.2 respectively (out of 5,  $N = 9$ , 1 = low). Evaluators included people in the mobile industry, academic researchers with proven expertise in the fields, students with at least one graduate level course in security and HCI.

*Equipment:* Participants completed the evaluations on their own smartphones and in the environment of their choosing. There was one Nexus S, one Galaxy Nexus, 2 iPhones 4, 4 iPhones 4S, and one Samsung Focus (SGH-i917). While evaluating a scheme, evaluators used a secondary device to compile feedback from the heuristics.

*Procedure:* Prior to the evaluation, evaluators responded to a demographics questionnaire. A host web page presented links to the four different target captchas hosted on third party demo sites. Thus experts evaluated and experienced live versions of the captcha schemes. Each evaluator conducted his/her assessment independently. For each scheme visited on the evaluator's smartphone, they assessed its merits based on the heuristics, noted in Limesurvey<sup>10</sup> any problems uncovered, rated each problem's severity, and provided an overall score based on compliance with the heuristic.

#### 4.6.1 Data Analysis

We summarized the usability problems identified by each evaluator and then generated an aggregate list of problems per scheme.

We used a variant of Thematic Analysis [17] to process, synthesize and categorize the reported problems. The categorization was conducted by two researchers: the thesis author plus another researcher. The steps were:

1) Problem synthesis (*open coding*). We separated and synthesized the *raw problems*. Raw problems include problem descriptions with compound issues which can be decomposed with better granularity. Compound issues are those that include or refer to more than one action, or where more than one problem is described, or where ambiguous problem descriptions can be better synthesized.

---

<sup>10</sup>LimeSurvey web based survey tool <http://www.limesurvey.org/>

2) Consolidate problems (*axial coding* and *selective coding*). We consolidated overlapping problems and identified *false alarms*. Consolidation started with an empty list, then a problem was added to the list if it did not yet exist in the list. Otherwise a problem-frequency counter was updated. In heuristic evaluations, it is not unusual for two evaluators to write two completely different descriptions for the same problem. For our analysis, we define *overlapping* problems as those identified by two or more evaluators. A *false alarm* was identified as a problem that “was not verified by any of the researchers” [66], or when “the reasoning of the evaluator in describing the problem was fallacious” [68]. The result is a list of *unique* problems found for each scheme.

3) Tagging the problems with heuristics. Two researchers reviewed each unique problem by verifying whether it fits in the assigned heuristic. Sometimes evaluators identified problems under a particular heuristic, but these problems fit better under a different heuristic; we refer to these as *misclassifications*. These misclassifications were re-tagged. Re-tagging was done by consensus (of the two researchers). We categorized the unique problems into major (2) and minor (1). *Major* problems were those that would significantly hinder usability, prevent the user from solving the challenge, or lead to mistakes. The original rating by evaluators was also considered when classifying.

All evaluators were asked to review the four schemes, but not every evaluator completed all schemes. Of the nine evaluators, eight evaluators completed reCaptcha, while six assessed Asirra, NuCaptcha and Animated. We asked them to spend one hour in total, but the evaluation took longer in some cases. Evaluators could take a rest break when tired, and we preferred that they take a rest break than give superficial evaluations.

#### 4.6.2 Results: Raw Problems

We analyzed raw problems to assess the evaluators’ consistency in finding problems and assigning severity ratings. The total number of raw problems per scheme across heuristics is as follow: reCaptcha (86), Asirra (60), NuCaptcha (31), Animated (79). Evaluators found problems relating to *MC3: Efficiency* and *MC4: Input Mechanisms*

most severe, and those that are related to *MC7: User Location* least likely to significantly impact usability. This is likely because most of the four schemes evaluated require zooming and panning to be able to fully see and answer the challenge thus affecting the efficiency of the captcha. Evaluators also found that restrictions on input mechanisms considerably hinder the usability of the evaluated captchas. The two heuristics that resulted in the most problems identified are *MC1: User control* and *MC3: Efficiency of use*, with 53 and 43 problems described across the four schemes. *MC7: User's localization* and *MC2: Learnability* led to the least discovery, with 31 and 23 problems described respectively. Based on severity ratings for *MC5: Solvability*, Asirra appeared most solvable (in the opinion of the evaluators) and Animated resulted in the most critical solvability problems. Figure 4.2 depicts the mean severity ratings for all raw problems reported. Lines are included for readability; data is not continuous. The gaps for NuCaptcha are because no problems were reported for that heuristic; where 1 = critical and 5 = minor.

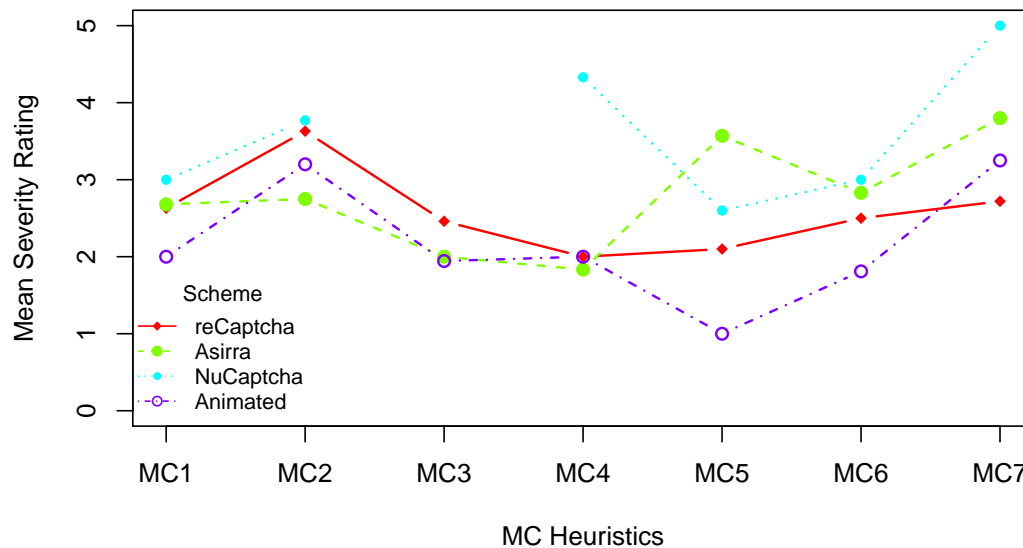


Figure 4.2: MC Heuristics. Mean severity ratings for all raw problem reports.

Scheme	Synthesized Unique Problems							Total	Evaluators
	MC1	MC2	MC3	MC4	MC5	MC6	MC7		
reCaptcha	6	9	8	6	8	3	3	43	8
Asirra	11	4	5	3	5	6	3	37	6
NuCaptcha	3	5	0	3	2	4	1	18	6
Animated	5	4	7	3	6	10	4	39	6
Total	25	22	20	15	21	23	11	137	

Table 4.3: MC Heuristics. Unique problems found by heuristic per scheme.

### 4.6.3 Results: Unique Problems.

The number of unique problems are more representative of real issues than raw problems, thus for the purpose of evaluating the heuristics, better conclusions can be obtained from this refined set of unique problems.

Table 4.3 summarizes the number of unique problems per heuristic. We remind the reader that our goal is to determine the usefulness of the heuristics, not scientifically compare the exact schemes. We provide these comparisons as complementary data. Fewer unique problems indicate that the scheme performed better than other schemes on that heuristic. The heuristics with the most unique problems were *MC1: User control* and *MC2: Learnability*, with 25 and 22, respectively. Fewer unique problems were discovered by *MC4: Input mechanisms* and *MC7: Localization*, with 15 and 17, respectively across all schemes. reCaptcha had more unique problems for *MC5: Solvability* than other schemes, however its mean severity rating for *MC5: Solvability* was 2.1 indicating a large number of relatively minor issues. This highlights the importance of including severity ratings as an evaluation measure. The number of unique problems for *MC3: Efficiency* are quite variable; reCaptcha gathered the most efficiency problems while NuCaptcha had none.

#### 4.6.4 Results: Evaluator Performance

Regardless of the heuristics, evaluators perform differently from each other. HCI background, expertise in the domain(s), time, incentives, and heuristic evaluation experience are some of the factors that influence their performance.

The distribution of unique problems across evaluators for each scheme is depicted in Figures 4.3(a) to 4.3(d). In addition to using Nielsen’s original graphing depiction [102, 109], our graphs use colours to show the severity rating assigned by evaluators. Each row represents an evaluator, each column represents a unique problem. Evaluators are presented in increasing order from bottom-to-top based on the number of unique problems identified. Stronger evaluators are at the bottom of the graph since they found the most problems. Each graph contains results for Nielsen’s heuristics above the black bar and the MC heuristics below the black bar; they should be interpreted separately. (This, and other figures, are best viewed in color.) The definition of strong-vs-weak evaluators is related to the evaluator’s ability to find problems. For readability purposes strong and weak labels are only depicted once per scheme. However, each set of heuristics depicts strong at the bottom and weak at the top of graph. The problems are also ordered, from left (easy) to right (hard). An *easy* problem is typically defined as a problem that is commonly found by multiple evaluators; it is an “easy” problem to identify, versus a “hard” problem that is rarely found. The characterization of easy-vs-hard problem and strong-vs-weak evaluator are common interpretations introduced by Nielsen and Molich [109].

We noticed that some easy problems are overlooked by strong evaluators, while some hard problems are only found by strong evaluators. This finding supports Nielsen’s argument that heuristic evaluation should be conducted with multiple evaluators, since no one evaluator will uncover all problems. In addition, the distribution of unique problems shown by the schemes is consistent with that of Nielsen’s and existing literature [68, 109]. The colouring of the graphs helps to visually identify the overall severity of problems per scheme. For example, the Animated scheme may have fewer unique problems (39) compared to reCaptcha (43), however the darker colours of the graph shows that evaluators found more severe problems for Animated. The colours also highlight the degree to which evaluators agree, or disagree, on the



severity of any given problem. It may also show patterns within each expert’s own rating of problems. That is, does expert *A* assign consistently high or low severity ratings to the problems they report?

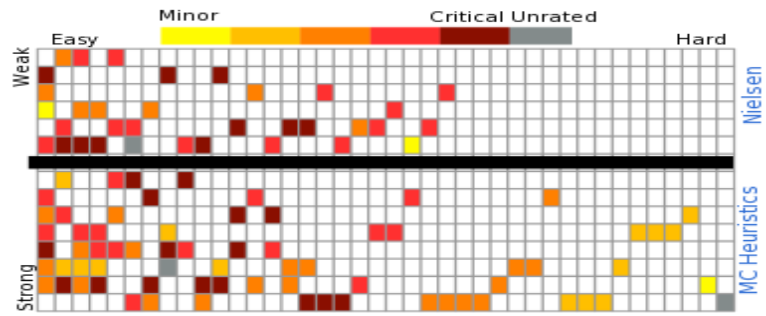
Within each scheme, evaluators tend to assign a higher severity to the ‘easiest’ problems identified, indicating that the heuristics are helpful in guiding evaluators at uncovering more severe problems.

#### 4.6.5 Results: Overall Rating per Heuristic

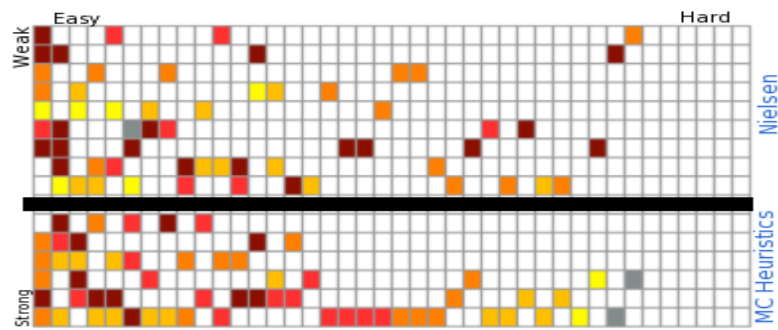
Evaluators provided an overall rating of each scheme per heuristic. The rating consisted of a Likert scale ranging from 1 (significant problems) to 10 (excellent). These ratings provided us with an overall perspective of the problems discovered for each of the heuristics and their severity (Figure 4.4). They further provide supporting evidence for the analysis of the unique problems.

*MC1: User control and freedom* helped evaluators identify significant issues with most schemes. For example, an evaluator mentioned: “Buttons on the side of the input box are very small to use on the phone” (reCaptcha). Heuristic *MC2: Learnability* uncovered less significant issues and comments support it: “I thought the instructions were really simple and understandable and the other features (the zoom) became very obvious as you used the system” (Asirra). Although three schemes used the virtual keyboard as input mechanism, not all schemes scored equally; *MC4: Input mechanisms* highlighted these differences. An evaluator expressed “When I type, I’m still able to see the Captcha, most of the time” (NuCaptcha). As well *MC6: Perception* underlined the importance of testing innovative captchas, one evaluator noted “these captchas made me nauseous while looking at them” (Animated).

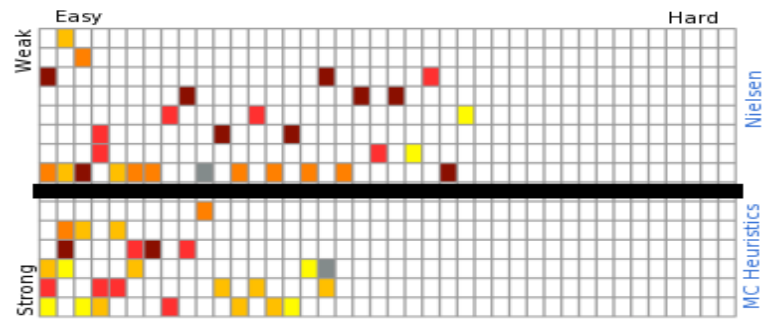
**MC HE Summary.** Based on the number of unique problems, severity ratings assigned and the overall performance ratings, the heuristics appear useful at identifying domain specific problems that may impact usability. As might be expected, the captchas performed similarly on some heuristics. However, other heuristics (*e.g.*, MC5) helped to identify significant differences in usability between schemes.



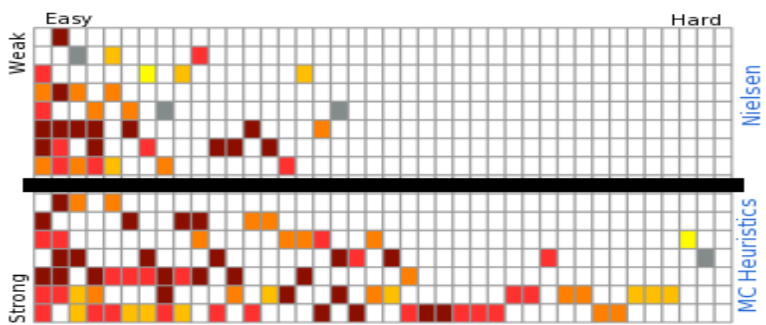
(a) reCaptcha Unique Problems



(b) Asirra Unique Problems



(c) NuCaptcha Unique Problems



(d) Animated Unique Problems

Figure 4.3: MC Heuristics. Unique problems per captcha scheme.

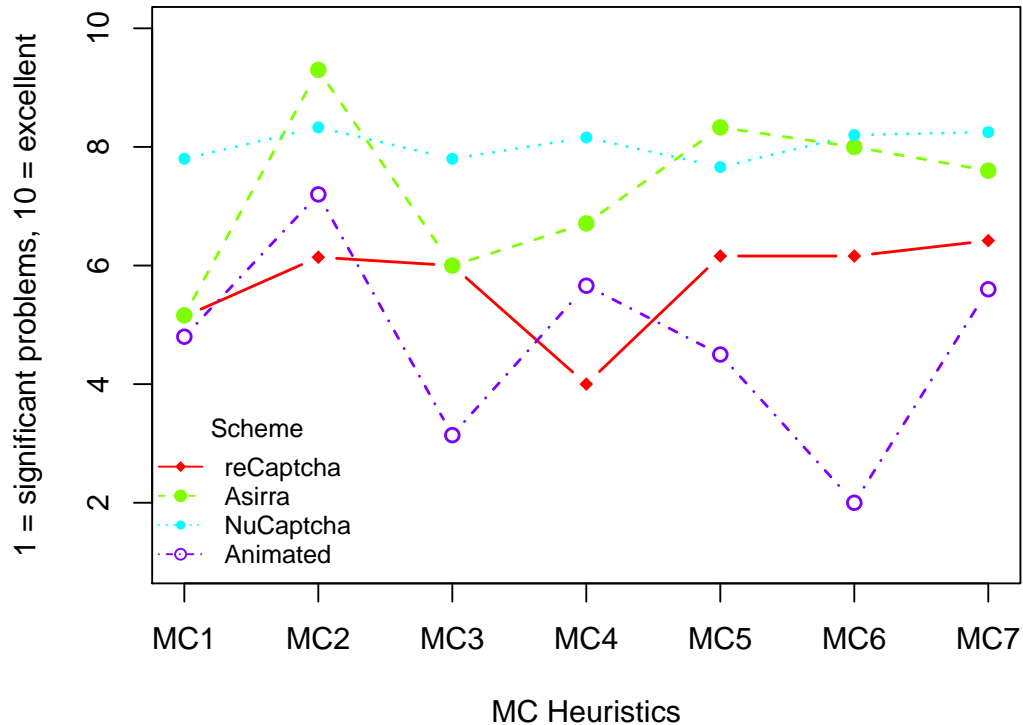


Figure 4.4: MC Heuristics. Mean Likert scale responses for overall performance ratings.

#### 4.7 Evaluation 2: Heuristic Evaluation - Nielsen and Molich Heuristics

A HE was carried out by asking evaluators to use Nielsen and Molich’s ten heuristics [103, 109]. The same methodology as for the MC heuristics was followed (Section 4.6).

*Evaluators.* To recruit evaluators, we e-mailed students who had finished at least one graduate level HCI course. We also recruited people from the LinkedIn ACM SIG CHI - Special Interest Group on Computer-Human Interaction. We limited participation to nine evaluators (3 females, 6 males). Evaluators rated their knowledge on security and HCI, their means were 3.44 in both (out of 5,  $N = 9$ , 1 = lowest). Volunteers were paid a CND\$15 Amazon gift certificate.

*Equipment.* Participants completed the evaluation on their own smartphones and in the environment of their choosing. There was one Nexus 5, one Samsung Nexus, two LG Nexus 4, one iPhone 5S, one iPod, one Windows Phone, and two Galaxy S III. While evaluating a scheme, evaluators used a secondary device to compile feedback about the heuristics.

*Procedure.* We followed the same procedure as described in Section 4.6.

#### 4.7.1 Data Analysis

The same procedure as in Section 4.6.1 was followed to perform the analysis. The categorization was also conducted by two researchers: the thesis author plus another researcher. We provide brief results here; detailed performance comparisons between MC heuristics and Nielsen’s are described in Section 4.8.

#### 4.7.2 Results: Raw Problems

The number of raw problems per scheme across heuristics is as follows: reCaptcha (36), Asirra (73), NuCaptcha (37), Animated (46). Similarly to MC heuristics, evaluators tended to describe most of the problems they encountered in the first heuristic *Visibility of System Status*: 44; we speculate evaluators used this heuristic as starting point and noted problems here even if it was not the best match. *Error Prevention* also promoted a high number of problem descriptions: 37. Figure 4.5 shows the mean severity ratings assigned by evaluators to all raw problems. Ratings are 1 = critical and 5 = minor, and the gaps shown are due to lack of severity ratings by evaluators.

#### 4.7.3 Results: Unique Problems.

Nielsen and Molich’s heuristics with most unique problems were *Help and documentation* and *Visibility of System Status*, with 21 and 10 respectively. Fewer unique problems were classified in *Aesthetic and minimalist design*, *Consistency and standards*, *Help users recognize diagnose, and recover from errors* with 4, 3 and 3 respectively. The number of false positives was: reCaptcha (6), Asirra (13), NuCaptcha (10), Animated (5).

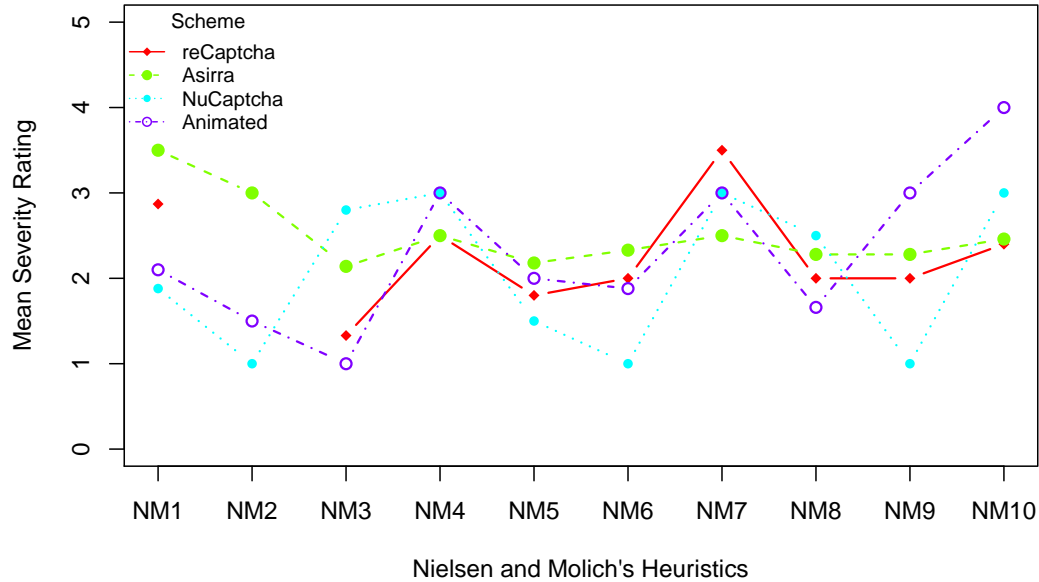


Figure 4.5: MC Heuristics. Nielsen & Molich’s Heuristics. Mean severity ratings for all raw problem reports.

#### 4.7.4 Results: Evaluator Performance

The distribution of unique problems across evaluators for each scheme is depicted in Figures 4.3(a) to 4.3(d). We see that using Nielsen and Molich’s heuristics the unique problems uncovered by the evaluators are more scattered (*i.e.*, have less agreement between evaluators) than the unique problems using our MC heuristics. This distribution may be due to the lack of specificity of Nielsen and Molich’s heuristics for this domain.

#### 4.8 Effectiveness of the MC Heuristics

Table 4.4 shows the classification of the problems from the MC and Nielsen’s HE. Naturally there are similarities between the two set of heuristics since our set has heuristics derived from Nielsen’s. However, MC heuristics facilitated uncovering more unique problems than Nielsen’s, despite the fact that Nielsen’s HE had more evaluators finish the evaluations. This suggests that our set of heuristics does not require

Scheme	Unique		False Positives		Major		Minor		Raw		Evaluators	
	MC	N	MC	N	MC	N	MC	N	MC	N	MC	N
reCaptcha	43	26	5	6	26	9	17	17	86	43	8	8
Asirra	37	32	3	13	14	14	23	18	60	82	6	9
NuCaptcha	18	27	3	10	6	10	12	17	31	46	6	9
Animated	39	18	2	5	24	7	15	11	79	52	6	8
All	137	103	13	34	70	40	67	63	256	223		

Table 4.4: Heuristic Evaluations. Problem classification.

a large number of evaluators to uncover a similar number of problems as Nielsen’s heuristics. The number of *false positives* was lower with MC heuristics and more major problems were identified. We attribute this to the lack of domain specificity of Nielsen’s heuristics and the customized nature of the our MC heuristics for this domain. Regarding the number of evaluators, we intended the evaluation of the four schemes to be independent from each other. Nielsen and Molich [109] specify that 3 to 5 evaluators is the recommended number for conducting HEs; we have more than the maximum number for each of our example evaluations.

It is desirable for heuristic evaluations to be *thorough*, *valid*, and *consistent* (for reliability) [63]. We compared the results of our HE and user study along these metrics. The norm is to visually compare these metrics [31, 63]. The ultimate criterion to assess usability evaluation methods is finding real usability problems (Section 4.5). We define the set of *real problems that exist* as the collection of unique problems derived from the MC HE and the user study. To test the validity and effectiveness of the MC heuristics we explore the following question: How well do the proposed heuristics help evaluators discover real problems? Table 4.5 clarifies the origin of the numbers used in the equations below.

#### 4.8.1 Comparison between MC and Nielsen

**Thoroughness:** The heuristics should find as many usability problems as possible. Thoroughness (T) indicates the proportion of problems found by the HE to the real problems in the scheme [31, 63], Equation 4.1:  $T_s^h$ . Where subscript  $s$  represents the scheme, and superscript  $h$  represents the heuristic employed:  $C$  for Mobile Captcha heuristics or  $N$  for Nielsen and Molich’s.

Scheme	MC Heuristics				Nielsen’s Heuristics			
	Only MC	Match	Only User Study	Total number of problems	Only Nielsen’s	Match	Only User Study	Total number of problems
reCaptcha	32	11	11	54	16	10	12	38
Asirra	30	7	15	52	20	12	10	42
NuCaptcha	18	3	9	30	20	7	5	32
Animated	30	9	14	53	10	8	15	33

Table 4.5: Unique problems for both HEs and user study. Details regarding the User Study can be found in Section 3.3.

$$\text{Thoroughness} = \frac{\text{Number of real problems identified}}{\text{Number of real problems that exist}} \quad (4.1)$$

$$\begin{aligned}
T_{reCaptcha}^{MC} &= \frac{32 + 11}{32 + 11 + 11} = 0.796 & T_{reCaptcha}^N &= \frac{26}{38} = 0.684 \\
T_{Asirra}^{MC} &= \frac{37}{52} = 0.711 & T_{Asirra}^N &= \frac{32}{42} = 0.761 \\
T_{NuCaptcha}^{MC} &= \frac{21}{30} = 0.700 & T_{NuCaptcha}^N &= \frac{27}{32} = 0.843 \\
T_{Animated}^{MC} &= \frac{39}{53} = 0.735 & T_{Animated}^N &= \frac{18}{33} = 0.545
\end{aligned}$$

Thoroughness ranges from 0 to 1, with higher values indicating more thoroughness. The MC heuristics score over 70% on all schemes. Other published domain specific heuristics [31, 68] have shown 71% and 55% thoroughness, respectively. We note that MC heuristics are consistently thorough across the evaluated schemes. Nielsen’s thoroughness values are similar but have considerable variability between schemes.

Validity: Validity (V) is the measure on how well the heuristics do in what it is intended to do; in other words, the proportion of problems found by the MC heuristics that are real usability problems (Equation 4.2) [31, 63]. The *Number of issues identified as problems* includes the number of *False positives* identified by the researchers and the *Number of real problems identified* (Table 4.4).

$$\text{Validity} = \frac{\text{Number of real problems identified}}{\text{Number of issues identified as problems}} \quad (4.2)$$

$$\begin{aligned}
V_{reCaptcha}^{MC} &= \frac{32 + 11}{32 + 11 + 5} = 0.895 & V_{reCaptcha}^N &= \frac{26}{32} = 0.813 \\
V_{Asirra}^{MC} &= \frac{37}{40} = 0.925 & V_{Asirra}^N &= \frac{32}{45} = 0.711 \\
V_{NuCaptcha}^{MC} &= \frac{21}{23} = 0.913 & V_{NuCaptcha}^N &= \frac{27}{37} = 0.730 \\
V_{Animated}^{MC} &= \frac{39}{41} = 0.951 & V_{Animated}^N &= \frac{18}{23} = 0.783
\end{aligned}$$

Validity ranges from 0 to 1. The MC heuristics perform consistently across the four schemes with a high number of real problems identified relative to the small number of false positives yields high validity. Nielsen's HE showed larger number of false positives. High Validity scores indicate that most of the problems are real, thus higher values are better. In comparison, Chattratchart and Linaard [31] obtained a validity of 65% for their proposed effectiveness evaluation method.

Reliability: It is important that heuristics help evaluators produce consistent results independently of the individual performing the heuristic evaluation [63]. In addition, it is desirable to identify major usability issues as they may seriously hinder the ability of the user to operate the scheme effectively and efficiently [68]. Reliability (R) can be measured by the mean number of the evaluators finding a problem (Equation 4.3) [31]. Our reliability scores are similar to the 0.178 reported by Chattratchart and Linaard [31]. Even though Nielsen and Moich's HE had more evaluators, its reliability ratings were lower than our set of heuristics. Lower values indicate better performance.

$$\text{Reliability} = \frac{\text{Number of evaluators}}{\text{Number of real problems identified}} \quad (4.3)$$

$$\begin{aligned}
R_{reCap}^{MC} &= \frac{8}{43} = 0.186 & R_{reCap}^N &= \frac{8}{26} = 0.307 \\
R_{Asirra}^{MC} &= \frac{6}{37} = 0.162 & R_{Asirra}^N &= \frac{9}{32} = 0.281 \\
R_{NuCap}^{MC} &= \frac{6}{21} = 0.285 & R_{NuCap}^N &= \frac{9}{27} = 0.333 \\
R_{Animated}^{MC} &= \frac{6}{39} = 0.153 & R_{Animated}^N &= \frac{8}{18} = 0.444
\end{aligned}$$



**Effectiveness Summary.** The *Thoroughness*, *Validity*, and *Reliability* results align with published work using the same metrics to determine the effectiveness of heuristics sets. Typically new sets of heuristics are validated against one piece of software. We validated against four applications (captcha schemes). The resulting outcomes for the different metrics show consistency across the evaluated schemes. We hypothesize that our heuristics will perform consistently when used on other schemes.

#### 4.8.2 Comparison between MC and User Study

To complement the heuristics validation, a user study was also conducted on the same schemes as in the heuristic evaluations. Details of this user study are given in Section 3.3. We compared the findings between the results from the MC heuristic evaluation and the user study. The user study provided insight into users' preferences and perception of the schemes. The MC heuristics gave us more detailed feedback on the problems found. A large number of issues uncovered by the MC HE and the user study were similar and confirmed each other even though they may have been expressed differently. For example, for Asirra one evaluator wrote: "zooming tool is annoying because it highlights everything. Easier to use the phone's zoom function". Comments and observations from the user study expressed: "participants were using the browser's zoom to see the images rather than using the scheme's enlarged image (zoomed)". Figure 4.6 highlights the overall effectiveness of the MC heuristics in finding problems compared to the user study. Overlapping problems identified in both evaluations are depicted in red. This visual representation summarizes the performance of schemes according to specific heuristics.

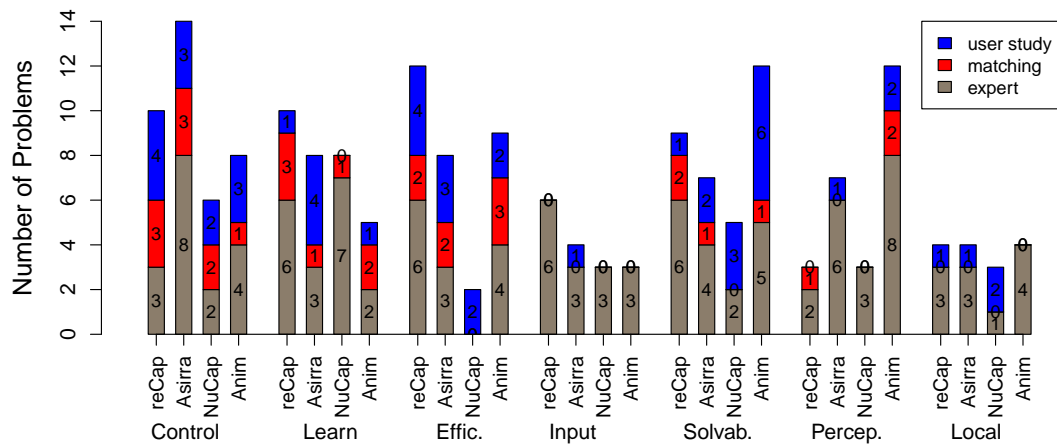


Figure 4.6: Unique problems from the user study and the MC HE.

#### 4.9 Discussion

Results show that the MC heuristics performed well in finding usability problems. Compared to the user study (Section 3.3), MC heuristics found more problems. We believe that the reason is that the heuristics guide and lead evaluators to discover problems. Despite user study participants being told to explore the schemes, they focused on the task at hand: solving challenges. Naturally, Nielsen’s heuristics also discovered more problems than the user study but fewer than the MC heuristics. As suggested by Nielsen [108], domain specific heuristics perform better at uncovering problems than their general heuristics. We found that a HE is more effective at uncovering usability problems quickly and cost effectively than a user study when evaluating non-complex software such as captchas (Tables 4.5, 4.4).

The overlap between unique problems found by MC evaluators and those reported by users is 20% for reCaptcha, 13% for Asirra, 10% for NuCaptcha, and 17% for Animated. This overlap is consistent with work done by Petrie and Power (14%) [113]. These percentages are relatively low, thus there are problems highlighted by the user study that the MC HE does not explicitly uncover, and vice versa. The overlap between unique problems found by Nielsen’s HE and those reported by the user study is 26% for reCaptcha, 28% for Asirra, 21% for NuCaptcha, and 24% for Animated. There is a greater overlap for Nielsen’s HE than the MC HE evaluation. Highlighting

the differences between domain specific heuristics versus general heuristics.

By reviewing the identified problems and analyzing the feedback of the user study, we realized that additional problems could align with the MC heuristics despite there were not exact matches. For example, the user study identified problems such as “when correcting typing errors, the participant pressed submit by mistake” which aligns with the “small control buttons” problem identified by evaluators. This would increase the overlap between the MC HE and the user study. In many cases, the MC HE identified a generalized version of a problem while the user study provided an instance of the issue.

The differences between schemes in the MC HE severity ratings, unique problems and overall performance also indicate that the heuristics performed well. Identical results for all evaluated schemes would have indicated flaws with the heuristics since the schemes are representatives of different captcha categories, each with different usability strengths and weaknesses.

The participants of the user study were not asked to assign a severity rating for the problems they came across, thus we cannot do a direct comparison between the MC HE and the user study. However, the results of the perception questionnaires coincide with the results of the MC HE. For example, the user study rated NuCaptcha as the “easiest to understand” closely followed by Asirra; the latter had the best rating for the Learnability heuristic. The user study indicated Animated as the most error-prone scheme, and the MC HE showed that Animated performed worst for the Solvability heuristic.

Several directions could be explored in future work. Additional analysis looking at the evaluators’ background may be useful in some circumstances; for example, when MC HE evaluators do not share a homogenous background. Exploration based on evaluator backgrounds could include similarities or differences on the severity of the problems, the types of issues reported, and the likelihood of finding common problems or discovering uncommonly reported problems.

The set of heuristics was used by people with well founded knowledge of HCI and security. Our future work includes further ecological validation. We plan to ask non-experts such as captcha developers, web administrators, web content managers,

and any person responsible for ensuring human-form interaction on a web site to use the MC heuristics in an evaluation. Heuristics taking into account audio schemes could be added to the present set of heuristics. The MC heuristics can be used to evaluate captchas developed for desktops, however we think that they could be further optimized for this context.

Evaluating the security of any captcha scheme is a non trivial task which requires significant expertise. The approach to “breaking” a scheme varies according to the captcha class. Assessing the security of character recognition captchas requires a different set of tools than an image recognition captcha. A heuristic evaluation cannot be designed to evaluate the security of any piece of software. Thus the MC heuristics are meant to be used in parallel with a security analysis.

The aim of this chapter was to propose and validate a set of heuristics for evaluating captcha schemes on smartphones. In order to validate the heuristics, we conducted two heuristic evaluations and used the results of our user study (Section 3.3). The MC heuristics can be used as a discount method to uncover usability problems in captchas meant to be used in web sites targeting mobile devices. Furthermore, the heuristics can be used at any stage of the design of new captcha schemes for mobile devices.

Developing the MC heuristics gave us the basis for the design and evaluation of new captcha prototypes which aim at addressing the usability issues uncovered by the studies presented in Chapter 3 and the MC heuristic evaluation. The captcha prototypes are presented in Chapter 5. We have conducted further analysis of these results and elaborated a series of design recommendations for captcha aimed at smartphones; these are presented in Chapter 6.

## Chapter 5

### Smartphone Captcha Prototypes

#### 5.1 Introduction

In Chapter 3, we have shown evidence that users interacting with captchas on smartphones experience significant usability issues. Even though virtual mobile keyboards have improved and several types exist, our empirical data shows that current keyboards continue to be a cause of errors and frustrations for users. Swapping between numeric and alphabetic keyboards is a source of mistakes while solving captcha challenges, for example [120, 121, 151].

For readability, we briefly review some existing captcha schemes intended for mobile devices; for a more complete overview refer to Section 2. Wismer *et al.* explore voice and touch input of existing captchas on mobile devices [148]. Their evaluation focuses on voice and touch input using Apple’s iPad and they found significant usability problems.

Gossweiler *et al.* [59] present a IR captcha scheme that could be adapted for mobile usage. Their scheme consists of rotating an image to its upright / natural position with a slider (Figure 2.7(b)). They suggest that the mobile version would allow direct image rotation with finger gestures.

Specific to mobile devices, Chow *et al.* [36] introduce the idea of presenting several CR captchas in a grid of clickable captchas (Figure 2.7(a)). The answer is input by using the phone’s (NOKIA 5200) keypad and selecting the grid elements which satisfy the challenge. For example, the user may have to identify in the grid a subset of captchas with embedded words, as opposed to random strings. Since the answer consists of selection by “clicking”, via the keypad, rather than typing, this scheme could be used on touch screen mobile devices. Despite showing possible benefits, this captcha scheme has not been made public nor implemented.

Shirali-Shahreza *et al.* [130] explore speech as an input mechanism for both CR challenges and AUD challenges on smartphones. Lin *et al.* [81] propose two captcha schemes. The first is an IR scheme called “captcha zoo” (Figure 2.7(c)). It requires users to choose certain 3D target animals from a set of containing two types of animals. For example, displaying dogs and horses, and the user clicks on the horses. This approach is similar to Asirra (Figure 5.1(c)). The second scheme is a four-character CR challenge with a custom eight-icon keyboard displaying characters from the challenge plus four additional characters.

On the commercial side, NuCaptcha offers a mobile version of their MIOR desktop captcha [110]. Solve Media [89] also offers a mobile version of their CR captcha scheme.

In this chapter, we investigate our own customizations to existing desktop captcha schemes; these customizations are intended for smartphone usage. We compare the usability of these customizations to the original designs. In total, we compare the usability of nine captcha schemes on smartphones. The evaluated captchas include prototype adaptations and implementations of existing schemes which offer an API, commercial captchas, and academic captchas. The security of some of these schemes has been explored in previous research [21, 54, 151]. Parts of this work was published at the 2015 USEC workshop [121].

In this chapter we employ the Wizard of Oz (WoZ, explained in Section 5.3) prototyping technique for some of the tested schemes. In our user study (Sections 5.4), the researcher unobtrusively observed the user’s drawn characters in real-time, and typed the character on the server. A JavaScript script relayed it to the user’s device and displayed it in the answer input field. The user would either verify the character and continue drawing characters, or would delete the character from the input field and draw again.

In Section 5.3, we describe our captcha prototypes and the existing schemes used in the evaluation. The details of the user study and our evaluation methodology are described in Section 5.4. Our results are presented in Section 5.5. A brief discussion is presented in Section 5.6. The conclusion is presented in Section 5.7.

## 5.2 Existing Schemes

For comparison in our user study, we chose five existing captcha schemes, representing each of the main captcha categories: CR, IR, and MIOR. We also included two mobile-friendly schemes. The target schemes are summarized below and depicted in Figure 5.1; more details on the schemes can be found in our Background (Chapter 2).

- (S1) *reCaptcha* [56] is widely deployed on the Internet; this CR challenge consists of recognizing and typing two words or numbers. The challenge for this scheme was updated during the summer of 2014; the recent captcha challenges consist of either interpreting and typing two distorted words or reading images of house numbers without any distortions or distractors. Most of the challenges randomly shown to participants consisted of the latter.
- (S2) *NuCaptcha* [110] is a commercial MIOR scheme consisting of either reading alphanumeric characters that overlap as they swing independently left to right (statically pinned at the centre of each letter), or reading a uniquely coloured code word in a phrase that loops endlessly in the captcha window.
- (S3) *Emerging* [151] is an academic proposal which addresses security flaws found in NuCaptcha. The challenge consists of recognizing three alphanumeric characters. The three characters move on a wave across a canvas in an endless loop. Both the challenge characters and the background consist of moving black and white pixels. The movement renders visible the characters, but observing any one frame does not reveal the characters from among the noise.
- (S4) *Asirra* [92] is a research IR captcha that was available<sup>1</sup> for deployment; the challenge consisted of selecting images of cats from a grid of 12 images containing dogs and cats. The image database comes from a pet adoption service called Petfinder [112]. These images are pre-labeled by the person uploading each pet's image.

---

<sup>1</sup>The Asirra captcha service was closed permanently in October 2014.

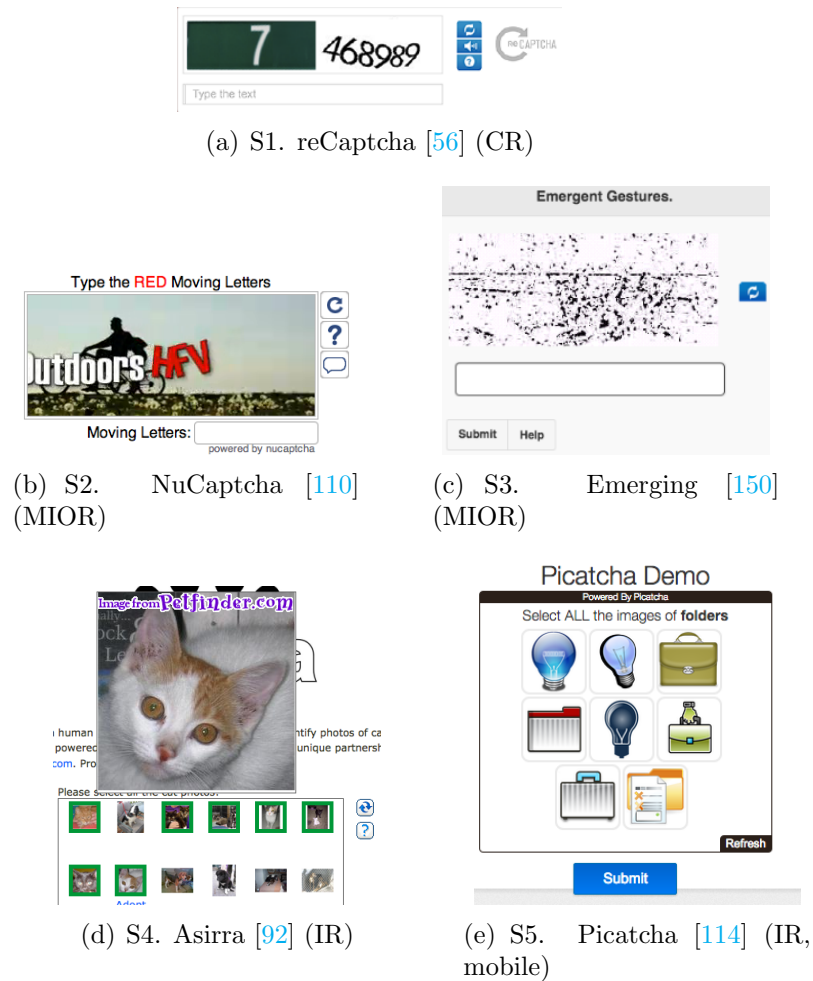


Figure 5.1: Screenshots of the five pre-existing evaluated captcha schemes.

(S5) *Picatcha* [114] is a commercial mobile-friendly IR captcha. This captcha shows a grid of eight images from which the user has to select a specific subset. For example, a user might be asked to select icons of horses, monitors, or hands. The number of correct target images varies from challenge to challenge.

### 5.3 Proposed New Input Mechanisms

As discussed in Chapter 3, captchas on smartphones (and on desktops) have well defined usability limitations. Larger screens and new keyboard layouts have not approached the input ease of full a keyboard, mouse and display. We designed mobile adaptations of existing schemes with the intention of simplifying the user interaction



on mobile devices. Our goal was to explore and evaluate whether adapting input mechanisms to solve captcha challenges helps usability on smartphones.

We chose existing schemes for which their security has been previously explored and tested. We selected five existing schemes representing the main captcha categories: CR, IR, and MIOR. For the CR and MIOR schemes, we eliminated the use of the standard keyboard as an input mechanism and incorporated touch gestures where possible. We define as gesture-based input the action of character-drawing by the user with the finger on the designated screen area (canvas). For the IR scheme, we arranged the images as a carousel rather than in a grid. The prototype descriptions and the rationale behind their design are described in the following section.

Wizard of Oz is a prototyping technique used in user studies. In this technique, the participant interacts with a medium-fidelity prototype as though interacting with the finished product and a human operator simulates the software's response to the user [127].

### 5.3.1 Mobile Prototype Design

The general design goals behind our modifications were to:

- Provide an alternative input mechanism to the virtual keyboard
- Minimize the occlusion of the challenge
- Reduce or eliminate zooming and panning
- Provide adequate captcha control mechanisms

### 5.3.2 Proposal 1 (P1) - Gesture reCaptcha

This scheme is a CR captcha which employs reCaptcha's API [56] as the source of its challenges. reCaptcha is a widely known and widely deployed captcha. The challenge image is displayed without modification, but the input mechanism is altered. In our prototype, a drawing canvas is included beneath the challenge. From the design perspective, the objective of this prototype was to replace typing on the virtual keyboard

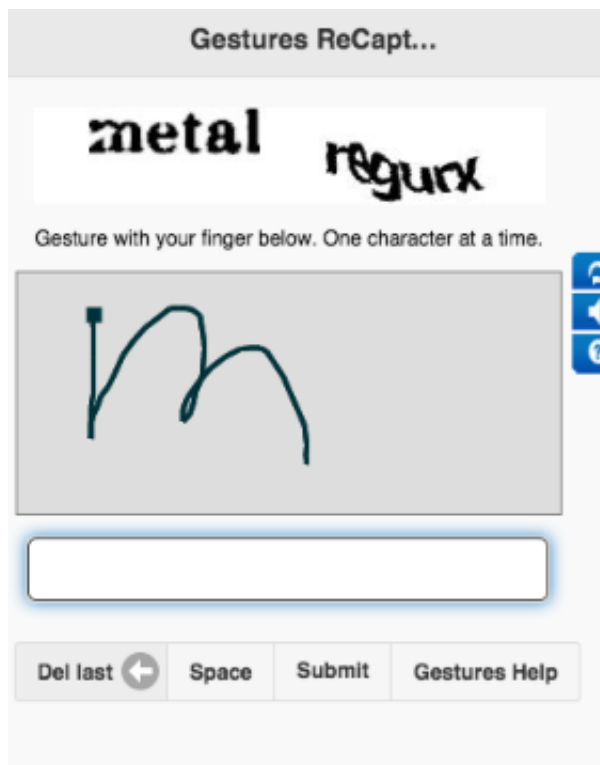


Figure 5.2: P1. Gesture reCaptcha.

as the input mechanism with characters drawn with gestures. We chose drawing gestures with the intention to eliminate the “fat finger problem” (accidentally touching the wrong field or button); gesturing also avoids swapping between alphabetic and numeric virtual keyboards. Figure 5.2 depicts our Gesture reCaptcha prototype and Figure 5.3 depicts the workflow of this prototype.

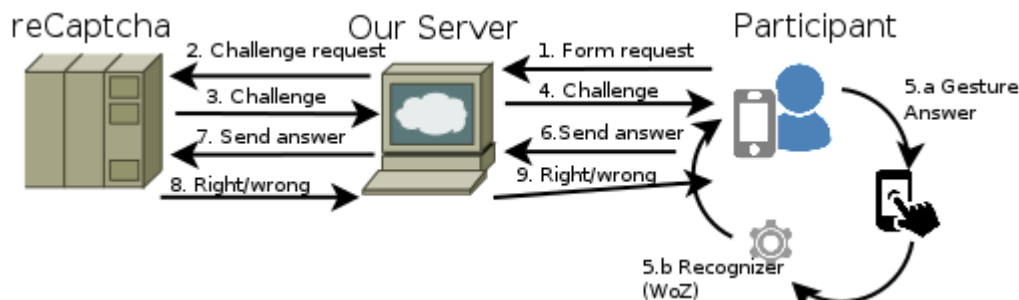


Figure 5.3: P1. Gesture reCaptcha’s message and activity flow.

1. The participant on her smartphone browser requests the page hosting the scheme from our server
2. Our server sends a request for a challenge to reCaptcha's servers, along with our public key
3. reCaptcha's servers return a challenge to our server
4. This challenge is sent to the participant's browser
5. The participant answers the challenge or requests a new challenge
  - (a) The participant gestures (draws in the canvas) each character one at a time; completion of a character is recognized by a mouse-up event (*i.e.*, finger-up)
  - (b) The recognizer, or the WoZ, recognizes the character and adds it to the input field
6. Once all the characters are in the input field, the participant submits the answer
7. Our server receives the answer and sends it to reCaptcha's servers for validation
8. We obtain a reply indicating whether the response was correct or incorrect, and
9. Pass the reply to the participant

*Input Mechanism.* The canvas area for gesturing the characters was 300px×140px. Further discussed in Section 5.3.6, we pilot tested some classifiers; however, these showed poor accuracy. Thus, we moved to testing with WoZ prototyping. In our case, the researcher injected the character gestured by the participant as input to the challenge answer. These actions went unnoticed by participants. This input character was visible to the participant and reflected the gesture drawn by the participant. Participants gestured each character on the canvas and a delay of 3.5 seconds was introduced to allow the evaluator to input the gestured character. During the pilot testing phase, with a recognizer, the character was recognized after a 3.5-second delay from the moment the participant started gesturing the character. Therefore this delay

in the WoZ was similar to that experienced with the actual recognizer. *Character set.* reCaptcha’s challenges are often improved and changed. During our testing, most of the challenges we observed were images taken from Google Street View cameras, mostly house street numbers. Occasionally, we observed two-word challenges. The character set includes the lowercase and uppercase English (26 upper case characters, 26 lower case characters, plus 10 digits). *Distractor.* House street numbers are not distorted. The two word challenge, one word is not distorted and the second has some warping and small overlapping. *Logging information.* We included logging on the client end (JavaScript) using `log4javascript` [45]. This framework allowed us to capture interaction between the user and the browser before data is sent to the server.

### 5.3.3 Proposal 2 (P2) - Gesture Emerging

Similarly, we adapted Emerging Captchas [151] to use a canvas for drawing each character using gestures. The challenge consists of only three characters in motion across a view finder. Using a similar approach as in Proposal 1, the evaluator recognizes what the user is trying to input, and then the interpreted character is manually entered by the evaluator into the input field that gets relayed to the server. Gesture Emerging has been proven to resist motion-based attacks [151]. The short, three-character, alphabetic challenge also made this a good candidate for testing gesture-character drawing. In terms of the design, this prototype uses a similar input mechanism as Gesture reCaptcha. The Gesture Emerging is shown in Figure 5.4.

For Gesture Emerging, we had direct access to a set of challenges and their answers. Thus, we implemented the server on our own infrastructure without the need to communicate with third-party servers. Figure 5.5 depicts the workflow for Gesture Emerging.

1. The participant requests the captcha-hosting page from our server
2. A challenge is sent to the participant’s browser on his smartphone



Figure 5.4: P2. Gesture Emerging.

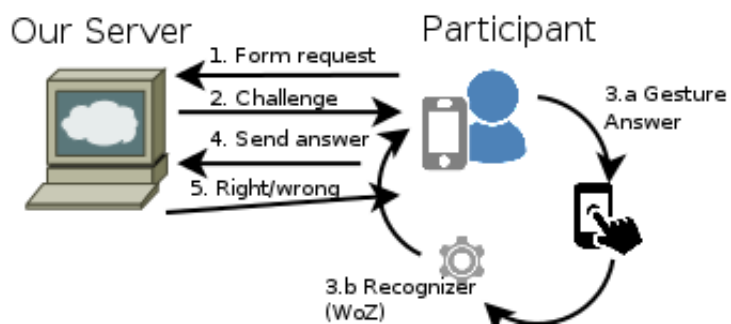


Figure 5.5: P2. Gesture Emerging's message and activity flow.

3. The participant answers the challenge or requests a new one
  - (a) The participant draws one character at a time on the canvas
  - (b) Using WoZ technique, the character is entered in the captcha's response input field
4. The participant submits the answer to the server
5. The server returns a reply to the participant indicating success or failure for the challenge

*Input Mechanism.* The canvas area for gesturing the characters was 300px×140px. Based on the previous experience with the recognizers and their poor accuracy, we directly implemented this prototype using a Wizard of Oz testing approach. Participants used their finger to draw each character on the canvas. A delay of 3.5 seconds was introduced to allow the evaluator to input the gestured character. *Character set.* The character set consisted of the uppercase English alphabet. *Distractor.* There is no distortion *per se* in this scheme but a distractor. The distractor is the challenge movement across the screen forming characters which can only be recognized based on the information aggregation principles [151]. *Logging.* Similarly to reCaptcha Gesture, we included log4javascript and logging at the server.

### 5.3.4 Proposal 3 (P3) - Asirra Slide

Using Asirra's publicly-available API<sup>2</sup>[92], we modified the user interface of the challenge. Asirra was a good representation of an IR captcha. The concept of sliding carousel is well known by users and employed in multiple mobile applications. Rather than showing a grid of images, we updated Asirra to display each of the images individually in a carousel. The design was changed so participants could slide the images back and forth, and select the cat images with a long image tap. While the underlying task remained the same, the modified user interface allowed for larger images that fit on the mobile screen without the need for zooming and provided a larger area for clicking on items. This design also took less device screen real estate than the

---

<sup>2</sup>The Asirra captcha service was closed permanently in October 2014.

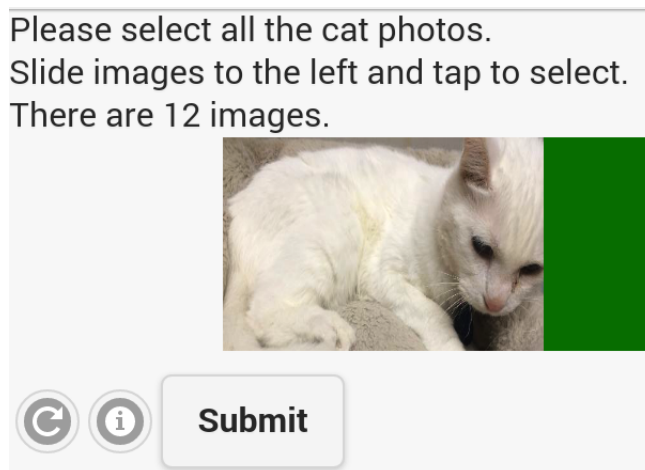


Figure 5.6: P3. Asirra Slide.

original design and the challenge was not obfuscated by the enlarged image of the current selection. Figure 5.6 depicts the prototype.

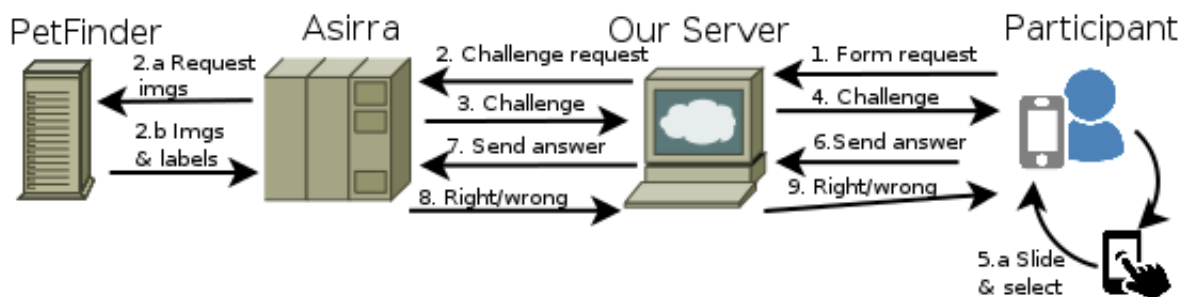


Figure 5.7: P3. Asirra Slide’s message and activity flow.

Asirra’s workflow is depicted in Figure 5.7. Asirra provided a simple API with few configurable options. We deconstructed parts of their API and adapted them for our prototype. Asirra’s captcha service involved a third-party communication from where they got their database of images to construct the challenges. This third party server is PetFinder [112].

Our prototype was structured as follows.

1. The participant requests a challenge from our server
2. Our server sends the request, along with the security public key, to Asirra's server
  - (a) Asirra's server communicates with PetFinder and
  - (b) Receives a set of 12 random images of dogs and cats with associated labels identifying the type of animal
3. Asirra sends the images and unique identifiers for each image to our server
4. Our server sends the challenge to the participant
5. The participant receives the images (challenge)
  - (a) The challenge is displayed in a carousel style, where each image is displayed individually and selected individually
6. Once the participant browse through all the images he submits the answer to our server
7. We submit the answer to Asirra
8. Asirra validates the participant's response
9. Our server sends back the response to the participant

*Input Mechanism.* The selection action consisted of a one-second tap on the cat images which resulted in a green rectangle appearing next to the image. The container holding the displayed image was 250px×250px, and of the image was 150px×100px. The selection indicator was included in the 250px×250px container. Participants could browse back and forth through the images in the carousel by swiping left or right with their finger. *Content set.* The artifacts for recognition on this scheme are images of cats and dogs uploaded to PetFinder by the general public. *Distractor.* The distractors are the images of dogs which users must distinguish from those of cats.



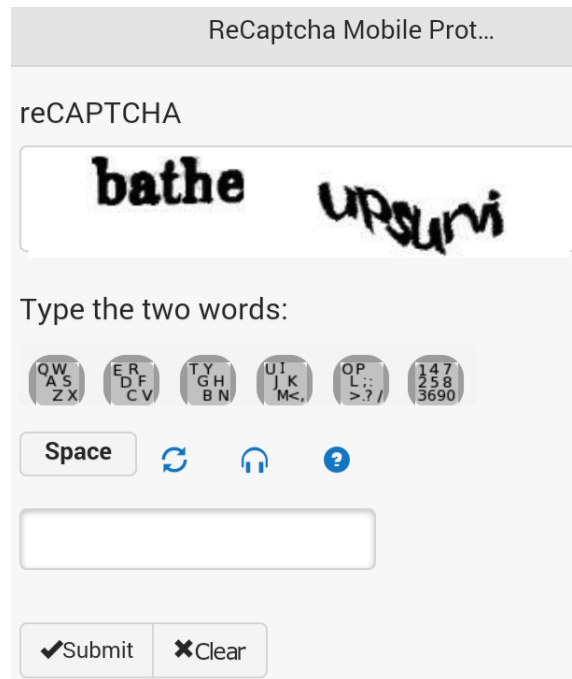


Figure 5.8: P4. reCaptcha Buttons.

No additional distortions or distractors are added to the scheme. *Logging.* Logging on the client side using `log4javascript` was implemented, as well as server side logging information. of selections, times, deselects and participant.

### 5.3.5 Proposal 4 (P4) - reCaptcha Buttons

Using the reCaptcha API, we created a second prototype. The challenge was displayed normally, but the answering mechanism was modified. This prototype aimed to enlarge the keyboard buttons by grouping the `[a-z0-9]` characters in six buttons. Five buttons grouped alphabetic characters and one grouped numeric values. When pressed, each button showed a pop-up menu containing the characters for that group and participants selected characters from the pop-up menu. The purpose of this design choice was eliminating challenge occlusion and eliminating the need for shifting the keyboard between upper case and lower case since the buttons showed both upper and lower case characters. This was perhaps our most unusual design, but our rationale was to reduce the real estate usage by the keyboard. Figure 5.8 shows the reCaptcha Buttons prototype.

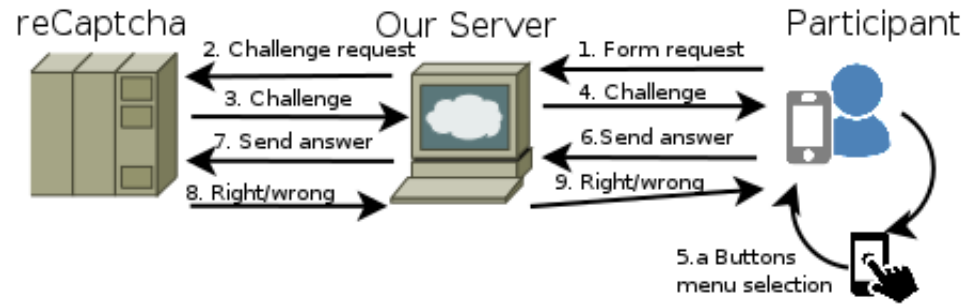


Figure 5.9: P4. reCaptcha Buttons' message and activity flow.

Our pilot prototype of reCaptcha Buttons' message and workflow is the same as in Gesture reCaptcha with the exception of the input mechanism; as described below. The workflow is shown in Figure 5.9.

1. The participant, from her browser, requests the page hosting the scheme from our server
2. Our server sends a request for a challenge to reCaptcha's servers
3. reCaptcha's servers returns a challenge to our server
4. This challenge is forwarded to the participant's browser
5. The participant answers the challenge or requests a new challenge
  - (a) The participant uses the buttons grouping characters and then selects each of the characters in a pop-up menu
6. Once all the characters are in the input field, the participant submits the answer
7. Our server receives the answer and sends it to reCaptcha's servers for validation
8. We obtain a reply indicating whether the answer was correct or incorrect
9. Our server forwards the reply to the participant

*Input Mechanism.* Buttons and multiple selection menus were used in this prototype. Each button grouped a sequence of between 6 and 10 characters. The button size was 36px×36px. *Content set.* The character set was reCaptcha's alphabet which

usually includes the English alphanumeric set; similar to Proposal 1 5.3.2 . While testing this prototype, we did not encounter Google Street View images; *i.e.*, house street numbers. The challenges were two words, one distorted and one not. *Distractor*. The same distractors that we encounter in Gesture reCaptcha we found on this scheme. *Logging*. Logging in this scheme was limited to server interactions which also recorded the user’s input.

### 5.3.6 Recognizers’ Accuracy

Gesture and character drawing recognition is well advanced in native applications. Applications from Google [57] and Evernote [133] include features which employ character recognition using gestures. However, publically available gesture recognition implementations for browsers are rare and lack accuracy. We tested two recognizer implementations, the  $\$P$  and  $\$N$  recognizers [119]. Both of these recognizers are based on a nearest-neighbour classifier with an Euclidean scoring function. We generated a dictionary with multiple variations for each letter and number of the English alphabet, both uppercase and lowercase. The  $\$N$  recognizer has known limitations [82]. Among those is that order and directionality are important; some gestures can be done in a single-continuous stroke or multiple strokes and the recognizer may recognize these same characters as associated with different matches. The  $\$P$  recognizer addresses some of the issues found in the  $\$N$ , such as the ordering and directionality problem. In our studies, the  $\$P$  recognizer was pilot tested by four participants. We included a help image showing users how to draw confusable characters; *e.g.*, 2/Z/z, 5/S/s, 9/g/q. Despite using the training data and asking the participants to draw gestures as the recognizer was trained, the accuracy was poor.

The fields of gestures, and handwriting recognition, and machine learning offer a wide selection of algorithms which can be explored and implemented to serve as more accurate recognizers than the few tested here. However, this is beyond the scope of the present thesis. Given the state of current native recognizers, we expect that publicly available browser gesture recognizers will show considerable improvements in the near future.

### 5.3.7 Design Challenges

While designing our novel prototypes, we encountered several design challenges. Captchas are typically found in web pages; few other applications make use of them. Thus, the designs considered the fact the captchas would be part of a web form and running in a browser. Below we describe some of the design challenges faced while developing the prototypes.

- *Double-tap.* A double tap on a mobile device is not the same as a double-click on the desktop, on the desktop a double-click is for selection. Some mobile browsers implement the double-tap action to zoom in; while others use it for selection. During development of our prototypes, we initially used double-tap as selection but sometimes this action would trigger a zooming behaviour during cross platform testing. We then settled for one-second tap for selection and deselection actions.
- *User expectations.* Users form mental models, or internal expectations, of how certain behaviours and interactions should respond. For many participants, it was the first time they encountered drawing or gesturing on canvas as an input method. Thus, their gesturing expectations varied in terms of recognition accuracy and style of gesturing. In addition, there was a general expectation of being able to draw the complete challenge on the canvas. Gesturing the full challenge at once faces the two main challenges: the canvas space may not allow entering long sequence of characters (*e.g.*, some of reCaptcha's challenges), and the recognizer would have the additional tasks of segmentation and post-segmentation; as opposed to pre-processing, recognition and post-processing for character by character in which the two extra steps are not needed. Segmentation and post-segmentation are two of the most complex tasks while recognizing handwriting and overlapping characters [25, 155]. Another user expectation related to usage of double-tap. Different participants were expecting a particular behaviour (*e.g.*, selection) from using the double-tap.

- *Moving challenges.* The original Emerging and Gesture Emerging variants used GIFs<sup>3</sup> to display their challenges. The original implementation targeted desktop browsers; there are more supported file types on these browsers than the mobile browser implementation. Typically, desktop users are no longer concerned with the amount of data transmitted to or from their computers. Hence, for some captcha schemes, the amount of data transferred is a non-functional requirement. However, mobile users are well aware of mobile network data communication. The available set of challenges for Emerging were large files, and the loading times were negatively impacted on mobile devices.
- *Gesture recognizer.* We pilot tested reCaptcha with a couple of recognizers (Section 5.3.6) but found their accuracy unacceptable. As a result, we discontinued their use and moved to a WoZ user study design. We remark that there are mobile applications such as Evernote [133] and some Google [57] apps which provide sophisticated gesture recognition support. This handwriting support leverages advanced libraries included in the applications. However, web form support for gesture and handwriting recognition is provided via web application, *e.g.*, Javascript implementation of a gesture recognizer algorithm. The computational power and availability is limited compared to native applications.
- *Wizard of Oz.* Using WoZ impacted the performance and perception of the gesture schemes. It would be desirable to do a full implementation with a robust gesture recognizer to more fully assess the usability of gestures as an input mechanism. This was beyond the planned scope of this thesis.

---

<sup>3</sup>GIF is a lossless format for image files that supports both animated and static images [58].

## 5.4 User Study

We conducted a usability evaluation comparing the nine captcha schemes (4 proposals and 5 pre-existing schemes). The goals of the study were to assess their usability on smartphones and identify the users' preferences and opinion of the various schemes.

We chose a controlled lab study to collect quantitative performance data and we collected participants' *impromptu* reactions and comments. Sessions lasted approximately 45 minutes. Participants were offered a \$15 honorarium for their time. This research has been approved by our institution's Research Ethics Board.

We chose to allow participants to use their own device for the study to cover a range of scenarios and to ensure that any problems uncovered were not due to unfamiliarity with the device itself. In the real world, users have a plethora of browsers and smartphone models. Different issues were uncovered because of the wider spectrum of smartphones being tested.

### 5.4.1 Methodology

Participants were divided into two groups; each group evaluated four captcha schemes. A within-subjects experimental design was used for each group.

Each participant was asked to solve ten challenges per scheme, however some participants kept solving challenges after being told they could stop, and a few stopped before completing the ten challenges. Participants completed an average of 35.6 challenges across their four assigned schemes. In total, 998 challenges were attempted. This process was not automated because we used the demo sites provided by the scheme owners whenever possible and not all sites provided APIs to embed the system.

The order of presentation for the schemes was counterbalanced according to a  $4 \times 4$  Latin Square for each group to eliminate biases from learning effects. For each scheme, challenges were randomly selected.

We collected performance data and subjective data. Performance was measured by noting the overall time, number of successes, refresh/skips, and errors while answering the challenges. The participants also responded to a demographics questionnaire and

a satisfaction survey. The questionnaires were implemented using Limesurvey<sup>4</sup>.

**Participants.** The 28 participants (16 females, 12 males, mean age 35.1, SD = 10) were graduate students (5), undergraduate students (6), professionals (9), research assistants (4) and faculty members (4). None had participated in prior captcha studies. The average self-reported expertise using smartphones was 6.8 out of 10, SD = 2.6. The average length of phone ownership was 32 months. All except three reported having encountered captchas before the study. Nineteen participants browsed the Internet on their smartphones *daily*, two browsed *once a week*, two browsed *several times a week*, two *less than once a week*, and two declined to answer.

**Procedure.** The same procedure as our previous user studies was followed (described in Section 3.3). Each participant completed a one-on-one session with the experimenter; sessions which included NuCaptcha and Picatcha schemes were video and audio recorded. Video taping was necessary since these two schemes do not provide APIs and thus we could not automatically log timings. The study was divided into two groups. Group (A) tested: Asirra, Asirra Slides, reCaptcha and Gesture reCaptcha. Group (B) tested: Emerging, Gesture Emerging, NuCaptcha, and Picatcha. The ninth prototype (reCaptcha Buttons) was discarded after pilot testing.

**Pilot testing.** The reCaptcha Buttons prototype offered a modified user interface for reCaptcha meant to enlarge buttons while leaving screen space for the challenge. However, early pilot testing showed that this user interface had significant usability issues. We decided that this was not a viable alternative and we discontinued testing after five users. Therefore, our results focus on the remaining eight schemes.

The study protocol consisted of the following components for each of the groups:

- Briefing session. The experimenter explained the goals of the study, detailing the study steps, and asking them to read and sign the consent form.
- Demographics questionnaire. Participants answered an online demographic questionnaire at the end of the first evaluated scheme.

---

<sup>4</sup>LimeSurvey: <http://www.limesurvey.org/>

- **Captcha testing.** Participants visited a host page with links to the captcha schemes from the smartphone. Participants in each group tested four schemes each.
- **Satisfaction questionnaire.** After completing the challenges for a scheme, participants completed an online satisfaction questionnaire collecting their opinion and satisfaction of the scheme.

***Experimental Setup.*** A simple web-based user interface was designed where users could enter their user name. Participants were directed by the experimenter as to which scheme to select next according to their prescribed presentation order.

Where possible, we used the live demo sites offered by the original developers. Visiting the original demo sites allowed testing of the latest version of the schemes and meant that the systems functioned exactly as intended by their developers. When this was not possible, alternative methods were used as described below. Customizations and implementations were made using PHP, HTML5, CSS3, JQuery and JavaScript.

The pre-existing captchas were presented as follows. For NuCaptcha and Picatcha, participants were redirected to the respective demo sites. We created a plain webpage that only included the embedded challenge for reCaptcha and Asirra and we used the available APIs to generate and evaluate the challenges. Emerging captcha is a prototype system. We created a plain webpage embedding the captcha scheme and had a pool of 20 challenges available from which the server would randomly select.

As detailed in Section 5.3.2, the Gesture reCaptcha prototype customized the user interface for the reCaptcha API by adding a gesture canvas and translating the user's input to a text string submitted as input to the API. Due to limitations of the available gesture recognizers and results of pilot testing, we used a WoZ prototyping technique (described in Section 5.1) to translate user gestures to text input. Participants thought that the researcher was taking notes relating to observations and were unaware of this aspect of the prototype.

The Gesture Emerging prototype used a similar gesture canvas and WoZ prototyping technique to capture user's drawn characters and translate it to text input.



Scheme	Number of Participants	Hosting	Mobile features	Data Collection	Group
S1. reCaptcha	18	API	✗	Instrumented	A
P1. Gesture reCaptcha	12	API	✓	Instrumented	
S4. Asirra	15	API	✗	Instrumented	
P3. Asirra Slides	15	API	✓	Instrumented	
P2. Gesture Emerging	10	Coded	✓	Instrumented	B
S3. Emerging	10	Coded	✓	Instrumented	
S2. NuCaptcha	10	Demo	✓	Video	
S5. Picatcha	10	Demo	✓	Video	
P4. reCaptcha Buttons	5	API	✓	Instrumented	Pilot

Table 5.1: Overview of experimental setup. Schemes with ✗ do not provide smartphone specific design features. For images of the schemes refer to Figures 5.2, 5.4, 5.6, 5.8, and 5.1.

The Asirra Slides prototype provided a custom user interface using the carousel slide component. It used the Asirra API in the back-end to serve and evaluate challenges. The slide images were 150px×150px in size.

Table 5.1 summarizes the evaluated schemes. The “Hosting” column refers to the site which participants visited to solve challenges. *API* refers to a webpage on our lab server using the scheme’s public API. Schemes in which participants were redirected to another server are noted as *Demo*, while those denoted *Coded* were custom prototypes served from our lab server. The *Mobile* column identifies whether the scheme has design features provisioning for smartphone usage. The *Data Collection* column summarizes how we gathered performance data for the scheme. We were able to *Instrument* the webpages so that most of the data collection was accomplished programmatically. However, this was not possible for the two schemes denoted with *Video*; for these, we video recorded the interaction and manually extracted the necessary data after the session.

### 5.4.2 Data Collection

Three methods were used to collect data: logs, questionnaires and observations. Unless otherwise indicated in Table 5.1, each system was instrumented to log users' interactions with the system. We recorded the overall time (receiving, answering, submitting, and getting the reply) for each challenge. We also tracked the frequency counts for success, refresh/skip, help requests, and errors while answering the challenges. For NuCaptcha and Picatcha, this information was manually extracted from the videos.

### 5.4.3 Data Analysis

**Outcomes:** to be identified as a *Success*, the user's response had to be entirely correct. An *Error* occurred when the user's response did not match the challenge's solution and was indicated as incorrect by the captcha site. A *Skipped* outcome occurred when the participant pressed the "Get Another Challenge" button and was presented with a different challenge.

**Overall Time:** The overall time was measured as the time between when the challenge was displayed to when the response of success or failure was received on the client side. This included the time to input the answer, as well as the time it took the form to receive the reply. Times for skipped challenges were not included since we observed users making the "skip" decision very quickly and this may unfairly skew the results towards shorter mean times. However, we include times for challenges that resulted in errors because in these cases participants actively tried to solve the challenge. The total success, error, skip rates and time were calculated based on the average of each participant's averages to avoid biasing towards the performance of users who completed extra challenges.

The Gesture Emerging and Gesture reCaptcha schemes used the Wizard of Oz technique. The WoZ overhead time (3.5 seconds) was included in the Overall Time calculations and related statistical analysis.

**User Perception:** Our satisfaction questionnaire used 10-point Likert-scales to evaluate agreement or disagreement with the questions (1 - Strongly Disagree, 10 - Strongly Agree).

Captcha category	Scheme	N	Success	Error	Skips	Mean Time in seconds (SD)
IR	S4. Asirra	97	0.80 (0.16)	0.19 (0.17)	0.10 (0.00)	29.2 (9.83)
IR	P3. Asirra Slides	103	0.75 (0.22)	0.25 (0.22)	0.00 (0.00)	30.6 (12.98)
CR	S1. reCaptcha	190	0.91 (0.18)	0.90 (0.22)	0.00 (0.00)	25.2 (17.50)
CR	P1. Gesture reCaptcha	102	0.87 (0.40)	0.30 (0.10)	0.10 (0.65)	55.3 (12.49)
MIOR	S3. Emerging	116	0.98 (0.60)	0.20 (0.00)	0.00 (0.00)	22.4 (6.46)
MIOR	P2. Gesture Emerging	115	0.88 (0.22)	0.12 (0.27)	0.00 (0.00)	44.5 (12.65)
MIOR	S2. NuCaptcha	155	0.98 (0.40)	0.20 (0.30)	0.00 (0.00)	8.5 (2.92)
IR	S5. Picatcha	120	0.80 (0.10)	0.17 (0.24)	0.30 (0.00)	12.3 (4.97)

Table 5.2: Summary of performance results.

## 5.5 Outcomes

We now report the result of the user study. Our goal was to explore what worked well and identify areas where usability was problematic. In particular, we did not include detailed statistical analysis. We felt that these would be misleading given that some of the schemes used WoZ. For completeness, post-hoc statistics are included in Section 5.5.2.

### 5.5.1 Performance

Table 5.2<sup>5</sup> summarizes the performance outcomes for the evaluated schemes. Success, error and skips are presented as percentages. The mean time and its standard deviation are expressed in seconds<sup>6</sup>. Figure 5.10 visually summarizes the mean percentages of success, error and skips for the evaluated the schemes.

**Success:** As shown in the table, users were most successful at solving the text-based (CR & MIOR) Captchas. At 98%, NuCaptcha and Emerging resulted in the most successful outcomes, but all text-based schemes had success rates of 90% or

<sup>5</sup>reCaptcha Buttons is not reported since it was eliminated earlier on.

<sup>6</sup>Our first Gesture reCaptcha participant used an open source handwriting recognizer and its accuracy was unacceptable. As a result, the solving times for this participant were removed for these calculations.

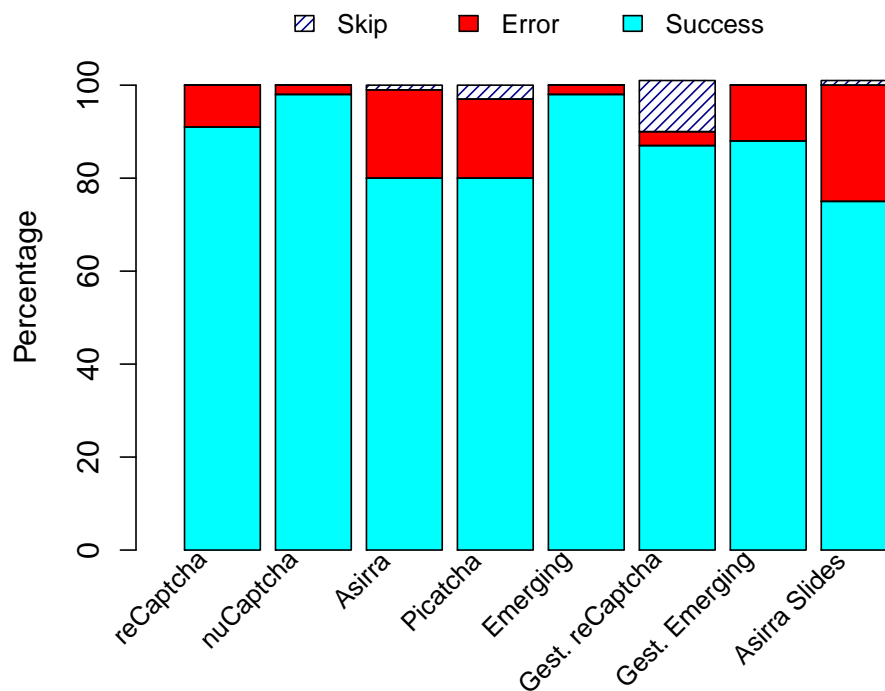


Figure 5.10: Percentages of Successes, Errors and Skips per scheme.

above. Looking at reCaptcha and Emerging, the variants with virtual keyboards had slightly higher success rates than their gesture-based counterparts. It is worth noting that we are comparing usability aspects, but not security. Therefore, schemes on which users are more accurate (more correct answers) and which are preferred by users are not necessarily the most *recommended* schemes. For example, the schemes might also be much more easily defeated by automated programs than other alternative schemes. An alternative with suitable security and usability for the context of use should be selected.

**Errors:** Although success rates were relatively high, we examine the sources of errors to gain insight into where problems occur. Asirra and Asirra Slides resulted in the highest number of errors. For Asirra, the small image size, the quality of images and the need for panning were the likely sources of errors. For Asirra Slides, many participants initially had correct responses but accidentally deselected images by tapping and sliding before submitting. Picatcha had images which often confused

participants and they would fail to recognize some of the target images. Gesture reCaptcha, NuCaptcha, and Emerging had extremely low error rates.

**Skips:** Very few skips were observed. Five schemes had no skips at all, and the remainder had skip rates under 10% (shown in Table 5.2). Although it happened rarely, participants were quick to decide if they were skipping a challenge or attempting to solve it. Once committed to solving a challenge, participants followed through and we saw no skipping partway through completion.

**Overall Time:** Mean times are also summarized in Table 5.2. NuCaptcha demonstrated the shortest overall solving time. Although Emerging and Gesture Emerging challenges were also three characters in length, these challenges could not be read and solved until the animation started. Unfortunately, loading times for these prototypes were slow<sup>7</sup> and this affected the overall solving times. This could be addressed in an actual implementation. Similarly, the gesture schemes using WoZ had significantly longer solving times due to manual translation of each drawn gesture into text. Although we provide a summary of the times for completeness, we do not consider our timing information to represent a realistic comparison because of these limitations.

### 5.5.2 Post-hoc Statistical Analysis of Correct Answers

We remind the reader that our user study design consisted of two within-subjects groups. That is, all participants in group (A) performed all the conditions for that group, and similarly for group (B). Although no statistical test exactly matches these conditions, we present post-hoc statistical analysis using the most appropriate tests to our knowledge. Results should be confirmed in future work with a slightly different study design. One possible study design could have each participant try all of the prototypes and schemes, providing a fully within-subjects design.

We used ANOVA to compare of the means for the schemes in each of the groups. For group (A), a one-way repeated-measure ANOVA for correct answers showed no significant differences between Asirra, Asirra Slide, reCaptcha, and reCaptcha

---

<sup>7</sup>In some cases loading times were more than 10 seconds while the loading times for the the rest of the schemes were under 3 seconds.

Scheme pairs		p
P2. Gesture Emerging	S3. Emerging	0.88
S2. NuCaptcha	S3. Emerging	0.03
S5. Picatcha	S3. Emerging	0.98
S2. NuCaptcha	P2. Gesture Emerging	0.005
S5. Picatcha	P2. Gesture Emerging	0.98
S5. Picatcha	S2. NuCaptcha	0.01

Table 5.3: Tukey comparisons for each scheme in group (B).

( $F(3, 44) = 1.21, p = 0.317$ ). For group (B), one-way repeated-measure ANOVA for correct answers showed significant differences between Emerging Gesture, Emerging, Picatcha and NuCaptcha ( $F(3, 34) = 5.42, p = 0.00371$ ). We then used post-hoc Tukey HSD tests to see where the differences were.

The Tukey pairwise comparison revealed significant differences between NuCaptcha and Emerging ( $p < 0.05$ ), NuCaptcha and Emerging Gesture ( $p < 0.01$ ), and NuCaptcha and Picatcha ( $p < 0.01$ ) (summarized in Table 5.3); no other differences were significant<sup>8</sup>. The results indicate that NuCaptcha has significantly fewer wrong answers compared to the other schemes. The differences between Emerging Gesture, Emerging and Picatcha were not significant.

We also used ANOVA to compare the means for the schemes based on their input mechanisms: virtual keyboard and gestures. We compared reCaptcha and Emerging, to Gesture reCaptcha and Emerging Gesture. A one-way repeated-measure ANOVA for input mechanisms showed no significant differences between keyboard and gesture input mechanisms ( $F(1, 46) = 0.482, p = 0.491$ ).

---

<sup>8</sup>Note that p-values greater than 0.05 are considered not significant.

8.19	8.81	9.88	7.19	7.38	7.19	7.46	7.57	reCaptcha
9.6	9.6	9.8	9.3	9.7	9.2	9.2	8.6	nuCaptcha
7.13	8.53	9.53	6.07	7.6	5.07	5.57	7.31	Asirra
8.11	7.67	8.56	7.33	8.11	6	6.89	5.56	Picatcha
8.2	8.2	9	7.4	8.7	6.3	6.4	7.78	Emerging
5.81	7.88	9.06	5.25	6.25	5.12	4.6	4.47	Gest. reCaptcha
5.27	6.64	7.73	4.91	5.09	4.6	3.18	4.64	Gest. Emerging
8	8.8	9.6	7.67	7.07	8	6.43	6.25	Asirra Slides
								Accurate solving
								Understandability
								Memorability
								Pleasant
								Solvability*
								Suitability
								Preference
								Input Mecahnism*

Figure 5.11: Mean scores for the Likert-scale responses. 1 = most negative, 10 = most positive. X-axis represents the Likert scale questions, Y-axis represents the schemes. \* represents inverted questions.

### 5.5.3 User Perception

Participants provided feedback through Likert-scale responses, free-form questionnaire responses, and verbal comments during the sessions.

**Likert-scale questions:** Figure 5.11 reports the mean Likert-scale responses assessing each scheme. The Likert-scale ranged from 1 = Strongly Agree (dark green) to 10 = Strongly Disagree (dark red). The questions marked with (\*) were inverted to avoid bias; as a result the scores for these statements were reversed before calculating the means. Thus, a higher score always refers to a positive opinion for the scheme.

The statements are as follows:

1. *Accurate solving*: It was easy to accurately solve the challenges
2. *Understandability*: The challenges were easy to understand
3. *Memorability*: If I didn't use this mechanism for a few weeks, I would still remember how to answer the challenges
4. *Pleasant*: This captcha mechanism was pleasant to use
5. *Solvability\**: I found it hard to solve challenges presented on this captcha scheme
6. *Suitability*: I found this mechanism well suited for the smartphone
7. *Preference*: On a smartphone, I would prefer using this captcha mechanism compared to other captchas
8. *Input mechanism\**: The input mechanism is more prone to mistakes than traditional input mechanisms (*e.g.*, virtual keyboard)

As shown in the Figure 5.11, both gesture-based schemes were scored poorly. Gesture Emerging ranked the lowest for preference. From our observations, participants were especially unhappy with challenge loading times and canvas response. The latter was due to the WoZ experimental design, and the former due to file size. These two factors had a major impact on the participants' perception of the scheme. At the opposite end, participants particularly enjoyed NuCaptcha and rated it highly on all aspects. Both Asirra Slides and Emergent scored positively on all questions, indicating that they were also fairly well-received.

**Comments:** The perception questionnaire included free-form space for comments. Sample comments from participants are included in Table 5.4. We organize comments according to the main themes uncovered.

*Input mechanisms:* Many participant comments related to the scheme's input mechanisms. Many wanted the input mechanism to be tailored specifically for the captcha task. For example, some participants wanted a numeric virtual keyboard to be automatically activated if the challenge was number-based. One participant



commented “The keyboard is too big on this” after the keyboard pushed the challenge partially out of the viewport. Others commented on the responsiveness of the gesture-based schemes. There was concern over the recognizer and the time it takes to process characters: “it was slower than typing”. This is clearly related to the WoZ technique, but participants were unaware of this fact.

*Preferences:* Participants felt that preferred schemes and input mechanisms would vary for different user groups. For example, “If the recognizer would work I think younger people may prefer this method. I think they are not used to keyboards.” Although not age-related, we have some evidence of such polarized preferences within our study. Our participants were almost equally divided in their preferences regarding the grid layout and the sliding carousel for Asirra. Several participants remarked that some of the schemes may have cultural and language dependencies which would limit broad adoption and usage.

*Challenge content:* The content of the challenges was highly important in users’ perception. For example, participants expressed preference for selecting cats as opposed to dogs. Unsurprisingly, participants also expressed a clear preference for simpler, shorter challenges with little or no distortion. They voiced displeasure with highly distorted and time-consuming challenges. Participants also preferred words over random letters or a mix of alphanumeric characters, and noted that image quality significantly affected the difficulty of Asirra challenges. The simplest, shortest challenges would clearly be the most usable but security is an important consideration as well. Simpler might translate to being less secure; this is certainly the case for simpler versions or simpler instances from a single scheme, e.g., recognizing 1 letter vs. recognizing 5 letters. Regarding bandwidth usage, users did not want to waste their data plan on captchas and quickly got impatient with slowly loading challenges and slow response times.

#### 5.5.4 Experimenter Observations

The experimenter noted verbal comments as participants solved challenges and noted any behavioural observations. We have categorized the comments and observations according to the following themes.

Scheme	Comments
S3. Asirra	I prefer seeing all the pictures of cats and dogs at the same time.
P3. Asirra Slides	The implementation was pretty buggy, but I think the idea was alright.
S1. reCaptcha	Important on a smartphone to only ask for either digits or text (as was the case here)
P1. Gesture reCaptcha	Sometimes difficult to get the intended results [recognizer]
S3. Emerging	The video was slow to load sometimes
P2. Gesture Emerging	It was mainly hard because of the time between accepting the gesture, and the canvas not responding.
S2. NuCaptcha	The colour helped in terms of making it easier to read.
S5. Picatcha	Images could be confusing for some, culturally specific.

Table 5.4: Sample comments from participants.

**Phone handling:** Some participants varied the position of the phone depending on the type of challenge. Some placed the smartphone on the table when typing was involved but held the phone in their hands when using gestures, selecting or sliding images. Secondly, although most of the participants kept the phone in portrait mode, some participants rotated their phones from portrait to landscape when attempting to avoid challenge obfuscation. Obfuscation was most problematic with Asirra. We observed as well that pinching and zooming sometimes triggered other phone features. For example, while zooming and panning, one participant wanted to drag the page but instead the OS drawer (top bar) was activated, increasing the solving time for that challenge. Several participants had protective cases or screen overlays on their smartphones; this did not seem to interfere with the solving of the challenges.

**Software observations:** Many errors (challenge responses marked as incorrect) were due to implementation issues and browser incompatibilities. For example, we noted that two phone models by Lenovo and Samsung, using the Mobile Safari/534.30 browser resulted in a second shadow line drawn outside the canvas in response to gestures on the drawing canvas (Figure 5.12). This was obviously a source of confusion for participants in the gesture-based schemes. We did not observe the same problem

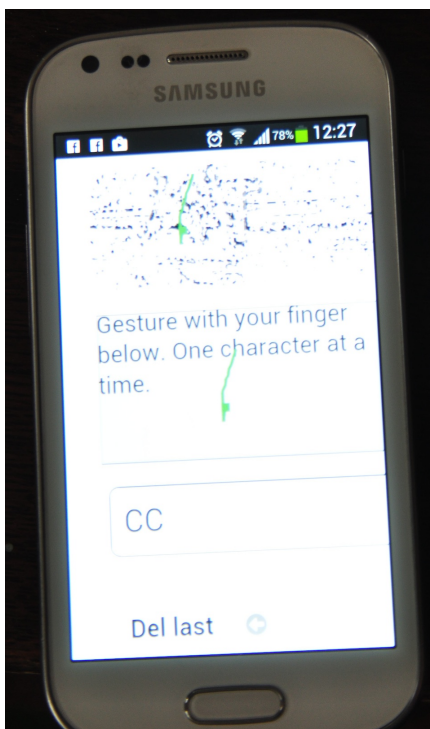


Figure 5.12: Prototypes. Wrong canvas rendering.

on any other browsers. Similarly to desktop solving, many participants were not aware that CR answers are generally case insensitive. Some participants typed each character in capital letters, activating the shift key each time. The help buttons were never used by participants throughout the entire study.

**Challenges and schemes:** We noticed several scheme-specific issues and discuss our main observations. The most obvious difficulties were observed with Asirra. Asirra used a pre-tagged public image database for its challenges. The low quality of some images negatively impacted participants' performance and experience; although we expect that this would also occur on a desktop. Some participants accidentally clicked up to three times on the "Adopt Me" link. In response, the browser left the current page and opened Petfinder's website. Upon returning to the scheme's challenge page, a new challenge would be presented. Furthermore, a couple of participants were confused about which image to select; they attempted to select the enlarged image rather than the image from the grid.

Asirra Slides resulted in more positive comments than Asirra. Comments included

“This is fun” or “Sliding is easier for this type of input, but the keyboard suits me fine”. However, participants were ultimately divided on which variant was best. Four participants expressed a preference for seeing all the images at once in a grid and three participants said they liked sliding through the images. The remainder did not voice an opinion.

While using reCaptcha, one participant commented “it’s frustrating [referring to the challenge], can I hear it?”. Then the participant tried an audio challenge but rather than function as expected, the browser wanted to download an MP3 audio file. This highlights some of the shortcomings and challenges audio captchas still face on smartphones.

***Distraction from main task:*** While participants generally accepted the need for captchas, they wanted it to be as quick and unobtrusive as possible. One participant noted for Picatcha “I didn’t like it, it’s like a game, you don’t find a purpose, it’s a distractor, you want to do your main task, not play. You want to send your form”. Participants were unhappy with schemes that annoy or distract the user from their main task. This observation is especially noteworthy for designers of game-based captchas or those choosing what scheme to deploy on a websites.

Other distractions were also noted. Asirra and Picatcha could take participants to external sites. Asirra had the “Adopt Me” link under each of the images and Picatcha’s success notification image would take participants to external sites if it was clicked. This clearly would affect the user’s primary task.

***Gesture input:*** On a few smartphones, the gesture canvas pushed part of the challenge off of the visible portion of the interface. However, this was more problematic for Gesture reCaptcha than Gesture Emerging. Gesture Emerging has a three-letter challenge that allows the participant to observe the challenge and answer it without continuously having to see it because they can remember three characters.

The gesture interface was new to participants and they needed some time to learn how to use it effectively. Most tried to write the complete answer on the canvas at once even though we told participants to draw the characters individually. Participants voiced their uncertainty: “With the gestures I have to learn how to use it”. It is likely that the recognizer would need to either offer personalized gesture training

or the user would need to learn to “properly” gesture each character. We received comments including “easy challenges, but the gestures were tricky”, “easy to gesture” and “I would prefer the gestures if properly recognized’. In spite of their uncertainty, participants did not venture to click on the help button where a table on how to draw characters was available, highlighting the importance of an intuitive, self-explanatory interface.

### 5.5.5 Summary of Results

We recorded the following performance measures: success rates, errors and overall solving times. Participants were able to solve the majority of challenges in all schemes. As we described above, NuCaptcha showed the most successful outcomes for both the user study and was favoured by participants in the questionnaire responses. The Emerging scheme also showed good overall performance, although its loading times were an issue for user satisfaction and performance. Both were text-based schemes.

We tested several different modalities for input mechanisms: virtual keyboard, drawn gestures, carousel slides, or tap-based image selection. Surprisingly, given the difficulties with smartphone keyboards, we found that keyboard input was most accurate. NuCaptcha and Emerging both had 98% success rate. It appears that familiarity with the keyboard was enough to overcome known difficulties with typing on small touch screens.

Participants found the gesture input interesting, but performance issues relating to the WoZ implementation led to more negative perceptions than initially expected. In their current state, participants felt that gestures added complexity compared to typing. Users may make errors while typing, but they knew exactly how to resolve the problem when it occurred.

Participants preferred simple and quick schemes such as NuCaptcha and Emerging. For schemes where concerns were noted, participants’ questionnaire responses and experimenter observations revealed valuable feedback regarding implementation and deployment.

## 5.6 Discussion

As part of our investigation of captchas on smartphones, we designed four modified captcha schemes intended to be more usable for this environment. Unfortunately, our studies found that these prototypes were not as successful as we had hoped. Both pre-existing NuCaptcha and Emerging performed well overall in terms of usability. Given that NuCaptcha has been broken [151], perhaps Emergent captcha would serve as a reasonable alternative if implemented in a manner that takes into account our proposed recommendations (Section 6.2).

Bursztein *et al.* [26] noted a disconnect between users' preferences and their ability to accurately solve challenges. Our work supports this finding across a wide range of captcha schemes. Although our participants could solve challenges successfully; they did not necessarily like the schemes. For example, Gesture Emerging had similarly high successful outcomes to NuCaptcha, but Gesture Emerging scored considerably lower on the perception questionnaires. It is possible that preference could be due to shorter versus longer solving times. Nevertheless, this reinforces the need for considering both correctness and perception when evaluating schemes. Security is also essential to any captcha scheme, but not studied within the context of this thesis.

Regardless of the variant tested, participants showed some reluctance to learn new input mechanisms. Most of the participants preferred the now-familiar virtual keyboard despite its small size and known inconveniences. Participants felt that they at least knew how to cope with the keyboard when something went wrong; they had not devised such coping mechanisms for the new input mechanisms. We are reluctant to completely disregard gestures as an input mechanism, considering a number of recent applications (Evernote and Google apps [57, 133], for example) now include gesture recognition for character input because it is more convenient than typing in several scenarios. We believe that it is likely that over time, gesture input may become as familiar as the virtual keyboard and may mature into a viable alternative for captchas.

Although gesture-recognition technology is reasonably advanced, web browser implementations of gestures are still lacking. Currently gesture recognition must be

implemented as part of the web application, as opposed to OS-natively supported for mobile applications such as Evernote. Gesture recognition in current mobile browsers have several constraints. For example, browser support for W3C standards varies, there are limitations in HTML specifications, and the availability and implementation of efficient and robust multi-stroke recognizers is lacking. Until some of these constraints are addressed, it will be difficult to use gestures for web-based captchas.

## 5.7 Conclusion

The aim of this chapter was to explore whether alternative input mechanisms help improve the usability of captchas on smartphones, and to evaluate the usability of the modified schemes intended to be more suitable for smartphones. In total, we compared the usability of nine captcha schemes on smartphones, including four proposed alternatives. The results show that although participants find virtual keyboards prone to errors, they prefer them for solving character recognition challenges over the other input alternatives studied herein. We believe that including the results of the proposed alternatives is valuable to the community even if their usability was lower than we had hoped since the lessons learned could help with future development. Negative perceptions might be greatly improved with alternative implementations. Our recognizer implementation was not robust nor reliable. However, we believe that having a robust and reliable implementation might have allowed participants to accept it as a viable option for solving captcha challenges. Perhaps unsurprisingly, participants' preferences were dominated by simple, short challenges with little or no distortion. Character recognition challenges were preferred over image recognition challenges. Another key finding was the disconnect between users' preferences and their ability to correctly solve challenges. Thus a sole measure of captcha solving success and failures does not accurately indicate usability. Our studies suggest that although alternative or novel input mechanisms are approached with caution by users, there is room for more research and improvement. This work contributed to informing the set of recommendations and suggestions, offered in Chapter 6, to adapt captchas schemes for websites catering to mobile users.

## Chapter 6

### Design Recommendations and Conclusion

#### 6.1 Introduction

In Chapters 3 and 5, we have shown how the usability of current captchas on smartphones has a negative impact on users' perception of the overall experience of captchas on smartphones and, in some schemes, an impact on users' performance. Chapter 4 provides an instrument for the evaluation of captchas on smartphones. Heuristics as inspection tool can be developed following multiple approaches (Section 4.2); one approach to developing domain specific heuristics is to turn design guidelines into heuristics. Thus the heuristics form the basis of recommendations. Our heuristics (Section 4.4) and the experience obtained by conducting the usability studies [120, 121, 150] led to the design suggestions presented in this chapter.

We provide a series of improvements and suggestions to adapt and deploy captcha schemes on mobile websites or responsive websites<sup>1</sup> concerned with bots. While our recommendations focus on usability aspects, it is important that security considerations [23, 155, 161] are also taken into account before deploying a captcha scheme. For example, features such as simple distortions, colours, and lines have been shown to offer little additional security in the face of a determined attacker [21, 155, 157]. Such features are often used in simplistic schemes failing to increase the security and decreasing the usability of captcha scheme. Our Design Recommendations aggregate work published at USEC 2015 [121] and SECRIPT 2013 [120].

This chapter is organized as follows, Section 6.2 lists design recommendations for designing captchas for mobile devices. Section 6.3 recalls our research objectives and thesis topic. Section 6.4 summarizes the research contributions addressing the objectives set forth in this thesis. Finally, Sections 6.5 and 6.6 discuss further research

---

<sup>1</sup>Responsive websites are designed to offer an optimal viewing and interaction experience regardless of the device used by presenting layouts based on device type.



directions resulting from this thesis and the conclusions of this work, respectively.

## 6.2 Design Recommendations

We offer 13 design recommendations for mobile captchas based on our research. We group the design recommendations into: design of challenges, screen layout, and environment of use and deployment. Design of challenges recommendations pertain to the content of the challenge, the task, and the challenge evaluation output. Screen layout recommendations discuss how to incorporate the captcha design within the overall web form. Finally, environment of use and deployment recommendations address issues relating to non-functional requirements and deployment of captchas. We organize the recommendations into two groups: a) focusing on the captchas themselves, and b) considering the broader context of the environment in which captchas are used.

### 6.2.1 Recommendations for design of challenges

Although these recommendations are aimed at small-screen devices, some apply equally to captchas for desktops. In these recommendations, we address the captcha challenge, the control buttons, input methods and other distractors that may not directly relate to solving the challenge.

**R1.** *Choose a scheme which avoids keyboard swapping (e.g., between numeric and alphabetic) and confusable characters (e.g., 1/1, 0/0, 6/G/b, 2/Z/z, 5/S, nn/m) since these are especially problematic on smartphones, regardless of the input mechanism.*

Captcha security research has shown that eliminating characters does not significantly impact the security of CR schemes [25]<sup>2</sup>. Thus eliminating some of the confusable characters does not negatively impact the security of a CR scheme but improves its usability. In addition, the captcha character set should be case constant; *i.e.*, either the whole challenge is in upper case or lower case. This eliminates the

---

<sup>2</sup>Bursztein *et al.* do not specify the limit to how many characters can be eliminated without impacting the security of the scheme.

need for shifting between keyboards. For example, NuCaptcha consistently uses three upper case characters for one of their offered captcha formats, and a version of reCaptcha used images taken from Google Street View cameras which consisted only of numbers. On the other hand, Microsoft's Live.com [93] captcha challenges are upper case but include numbers in some challenges, forcing the user to swap keyboards.

**R2.** *Design with captcha-solving focus.*

While additional features may appear desirable for other reasons, such as generating advertising revenue, they significantly hinder usability on small screens because solving captchas is already more difficult and time-consuming than on a desktop. An example of captchas with distractions is Solve Media. Solve Media [89] includes the advertisement text and the challenge text (Figure 2.3(c)) which may leave users confused about the required task.

**R3.** *Use input mechanisms that are cross-platform compatible and that do not interfere with normal operation of the browser.*

In particular, touch events are often handled differently depending on the browser. For example, the default behaviour of double tapping in some browsers is to trigger a zoom-in, and thus this gesture should not be mapped to another action within the captcha.

**R4.** *Provide an input correction mechanism which adequately supports the input mechanism.*

In case the captcha scheme utilizes input mechanisms different than the virtual keyboard, the input error correction mechanism should allow input clearing of the user's answer to the captcha; avoiding complete web form clearing. For example, if the input mechanism is a canvas and the challenge consists of drawing in a canvas then clearing of the canvas to correct the drawing should be supported.

**R5.** *Make sure error messages, skip, help buttons, and dialogues do not distract the user from the main task or force the user to restart the main task.*

Error messages from captchas should not be disruptive to the main task, *e.g.*, filling a form. Ideally, the scheme should simply indicate to the user that the challenge was not successfully passed and allow another attempt without undoing anything from the main task. For instance, Picatcha and Asirra error messages distracted users from the main task during our user testing and heuristic evaluation (Sections 3.3.4, 5.5.3). In the case of Picatcha, the error message was a cartoon displayed on a different page. Asirra’s wording was found to be simplistic or childish by many participants. This observation is especially noteworthy for designers of game-based captchas because unclear error messages could be confused as part of the game. The MC heuristics revealed that some schemes’ help option would take the user away from the main task by opening new browser tabs with the help description, leading users to lose context for their main task.

**R6.** *Take into account browser capabilities and limitations.*

MIOR, audio, and game-based captcha schemes should be particularly attentive to current mobile browser limitations since most of these schemes rely on browser capabilities to be able to play or display the challenge. For example, certain animation file formats may not be supported by all mobile browsers. In general, we should avoid designs which require non-standard software, fixed-size screens, extra plug-ins or additional tools on the client mobile device.

### 6.2.2 Recommendations for screen layout

The following recommendations are targeted toward the overall captcha layout of the hosting page and its behaviour in relation to the form. Captchas have become an integral part of input forms and as such they should be considered in context, rather than as an additional API or plug-in.

**R7.** *Consider isolating the captcha task from the rest of the web form.*

Isolation maximizes screen real-estate for the captcha and makes it easier to ensure that incorrect challenges do not unnecessarily disturb the main task of completing the web form. This alternative can be particularly useful in case the website is not designed as a responsive website. For example, we could create a wizard in which solving the challenge is one of the steps to complete the web form; or create a pop-up layer with the captcha in it and nothing else.

**R8.** *Strive for a minimalist interface. Avoid optional features on the captchas.*

While Recommendation 2 focuses on the content of the challenge this recommendation pertains to the interface for the captcha. Avoid cluttering the captcha challenge with non-essential buttons and instructions, in particular because the input interface may require a significant portion of the screen. Captcha schemes typically include controls for requesting new challenges, help, and audio challenges. Additional controls, buttons, or information reduce the already limited screen real-estate. Asirra’s “Adopt Me” [92] link had benevolent intentions but it often caused users to lose their current image selections as they are redirected to another site (Figure 2.4 (a))

**R9.** *Keep orientation and size of the captcha consistent with the rest of the web form.*

In case that the captcha is not isolated (Recommendation 7), users should not need to change the device orientation or need to zoom/pan in order to solve challenges. Our studies (Chapter 3) showed that many existing captchas do not follow established mobile design and responsive website conventions. For example, a mobile device browser, by default, will scale down the website to fit the viewport<sup>3</sup> A responsive website could use the HTML tag *viewport* to render the page to the actual width of the device. Large challenges cause the user to lose overview, while small challenges force zooming. Thus, users found themselves zooming, panning and often trying to find the best fit for looking at the challenge by flipping the phone horizontally and vertically.

---

<sup>3</sup>The viewport is the area on the device screen where you are viewing a web page.

### 6.2.3 Recommendations for environment of use and deployment

The captcha scheme should be suitable for the range of situations and environments in which users may access the scheme. Based on what we observed from our user studies and heuristic evaluations, it is likely that most users keep the factory installed browser. Nevertheless, the perennial problem of browser cross-compatibility permeates to mobile devices. That is, not all browsers render equally the same web sites. These recommendations consider the captcha in an ever broader context than the challenge or its hosting web form.

**R10.** Perform adequate cross-platform testing to ensure that all web elements work across a majority of browsers and versions.

Testing on a myriad of hardware devices can become a daunting task but it is critically important to the usability of captchas given the wide range of potential platforms. Several tools facilitate the task of cross-platform testing [91, 149] to identify incompatibilities. For example, a canvas web element to capture gestures may behave differently or not work in some browsers, and some audio captchas may be non-functional. We exposed one particular problem in Section 5.3.7 (Figure 5.12) in which one browser configuration caused the canvas element to generate double, or ghost lines, but the same hardware worked perfectly with a different browser.

**R11.** Avoid features and challenges which may fail in commonly expected environmental conditions.

Environmental conditions can significantly impact the usability of captchas. Given that mobile devices are used in a wide range of environments, this is an important consideration. Examples of such contexts are noisy environments which are bad for audio; bright or dim conditions which are bad for low contrast challenges; and social interactions which are bad for a cognitively demanding challenge.

**R12.** Minimize bandwidth usage as well as image, animation, game and audio challenge file sizes.

Design options with acceptably small bandwidth requirements should be selected. Both our user studies (Chapter 3) and our MC heuristic evaluation (Section 4.6) capture concerns regarding bandwidth usage and data plans. Users quickly got impatient with slowly loading challenges and response times. If the answer to the captcha consists of collecting data (*e.g.*, accelerometer time-series data) in which the analysis is done on the server then the size of response data files should also be considered.

**R13.** Choose input mechanisms with consideration to the ever-changing usage context of mobile devices.

While Recommendation 11 discerns the challenge and the context of use, this recommendation focuses on the input mechanisms and the input process to answer challenges. Mobile devices are used in a variety of contexts, such as standing, sitting, or walking; these conditions may have a negative impact on the input process and mechanisms employed. For example, a captcha scheme requiring accelerometers may be hindered in crowded spaces such as buses or metros. These factors impact the input process, and many lead to input mistakes.

**Summary.** The outcomes of our multiple usability studies identified recurring problems pertaining to captchas on smartphones. These design recommendations provide guidance for captcha designers and web masters when deciding which captcha schemes to include on web sites catering to mobile devices. The recommendations could, as well, help researchers evaluate novel and experimental captcha schemes.

### 6.3 Research Objectives

The overall research topic addressed in this thesis was to assess and improve the usability of captchas on smartphones. In particular, our research revolves around three primary, inter-related questions:

1. How can we effectively assess usability issues of captchas accessed on smartphones?
2. What are the most prevalent usability issues of captchas accessed on smartphones?
3. How can we improve captchas for smartphone usage?

Based on these questions, we defined four main objectives, recalled below.

**Objective 1:** Develop a set of evaluation heuristics specific to captchas, with a focus on mobile interaction.

**Objective 2:** Empirically validate the effectiveness<sup>4</sup> of the proposed set of heuristics.

**Objective 3:** Identify promising interaction methods for captchas using touch screens, then create and evaluate an alternative captcha prototype employing these new interaction methods.

**Objective 4:** Identify underlying design characteristics of successful mobile captcha schemes and generalize to develop design guidelines for captchas on smartphones.

We present below how our research contributions addressed the objectives set forth in this thesis. These contributions advance knowledge in the field of usable security through captcha research, input mechanisms, empirical and heuristic evaluations.

---

<sup>4</sup>We refer to effectiveness as the evaluators' ability to uncover problems [68].

## 6.4 Research Contributions

To meet the *first* and *second* objectives, we reviewed the academic literature and identified the most relevant papers in the area of heuristic evaluations. We summarized documentation on interaction paradigms, input methods, mobile device interaction design, captcha usability concerns, general captcha security considerations, mobile internationalization and cultural concerns, and contextual usage (Chapter 2). We developed and refined a set of mobile captcha heuristics in an iterative process [122]. We conducted two heuristic evaluations, one using our heuristics and the second using Nielsen’s heuristics. To evaluate the effectiveness of the proposed heuristics, we analyzed and compared results from these heuristic evaluations (Chapter 4). We further compared the results obtained from a user study conducted on the same schemes as in the heuristic evaluations [120, 121] (Chapter 3). The MIOR user study contributed to establishing research directions for developing the heuristics (Section 3.2). The heuristics contribute to the field of captchas by providing a tool for quick evaluations compared to a user study that spans several days or weeks. In addition to the heuristics themselves, we have shown how validation of new sets of heuristics can also be conducted by comparing between the outcomes of a user study and the outcomes of a HE using the proposed set. This contribution advances the field of usability evaluation methods.

The *third* objective was met by researching, prototyping and conducting user testing on various input methods for schemes in multiple captcha classes [121] (Chapter 5). In addition, we tested gesture-recognition technology; however, web browser implementations of these are still lacking. Gesture recognition in current mobile browsers have several constraints: browser support for W3C standards varies, there are limitations in HTML specifications, and the availability and implementation of efficient and robust multi-stroke recognizers are lacking. As part of meeting this third objective, we designed, prototyped, and tested four novel captcha scheme prototypes.

1. Gesture reCaptcha used the reCaptcha API to display challenges but used a drawing canvas to input gestured characters.
2. Gesture Emerging used Emerging Captcha character-moving challenges and a



canvas for drawing characters.

3. Asirra Slide was implemented using Asirra’s original challenges by enlarging images and presenting them in a carousel of images.
4. reCaptcha buttons displayed reCaptcha challenges normally, but input was done with buttons and menus as keyboard.

We feel that our prototypes were valuable and offered important insight into the usability of captchas on smartphones. Our results suggest that although alternative or novel input mechanisms are approached with caution by users, there is room for more research and improvement. The results are useful to the community by providing lessons learned through experience and opening doors to further research.

The prototypes, the user studies, and the new heuristics and their evaluation enabled us to provide a set of recommendations and suggestions to adapt captcha schemes for websites catering to mobile users (Chapter 6). This helped us to meet the *fourth* objective.

## 6.5 Directions for Further Research

The work compiled in this thesis has contributed to the usable security literature in the area of captchas. It has also raised further questions. In this section, we discuss other potential projects resulting from this thesis.

**Extend the Mobile Captcha heuristics.** Our evaluation heuristics for captchas could be extended to evaluate captchas for desktops. Additional considerations, such as audio schemes, could be taken into account and added to the existing heuristics, and new heuristics could be added to the existing set.

**Gesture input.** Using WoZ impacted the performance and perception of the gesture schemes. It would be desirable to do a full implementation with a robust gesture recognizer to comprehensively assess the usability of gestures as an input mechanism. We pilot tested reCaptcha (Section 5.3.1) with the \$P Point-Cloud Recognizer [119] but found its accuracy unacceptable. On the other hand, robust gesture recognition implementations can be found as native applications to the operating system (Evernote and Google apps, for example), but not browser implementations.

This implementation would add robustness and confidence to users when gesturing answers.

**Additional usability studies.** Several complementary user studies could be added to expand our current usability studies. For example, a heuristic evaluation using our MC heuristics to evaluate our prototypes could be conducted. Additionally, a user study could test schemes embedded in a web form. Such user study would provide an ecological valid assessment of the schemes. Users would get to test the schemes in the context of the main task; *e.g.*, completing a form, poll, or bid. Another study could test users' perceptions regarding gesturing the complete answer to a CR challenge versus gesturing character by character. Currently there is limited data on users' opinion regarding gesturing full answers as opposed to character by character. This testing would highlight potential technical complications for captcha designers to evaluate captcha answers. An answer including all the characters at once, as opposed to by character by character, would involve character segmentation and recognition. This study could underscore potential perceptual and technical issues regarding the challenge length versus the canvas space for gesturing.

**The use of smartphone sensors for a novel captcha class.** The use of accelerometer and gyroscope could be leveraged to create novel classes of captchas. These sensors could be used both as input mechanism to solve the character recognition challenges; and to solve a new type of movement captcha class in which the challenge would consist on tracing movement with the smartphone. The benefit of such a scheme is that it could potentially be more secure and usable than existing schemes for smartphones.

**Security and task abandonment analysis.** It would be interesting to conduct a thorough analysis of sites purposely avoiding the use of captchas when it is detected that the request comes from a mobile device. We speculate that some sites purposely do not include captchas in their web forms when these are accessed from mobile devices. The justification for not including captcha on these sites could be the abandonment of the task by users who encounter captchas while on their smartphones. In addition, currently there is no published empirical data regarding the number of users that abandon the primary task due to captchas; thus an empirical study could support and justify the use of captchas aimed at mobile devices.

## 6.6 Conclusion

In this thesis we surveyed and summarized captcha schemes, and we developed and evaluated a set of heuristics for assessing the usability of captchas aimed at smartphones. We explored whether alternative input mechanisms help improve the usability of captchas on smartphones. We focused on gesture drawing characters for solving character recognition challenges and tapping for selecting images presented as carousel of images. Our studies and work suggest that although alternative or novel input mechanisms are approached with caution by users, there is room for more research and improvement.

Returning to our research questions, our empirical work suggests that domain-specific heuristics can provide an effective method of quickly assessing captchas for use on smartphones and we recommend their use before deploying new schemes or as formative evaluation when creating new schemes. The most prevalent problems identified through our work were zooming and panning, loss of position, excessively long challenges, and small control buttons. Based on our research, we provide a set of 13 Design Recommendations to help improve the design of captchas for smartphones. Given the ubiquitous nature of mobile interactions, this will remain an important consideration for the foreseeable future.

## Bibliography

- [1] *24th IEEE International Conf. on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April 2010*. IEEE Computer Society, 2010.
- [2] Are you a human? <http://areyouahuman.com/>, 2012. Last accessed: Aug 2015.
- [3] 2captcha. <https://2captcha.com/>, 2015. Last accessed: Aug 2015.
- [4] Akismet. <http://akismet.com/>, 2013. Last accessed: Aug 2015.
- [5] N. Asokan and C. Kuo. Usable mobile security. In R. Ramanujam and S. Ramaswamy, editors, *Distributed Computing and Internet Technology*, volume 7154 of *Lecture Notes in Computer Science*, pages 1–6. Springer Berlin Heidelberg, 2012.
- [6] E. Athanasopoulos and S. Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In H. Leitold and E. Markatos, editors, *Communications and Multimedia Security*, volume 4237 of *Lecture Notes in Computer Science*, pages 97–108. Springer Berlin / Heidelberg, 2006.
- [7] Baidu. Baidu search engine. <http://tieba.baidu.com/index.html>, 2013. Last accessed: Aug 2015.
- [8] H. S. Baird and M. Luk. Protecting websites with reading-based captchas. In *Second International Workshop on Web Document Analysis (WDA2003)*, Edinburgh, UK, August 2003.
- [9] K. Baker, S. Greenberg, and C. Gutwin. Empirical development of a heuristic evaluation methodology for shared workspace groupware. In *Proc. of the 2002 ACM Conf. on computer supported cooperative work, CSCW '02*, pages 96–105, New York, NY, USA, 2002. ACM.
- [10] A. Basso and F. Bergadano. Anti-bot strategies based on human interactive proofs. In P. P. Stavroulakis and M. Stamp, editors, *Handbook of Information and Communication Security*, pages 273–291. Springer, 2010.
- [11] A. Basso and S. Sicco. Preventing massive automated access to web resources. *Computers & Security*, 28(3-4):174 – 188, 2009.
- [12] BBC. Ticketmaster dumps ‘hated’ Captcha verification system. <http://www.bbc.co.uk/news/technology-21260007>, Accessed: Feb 2013.

- [13] J. Bergman and J. Vainio. Interacting with the flow. In *Proceedings of the 12th int. conf. on human computer interaction with mobile devices and services*, MobileHCI '10, pages 249–252, New York, NY, USA, 2010. ACM.
- [14] E. Bertini, S. Gabrielli, and S. Kimani. Appropriating and assessing heuristics for mobile computing. In *Proc. of the working conf. on advanced visual interfaces*, AVI '06, pages 119–126, New York, NY, USA, 2006. ACM.
- [15] M. Blum, L. von Ahn, N. Hopper, and J. Langford. The official captcha site. <http://captcha.net/>, 2011. Last accessed: Aug 2015.
- [16] A. Botha, M. Herselman, and D. van Greunen. Mobile user experience in a mlearning environment. In *Proceedings of the 2010 Annual Research Conf. of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '10, pages 29–38, New York, NY, USA, 2010. ACM.
- [17] V. Braun and V. Clarke. *Thematic analysis*, pages 57–71. APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological. American Psychological Association, 2012.
- [18] J. Brenner. Pew internet: Mobile. <http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>, 2014. Accessed: May 2015.
- [19] G. Buchanan, S. Farrant, M. Jones, H. W. Thimbleby, G. Marsden, and M. J. Pazzani. Improving mobile internet usability. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 673–680, New York, NY, USA, 2001. ACM.
- [20] S. Burigat, L. Chittaro, and E. Parlato. Map, diagram, and web page navigation on mobile devices: the effectiveness of zoomable user interfaces with overviews. In *Proceedings of the 10th int. conf. on human computer interaction with mobile devices and services*, MobileHCI '08, pages 147–156, New York, NY, USA, 2008. ACM.
- [21] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell. The end is nigh: Generic solving of text-based captchas. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. USENIX, 2014.
- [22] E. Bursztein, R. Beauxis, H. Paskov, D. Perito, C. Fabry, and J. Mitchell. The failure of noise-based non-continuous audio captchas. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 19–31, May 2011.
- [23] E. Bursztein and S. Bethard. Decaptcha: breaking 75% of eBay audio CAPTCHAs. In *Proc. of the 3rd USENIX Conf. on Offensive Technologies*, WOOT'09, 2009.

- [24] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky. How good are humans at solving CAPTCHAs? A large scale evaluation. In *IEEE Symposium on Security and Privacy*, pages 399–413. IEEE Computer Society, 2010.
- [25] E. Bursztein, M. Martin, and J. C. Mitchell. Text-based captcha strengths and weaknesses. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM Conf. on Computer and Communications Security*, pages 125–138. ACM, 2011.
- [26] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, J. C. Mitchell, and D. Jurafsky. Easy does it: More usable captchas. CHI '14, pages 2637–2646. ACM, 2014.
- [27] J. Carroll and H. van der Meij. Ten misconceptions about minimalism. *IEEE Transactions on Professional Communication*, 39(2):72–86, jun 1996.
- [28] L. Cerejo. The elements of the mobile user experience. <http://mobile.smashingmagazine.com/2012/07/12/elements-mobile-user-experience/>, 2012. Last accessed: Jun 2013.
- [29] A. Charland and B. LeRoux. Mobile application development: Web vs. Native. *ACM Queue*, 9(4):20:20–20:28, Apr. 2011.
- [30] K. Charmaz. *Constructing grounded theory: A practical guide through qualitative analysis*. Sage Publications Limited, 2006.
- [31] J. Chattratichart and G. Lindgaard. A comparative evaluation of heuristic-based usability inspection methods. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, pages 2213–2220, New York, NY, USA, 2008. ACM.
- [32] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Building segmentation based human-friendly human interaction proofs (HIPs). In H. Baird and D. Lopresti, editors, *Human Interactive Proofs*, volume 3517 of *Lecture Notes in Computer Science*, pages 173–185. Springer Berlin / Heidelberg, 2005.
- [33] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing human friendly human interaction proofs (HIPs). In *Proc. of the SIGCHI conf. on human factors in computing systems*, CHI '05, pages 711–720, New York, NY, USA, 2005. ACM.
- [34] M. Chew and H. S. Baird. Baffletext: a human interactive proof. In T. Kanungo, E. H. B. Smith, J. Hu, and P. B. Kantor, editors, *DRR Document Recognition and Retrieval X*, volume 5010 of *SPIE Proc.*, pages 305–316. SPIE, 2003.
- [35] M. Chew and J. D. Tygar. Collaborative filtering CAPTCHAs. In H. S. Baird and D. P. Lopresti, editors, *Human Interactive Proofs: Second International*

- Workshop, HIP 2005*, volume 3517 of *Lecture Notes in Computer Science*, pages 66–81, Bethlehem, PA, USA, May 2005. Springer-Verlag, Berlin Germany.
- [36] R. Chow, P. Golle, M. Jakobsson, L. Wang, and X. Wang. Making captchas clickable. *HotMobile '08*, pages 91–94. ACM, 2008.
- [37] Confident Technologies. Confident CAPTCHA. <http://www.confidenttechnologies.com/products/confident-captcha>, 2013. Accessed: Jun 2013.
- [38] J. Cui, W. Zhang, Y. Peng, Y. Liang, B. Xiao, J. Mei, D. Zhang, and X. Wang. A 3-layer Dynamic CAPTCHA Implementation. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 1, pages 23–26, March 2010.
- [39] J.-S. Cui, J.-T. Mei, X. Wang, D. Zhang, and W.-Z. Zhang. A CAPTCHA Implementation Based on 3D Animation. In *Multimedia Information Networking and Security, 2009. MINES '09. International Conf. on*, volume 2, pages 179–182, Nov. 2009.
- [40] J.-S. Cui, J.-T. Mei, W.-Z. Zhang, X. Wang, and D. Zhang. A CAPTCHA Implementation Based on Moving Objects Recognition Problem. In *E-Business and E-Government (ICEE), 2010 International Conf. on*, pages 1277–1280, may 2010.
- [41] Y. Cui and V. Roto. How people use the web on mobile devices. In *Proceedings of the 17th int. conf. on World Wide Web, WWW '08*, pages 905–914, New York, NY, USA, 2008. ACM.
- [42] M. Dailey and C. Namprempre. A text graphics character CAPTCHA for password authentication. In *TENCON 2004. 2004 IEEE Region 10 Conf.*, volume 2, pages 45–48, Nov. 2004.
- [43] A. Desai and P. Patadia. Drag and drop: A better approach to captcha. In *India Conf. (INDICON), 2009 Annual IEEE*, pages 1–4, Dec. 2009.
- [44] R. Dhamija and J. Tygar. Phish and hips: Human interactive proofs to detect phishing attacks. In H. Baird and D. Lopresti, editors, *Human Interactive Proofs*, volume 3517 of *Lecture Notes in Computer Science*, pages 69–83. Springer Berlin / Heidelberg, 2005.
- [45] T. Down. Log4javascript logging framework. <http://log4javascript.org/>, 2013. Last accessed: Oct 2014.
- [46] H. B.-L. Duh, G. C. B. Tan, and V. H.-h. Chen. Usability evaluation for mobile device: a comparison of laboratory and field tests. In *Proc. of the 8th conf. on*

*human-computer interaction with mobile devices and services*, MobileHCI '06, pages 181–186, New York, NY, USA, 2006. ACM.

- [47] eBay. Send a message to a seller. <http://www.ebay.com>, 2013. Last accessed: Nov 2013.
- [48] M. Egele, L. Bilge, E. Kirda, and C. Kruegel. Captcha smuggling: hijacking web browsing sessions to create captcha farms. In *Proc. of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1865–1870, New York, NY, USA, 2010. ACM.
- [49] A. El Ahmad, J. Yan, and W. Ng. Captcha design: colour, usability and security. *Internet Computing, IEEE*, 16(2):44–51, 2012.
- [50] A. S. El Ahmad, J. Yan, and L. Marshall. The robustness of a new captcha. In *Proc. of the Third European Workshop on System Security*, EUROSEC '10, pages 36–41, New York, NY, USA, 2010. ACM.
- [51] J. Elson, J. R. Douceur, and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proc. of the 14th ACM conf. on computer and communications security*, CCS '07, pages 366–374, New York, NY, USA, 2007. ACM.
- [52] Facebook. Facebook, comment protection. <https://www.facebook.com>, 2013. Last accessed: Mar 2013.
- [53] H. Gao, D. Yao, H. Liu, X. Liu, and L. Wang. A novel image based captcha using jigsaw puzzle. In *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conf. on*, pages 351–356, dec. 2010.
- [54] P. Golle. Machine learning attacks against the Asirra CAPTCHA. CCS '08, pages 535–542. ACM, 2008.
- [55] Google. Gmail account creation. <https://accounts.google.com/NewAccount>, 2011. Last Accessed: Oct 2012.
- [56] Google. reCaptcha: Stop Spam, Read Books. <http://www.google.com/recaptcha>, 2013. Last Accessed: Oct 2012.
- [57] Google. Handwrite. <http://www.google.com/insidesearch/features/search/handwritinginput/>, 2014. Last Accessed: Dec 2014.
- [58] Google. Gif definition. <https://www.google.com/search?q=gif+definition&ie=utf-8&oe=utf-8>, 2015. Last Accessed: Jan 2015.
- [59] R. Gossweiler, M. Kamvar, and S. Baluja. What's up CAPTCHA?: a CAPTCHA based on image orientation. WWW '09, pages 841–850, New York, NY, USA, 2009. ACM.



- [60] S. Greenberg, G. Fitzpatrick, C. Gutwin, and S. M. Kaplan. Adapting the locales framework for heuristic evaluation of groupware. *Australasian Jnl of Information Syst.*, 7(2), 2000.
- [61] P. Hagen, T. Robertson, M. Kan, and K. Sadler. Emerging research methods for understanding mobile technology use. In *Proceedings of the 17th Australia conf. on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future*, OZCHI '05, pages 1–10, Narrabundah, Australia, Australia, 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.
- [62] R. V. Hall Jr. Captcha as a web security control. <http://www.richhall.com/captcha/>, Dec 2005. Last Accessed: Feb 2012.
- [63] H. R. Hartson, T. S. Andre, and R. C. Williges. Criteria for evaluating usability evaluation methods. *Int. Journal of Human-Computer Interaction*, 13(4):373–410, 2001.
- [64] J. M. G. Hidalgo and G. Alvarez. Captchas: An artificial intelligence application to web security. In M. V. Zelkowitz, editor, *Advances in Computers*, volume 83, pages 109 – 181. Elsevier, 2011.
- [65] M. Hiltunen, M. Laukka, and J. Luomala. *Mobile user experience*. IT press, 2002.
- [66] E. T. Hvannberg, E. L.-C. Law, and M. K. Lrusdttir. Heuristic evaluation: Comparing ways of finding and reporting usability problems. *Interacting with Computers*, 19(2):225–240, 2007.
- [67] P. Jaferian, K. Hawkey, A. Sotirakopoulos, and K. Beznosov. Heuristics for evaluating IT security management tools. In D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, editors, *CHI Extended Abstracts*, pages 1633–1638. ACM, 2011.
- [68] P. Jaferian, K. Hawkey, A. Sotirakopoulos, M. Velez-Rojas, and K. Beznosov. Heuristics for evaluating it security management tools. In *Proc. of the Seventh Symposium on Usable Privacy and Security*, SOUPS '11, pages 7:1–7:20, New York, NY, USA, 2011. ACM.
- [69] M. Jakobsson. Captcha-free throttling. In D. Balfanz and J. Staddon, editors, *AISec 2009*, pages 15–22. ACM, 2009.
- [70] L. James L. User interface design for the mobile web. <http://www.ibm.com/developerworks/web/library/wa-interface/index.html>, 2012. Last accessed: Jan 2013.

- [71] J. Kjeldskov. “Just-in-Place” information for mobile device interfaces. *Lecture Notes in Computer Science*, 2411:271–275, 2002.
- [72] J. Kjeldskov and C. Graham. A review of mobile HCI research methods. In *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 317–335. Springer Berlin / Heidelberg, 2003.
- [73] J. Kjeldskov and J. Stage. New techniques for usability evaluation of mobile systems. *Int. Journal of Human-Computer Studies*, 60(56):599 – 620, 2004.
- [74] K. A. Kluever and R. Zanibbi. Balancing usability and security in a video captcha. In *Proc. of the 5th Symposium on Usable Privacy and Security, SOUPS '09*, pages 14:1–14:11, New York, NY, USA, 2009. ACM.
- [75] H. Korhonen and E. M. I. Koivisto. Playability heuristics for mobile games. In *Proc. of the 8th conf. on human-computer interaction with mobile devices and services, MobileHCI '06*, pages 9–16, New York, NY, USA, 2006. ACM.
- [76] J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. John Wiley and Sons, 2010.
- [77] I. Lee, J. Kim, and J. Kim. Use contexts for the mobile internet: A longitudinal study monitoring actual use of mobile internet services. *International Journal of Human-Computer Interaction*, 18(3):269–292, 2005.
- [78] S. Li, S. A. H. Shah, M. A. U. Khan, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz. Breaking e-banking captchas. In *Proc. of the 26th Annual Computer Security Applications Conf., ACSAC '10*, pages 171–180, New York, NY, USA, 2010. ACM.
- [79] C.-J. Liao, C.-J. Yang, J.-T. Yang, H.-Y. Hsu, and J.-W. Liu. A game and accelerometer-based captcha scheme for mobile learning system. In J. Herington, A. Couros, and V. Irvine, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2013*, pages 1385–1390, Victoria, Canada, June 2013. AACE.
- [80] W.-H. Liao and C.-C. Chang. Embedding information within dynamic visual patterns. In *Multimedia and Expo, IEEE International Conf. on*, volume 2, pages 895–898, june 2004.
- [81] R. Lin, S.-Y. Huang, G. B. Bell, and Y.-K. Lee. A new captcha interface design for mobile devices. In *ACSW 2011: Australasian User Interface Conf.*, 2011.
- [82] A. Lisa and W. Jacob O. \$N\$ limitations. <http://depts.washington.edu/aimgroup/proj/dollar/limits/index.html>, 2014. Last accessed: Feb 2015.

- [83] Y. Liu and K.-J. R aih a. Rotatxt: Chinese pinyin input with a rotator. In *Proceedings of the 10th int. conf. on human computer interaction with mobile devices and services*, MobileHCI '08, pages 225–233, New York, NY, USA, 2008. ACM.
- [84] R. Lowry. *Concepts and Applications of Inferential Statistics*. Vassar College, <http://faculty.vassar.edu/lowry/webtext.html>, 1998.
- [85] I. MacKenzie and R. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17(2-3):147–198, 2002.
- [86] J. Mankoff, A. K. Dey, G. Hsieh, J. A. Kientz, S. Lederer, and M. Ames. Heuristic evaluation of ambient displays. In G. Cockton and P. Korhonen, editors, *CHI*, pages 169–176. ACM, 2003.
- [87] P. Matthews and C. C. Zou. Scene tagging: image-based captcha using image composition and object relationships. In *Proc. of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 345–350, New York, NY, USA, 2010. ACM.
- [88] A. McMenamin. The end of the CAPTCHA? <http://cartesianproduct.wordpress.com/2012/02/20/the-end-of-the-captcha/>, 2012. Last accessed: Feb 2013.
- [89] S. Media. Solve media - effective ad campaigns. <http://solvemedia.com/security>, 2014. Last accessed: Dec 2014.
- [90] H. Meutzner, V.-H. Nguyen, T. Holz, and D. Kolossa. Using automatic speech recognition for attacking acoustic captchas: The trade-off between usability and security. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, pages 276–285, New York, NY, USA, 2014. ACM.
- [91] Microsoft. Visual C++ cross-platform mobile. <https://www.visualstudio.com/en-us/features/cplusplus-mdd-vs.aspx>, 2015. Accessed: Mar 2015.
- [92] Microsoft. Asirra (Animal Species Image Recognition for Restricting Access). <http://research.microsoft.com/en-us/um/redmond/projects/asirra/>, 2012. Last Accessed: Jan 2013.
- [93] Microsoft. Windows Live, Hotmail account creation. <https://signup.live.com/signup.aspx>, 2015. Last accessed: Feb 2015.
- [94] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or. Emerging images. *ACM Transactions on Graphics*, 28(5), 2009.

- [95] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W.-B. Chen. A three-way investigation of a game-captcha: Automated attacks, relay attacks and usability. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 195–206, New York, NY, USA, 2014. ACM.
- [96] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In *Conf. on Computer Vision and Pattern Recognition, 2003. Proc.. 2003 IEEE Computer Society*, volume 1, pages I–134 – I–141 vol.1, June 2003.
- [97] G. Mori and J. Malik. Breaking a visual captcha. <http://www.cs.sfu.ca/~mori/research/gimpy/>, 2013. "Last accessed: Feb 2015".
- [98] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: CAPTCHAs-understanding CAPTCHA-solving services in an economic context. In *USENIX Security Symposium*, pages 435–462. USENIX Association, 2010.
- [99] G. Moy, N. Jones, C. Harkless, and R. Potter. Distortion estimation techniques in solving visual captchas. In *Conf. on Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proc. of the 2004 IEEE Computer Society*, volume 2, pages II–23 – II–28 Vol.2, June-2 July 2004.
- [100] M. Naor. Verification of a human in the loop or identification via the Turing Test, Sept. 1996.
- [101] J. Nielsen. Introduction to usability. <http://www.useit.com/alertbox/20030825.html>, 2011. Last accessed: Apr 2013.
- [102] J. Nielsen. Heuristic evaluation: How to. <http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/>, 2012. Last accessed: May 2015.
- [103] J. Nielsen. Ten usability heuristics. <http://www.nngroup.com/articles/ten-usability-heuristics/>, 2012. Last accessed: May 2015.
- [104] J. Nielsen. Heuristic evaluation. <http://www.nngroup.com/topic/heuristic-evaluation>, 2013. Last accessed: May 2015.
- [105] Nielsen Co. The asian mobile consumer decoded. <http://www.nielsen.com/us/en/newswire/2013/the-asian-mobile-consumer-decoded0.html>, 2013. Last accessed: Jul 2013.
- [106] Nielsen Co. Mobile majority: U.S. smartphone ownership tops 60%. <http://www.nielsen.com/us/en/newswire/2013/mobile-majority--u-s--smartphone-ownership-tops-60-.html>, 2013. Last accessed: Jul 2013.

- [107] Nielsen Co. Who is the mobile shopper? <http://www.nielsen.com/us/en/newswire/2013/who-is-the-mobile-shopper-.html>, 2013. Last accessed: Jul 2013.
- [108] Nielsen, Jakob, Mack, and R. L., editors. *Usability Inspection Methods*. Number 14. John Wiley & Sons, New York, 1994.
- [109] Nielsen, Jakob and Molich, Rolf. Heuristic evaluation of user interfaces. In *Proc. of ACM CHI'90 Conf. on Human Factors in Computing Systems*, pages 249–256, 1990.
- [110] NuCaptcha. White paper: Nucaptcha and traditional captcha. <http://www.nucaptcha.com/resources/whitepapers>, 2012. Last accessed: Feb 2013.
- [111] T. Paek, K. Chang, I. Almog, E. Badger, and T. Sengupta. A practical examination of multimodal feedback and guidance signals for mobile touchscreen keyboards. In *Proceedings of the 12th int. conf. on human computer interaction with mobile devices and services*, MobileHCI '10, pages 365–368, New York, NY, USA, 2010. ACM.
- [112] PetFinder. Petfinder, Pet Adoption. <https://www.petfinder.com/>, 2014. Last accessed: May 2014.
- [113] H. Petrie and C. Power. What do users really care about?: a comparison of usability problems found by users and experts on highly interactive websites. In *Proc. of the 2012 ACM annual conf. on Human Factors in Computing Systems*, CHI '12, pages 2107–2116, New York, NY, USA, 2012. ACM.
- [114] Picatcha. PICATCHA: Image Captcha. <http://www.picatcha.com>, 2014. Last accessed: Jun 2015.
- [115] D. Pinelle, N. Wong, and T. Stach. Heuristic evaluation for games: usability principles for video game design. In *Proc. of the twenty-sixth annual SIGCHI conf. on human factors in computing systems*, CHI '08, pages 1453–1462, New York, NY, USA, 2008. ACM.
- [116] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *Proc. of the 9th ACM conf. on computer and communications security*, CCS '02, pages 161–170, New York, NY, USA, 2002. ACM.
- [117] S. Po, S. Howard, F. Vetere, and M. B. Skov. Heuristic evaluation and mobile usability: Bridging the realism gap. In S. A. Brewster and M. D. Dunlop, editors, *Mobile Human-Computer Interaction - Mobile HCI, 13-16, 2004.*, volume 3160 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2004.

- [118] W. W. Preston, J. Goulding, G. Burnett, and M. Jackson. Heuristics & usability of virtual attack points for pedestrian navigation: User study using paper-prototyping. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS '12*, pages 65–72, New York, NY, USA, 2012. ACM.
- [119] V. Radu-Daniel, A. Lisa, and W. Jacob O. \$P Point-Cloud Recognizer. <http://depts.washington.edu/aimgroup/proj/dollar/pdollar.html>, 2014. Last accessed: Oct 2014.
- [120] G. Reynaga and S. Chiasson. The usability of CAPTCHAs on smartphones. In *International Conf. on Security and Cryptography, (SECRYPT 2013)*, pages 427–434. SCITEPRESS, August 2013.
- [121] G. Reynaga, S. Chiasson, and P. C. van Oorschot. Exploring the usability of captchas on smartphones: Comparisons and recommendations. In *Workshop on Usable Security USEC*. Internet Society, February 2015.
- [122] G. Reynaga, S. Chiasson, and P. C. van Oorschot. Heuristics for the evaluation of captchas on smartphones. In *British HCI, BHCI '15*. ACM, July 2015.
- [123] Y. Rogers, H. Sharp, and J. Preece. *Interaction Design: beyond human-computer interaction, 3rd Edition*. John Wiley & Sons, 2011.
- [124] Y. Rui and Z. Liu. Artificial: Automated reverse turing test using facial features. *Multimedia Systems*, 9:493–502, 2004.
- [125] J. Sauro and E. Kindlund. A method to standardize usability metrics into a single score. In *Proc. of the SIGCHI conf. on human factors in computing systems, CHI '05*, pages 401–409, New York, NY, USA, 2005. ACM.
- [126] A. Saxena, N. S. Chauhan, K. R. Sravan, A. S. V. Vangal, and D. P. Rodriguez. A new scheme for mobile based captcha service on cloud. In *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conf. on*, pages 1–6, Oct. 2012.
- [127] H. Sharp, Y. Rogers, and J. Preece. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Indianapolis, IN, 2 edition, 2007.
- [128] M. Shirali-Shahreza and S. Shirali-Shahreza. Question-based captcha. In *Conf. on Computational Intelligence and Multimedia Applications, 2007. International Conf. on*, volume 4, pages 54–58, dec. 2007.
- [129] M. Shirali-Shahreza and S. Shirali-Shahreza. Motion CAPTCHA. In *Conf. on Human System Interactions*, pages 1042–1044, May 2008.

- [130] S. Shirali-Shahreza, G. Penn, R. Balakrishnan, and Y. Ganjali. Seesay and hearsay captcha for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2147–2156, New York, NY, USA, 2013. ACM.
- [131] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, aug. 1983.
- [132] S. Shrestha. Mobile web browsing: usability study. In *Proceedings of the 4th int. conf. on mobile technology, applications, systems and the 1st int. symposium on computer human interaction in mobile technology*, Mobility '07, pages 187–194, New York, NY, USA, 2007. ACM.
- [133] A. Sinkov. Handwriting Arrives in Evernote for Android. <https://blog.evernote.com/blog/2014/03/05/handwriting-arrives-evernote-android/>, 2014. Accessed: Apr 2015.
- [134] Snapchat. Snapchat. <https://www.snapchat.com/>, 2014. Last accessed: Nov 2014.
- [135] C. Stephanidis and D. Akoumianakis. Universal design: towards universal access in the information society. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 499–500, New York, NY, USA, 2001. ACM.
- [136] C. Strilechi and M. Vaida. A web 3.0 solution for restraining the web-bots access to the on-line displayed content. In *Information Technology Interfaces, 2009. ITI '09. Proc. of the ITI 2009 31st International Conf. on*, pages 633–638, june 2009.
- [137] A. G. Sutcliffe and B. Gault. Heuristic evaluation of virtual reality applications. *Interacting with Computers*, 16(4):831–849, 2004.
- [138] J. Tam, J. Simsa, D. Huggins-daines, L. V. Ahn, and M. Blum. Improving audio CAPTCHAs. In *The Symposium on Accessible Privacy and Security (SOAPS '08)*, USA, July 2008.
- [139] A.-J. Thompson and E. A. Kemp. Web 2.0: extending the framework for heuristic evaluation. In *Proc. of the 10th Int. Conf. NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, CHINZ '09, pages 29–36, New York, NY, USA, 2009. ACM.
- [140] K. Vaananen-Vainio-Mattila and M. Waljas. Evaluating user experience of cross-platform web services with a heuristic evaluation method. *Int. J. of Arts and Technology*, 3:402–421, Oct. 05 2010.

- [141] P. C. van Oorschot and S. Stubblebine. On countering online dictionary attacks with login histories and humans-in-the-loop. *ACM Trans. Inf. Syst. Secur.*, 9:235–258, August 2006.
- [142] Vappic. 4D CAPTCHA. <http://www.vappic.com/moreplease>, 2012. Last accessed: Nov 2013.
- [143] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *EUROCRYPT*, volume 2656. 2003.
- [144] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Commun. ACM*, 47:56–60, February 2004.
- [145] W3C. Localization vs. Internationalization. <http://www.w3.org/International/questions/qa-i18n>, 2012. Last accessed: Oct 2012.
- [146] W3C. Mobile Web Best Practices 1.0. <http://www.w3.org/TR/mobile-bp/>, 2012. Last accessed: Oct 2012.
- [147] L. Wang, X. Chang, Z. Ren, H. Gao, X. Liu, and U. Aickelin. Against spyware using CAPTCHA in graphical password scheme. In *AINA* [1], pages 760–767.
- [148] A. J. Wismer, K. C. Madathil, R. Koikkara, K. A. Juang, and J. S. Greenstein. Evaluating the usability of captchas on a mobile device with voice and touch input. In *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, volume 56, pages 1228–1232. SAGE Publications, 2012.
- [149] Xamarin Inc. Xamarin Test Cloud. <http://xamarin.com/test-cloud>, 2015. Last accessed: May 2015.
- [150] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monroe, and P. C. van Oorschot. Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion. In *Proc. of the 21st USENIX Security Symposium*, Berkeley, CA, USA, 2012. USENIX Association.
- [151] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monroe, and P. C. van Oorschot. Security analysis and related usability of motion-based captchas: Decoding codewords in motion. *IEEE TDSC*, 11(5):480–493, Sept 2014.
- [152] Yahoo! Yahoo! Account Creation. <https://edit.yahoo.com/registration>, 2011. Last accessed: May 2012.
- [153] T. Yamamoto, J. D. Tygar, and M. Nishigaki. CAPTCHA using strangeness in machine translation. In *AINA* [1], pages 430–437.
- [154] J. Yan. Bot, Cyborg and automated Turing test. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols Workshop*, volume 5087 of *Lecture Notes in Computer Science*, pages 198–201. Springer, 2006.



- [155] J. Yan and A. El Ahmad. Breaking visual CAPTCHAs with naive pattern recognition algorithms. *ACSAC*, pages 279–291, Dec. 2007.
- [156] J. Yan and A. El Ahmad. Captcha security: A case study. *Security Privacy, IEEE*, 7(4):22–28, July-Aug. 2009.
- [157] J. Yan and A. S. El Ahmad. A low-cost attack on a Microsoft Captcha. *ACM CCS '08*, pages 543–554.
- [158] J. Yan and A. S. El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. *SOUPS '08*, pages 44–52, New York, NY, USA, 2008. ACM.
- [159] S. Zhai and P. O. Kristensson. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM*, 55(9):91–101, Sept. 2012.
- [160] A. Zhou, J. Blustein, and N. Zincir-Heywood. Improving intrusion detection systems through heuristic evaluation. In *Canadian Conf on Electrical and Computer Engineering, 2004.*, volume 3, pages 1641 – 1644, May 2004.
- [161] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and design of image recognition CAPTCHAs. *CCS '10*, pages 187–200. ACM, 2010.

## Appendix A

### Nielsen and Molich's Heuristics

Nielsen and Molich proposed a set of heuristics for evaluating the usability of user interfaces [109]. The original set of heuristics have been refined since they were originally proposed. Below we list the most recent iteration of Nielsen's heuristics [103].

1. **Visibility of system status.** The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. **Match between system and the real world.** The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom.** Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. **Consistency and standards.** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention.** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. **Recognition rather than recall.** Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use.** Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. **Aesthetic and minimalist design.** Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. **Help users recognize, diagnose, and recover from errors.** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10. **Help and documentation.** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

## Appendix B

### Bertini *et al.* Heuristics

Bertini *et al.* [14] propose a set of heuristics to evaluate usability issues in general applications for *mobile computing*, with emphasis on contextual usage.

1. **Visibility of system status and losability/findability of the mobile device.** Through the mobile device, the system should always keep users informed about what is going on. Moreover, the system should prioritize messages regarding critical and contextual information such as battery status, network status, environmental conditions, etc. Since mobile devices often get lost, adequate measures such as encryption of the data should be taken to minimize loss. If the device is misplaced, the device, system or application should make it easy to find it back.
2. **Match between system and the real world.** Enable the mobile user to interpret correctly the information provided, by making it appear in a natural and logical order; whenever possible, the system should have the capability to sense its environment and adapt the presentation of information accordingly.
3. **Consistency and mapping.** The users conceptual model of the possible function/interaction with the mobile device or system should be consistent with the context. It is especially crucial that there be a consistent mapping between user actions/interactions (on the device buttons and controls) and the corresponding real tasks (e.g. navigation in the real world).
4. **Good ergonomics and minimalist design.** Mobile devices should be easy and comfortable to hold/carry along as well as robust to damage (from environmental agents). Also, since screen real estate is a scarce resource, use it with parsimony. Dialogues should not contain information which is irrelevant or rarely needed.

5. **Ease of input, screen readability and glancability.** Mobile systems should provide easy ways to input data, possibly reducing or avoiding the need for the user to use both hands. Screen content should be easy to read and navigate through notwithstanding different light conditions. Ideally, the mobile user should be able to quickly get the crucial information from the system by glancing at it.
6. **Flexibility, efficiency of use and person- alization.** Allow mobile users to tailor/personalize frequent actions, as well as to dynamically configure the system according to contextual needs. Whenever possible, the system should support and suggest system-based customization if such would be crucial or beneficial.
7. **Aesthetic, privacy and social conventions.** Take aesthetic and emotional aspects of the mobile device and system use into account. Make sure that users data are kept private and safe. Mobile interaction with the system should be comfortable and respectful of social conventions.
8. **Realistic error management.** Shield mobile users from errors. When an error occurs, help users to recognize, to diagnose, if possible to recover from the error. Mobile computing error messages should be plain and precise. Constructively suggest a solution (which could also include hints, appropriate FAQs, etc). If there is no solution to the error or if the error would have negligible effect, enable the user to gracefully cope with the error.