ACRAS
A Hybrid Graphical User-Authentication System

H Ryan Harasimowicz

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Cyber Security Engineering

University of Washington

2018

Reading Committee:

Brent Lagesse, PhD., Chair

Marc Dupuis, PhD.

David LeBlanc, PhD.

Program Authorized to Offer Degree:
Computing & Software Systems

University of Washington

Abstract

ACRAS
A Hybrid Graphical User-Authentication System

H Ryan Harasimowicz

Chair of the Supervisory Committee:
Associate Professor Brent Lagesse, PhD.
Computing & Software Systems

The traditional text-based password is ubiquitous in today's computing environment, yet creation and maintenance of both usable and secure passwords remains one of the largest challenges in modern computing. This project investigates an alternative authentication mechanism to the traditional static text-based password. The Algorithmic Challenge/Response Authentication System (ACRAS) is a single-factor, one-time-password system based on the accurate recognition and interpretation of user-defined graphic characteristics within a set of challenge graphics. There is broad consensus that the human mind excels at graphic recognition and cued recall when compared to the abstract rote memorization of a complex string of text. ACRAS leverages this innate ability of the human mind; providing a framework for system users to define a set of rules for the recognition and processing of select characteristics of graphic challenges. Application of these easily-recalled rules deterministically generates a one-time-password string that is dependent upon the session's randomly selected set of challenge graphics. As a one-time-password system, ACRAS is inherently resistant to some of the more common attacks on traditional authentication systems and suggests an increase of protection against others as compared to these static systems. A series of user-experiments have been conducted with an ACRAS prototype to gauge usability and overall user impression of the system.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

BRUTE-FORCE ATTACK: An attack on a password authentication system where an attacker conducts serial attempts to guess a password by exhaustively attempting all possible variations.

CHALLENGE/RESPONSE AUTHENTICATION: A family of protocols for authentication purposes wherein a challenge question or prompt is presented to a user who in-turn provides a specific response to be authenticated.

FAMILIARITY-BASED GUESSING ATTACK: A means of narrowing the scope of a brute-force attack by focusing on subjects related to the user whose account is being attacked.

GRAPHICAL PASSWORD AUTHENTICATION: An authentication system based on the accurate selection of one or more graphical images in a specific order.

ONE TIME PASSWORD: An authentication system which uses a dynamic password that is valid for only a single session and is changed after each subsequent session.

MOVING TARGET DEFENSE: The appropriation of typically offensive-oriented dynamism of configuration, environment, etc. applied to defensive measures to increase uncertainty and complexity while reducing the window of availability for attack.

MULTI-FACTOR AUTHENTICATION: An enhanced authentication protocol where a successful authentication requires multiple factors: something you know (e.g. password); something you have (e.g. hardware device); something you are (e.g. biometric). As opposed to single-factor authentication which may simply consist of one of the aforementioned properties: something one knows, has or is.

SYMBOL DIMENSIONALITY: The plurality of representative characteristics exhibited by the presentation of a symbol (alpha-numeric character, shape, etc.). Examples include: color, size, outline/fill, value, English representation, etc.

SHOULDER-SURFING ATTACK: An attack on a password authentication system where an adversary is able to observe the successful entry of a password. May include challenge data if relevant.

SPYWARE: A malicious program installed on a user's computer that tracks user activity including taking screenshots, and logging keystrokes.

ACKNOWLEDGMENTS

First, I would like to thank my committee members who have been incredibly generous with their expertise and time. A special thank you to Professor Brent Lagesse of the Computing & Software Systems Department at the University of Washington. Dr. Lagesse's encouragement and guidance over the last few years has consistently led to fascinating corners of emerging computing research, and has in-turn allowed me to develop my own rewarding research trajectory. In addition, I would like to thank Professors Marc Dupuis and David LeBlanc for agreeing to serve on my thesis committee. Your support and advice has helped immensely. Finally, a thank you to my family for their unconditional support and patience while chasing after this crazy aspiration.

DEDICATION

to Ivy, Catherine, and Eileen, who's unwavering love,
patience, and support has made this possible

Chapter 1

INTRODUCTION

As contemporary society's utilization and reliance upon computing systems continues to grow, the security demands of these systems and system-access grows in a continual lock-step. At the same time, the resulting attack surface expansion coupled with the traditional model of relatively static system-configurations present easy marks for would-be attackers to target. The recurrent security shortcoming found within traditional computing systems (as evidenced by media announcement of breach after breach) suggests alternative models are needed to address the present imbalance. As the importance and prevalence of computing systems within our society is not likely to lessen in the near-term, one promising alternative to traditional static systems falls under the classification of Moving Target Defenses (MTDs). MTDs appropriate the dynamic principles commonly found in assailant system-configurations, and reflect them back as a means of introducing system variability and resulting uncertainty into target systems, thereby complicating the attack process.

One area where such an MTD approach has potential to have an impact on system security is user-authentication. The traditional textual username/password is ubiquitous within our computational systems and yet it is inherently weak and subject to many potential attacks. The static nature of traditional text-based passwords makes them susceptible to many types of attack. Common attacks on password authentication include: brute-force attacks, where an attacker exhaustively tries every permutation of available characters until the correct password is identified; dictionary, and familiarity-based attacks where an attacker intelligently narrows the field of possible permutations; shoulder-surfing, where an attacker is able to observe password entry in-person; and spyware such as a key-loggers. Further, the traditional password model does not scale well. Rote memorization and recall of pass-

words of sufficient complexity is generally an arduous task. In common practice, users have too many accounts to maintain unique passwords for each which frequently results in users writing them down or reusing passwords for multiple accounts thereby weakening their security. Alternatives to traditional username/password systems including graphical-passwords, and challenge-response systems have been developed to capitalize upon the human mind's facility with graphic recognition as compared to recall [12]. Unfortunately, the state-space for such systems is typically not very large, leaving it vulnerable to brute-force similar to the static textual password. The introduction of two-factor authentication addresses some of the fundamental weaknesses of static passwords, yet such augmentations are reliant upon either special hardware (mobile-devices or hardware-keys for instance) or present the risk of associating and potentially losing the privacy of biometric information which cannot be changed.

Previously mentioned MTD characteristics can be applied to enhance the traditional user authentication domain via one-time passwords (OTP), wherein a user's password is valid only for a single session and each subsequent authentication requires a new, unique password. OTP schemas offer improved resistance to attack over that provided by traditional alternatives. Brute-force attacks become single-guess chances as the password changes after each attempt. Educated guessing attacks (familiarity with user preferences, tendencies, etc.) may be less effective depending on the implementation of the system. The dynamic nature of OTPs could result in sufficient variety beyond a user's predilections although breadth of password space need be balanced by a reasonable means of producing the password. Short of introducing a second-factor like a hardware token, time-based token, or predetermined second-channel, some target familiarity could be telling if inappropriately incorporated into the system. Shoulder-surfing attacks are less likely to succeed with an OTP scheme than with a traditionally static credential, simply by the fact that a given password will not be valid on a subsequent authorization.

1.1 Motivation

Development of an easy to use OTP system that does not require a second factor would provide a more secure alternative to the traditional static password system which despite all of its faults, remains prevalent in contemporary computing. Ultimately the stakeholders for such a system are not limited to any one specific group. As previously mentioned, passwords are pervasive throughout society's computing systems - we all use them, and as a result we are all subject to the same weaknesses and shortcomings within the system. By extension, society as a whole could benefit from an improvement in user authentication security.

The following thesis presents the Algorithmic Challenge/Response Authentication System (ACRAS), a user-defined algorithmic approach to an OTP system. Capitalizing on the relative ease of graphic recognition and methodical application of patterns as compared to rote memorization/recall, the ACRAS system employs a graphic challenge prompting a textual response which allows the system to be functional across multiple platforms (mobile devices, PC, etc.). The ACRAS system leverages the plurality of symbol (alpha-numeric character, shape, word, etc.) representation and dimensionality to increase the potential state space of passwords within a limited component set. As an OTP system, the proposed system is inherently resistant to brute-force attacks. Additionally, the system presents resistance to familiarity-oriented guessing attacks through the use of common shapes and symbols and demonstrates additional protections against shoulder-surfing attacks.

Success of the ACRAS system will be considered based on the two underlying principles of security and usability. Security in this context relates to the ACRAS system's resistance to common attacks. A series of quantifiable analyses will be described in which resistance to these common attacks is evaluated. Usability, in-turn relates to how easily a system user can register, recognize, recall and ultimately respond to the ACRAS challenge. A series of user-experiments will be detailed in which these usability aspects are assessed. Experiments will include evaluation of duration-oriented metrics for different operations and accuracy figures for subject responses. Additionally, the results from a short user-survey on ACRAS system

impression will be presented. The remainder of this thesis document is organized according to the following: Background and related work are discussed in Chapter 2. Chapter 3 establishes an attacker model and benchmarks for system success. Project methodology follows in Chapter 4, and the results to user-experiments are presented and discussed in Chapter 5. Chapter 6 includes discussion of the success of the ACRAS system as a whole, and opportunities for future work. Chapter 7 concludes the thesis document.

Chapter 2

BACKGROUND & RELATED WORK

2.1 The Problem(s) with Passwords

As previously introduced, there are numerous shortcomings within traditional static user-name/password authentication mechanisms. For motivated adversaries, passwords have proven to be relatively easy to ascertain, either by means of brute-force attacks, dictionary-attacks or social-engineering efforts. Secure passwords which are resistant to these types of attack are difficult to create, and even more difficult to remember which frequently results in users re-using passwords. A 2016 survey sponsored by TeleSign[22] suggests that as many as 73% of users utilize the same password across multiple accounts (likely also associated with the same user-name). The probability of compromise for such a password is increased with the additional exposure; and once compromised, other accounts which share the credentials are endangered as a result.

In their 2014 work, The Password Life Cycle: User Behaviour in Managing Passwords[25], Stobert and Biddle analyze real-world user strategies and techniques for management of traditional password credentials. Their research includes the categorical organization of common user practices and a means for comparing these real-world applications with current best practices. Stobert and Biddle propose such a comparison lends itself to identifying gaps, with opportunities for focused improvement in authentication systems which may be tailored to better suit user practices rather than forcing users to conform to ill-suited frameworks. Through the course of their research, they distilled user practice into a generalized pattern for the lifecycle of user credentials which includes multiple forks leading to password reuse. The conclusions drawn include support for single-source management tools, either password managers or single sign-on services, and a suggestion for the incorporation of non-secret

graphical cues to help users recall a password. According to their research, the graphical information need not be directly related to the password, yet the recognition of the image cue would help users to associate their password with the current site. This graphical recognition is the same memory mechanism proposed for the ACRAS system, only extending it from recognition to interpretation in order to introduce a dynamic element into the authentication process.

2.2 Moving Target Defenses

Within information security, there is a burgeoning classification of defensive strategies referred to as Moving Target Defenses (MTDs). Historically associated with guerrilla warfare tactics, MTD strategies are increasingly gaining traction within the information security field as a means to reset or invert the current imbalance of traditional attacker/target exposure. MTDs rely upon the added complexity of a dynamic environment to make target systems more difficult to analyze and prepare-for, in addition to limiting their exposure to attack. In their paper, Finding Focus in the Blur of Moving-Target Techniques[19], Okhravi, Hobson, Bigelow, and Streilein propose a framework for classifying and reviewing MTDs according to primary system stack domains: Dynamic Networks, Dynamic Platforms, Dynamic Runtime Environments, Dynamic Software, and Dynamic Data. Within these classifications, Okharvi, et al. review developing research, trends, and opportunities to evolve traditionally static systems into dynamic ones to increase the fundamental cost of attack. Dynamic Data systems specifically introduce variable data representation, including encryption, formatting, and encoding of data to tie it to a specific system state for it to be correctly interpreted and utilized. By abstracting this notion of the variable representation of data, one can apply similar techniques to data-content rather than simply its format to address some of the fundamental issues that have been identified with traditional authentication systems. Such dynamic password systems are aptly referred to as One-Time-Password (OTP) systems.

2.3 One Time Passwords

OTPs are a logical response to the weaknesses of traditional, static username/password authentication. OTPs are inherently resistant to brute-force and dictionary attacks as each attempt triggers a new password (note some time-based OTPs operate on a window of time sufficient to allow for multiple passwords submittals which employ the same time-based token). OTPs complicate shoulder-surfing attacks by introducing variability in response and generally countering replayed attempts for observed passwords. Coordination, management and presentation of OTPs present their own complications however, requiring a means to accurately construct the unique password. There are several methods to do this currently in practice. Google Authenticator[15] is one such tool: a mobile app which provides a unique time-based and HMAC-based OTP for authentication into applications using the Google service. Yubico markets another OTP tool; the YubiKey[10] which is a hardware-based tool that employs similar OTP protocols as the Google Authenticator app. In both cases however, these tools represent a second factor within the authentication protocol i.e., something a user has in addition to something the user knows, which presents limitations in the event a user does not have their mobile device or their hardware key with them when they want to authenticate.

2.4 Multi Factor Authentication

Multi factor authentication (MFA) systems are a useful mechanism for introducing a dynamic element into the authentication process. In addition to the two second factor solutions discussed in the previous section, there are several technologies which may be used to provide OTPs, either standalone or as an augmentation to more traditional static passwords. In the 2017 internet article, A Guide to Common Types of Two-Factor Authentication on the Web[14], Jacob Hoffman-Andrews and Gennie Gebhart describe four categories of MFA solution in common practice: SMS 2FA, TOTP 2FA, Push-based, and Fido U2F. SMS based second factors leverage an out-of-band communication channel with a user's mobile phone

for receipt of an OTP. Hoffman-Andrews and Gebhart point out some common drawbacks to the solution related to privacy concerns associated with linking the user's mobile number to an account, the need to have a functional device available, and the SIM-swap attack. This attack, which has seen considerable activity recently is where bad-actors convince mobile providers to reassign a user's number to a different SIM card in their control, effectively diverting the OTP data directly to the attacker. TOTPs, or Time-Based One Time Passwords are the system previously described which is utilized by the Google Authenticator app, which again is reliant on a user having a functional device available to them when they want to authenticate. Push-based MFA solutions also leverage an out-of-band channel but instead of sending OTP data, they send a notification that an authentication request has been made with an approximated IP-Address based location. A user responds to the Push-based request on the secondary channel, either confirming the authentication attempt is valid, or directing the service to deny the attempt. Hoffman-Andrews and Gebhart point out shortcomings of these system involve a lack of standardization, the functioning second device and additionally requires an active data-connection for IP-address location services. Finally, the Fido U2F is an open-source specification for Universal Second Factor hardware keys. The specification is becoming increasingly recognized and the hardware requirements are generally low when compared to the other MFA systems discussed. However, the authors point out that there are presently some standardization issues and the additional coast associated with the device may be a prohibitive factor. Fundamentally, as an MFA device, the possession of said device remains a strict requirement.

2.5 Challenge/Response

Challenge/response authentication (CRA) systems offer an alternative means of construction for an OTP which does not require a user to have a second factor device. In a CRA system, a user is presented with some challenge and prompted to provide an accurate response in order to be authenticated. CRA challenges may be presented in a variety of formats. A text-based question (mother's maiden-name, make of first-car, favorite pet's name, etc.) is a classic

CRA challenge frequently employed for resetting one's forgotten password in traditional username/password authentication schemes. In [24], Skrai et al. propose a CRA challenge based on transactional data recorded during the course of the user's everyday activities. Examples of challenges include inquiries about the approximate cost of lunch purchased at a particular restaurant, or the title of a recent movie purchased through a streaming media app. The foundational assumption of the proposed system being that a user's various transactions could be cataloged according to a common set of metrics and that users would be receptive to a central repository for all of their potentially private transactions. Such a system might also be susceptible to social engineering attacks simply through observation. Skrai's system does however present a novel source for challenge data. Another unique source of challenge material is employed in [27]. Sun, et al. derive their CRA challenge queries from a user's own mobile app environment. PassApp presents a user with a series of app icons including some key apps which are installed on the user's mobile device intermixed with decoy apps for noise, or cover. A user authenticates themselves by correctly identifying the set of installed key apps and avoiding the decoy apps. Each time a session is initiated a new subset of key apps is presented to the user for selection. Similar to [24], the PassApp CRA is potentially subject to social engineering attacks but presents an interesting source pool for CRA challenges and an important mode of challenge through graphic representation.

2.6 Graphical Passwords

Graphical password authentication (GPA) systems comprise a class of authentication mechanisms that differ from traditional textual username/password systems through their representation of challenge information graphically. It is broadly recognized that the human mind is particularly well suited to recognize graphic information as opposed to memorization and subsequent recall of textual data[20]. In Memory Recognition and Recall in User Interfaces [5], Budiu cites the specific impact of graphic cues which serve to contextualize and trigger memory recall. Indeed, this is supported by anecdotal examples such as recognizing a face in a crowd without the ability to recall a name or how that person is familiar.

GPA systems may also present with a series of distinct graphics, often requiring the user to distinguish specific examples from the set as relevant. The PassApp system presented in [27] is an example of a simple, selection-based system. Other GPA systems demonstrate considerable variety of response mechanisms. In Shen, et al.'s work [23], a 3x3 matrix of distinct graphics presenting the digits 1-9 are randomly assigned position on a user's mobile device as each session initiates. Users then provide a response to the positional challenge by sequentially connecting-the-dots of an n-digit PIN. The random position complicates shoulder-surfing attacks from a distance but an astute observer who was able to identify the numeric sequence would be able to replay the PIN however the digits lay out. An interesting side-note, Shen et al, chose to build in a degree of tolerance into the connect-the-dots for imprecise swiping, effectively expanding the correct PIN from one to many. Similar to the PIN in [23], in [18], Murugavalli et al. propose two variations of a GPA scheme where a sequential selection of constituent graphics (this time within a singular composition) is returned as a response. In ClickText and ClickAnimal, a sequence of alpha-numeric characters within a distorted matrix (similar to a 2d CAPTCHA), or a series of farm-animals within a farm-scene composition respectively are clicked in sequence corresponding with a predetermined secret (abc123, or goat, pig, pig, cow for instance). In the case of all three of these sequential-selection systems the state-space of the response is relatively limited, which in-turn limits the security of the sequence. Additionally, there is a fundamental reliance upon the simple correlation of response to challenge data which makes the sequence relatively straightforward to distinguish with reasonable observation despite random arrangements. Ultimately these systems would be better described as one-time-challenge systems rather than one-time-passwords. Our proposed system will also present as a GPA challenge although rather than a simple set-selection or even a sequential-selection, the response will change format as a textual response.

2.7 Graphical OTPs

In [3], Alsaiari et al. propose GOTPass, a multi-stage GPA with a corresponding OTP. The initial step in this system is a static (i.e. not changing between sessions) pattern-swipe which pre-authorizes a user to receive live graphical challenge data for the OTP stage of the system. In the event the initial swipe is not performed correctly, the system forks to a honey OTP challenge indistinguishable from the legitimate challenge where there is no correct answer. Assuming successful pre-authorization, the user is presented with a matrix of distinct graphics comprised of three sets of images: legitimate pass-image target images, related but incorrect distractor images, and random decoy images. Associated with each row, and each column of the matrix is also a three-digit numerical value. The user then selects the two legitimate target images from the matrix, and depending on the security level of the system chosen by the user at system setup, either selects the numerical values from the row(s) or the row and column of the correct images. These numerical values are then returned as the response for authentication. This system presents several unique and valuable attributes. First, the pre-authorization step serves to limit access to legitimate CRA challenges, reducing the likelihood of recurrence correlation studies. Secondly, the decoupling of the response from the selection of the correct images makes shoulder-surfing attacks more difficult and increases the potential state-space of the secret derived from the selected images. The limited quantity of numerical values within each matrix is however a significant limiting factor as employed. The decoupled relationship between the two could be leveraged further to increase the state-space. The proposed ACRAS system extends the decoupling from a simple association, into a logical derivation from representations of relevant member graphics.

2.8 Hybrid Systems

Similar to [3], Khan et al. propose a multi-stage, hybrid GPA sysem in their 2011 work [16]. In this case, the preliminary stage is a simple, traditional, text-based password which is

followed by a graphic component where a user redraws a pre-established symbol to complete the authentication. The two stages effectively stand on their own however, relying on two passwords in a single system to increase security over either one or the other.

In [28], Zangooei, Mansoori, and Welch propose another hybrid GPA system which combines two existing stand-alone GPA mechanisms into a single system. The first stage of their proposal includes a recognition-based technique similar to the Pass-Face system wherein a user identifies a series of valid images within a matrix. The second stage of Zangooei et al.'s proposal is similar to the Pass-Point system although instead of picking several points of interest within an image, involves momentarily remembering which of the slots within the matrix previously hosted which of the relevant images. During this phase, each of the images within the matrix are replaced with a short piece of text. The user then concatenates each of these pieces of text together according to the order prescribed by the previous images into a single password. The proposed ACRAS system relies on a similar hybrid graphic-interpretation to text-output technique to derive the session password.

2.9 Challenge Dimensionality

In Chen et al.'s patent [7], a graphical OTP scheme is described where the dimensionality of the graphic challenge is similarly extended beyond a simple yes/no membership. One of the fundamental characteristics of graphic presentation is the ability to employ multiple traits of a graphic simultaneously. In the case of their proposed system, Chen et al. identify three characteristics of each graphic: hatch-pattern, shape, and color which are then used to determine if a sequence of graphics corresponds with a user-defined pattern. Any of the three characteristics may be used independently or in concert with each other to establish the qualification of a given graphic symbol within the sequence. The challenge consists of eight sets of symbols, only one of which meets all the predefined criteria. The user selects the correct set of symbols to return as a response for authentication. While Chen et al.'s proposed system vastly increases the state-space of a relatively limited set of graphics with the addition of the 3x dimensionality, it should be noted that similar to [3], the actual

presentation of choices limits effective state-space by only including eight sets to pick from with a resultant 12.5% chance of guessing correctly.

A similar increase in dimensionality of presented graphic information is employed in the proposed ACRAS system. The increase will be asymmetrical with some characteristics applicable to all types of symbol, and some characteristics applicable only to a subset of the challenge symbols. This dimensionality increases the potential complexity of a user's pattern significantly and afford opportunities to layer representation such that a given symbol can carry a different value based on the user's algorithm. For example, a green digit 8 can carry a relative (noun) representation as (int) 8, (string) green, (string) eight, or (string) EIGHT. Each of these values could be acted upon (verb) or simply represented as a component of a textual response.

2.10 Implicit Pattern Recall

Within the larger field of Psychology, and specifically within the domain of Formal Language Theory, research has been conducted which further extends and formalizes this basic recognition vs. recall disparity. Studies related to human (as well as other species) facility with implicit memory and pattern recognition suggest that the distinction goes beyond simple graphic cues. In their summary work, Pattern perception and computational complexity: introduction to the special issue [12], Fitch, Friederici and Hagoort describe a series of foundational Artificial Grammar Learning studies which demonstrate human propensity towards implicit pattern recognition and application, even in the absence of recalled cues. This natural ability to apply patterns in new settings is precisely the behavior the proposed ACRAS system relies upon for generation of OTPs. Where the ACRAS system differs from these AGL studies, and many of the previously presented GPA systems however, is that rather than simply applying implicit memory patterns to identify boolean membership, a compound element of explicit memory is required to recall specifically how one is to interpret the recognized symbol. Similar to Budiu's recommendations [5], the graphical cues afforded by the initial implicit graphic recognition aid the user in the explicit recall.

2.11 Captcha GPA

One common GPA system most users are familiar with is the CAPTCHA [6] system. The Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) is a class of GPA systems that can take one of a variety of formats. In one case of the CAPTCHA a singular graphic is presented depicting text which is sufficiently distorted that computational systems are unable to distinguish the contents. Human beings on the other hand are expected to be able to recognize the contents and enter them into a text form as a response. CAPTCHAs are not intended to be a secure password for user account authentication as the response information is plainly presented and the instructions or formula for responding clearly stated for non-computers, but it does pose an interesting fundamental mechanism for a simple type of (human) authentication. The proposed ACRAS system acts in a similar fashion, however using a user-defined secret formula or algorithm for processing the information in the graphics acting as the primary point of security.

2.12 Misdirection and Deceit

A multiplicity of representation is particularly effective at creating increased variety within a limited set of available characters or symbols. Additionally, the increased dimensionality of the symbols affords opportunities to introduce misdirection or principles of deception into the CRA challenge. In [2], Almeshekah and Spafford classify varying levels of intentional deceit within a computer system to include a mixture of truth, naive, and intelligent deception. For example, in the proposed ACRAS system, the set of challenge graphics includes some symbols relevant to a user's pattern or algorithm (truth), and some irrelevant symbols together with a multiplicity of incorrect reads or interpretations of the relevant graphics (naive deception).

There is also the potential for some intentional orchestration of alternative patterns within irrelevant (or even otherwise relevant) symbols which could derive a similar result to the user's correct algorithm in order to intentionally create confusion for shoulder-surfing correlation analysis (intelligent deception). This functionality was in the end, not explicitly

implemented within the ACRAS prototype, however through analysis and observation it was noted that due to the extensive feature-set, all challenge sets naturally produce random plausible alternatives for one or more response stage. Particular care needs to be applied to this third technique; allowing for the viable alternative without disproportionately increasing the representation, or path(s) to a correct response similar to the swipe tolerance introduced by [23] in their PIN swiping CRA response.

Almesheka and Spafford further describe a series of techniques for creating deception within computer systems which can be used individually or in-concert with each other. Some techniques particularly relevant to proposed ACRAS system include *dazzling* where good or relevant information is hidden among irrelevant noise, *mimicking* principles are where false data is presented in the correct format in an effort to pass it off as legitimate, and *decoying* where false data is represented so as to draw an adversary's attention away from the most valuable portions of the system which are also present. In [1], Albayati, and Lashkari propose a recognition-based GPA system where a user is presented with a series of sub-image excerpts from a larger image, and a series of decoy images that are similar in image features yet not actually a part of the larger target image. A system user selects the appropriate images and re-positions them on a blank canvas to rebuild the larger image. The system is subject to guessing attacks however, as there is a necessary relational connectivity among the valid sub-image constituents, similar to the way in which adjacent, yet loose puzzle-pieces relate.

2.13 Graphic Authentication Attack Resistance

Much has been made of the improved memorability of graphic authentication mechanisms over traditional static text-based credentials. Such an improvement certainly warrants consideration for the class of authentication systems, though not if at the expense of system security. In their 2013 work [21], Renaud, Mayer, Volkamer, and Maguire consider the fundamental vulnerability to selection-oriented GPAs in terms of specific attack types. These types are commonly referenced as guessability, including brute-force, and dictionary attacks; observability, for shoulder-surfing and spyware; recordability, covering social-engineering and

theft of user-documented cues; and memorability which is generally regarded as more of a strength of GPAs. Through the analysis of several proposed GPA systems, Renaud et al. distill their observations into a series of recommendations for improved GPA security. Their recommendations for guessability include constraining user-choice when selecting images and defining points of interest so as to avoid hot-spots of conscious (or otherwise) concentration. observability resistance involves a decoupling of challenge from response mechanism, for instance, the manipulation of a series of matrix indices to identify a specific coordinate on-screen rather than pick-selection. The ACRAS prototype enjoys a similar challenge/response decoupling as a primary feature which helps protect it from simple observation. Interestingly, their work identifies recordability as one of the underlying weaknesses of GPA systems, suggesting that input obfuscation may provide some marginal benefit, albeit incomplete. Additionally, while memorability is clearly one of the strengths of GPA systems, other accompanying characteristics such as extended procedural duration may hinder a system's security by exposing the process to potential snooping for extended periods of time.

In [11], English and Poet analyze a series of countermeasures commonly employed within selection-oriented GPAs to protect against intersection attacks. The work details a series of experiments designed to support some logical hypotheses for increased resistance to these attacks. Experiments indicated a positive effect associated with increased valid challenge quantities in relation to challenge/response stages (or screens), and maintenance of a consistent set of distractor challenges with a given valid challenge in particular. A negative effect was found to be associated with higher numbers of challenge screens, potentially resulting from additional attacker exposure. The proposed ACRAS system differs from the class of GPA tested by English and Poet by employing a compounding effect of multiple valid challenges and a single screen of challenges with multiple reads.

2.13.1 Shoulder-Surfing

Another type of attack that GPAs are frequently vulnerable to is the shoulder-surfing attack. Shoulder-surfing is an attack where an adversary observes a user input credentials,

either by physically watching or via a technological means such as spy-ware, key-loggers, etc. Considerable research has been conducted with the aim to protect against shoulder-surfing attacks. None that we are aware of have addressed the attack in its entirety, rather different system implementations tend to exhibit stronger protections towards different variations of the attack.

In [9], Darbanian and Dastghaiby propose a hybrid recognition and recall-based GPA system. In their proposed system, upon registration a user selects 10 abstract images and assigns them each an unrelated alpha-numeric character value. During authentication, the system presents a 14 x 5 matrix of similar abstract images. A user then proceeds to analyze each of the 14 rows in top-down order looking for one of the 10 predetermined images. Each image is only displayed once, leaving for four rows of all decoy images. When a valid image is recognized, the user then types the corresponding assigned alpha-numeric value into the password entry field concatenating a 10-character long OTP. Resistance to shoulder-surfing and spyware attacks stem from the decoupling of challenge and response, with the response values having no directly-apparent correlation to the challenge images beyond the mental association made by the user at registration. While the decoupling is clearly a strong security element, this abstract association between the image and assigned alpha-numeric character does seem like it could present some recall-specific usability issues.

Man, Hong and Mattews propose a similar recognition/recall hybrid system in their work [17]. Their proposed system incorporates a similar abstracted character association with a series of valid challenge image variations, called perturbations which are used to build a response string as an OTP. These perturbations modify one small detail of an image which corresponds with a different output value for that particular variation. In addition to the perturbation-specific value, the positioning of the challenges relative to a pair of datum-points within a broader field of decoy challenges is responsible for an additional OTP constituent value. Similar to [9], the recall of what character value corresponds with a given image perturbation potentially presents a considerable usability issue. The proposed ACRAS system leverages a similar decoupling of challenge and response value via textual input of a

challenge-associated value, although the input value corresponds with a particular characteristic of the relevant challenge; relying on a graphically cued recall to aid in distinguishing an accurate response.

In [4], Wen, Safdar, Akbar, and Subramanian propose a GPA system, Wynd which leverages a combination of objects to determine functionality. The appearance or absence of specific user-defined classes of symbol indicates to users the mode of operation for which to interpret the graphics. When an image from one class is present it indicates to the user that they must pick-select a specified quantity of challenges from one set; in another case, a different quantity; in yet another case, the user must select a series of non-member symbols. In Wynd, Wen et al. introduces an element of temporality to the validity of challenges, i.e. the same challenges may be valid in one set, and not in another - all depending on the context of other challenges. Similar to other selection-oriented systems that have been reviewed, the Wynd system requires the serial passage of multiple stages in order to authenticate. The proposed ACRAS system shares a similar multiplicity of reading with challenges, though differs in that the staging of an ACRAS algorithm occurs all within a single screen, involving multiple reads of the same set of challenges.

In [26], Sun, Chen, Yeh, and Cheng propose another multi-stage authentication scheme called PassMatrix where users cycle one-time-secret X and Y coordinate index values to align with a predetermined image artifact for each stage. Each image is divided by into a 7x11 matrix and indices are aligned according to the artifact's coordinates. The *login indicator* is a one-time set of coordinate indices that are used to locate the image artifact. The *login indicator* may be delivered to the user in one of three ways, via device screen once a circled hand is presented to shield external view, via audio, or via a predefined separate image/index pairing. Maintaining the secrecy of the *login indicator* is critical to the security of the scheme. The dynamic ordering of indices between stages and the decoupling of index pair and image artifact are the primary mechanisms of PassMatrix' shoulder-surfing resistance.

In their 2013 work, [8], Chen, Ku, Yeh, and Liao propose an interesting dynamic graphical mechanism for entering an otherwise static text password. In their system, Chen et al.

provide a user with a dial interface broken into eight distinctly colored sections of equal size. Within the eight sections, a series of 64 characters are randomly distributed. To authenticate, a user must rotate the dial so that a specific color-zone corresponds with the next character in their password and confirm entry, repeating the process until the entire password has been entered. As with most GPAs, the system is subject to statistical analysis when complete data is available. Probability analysis with passwords of varying length indicate that the probability of a successful shoulder-surfing attack is nearly 50% with complete data from three successful logins and nearly 90% with four. Similar to [26] however, challenge decoupling from the response mechanism significantly increases the protection against casual shoulder-surfers or those with incomplete data. As previously stated, ACRAS relies on a similar decoupling of challenge from response forcing a considerably more focused analysis of complete challenge data correlated with the accompanying response to reverse engineer the user's set of rules.

Finally, in [13], Gao, Ren, Chang, Liu and Aickelin have implemented a system which, similar to [28], merges two existing GPA systems into a single authentication mechanism. Gao et al. propose the CDS system which appropriates the response mechanism from Draw A Secret techniques and the valid-challenge recall framework from a system called Story. In the Story system, users are encouraged to select their pass-images from a master-set in an order with which the user can build a mental story from their content. The CDS system essentially extends the Story model from a serial pick-select into a single-touch connect the dots response. Interestingly, Gao et al. chose to require the inclusion of distractor challenges not only at the necessary intermediate connections when drawing between valid challenges, but also to require users to start and end on a random distractor. The inclusion of distractor challenges along the course of a user's Story connection increases the work required for an observer to explicitly identify the valid challenges. A related mechanism is at play within the ACRAS framework wherein specific stages of response could be generated via several different challenge selection bases, one no more or less likely than another. This *plausible alternatives* mechanism is one of the primary means of shoulder-surfing resistance within the

ACRAS system.

Chapter 3

ATTACKER MODEL & BENCHMARKS FOR SUCCESS

3.1 Attacker Model

To gauge the success of any security related system it is important to define the attacker model and clarify assumptions. For the proposed algorithmic ACRAS system, it is assumed that the attacker has a strong knowledge of the system itself. An exploratory registration for the system would effectively give an attacker access to the bounds of the system, including the finite set of symbols or characters and the available rules for manipulation and derivation of challenge responses. The proposed system relies not on keeping the rules of the game a secret, rather it exploits the combinatorial volume of possible derivations provided by the dimensionality of symbol representation coupled with pre-established secrets which may be used to augment the presented challenge.

Renaud et al.'s appropriated framework [21] for authentication system vulnerability classifications serves well to organize assumptions and expectations of the proposed ACRAS system. Following are a series of assumed attacker capabilities and metrics for system success according to three of these classifications: Guessability, Observability, and Recordability.

3.1.1 Guessability

As an OTP system, the proposed ACRAS system is intrinsically resistant to brute-force attacks. For every session authorization attempt, a new password is constructed which prevents the reduction of possible responses through serial attempts and effectively turns brute-force into a repetitive set of one in many guesses. For this reason, brute-force attacks on this system are not particularly interesting and remain outside of the scope of analysis. Similarly, the system does not provide for the requisite reduction of possibilities necessary

for dictionary or familiarity-based attacks, which are really just a subset or means to increase the efficiency of a brute-force mechanism. As such, these two attack scenarios also remain outside the scope of analysis.

OTP systems are, however, subject to a type of reverse brute-force attack, where the same password is replayed over and over in the hopes of a corresponding challenge to be presented. This type of attack is straightforward to identify, and implementation of a simple fail to ban policy after a predetermined number of attacks can address this risk.

3.1.2 Observability

As has been covered at length in the Background and Related Work chapter, GPA systems in particular are frequently gauged against shoulder-surfing attacks. The forced variety of an OTP GPA lends itself well to complicating shoulder-surfing particularly when less than complete observation is assumed. If, however, complete observation of a successful authentication is assumed, the majority of the OTP systems are relatively straightforward to reverse engineer and reproduce with sufficient access to subsequent challenge/response pairs. Where complete shoulder-surfing observation becomes less effective is when a system embodies some degree of two particular characteristics: decoupling of challenge and response, and increased dimensionality of challenge.

In [9], Alsaiari et al, decouple a user's target pass-image from the numerical values representing rows and columns within the matrix of challenge images in their GOTPass system. A complete observation of challenge and response focuses the set of potential target candidate images to specific rows and columns but does not explicitly identify which images are the user's pass-images. Serial shoulder-surfing observations allow for a correlation of focused candidate image sets and will eventually lead an attacker to the correct identification of a user's pass-images. The question becomes how many observations are required to effectively achieve this focus, or perhaps more usefully - how much does each subsequent observation increase the likelihood of correctly identifying the target image(s)?

In [10], Chen et al. increase the dimensionality of challenge symbols sufficiently to di-

lute the targeted characteristics (hatch pattern, color, shape or some combination thereof). By employing the technique of Dazzling as described in [11], Chen et al. provide viable alternative patterns for each iteration of successful challenge selection. Similar to Alsaiari's GOTPass however, correlation of observations focuses the pattern relatively quickly when multiple successful challenge/selections are made available.

By extending shoulder-surfing attacks to technological variants, such as spyware capable of screen-captures and key-logging the complete capture requirement for efficient reverse-engineering becomes a more likely reality. In such cases, both the decoupling condition and the increased challenge dimensionality can serve to increase the required volume of complete data necessary for a successful reverse-engineering effort. Ideally, a user would conduct malware scans on their system with some frequency, hopefully on an interval sufficient to catch and remove said spyware prior to adequate data collection necessary to break a user's credentials.

3.1.3 Recordability

Recordability threats generally involve social engineering efforts or theft of user-documented credentials or cues for recall. Social engineering efforts such as phishing can be made more complicated to successfully coordinate when a sufficiently large set of all possible challenges is paired with an appropriately small set of valid challenges. If the conditions are right, a user may be alerted to the bogus challenge simply because the valid challenges required to generate a response are not present. For the purposes of ACRAS system analysis, it is assumed that attackers do not have sufficient knowledge of a user's algorithm definition to guarantee all of the requisite challenge members will be present within a challenge-set.

User documentation of authentication credentials is a risky behavior and broadly discouraged, yet remains surprisingly common. An authentication system may assist users by designing their system such that response mechanisms are easy to remember so that users do not feel compelled to document them. As this is a highly variable attack-vector based on many factors outside of the authentication system itself it is considered outside the scope of

analysis for this research.

3.2 Benchmarks for Success

With guessability attacks including brute-force and dictionary or familiarity-based attacks largely addressed by the inherent dynamic nature of graphical OTP systems, success for the system in this respect will be judged on the combinatoric volume of possible responses for a given algorithm. In addition to the abstract larger set of system-wide possible responses, user-experiments will document the complexity of actual user-defined algorithms and their associated possible response variations.

Observability attacks represent the most interesting class of attacks for the proposed ACRAS system. The ACRAS system aims to address shoulder-surfing attacks through compound application of the two aforementioned characteristics: decoupling of challenge from response and increased challenge dimensionality.

Figure 3.1: Plausible alternatives for ACRAS response generation

In addition to these naive deceptions, a coincidentally passive intelligent deception mechanism has been observed wherein short-term (i.e. single-session) plausible alternatives create decoy algorithmic possibilities. For example, in the screenshot of the ACRAS prototype shown in figure 3.1, given a textual response of "heart03", is the "3" a product of three capital letters being presented in the matrix, three filled shapes, or the result of the multiplication

of the filled digits "3" and "1" within the challenge set? Subsequent challenge/response pairs would naturally have a different value in the response with different plausible counting or mathematic explanations. As stated previously, care and balance are needed to prevent the system from creating too many paths to an accurate response while simultaneously allowing enough decoy to thwart easy reverse engineering of the algorithm.

Quantifiable success is difficult to explicitly define against casual shoulder-surfing. In order to provide some means for measurement of success, the following is proposed. Given common account time-out durations, we estimate a user might have to authenticate into an account twice at a single sitting where they could be observed by a shoulder-surfer. Add a third authentication for accidental browser closure, and yet another for the sake of conservativity, resulting in four authentications that a casual observer may be able to shoulder-surf from a user's activity at a single sitting. More technically oriented means of observability-related attacks, namely spyware and key-loggers would be capable of complete observation for the duration of their presence in a user's system. As alluded to previously, the frequency of related data-exfiltration, malware-scans and efficacy of malware-removal are outside variables which are beyond the scope of this research, save to say that any increase in quantities of data required for successful analysis will benefit an authentication system against compromise.

Chapter 4

METHODOLOGY

The primary objective of this thesis is to explore the feasibility of an algorithmic single-factor one-time password user-authentication system. During the course of the research, we have implemented a prototype for the algorithmic Challenge/Response Authentication System (ACRAS) and conducted a series of user experiments to both quantify and subjectively qualify the success of the system. The following is a detailed explanation of the system design and research methodology, generally organized according to the manner in which it has been conducted.

4.1 Phasing and Delivery Approach

Early planning efforts for the research called for a brief, yet intensive design-charrette intended to establish system design and direction. This design phase was to be followed with a series of agile implementation phases, subsequently building upon the system in an iterative manner. In practice, this evolved from the initially planned design-forward process to one of successive paired design and implementation phases with the interface and details of each component of the system being designed and developed in concert. A primary driver for the re-organization was the need for smaller, tangible features to play and experiment with in order to lead the design of subsequent system features. This methodology was determined to be a more effective means of development for a single developer than parallel development of multiple features in isolation from each other. Understandably, the serial nature of the process led to some inevitable re-work of previous system components to properly interface with additional features as they were developed. The lessons-learned along the way and the associated refactoring and feature-revisits ultimately made for a system which better

reflected and responded to the ideas being explored and developed in parallel, even if not according to the initial schedule.

4.2 Tools + Techniques

Early design and planning efforts for the ACRAS prototype system explored two different approaches: the native graphical user interface application, and the web application. Consideration was paid to the complexity of both the development and the operation environments as well as the fundamental nature of the ACRAS system. In the end, the prototype ACRAS system was developed using a web application framework to leverage the many available tools and libraries specifically designed to serve graphic content and capture user-input through form-data, the two primary tasks of the ACRAS system.

Given the current prevalence of web applications, there are numerous available frameworks to choose from depending on one's back-end programming language of choice. Python was selected as the programming language for the ACRAS back-end. Python is a popular, free, open-source programming language known for its simplicity and legibility of syntax as well as its multi-paradigmatic flexibility to behave in both object-oriented and functional programming manners. There are two basic versions of Python in common use, version 2.7 and version 3.x. Python 2.7, while still in common use, is technically a legacy version with Python 3.x serving as the current version which remains under active development. Python 3.6 (hereafter Python) was selected for the ACRAS system as a stable current release with broad support for third-party libraries and integrations. Python relies on a substantial standard library, and enjoys a strong development community which maintains a wide-variety of specialty libraries supporting functionality ranging from cryptography to front-end interface.

For web application frameworks, both Django and Flask were considered as candidates for the ACRAS application. Both frameworks are developed in Python, and similar to Python, each are popular, free and open-source. The two frameworks are generally quite similar, employing a Model, View, Controller (MVC) architectural pattern. The two frameworks differ in their fundamental approaches to delivering web application functionality however.

Django is designed to be an all-inclusive solution with everything a developer needs included in the core framework. Flask on the other-hand embraces a lighter-weight approach. Technically classified as a micro-framework Flask is meant to provide an underlying extensible framework that can be custom-tailored to the developer's needs by integrating with third-party extensions for desired web application functionality. Flask was ultimately selected over Django for the ACRAS prototype due precisely to its lighter weight; allowing one to focus specifically on the extended functionality required for the system without carrying the unnecessary bundled functionality preloaded into Django's core framework.

In keeping with the broader Flask philosophy, the web application's backend structure is configurable to whatever degree of complexity best suits the web application. A simple organizational structure was selected for the ACRAS system which allowed project resources to remain well organized yet was still suitable to the relatively small size of the prototype system. Several backend extensions and libraries were leveraged for required system functionality. These included the Secrets, and Hashlib utility libraries for Python to handle cryptographic functionality, the SQLAlchemy toolkit for database integration, and some experiments with the WTForms extension for management of HTML forms which was in-the-end, replaced in favor of JSON. Flask is not entirely without onboard resources however, as the Jinja2 templating engine comes bundled standard. The ACRAS system leveraged Jinja2 templates for efficient frontend implementation and configuration by backend resources.

On the frontend, the ACRAS system was designed specifically with a laptop interface in mind. There is no reason the system could not be extended to mobile platforms (in fact, it may be particularly well suited for the form-factor), but that level of development was decidedly outside the scope of the research project. Instead, efforts were made to produce a functional prototype to fit a common 13" laptop screen. As a web application, a number of different technologies were employed on the frontend of the ACRAS system. Frontend development occurred using common technologies, including HTML, CSS, and JavaScript. These fundamental web building-blocks were further extended using the Bootstrap frontend framework and jQuery library. Similar to Flask, Bootstrap is a popular, free, open-source

framework, focused solely on the frontend of a web application. Bootstrap contains templates for customization and control of common user-interface (UI) components including buttons and dropdowns, JavaScript extensions for component interaction, and modular content including icons and typographic resources. jQuery is another popular, free, and open-source library used for web application development. jQuery extends client-side JavaScript, facilitating UI component interaction and event handling and was used throughout the ACRAS frontend.

In addition to frontend and backend implementation, the ACRAS system prototype employs an SQLite database. The database is used solely for documentation of metrics collected during user experiments. SQLite is a popular embedded database which differs from other client-server database models insofar as it is entirely self-contained within a single database file that integrates with the ACRAS backend application rather than requiring a dedicated server process. The SQLite format is well suited for the ACRAS prototype as it is in keeping with the system's consistent application of lightweight extensible components. Further, the limited extent of data collected from user experiments of the prototype do not pose a challenge for the data size limitations of the database system.

The ACRAS system prototype has been designed as a stateful REST API. This deviates from traditional web applications which are typically stateless to afford system scalability. As a small prototype system intended for a single user at a time, the scalability requirement was not important to meet the goals of the prototype. This afforded the opportunity to simplify system implementation by maintaining required state within the backend for the duration of a user session. A multi-user implementation would require reworking the backend to integrate with a database system for state maintenance. Such modifications made for scalability would likely benefit from a transition from the system's current SQLite database implementation to a database system which better supports a similar scalability without the size limitations inherent to the SQLite platform.

The UI for the ACRAS frontend was designed to dynamically present and update data including graphic challenges and user-interface components. This design allowed the pro-

totype system to handle and represent real-time changes in user configurations. It also afforded an opportunity to maintain critical user-defined configuration information available to the user for reference throughout the user experiments. Asynchronous JavaScript and XML, or Ajax techniques are employed to provide this dynamic behavior without reloading the entire page every time a change is made. Critical to the web application model of the ACRAS prototype is a reliable mechanism for intercommunication between the various components. JavaScript Object Notation (JSON) was selected for several reasons, including a non-hierarchical, legible syntax which facilitates rapid development, as well as the ease with which JSON integrates with client-side JavaScript. Previous reference was made to some experiments in backend development with WTForms and HTML form data that were opted against in favor of JSON. In direct comparison, frontend to backend communication was relatively straightforward with both the form data and JSON. Implementation of the backend to frontend, and the backend to database communication channels proved to be particularly effective with JSON data however, and as a result the choice to consolidate on a single data model for communication in all directions was made.

4.3 Design.1 - Challenge

The first component of the ACRAS system to be designed was the challenge. As the basic building-block for the system, the challenge was a natural starting-point. The ACRAS system relies on the fundamental multi-dimensionality of the challenge to achieve adequate complexity, or the combinatorial explosion of possible readings for system "strength". It is precisely the instantiated representation of these abstract dimensions which a user interprets for both membership or inclusion in a qualifying set of challenges, and the resultant response. As such, critical to the successful interpretation of these challenges is the clear, unambiguity of dimensional representation, hereafter referenced as the critical clarity principle. Great care was taken in the selection and representation of possible dimensions in order to maintain this clarity and will be discussed further in the following sections. These challenge dimensions can be broken down into three distinct categories: common characteristics, shared characteristics,

and unique characteristics.

4.3.1 Common Characteristics

Common characteristics are those characteristics which are universal across all challenge instances. Every challenge possesses these characteristics, making them particularly useful for both distinguishing membership and constructing responses. Common characteristics include *color*, *fill*, *English*, and *object*.

The *color* characteristic is comprised of seven basic, unique colors, and three subsets of the larger set of all *colors*. The *color* set includes [*black, red, orange, yellow, green, blue*, and *purple*]. These seven *colors* were selected as clearly distinct colors with unambiguous identity. *Black* is black, *red* is red, there is little practical dispute of the distinction between the two. In keeping with the critical clarity principle, the traditional seven colors of the rainbow were not selected as unnecessary complexity is introduced when dealing with colors that could be blue, or indigo, violet, or simply purple. The point has further been raised that color interpretation is subjective and very much in the eye of the beholder, potentially weakening the adherence of *color* to the critical clarity principle. This subjectivity in color experience between individuals is acknowledged, yet ultimately not relevant to the issue at hand. As previously stated, the colors that have been chosen for inclusion are distinct, and while one individual's experience or perception of red may differ qualitatively from another individual's, both would still identify the color in question as red, which serves the purpose for the ACRAS system. Similarly, as an ACRAS user's hardware is beyond control of the system, there is little that can be done to control an accurate rendering of a color on the client-side. The best one can expect is sufficient distinction between colors which is generally regarded as reasonably attainable with commonly available hardware. Another concern that has been raised as a potential problem is one of colorblindness. In this case, the *colors* may not be distinct to an affected individual, expressly violating the critical clarity principle. This too, is acknowledged with the understanding that *color* is only one of several dimensions or characteristics of challenges. As a user of the ACRAS system is afforded the opportunity

to define which dimensions are relevant to their particular application; a colorblind user may simply opt-out of distinctions based on *color*, choosing instead to base distinction on another more easily-interpreted characteristic. In addition to the seven unique *colors*, three additional subsets have been overlaid on the larger set of challenges: *warm* colors [*red, orange,* and *yellow*], *cool* colors [*blue, green,* and *purple*], and *primary* colors [*red, yellow,* and *blue*]. In the spirit of multi-dimensionality, a challenge with the *color* characteristic *red*, also has the *color* characteristic *warm*, and the *color* characteristic *primary*.

The *fill* characteristic is binary in nature. A challenge is either *filled* or *outlined*, there are no additional dimensionality or overlay groups for this characteristic.

The *object* characteristic defines a challenge's fundamental typology. This is particularly important as it inherently defines additional degrees of dimensionality attributed to a challenge. Both shared characteristics and unique characteristics are defined by a challenge's underlying *object* characteristic. *Objects* include *shape, digit, letter* and *word*. Each challenge is explicitly one of these *objects*. Additional subsets further exist for each of these *object* classifications, yet by definition, are considered unique characteristics.

The *English* characteristic provides a considerable degree of complexity to the ACRAS system. How a challenge represents its *English* characteristic is the result of a challenge's *object*, and further defined by the challenge's unique characteristic which more specifically defines an *object's* sub-type. For instance, a *shape* challenge might be more specifically defined by a *primitive* characteristic of *circle* or *star*. The *English* characteristic of those challenges would then be "circle" and "star" respectively. *Digit:English* characteristics are represented by the *English* word for the *digit's* value, for example: "one", "two", "seven", etc. A *letter's English* characteristic is represented simply by the *letter's English* representation, for example: "a", "b", "q", etc. Note that *letter* challenges will have an additional level of dimensionality specifically related to the *case* of the *letter* which will be covered under the shared characteristics section which follows. Finally, *word* challenges have an *English* characteristic that reflects the actual word. As with *letter* challenges, *word* challenges also exhibit the *case* characteristic. *Words* are not simply constrained to either *lowercase* or

uppercase, a *word* challenge may also be *mixed-case*; for instance: "penguin", "PeNgUiN", "PENGUIN", "tacos", "TaCoS", and "TACOS".

4.3.2 Shared characteristics

Shared characteristics are comprised of characteristics which are shared by more than one challenge Object type, yet not common to all challenges. Some of these shared characteristics are mutual to entire *object* typologies, while others exist exclusively within a subset of the parent set of an *object*, typically corresponding with another characteristic exclusive to the *object*. Shared characteristics include: *case* for both *letters* and *words*; and *value*, *evenness* and *primality* for both *digits* and *words* within the *category number* (*word:number*).

The *case* characteristic shared between *letters* and *words* relates to the distinction between characters which are lowercase or uppercase. A *letter* challenge may be exclusively one or the other. A *word* challenge on the other-hand, may be either *lowercase, uppercase* or even *mixed-case* with alternating lowercase and uppercase characters in a single Word.

Value is an inherently quantifiable challenge characteristic associated with both *digits* and *word:number*. The *value* of either is simply the numerical representation of that particular challenge; for instance, the *value* of *digit* "9" is 9, simultaneously, the *value* of *word:number* "nine" (or "NiNe", or "NINE") is also 9. *Value* characteristics maintain additional value-based properties of numbers which by extension, are also shared characteristics. Such extended shared characteristics include a *value's* inherent evenness (*even* or *odd*), or primality (*prime* or *not-prime*). Additionally, *value* characteristics of have the unique property of being capable of mathematic operations which will be addressed in the section on *algorithms*.

4.3.3 Unique characteristics

Unique characteristics, as nomenclature suggests, are characteristics unique to a specific *object* typology. Generally, each of the different *object* types possesses a set of unique characteristics that it does not share with any other *object* types. The exception being

digit challenges, insofar as their characteristics are *value* based and as discussed previously, *word:number* challenges also possess *value*. Unique characteristics for other *object* types include such traits as *primitive*, and *direction* for *shape*, *vowel* (or *consonant*) for *letter*, and *category* for *word*.

Shape challenges are comprised of a series of six easily recognized shapes with distinct identities that are represented in the *primitive* trait: *circle, triangle, square, heart, star*, and *arrow*. In keeping with the critical clarity principle, each of these *primitives* are easily distinguished from one-another, recognizable in isolation, and directly mapped to a simple identification. Potentially ambiguous shapes were avoided when defining the set of *primitives* so that a user would not have to opt between confusing choices such as crescent or moon, an oval or the letter "O", or negotiate duplicative classifications such as a square also technically being a rectangle. *Direction* is another unique characteristic belonging to the *object:shape* typology; specifically unique to the *primitive:arrow* Challenge-set. *Arrow* challenges fundamentally point in some *direction*, and in the ACRAS system were designed to point *up, right, down*, or *left*.

The shared characteristic of *value* between *digit* and *word:number* challenges has already been identified but it is important to clarify that the design of the ACRAS prototype system has largely reserved the identification functionality of extended characteristics for *value*; i.e. evenness, primality and comparative-value for the *object:digit* typology. As a result, any *digit* challenge may be identified via these extended characteristics, whereas *word:number* challenges may not. The comparative-value characteristic for *digit* relies on multiple user-provided inputs including a comparison operator: *greater-than, less-than* or *exactly* and a relative setpoint value between 0 and 9. Details on how this functionality is manifested follow in the algorithm design section, for now it is sufficient to state that a *digit* may possess a comparative-value characteristic.

The *vowel* or *consonant* characteristics associated with *letter* challenges is reasonably straightforward. The one caveat being that in common practice within the English language, the letter "y" is sometimes a vowel, and sometimes a consonant. In the interest of maintaining

the critical clarity principle, a decision was made that the letter "y" is always a *vowel* within the ACRAS system. Similar to the comparative-value characteristic of the *digit* challenge, *letter* may also represent a relative-positioning of *before, after*, or *exactly* a user-defined setpoint within the strict order of the English alphabet.

The Category characteristic for *word* challenges has been previously referenced in this section when detailing the shared Value characteristic between *digit* and *word:number*. Six *category* distinctions have been incorporated into the design of the ACRAS prototype: *animal, color, food, number, sport*, and *vehicle*. Each *category* has 20 member *word* challenges; the various words having been selected with care to reduce the likelihood of categorical overlap which could result in an ACRAS user mistakenly identifying a *word's* category membership. For example, the word "orange" could qualify for either *color* or *food*, or the word "chicken" could be understandably interpreted to reside under either the *animal* or *food* category. Words such as "orange" and "chicken" have been omitted from the ACRAS system in favor of *word* challenges with clear *category* membership.

4.4 Implementation.1 - Challenge

Section 4.1 Tools + Techniques covered some of the broader details of the ACRAS prototype development, the following will cover implementation details specific to the ACRAS challenge. The challenge is implemented as a polymorphic series of inherited classes within Python. The Challenge superclass contains member variables used to account for the various common, shared and unique characteristics detailed in the previous challenge design section. Basic getter and setter member functions are implemented for common characteristics within the Challenge superclass. Challenge is extended by Shape, Digit, Letter and Word subclasses. Each subclass defines its own set of getter and setter member functions for management of shared and unique challenge characteristics.

The Challenge classes are implemented along-side a Challenge Factory class which instantiates the individual Challenge objects. The disparity of member-populations between the different *challenge:objects* initially presents a complication for even distribution when

randomly selecting the challenge objects to satisfy the challenge/response process. Unique challenge populations within the ACRAS prototype system accounting for the various combinations of *color, fill*, and *case* (where applicable) are as follows: *shape* (non-*arrow*): 70; *arrow*: 56; *digit*: 140; *letter*: 720; and *word*: 5,040. The distinction between *arrow* and non-*arrow* shapes is made as with each direction [*up, right, down, left*], there are four times as many representations of the *arrow* than other *shape* challenges. In order to address the population disparity and optimize the probability of an even *object* type distribution, a series of multiplicative factors are applied to each of the challenge types during instantiation. The resulting complete challenge set includes (60) instances of each non-*arrow* shape; (15) of each *arrow*; (36) of each *digit*; and (7) of each *letter* to bring the various populations even with the unique 5,040 *word* challenges.

Relevant implementation details beyond the basic structure of the programmatic interface and instantiation details are largely related to graphics and presentation to support the critical clarity principle. The ACRAS system is fundamentally reliant on graphic information as accurate responses are predicated on the correct interpretation of challenge data. For the system prototype, challenge graphics were constructed using commercially available Adobe Illustrator and Photoshop software, utilizing a time-consuming combination of manual and automated techniques. Each challenge graphic was manually constructed and exported from Illustrator as a 1600 pixel wide (with varying height) image as a black figure on a white background. After the set of representative black graphics were finished, a series of automated actions in the Photoshop software was applied to the directory to generate the complete set of challenge graphics. Automated actions included modifying the aspect ratio of images to 1:1, replacing black with the additional colors in the set and finally reducing the overall image scale to a final 400 pixel x 400 pixel image. This size was selected as a reasonable balance between file size and available detail for *word* challenges in particular as graphic scale was reduced to fit longer words.

In the previous section, some discussion was dedicated to the requirements for challenge color in the ACRAS system. Ease of distinction between colors and identification of a

color in isolation was identified as of critical importance to the system. The actual RGB values selected to represent each of the colors were selected from standard Photoshop swatch templates which met the distinction and identification requirements on the computer where the graphic software was used. Several versions of yellow were explored in order to create as much contrast with the white background as possible while still adhering to the other requirements. After the final set of challenge graphics was completed and cloned to the computer system where the balance of development and testing was to occur, it was noted that the chosen value for yellow did not present in the same way on the both machines. Yellow ended up appearing to have more of a chartreuse-hue on the development and testing system than was intended which resulted in several comments from users during user-experiments, although still easily meeting both the distinction and identification requirements. Figure 4.1 depicts the seven ACRAS colors implemented in the system prototype.

Figure 4.1: The seven colors of ACRAS challenge graphics

In addition to *color*, Challenges each graphically represent a state of *filled* or *outlined*. Distinction between the two is typically straightforward, however as challenge graphics decrease in scale, outline challenges can become harder to read. In the ACRAS prototype, *word* challenges have a variable scale to them in order to make the most use of the available space for a challenge graphic. Longer words require a greater reduction in scale to remain within the available space. As the scale of the challenge reduces, the stroke thickness of an outlined challenge reduces as well, making color distinction of a thin outline more difficult. At the same time, the stroke thickness for the outline of a letter can only be increased so far to compensate for the reduction in scale before the characters begin to appear similar to their filled counterparts. Ultimately, through a series of iterative trials, a balance was

struck between maximizing stroke thickness for legibility of character and color at smaller scales and maintaining a clear distinction between *filled* and *outlined* challenges. Figure 4.2 depicts the difference between *filled* and *outlined* challenges for each of the ACRAS *object* types.

Figure 4.2: The four object types of ACRAS challenges: Shape, Digit, Letter & Word

In addition to the accurate graphical representation of color and fill, the ACRAS system relies on the correct interpretation of challenge types. Distinction between *shape* and *word* challenges are not particularly difficult; however the correct interpretation of *digits* vs. *letters* has the potential to present a more complicated and interesting task for certain characters. There are two specific sets of lookalike characters where identification can be a challenge, namely: [uppercase "I" (i), lowercase "l" (L), and the number 1], and [lowercase "o", uppercase "O", and the number 0]. In order to aid in distinction between these potentially troublesome characters, the ACRAS system uses a combination of scale and fonts which exhibit different styles or typographic characteristics. Figure 4.3 illustrates the similarities between the two sets of lookalike characters.

Figure 4.3: Lookalike characters

A pool of several-hundred fonts was reduced to a series of possible candidates for the ACRAS prototype based on the relative ease of distinguishing between the identified lookalike

characters. Each of these candidate fonts was reasonably easy to discern lowercase "i" from uppercase "I" (i), from lowercase "l" (L), from the digit 1 in the immediate context of the other possible characters. The important task remaining however, was ensuring the ability to identify each of the characters in isolation, without the relative comparison afforded the characters in a side-by-side review. Investigation of sans-serif fonts frequently resulted in some ambiguity with respect to the lowercase "l" (L), often presenting in a format which could easily be mis-identified as either the digit 1, or possibly the uppercase "I" (i). Due to the ease of misinterpretation, sans-serif fonts were eliminated from the pool of candidates. Serif fonts on the other hand, represent characters with typographic feet, the non-structural details at the ends of some strokes. These feet aid in the distinction between lookalike characters and digits. To further maximize the likelihood of accurate character identification within the ACRAS prototype, two different serif fonts were selected for character and digit representation, each from different font-families; Clarendon and Geometric Slab-Serif fonts. These two font-families represent the feet of serif characters with contrasting styles, Georgia (Clarendon typeface) with stylistic, tapered feet, and Consolas (Geometric typeface) with block feet. Additionally, the slashed zero character in the Consolas font further differentiates the 0 digit from both the lowercase "o", and uppercase "O" characters of the Georgia font. Due to these contrasting details, Georgia and Consolas were selected for Letter and Digit Challenges respectively in order to clarify and emphasize the distinction between the similar-shaped lookalike characters.

4.5 Design.2+3 ACRAS Frontend + User Algorithm

Following design and implementation of the challenge, design and development of the ACRAS system shifted focus to the frontend user interface (UI) and the user-defined algorithm framework. The algorithm framework is the abstract mechanism or framework upon which a user defines the rules that their particular challenge/response session(s) will follow. The UI is of course the system frontend with which a user defines (and tests) their personal algorithm. While it may seem like a simple one-way relationship would follow, i.e. the algorithm in-

fluences UI design without reciprocal effect, the truth is that the two greatly informed each other's design. The early stages of UI design were especially effective for contextualizing and structuring the algorithm framework and definition process. The graphical elements from the UI design became useful tools for consolidating the loose collection of ideas and spreadsheets trying to map the algorithm functionality into a real-world cohesive system that could be used to test the concepts which govern the system. As one might expect, the algorithm framework design in-turn organized the structure and functionality of the UI.

Figure 4.4: Candidate fonts for ACRAS challenges

The ACRAS algorithm framework can be broken down into a series of smaller components for the purpose of system description. At a high-level, the algorithm framework is a multi-stage challenge/response system where a user serially interprets a series of Challenge graphics according to their personal set of rules (or their algorithm) which they have previously defined and generates a textual response. Zooming in on the process, the multi-stage aspect of the framework essentially means a user can generate between one and n textual password subcomponents that will be concatenated together to form a singular textual one-time password (OTP) which corresponds with both the user's personal algorithm definition and the set of Challenges that have been presented/interpreted. For the purposes of the ACRAS prototype, n has been limited to four stages with a corresponding UI quality indicator for the strength of a user's algorithm mapped to the n stages as follows: 1 stage: weak, 2

stages: good, 3 stages: better, and 4 stages: best. The UI strength metric does not represent any quantitative analysis of a user's algorithm, rather it is merely a simple means to associate relative algorithm strength with the number of defined stages, or algorithm depth. It is noteworthy however, that the increased depth of a user's algorithm generally corresponds with a greater combinatorial state-space for potential challenge responses which can have a positive effect on the strength of a user's algorithm against certain attack-models.

Each stage of a user's algorithm is defined independently although it is the same process for each and a user's larger algorithm is incomplete until all included stages have been defined. The skeleton of an algorithm stage broadly corresponds with the following outline: select an Action to be performed on some Objects which are selected from the set of Challenges according to a specific Selection Basis, optionally Transform that result according to a final manipulation and finally concatenate the end result onto the OTP. This process is the same for each stage in a user's algorithm, and results in an OTP that is somewhere between one stage and four-stages worth of concatenated text. The UI design mirrors this serial stage-definition process literally with four identical portions of the interface to enter the details of each stage. The individual stage sub-interfaces are intentionally identical in order to impart a sense of familiarity to the user as they navigate the system, and to emphasize the modularity of they system to break down the scale of what could be misinterpreted as an overwhelming task to define and subsequently recall the specifics of multiple complex stages. Following, we explore the specifics related to the skeletal process outline above.

4.5.1 Action

The first part of defining or applying a user's algorithm is the Action phase. An Action is basically just that; the action a user is going to apply to a (some) specific Challenge(s) presented to them. Within the context of the ACRAS prototype, Actions have a corresponding

Subject[1] which elaborates on the precise target of the Action. Actions include Count, Echo, and Math. Count is the simplest of the Action options and has only one Subject: Occurrences. For a Count:Occurrences Action, a user would count the occurrences of Challenges which meet a specific set of criteria defined further downstream in the algorithm stage and respond with the numerical value for the count. For example, if a user's algorithm stage was defined to Count:Occurrences of Red Shapes, a user would scan through the Challenge graphics presented, identify any Shape Challenges that were Red and concatenate the number of red shapes to the OTP.

The next Action, Echo is considerably more complex and has several options for Subject: Color, English, Value(#), Even/Odd, and Direction. With an Echo Action, a user identifies the relevant Challenge, again according to the criteria defined downstream and responds to the OTP with the Subject of that particular Challenge. Consider the following example where a Red Digit is the target Challenge and its number is 7: Echo:Color would result in "red"; Echo:English would result in "seven"; Echo:Value(#) would result in "7"; Echo:Even/Odd would result in "odd"; and Echo:Direction would not be an option for a Digit Challenge, as only Shape:Arrow Challenges have Direction. Another example where a Purple Letter is the target and the letter is a capital "X": Echo:Color would result in "purple"; Echo:English would result in "X" (note the case); Echo:Value(#), Echo:Even/Odd, and Echo:Direction are again not allowed.

Finally, the Math Action is applied the accompanying Subject: Add or Multiply to a subset of presented Challenges. These two mathematic operations were selected due to their relative ease of mental calculation, and most importantly because they are order-agnostic. In the ACRAS prototype, Challenge graphics are presented in a randomized order in a two-dimensional array, the order of qualifying Challenges could be a potentially confusing detail to establish (particularly if future versions extend the system to graphics-presentation arrays

[1]It is worthwhile to note that Subject in the ACRAS context does not correspond with subject in the study of linguistics. While there are considerable correlations to the structure of an ACRAS algorithm stage and linguistic structure, the similarities and resulting potential analysis of such similarities remain outside the scope of this research project.

beyond 2x10). Addition and Multiplication both hold the commutative property, so order of presentation is irrelevant.

4.5.2 Object(s)

The next phase of the ACRAS algorithm stage definition process is Object selection. In this phase of the process, a user defines which Objects the previously defined Action will be applied to. Options for Object include: Shape, Digit, Letter, Word or any combination of these four. Depending upon what Action and Subject have been already selected, the set of Objects may be constrained within the UI to ensure a valid stage is defined. For example, Count:Occurrences works for all Objects. Echo:Value(#) only works for Digits or Word:Numbers (the same holds for Echo:Even/Odd, and either Math Action). Echo:Direction is only available for Shape:Arrow, so Digit, Letter, and Word would all be unavailable (as would all Primitives except Arrow, but that will be discussed in the following section).

4.5.3 Selection Basis

The third phase of the ACRAS stage definition is the selection basis. Previous sections have made reference to Challenges that would be identified based on criteria which would be defined downstream; the selection basis is where these criteria are set. The selection basis is divided into a series of smaller basis groups which may or may not be relevant to a stage definition depending upon which Objects have been selected in the previous phase. Selection basis sub-groups include: General, which applies to all Challenges: Shape, which applies to Shape Challenges; Digit for Digits; Letter for Letters; and Word for Words. Within the ACRAS UI, each of these sub-groups have been organized in corresponding collections of drop-downs that keep related functionalities near one another for quick reference and access.

Within the General selection basis group, there are two sub-groups: Color, and Fill. Color is where a Challenge may be specified based on its Color characteristic. In addition to the seven colors explicitly defined in the abstract Challenge, non-exclusive color groups

have also been defined which allow for a broader set of qualifying challenges to be selected based on Color. These non-exclusive sets include: (any); warm [red, orange, yellow]; cool [green, blue, purple]; and Primary[2] [red, yellow, blue]. Selection sets defined with one of these non-exclusive groups will consider qualifying Challenges with any of the member Color characteristics. The Fill selection basis allows a user to define a selection basis predicated on a Challenge's Fill characteristic. Options for Fill include: (any), fill, and outline.

Shape-specific selection basis accounts for different Shape Primitives, and Direction, if Shape:Primitive is set to Arrow. Shape:Primitive options include: (any), Circle, Triangle, Square, Heart, Star, and Arrow. Shape:Direction options include: (any), Up, Right, Down, and Left.

Digit-specific selection basis affords an ACRAS user the opportunity to constrain a qualifying Digit to specific value-ranges or define a value requirement with respect to evenness or primality. For value-ranges, a user may define the Digit:Value to be (any), Exactly, Less-Than, or Greater-Than, with the latter three relating to a user-defined setpoint value. Digit:Misc in turn, controls whether a qualifying Challenge Value is required to be Even, Odd, Prime, or Not-Prime.

Selection basis for Letter Challenges are similar to those of Digit. Letter:Character constrains the character value (or English representation) of the Letter Challenge positionally within the English alphabet, and Letter:Vow/Case controls either a Letter's membership between vowel and consonant sets, or character casing. Like the Digit:Value attribute previously described, Letter:Character lets a user select between: (any), Exactly, Before, or After a user-defined character within the English alphabet. Letter:Vow/Case affords the ability to define if a Letter Challenge's character must be a Vowel, a Consonant, Lowercase, or Uppercase.

---

[2]Several different primary color sets exist in different usages, these sets include Additive Primary Colors, Subtractive Primary Colors, and the Primary Colors in common usage within the U.S. Educational System. The Primary Color set selected for inclusion in the ACRAS prototype system correspond with the Primary Colors in common usage within the U.S. Educational System and the National Gallery of Art https://www.nga.gov/education/teachers/lessons-activities/elements-of-art/color.html.

Finally, the selection basis options for Word Challenges include Category and character Case options. The Case options are similar to those available for Letter Challenges: Lowercase, Uppercase, and the addition of Multi-case for Words with both lowercase and uppercase characters. Word:Category is a selection basis unique to Word Challenges and reflects a Word's membership within a series of Category sets: Animal, Color, Food, Number, Sport, or Vehicle.

4.5.4 Transformation

The ACRAS Challenge Transformation is the final component of the algorithm stage. This optional phase to the stage definition is more closely related to the first phase where the Action is set than the phases which have followed although it has been designed to occur at the end where the details of the Object(s) and selection basis help to contextualize its function. The underlying purpose of the Transformation phase is to augment the possible responses to presented Challenges by overlaying an additional known complexity. In basic terms, a Transformation modifies the identified and interpreted Challenge in some way that the ACRAS user is aware of in the abstract sense but is not represented graphically by the presented Challenge. Unique Transformations are technically possible for each Action type, frequently with additional options corresponding with the various Subject or Object types. A limited subset of these possibilities was explored and implemented for the purposes of the ACRAS prototype; all within the Echo Action. Additional promising Transformations which were not investigated are discussed further in the Future Work section of this document. Following are descriptions of the Transformations which were implemented in the prototype system.

Shape Transformations are limited to Shape (specifically Arrow) rotation which would modify the interpreted Direction associated with the Challenge. Options include Rotate Clockwise and Rotate Counterclockwise. These would have the effect of changing the Shape:Direction value of an Arrow Challenge which pointed up to "right" or "left" respectively.

Digit and Letter Challenges are both capable of Sort Transformations for Echo:Value and Echo:English Action and Subject stages. Sorts increase the possible representation of qualifying Challenges from only a single instance to as many as three and require a compound response sorted according to the available options: Low –> High, or High –> Low for Digits; and A –> Z, or Z –> A for Letters. As a result of combinatorial explosion, the various permutations associated with Sort Transformations have the effect of increasing the possible response state-space for these Echoes from 10 to 1,000 for Digits, and from 52 to over 143,000 for Letters.

Letter Challenges with an Echo:English Actions are additionally capable of character-related Transformations which include the following: Capitalize, which has the effect of transforming the echoed response from whichever case the Challenge represents to a capitalized response; Lowercase, which works the same way with a resultant lowercase response; Invert-Case, which has the effect of changing lowercase responses to uppercase, and vice-versa; Double-Vowel, which has the effect of doubling the echoed response if the qualifying Challenge represents a vowel character (for example "a" becomes "aa", while "b" would remain "b"); and Double-Consonant which again has the opposite effect from the previous Transformation.

Within the context of the ACRAS system, Word Challenge responses for the Echo:English Actions are essentially just collections of characters. As such, they are subject to the same set of Transformations as Letters with a few modifications to suit their relatively extended content. Sorts are not supported for Word Challenges due to the inconvenience of requiring an ACRAS user to type three words for a single stage response. In addition to the option to Double Vowels or Consonants, the additional characters of a Word Challenge provide the opportunity to Omit Vowels or Consonants and still have a response. Similarly, the multi-character nature of Words affords the opportunity to apply a Reverse Transformation where, for example, a Challenge with an English value of "cow" would elicit the response "woc".

4.6 Implementation.2 ACRAS Frontend

As previously described in the Tools + Techniques section, the ACRAS prototype has been implemented using the Flask web application micro-framework. As a web application, the client-side, or frontend has been designed and developed for operation within a standard web browser. During development, current versions (each checks for updates upon launch) of both Mozilla Firefox, and Google Chrome browsers were used to debug and test functionality. Google Chrome was ultimately settled upon for the operating environment of the user experiments due to some minor unresolved bugs related to input validation within Firefox, and a general preference for some rendering details (stroke thickness for example).

Client-side functionality was implemented using JavaScript to manipulate UI components and interact with data sent-to and originating-from the ACRAS backend. The frontend UI was implemented largely from Bootstrap components, with some third-party extensions including the bootstrap-select dropdown and some external content such as icons and fonts. The UI was designed to separate presentation from interface, so a top-level bootstrap card container was defined as the master presentation scaffolding where graphical content is updated to reflect the state of the algorithm stage definition process, or the Challenge graphics during the challenge/response procedure. A series of collapsible cards for Stage registration was arranged below the presentation card. In accordance with the "weak", "good", "better" and "best" labels to encourage increased degrees of algorithm stage depth described in the algorithm Design section, the Stage Registration cards were colored according to a corresponding danger (red) –¿ warning (yellow) –¿ safety (green) scheme.

Each of the Stage Registration cards were implemented using the Jinja2 templating engine to facilitate efficient code-reuse for these nearly (aside from color) identical interfaces. The card-layouts reflect the work-flow for defining an algorithm stage moving from left to right across the card, interfacing with bootstrap-select dropdown menus which allow ACRAS users to select from a series of predefined options at each level. Dropdown menus are dynamically populated with JSON-defined values and activated or deactivated via jQuery as upstream

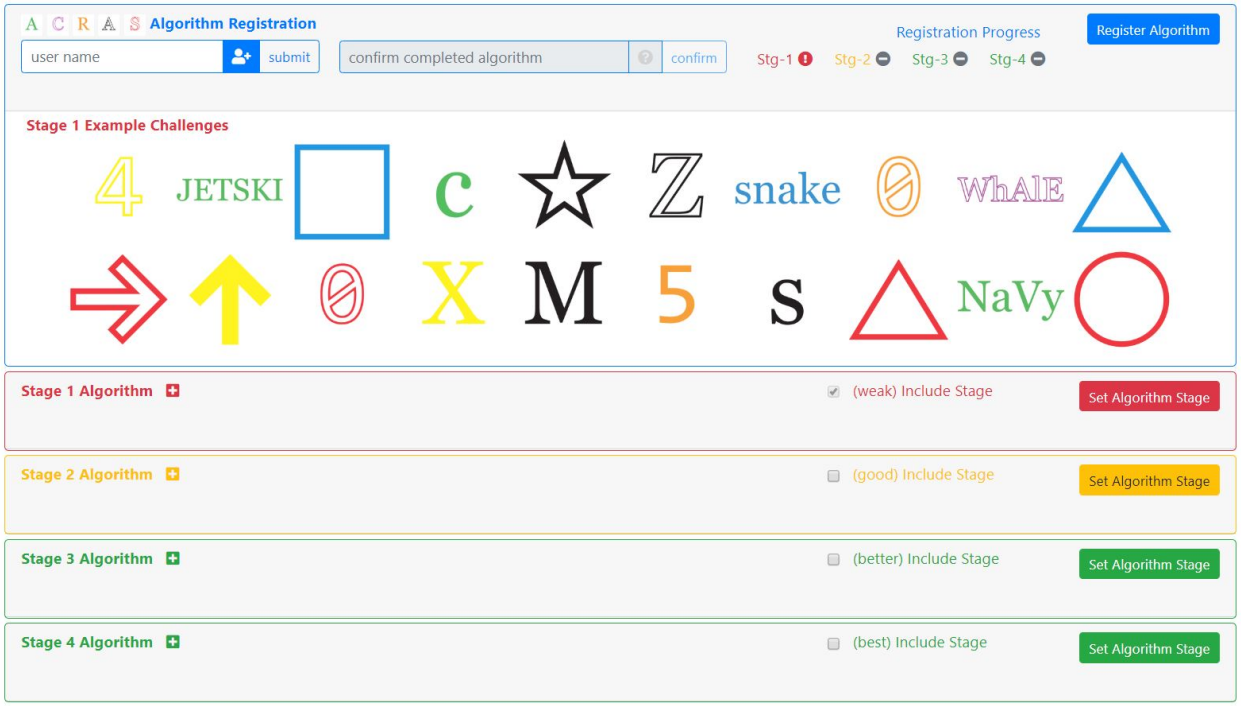

Figure 4.5: ACRAS algorithm registration stages

selections refine the Stage definition process.

4.7 Implementation.3 - ACRAS algorithm Backend

During the course of ACRAS frontend implementation, a minimal backend framework was developed in order to handle and validate frontend communication. Once the prototype frontend was largely complete, attention was shifted to the implementation of the ACRAS backend logic. As discussed earlier in the Tools + Techniques section, backend functionality was implemented using Python and the Flask micro-framework. The backend framework began by taking the functional incoming data traffic and extending it to bi-directional communication with the web application client-side. Upon establishing the proper communications pipeline between the two components, a series of incremental stages of backend ACRAS Algorithm processing were developed to handle backend logic.

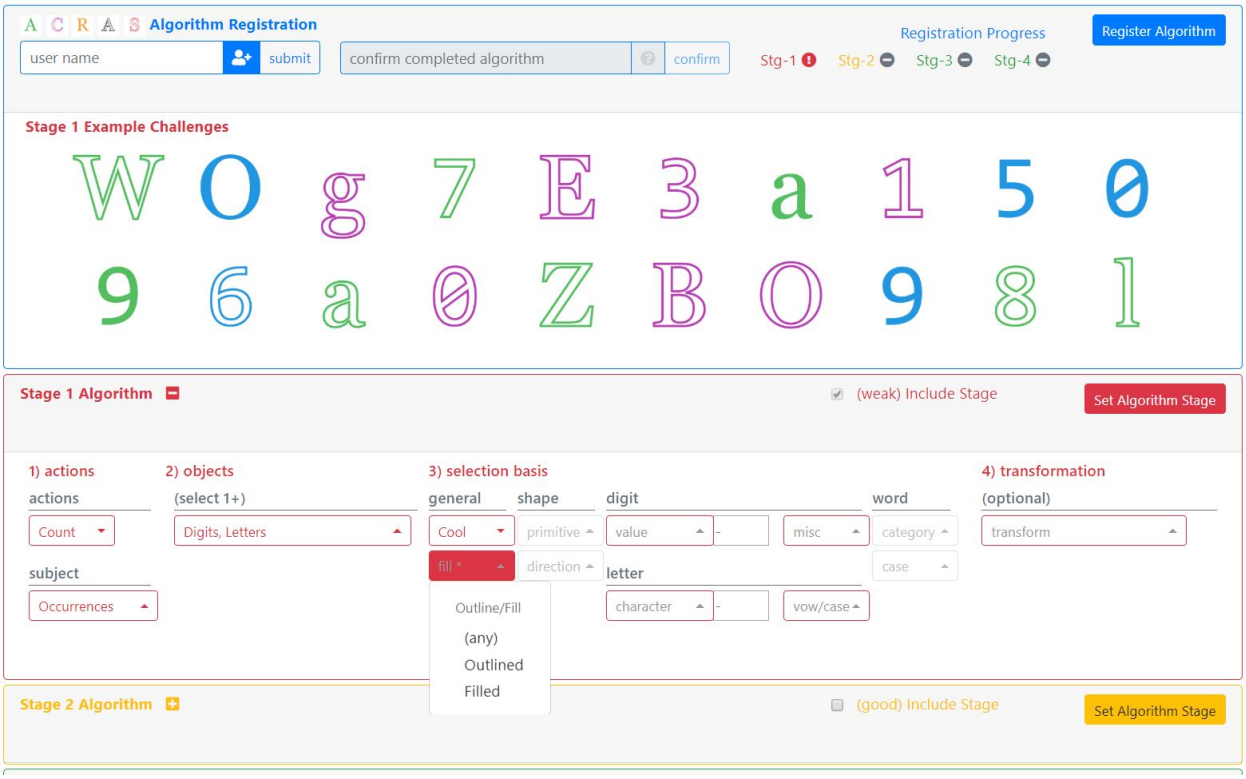

Figure 4.6: ACRAS stage registration interface

The ACRAS backend was organized according to several hierarchical classes to handle the logic and processing of user algorithms. The Challenge Factory module, which was developed in the first phases of system design and implementation was imported into the web application to handle all of the Challenge representation. In addition to the Challenge Factory and associated Challenge classes, two primary classes were constructed: AlgorithmStage, which is responsible for several higher-logic functionalities; and ChallengeSet, which coordinates and manages sets of Challenge objects.

The AlgorithmStage class's primary function is to handle the programmatic algorithm representations once defined by the ACRAS user and passed to the backend. This includes higher-level functionality such as parsing the JSON Algorithm and determining the Challenge-set requirements. Specific requirements and functionalities include determining

Algorithm stage quantities and Challenge populations for each, filtering valid qualifying Challenge pools, and constructing the expected response string from the final Challenge-set once it is built. The ChallengeSet class handles lower-level Challenge-related activities. These functions include selection of actual Challenges according to the Stage requirements; selection of non-qualifying, or noise Challenges; and Challenge-Set validation once the complete set has been assembled.

A secondary class, SelectionSet was implemented to support the Algorithm registration process on the ACRAS frontend. This class filters through the larger global set of Challenges for those which qualify for the current-level of Algorithm stage registration and passes a representative set of challenge graphics to the frontend UI. Several additional global support functions were developed during the course of the prototype as well, typically as ad-hoc quick and dirty solutions which ended up remaining in service. Future work on the system should include refactoring these methods into either an existing class as appropriate or a dedicated helper class.

4.7.1 Backend Activities

The ACRAS backend handles the logic which translates a user's Algorithm definition into a validated series of Challenges and generates a textual response. Following is a summary of related activities that occur in this process and noteworthy details, 4.7 illustrates the process flow.

Upon receipt of a user's Algorithm definition, the ACRAS backend first parses the definition to determine the number of stages, and sort them (in increasing order) according to their exclusivity requirement. Each ACRAS Algorithm stage has a specific requirement related to the degree of exclusivity the stage requires; for example, *echo* actions, such as an *echo:English* of a *shape* challenge impose a strict exclusivity requirement on qualifying Challenges. With the *shape* example, if there are more than one qualifying Shape in the Challenge-set, the ACRAS user has no way to tell which *English* representation should come first when concatenated into the OTP. Other Actions have their own functionality-specific

exclusivity requirements. *Echo* actions typically impose strict exclusivity, with only a single qualifier allowed unless a *sort* transformation is applied to the stage as is an option with *echo:Value(#)* for *digits* and *word:Numbers*, and *echo:English* for *letters*. The *sort* transformation provides a user with a prescribed order in which to enter the response into the OTP, so the ACRAS system responds by expanding the strict exclusivity to up to three qualifying challenges. As previously discussed in the *transformation* design section, this expansion has the effect of significantly increasing the state-space of the *echo* response. *Math* actions have different degrees of required exclusivity depending on the *subject*. *Add* may entertain anywhere from zero to four qualifying challenges; zero values added together result in a sum of 0. *Multiply* was implemented to impose a strict semi-exclusivity, always requiring two qualifying members to limit the complexity of mental-math required of users. *Count* actions have the least restrictive exclusivity requirement and may include up to five qualifying challenges.

Once the Algorithm stages have been sorted according to their exclusivity requirements, a set-population is generated for each stage meeting these exclusivity conditions and together with the sorted Algorithm definition is sent to the ChallengeSet class for Challenge-set construction.

For each stage in the user's algorithm, the same process is followed to generate a valid Challenge subset. First, a valid Challenge pool is prepared by filtering out any non-qualifying Challenges; these may simply be Challenges which do not meet the characteristic requirements of the stage, or may meet the requirements but be on an Algorithm-blacklist in response to a previous sStage's exclusivity mandate. Following the valid Challenge pool preparation, for each of the stage's set-population count, a Challenge is randomly selected from the valid Challenge pool and added to the stage's valid Challenge subset. Once the predetermined population count for the Stage has been met, any Challenges which if included further downstream would break the stage's exclusivity requirement are added to the Algorithm-blacklist, and the stage's valid Challenge subset is added to the greater Algorithm valid Challenge-set. The process is then repeated for each stage in the user's algorithm.

Once completed, the algorithm's valid Challenge-set is only a portion of a final Challenge-

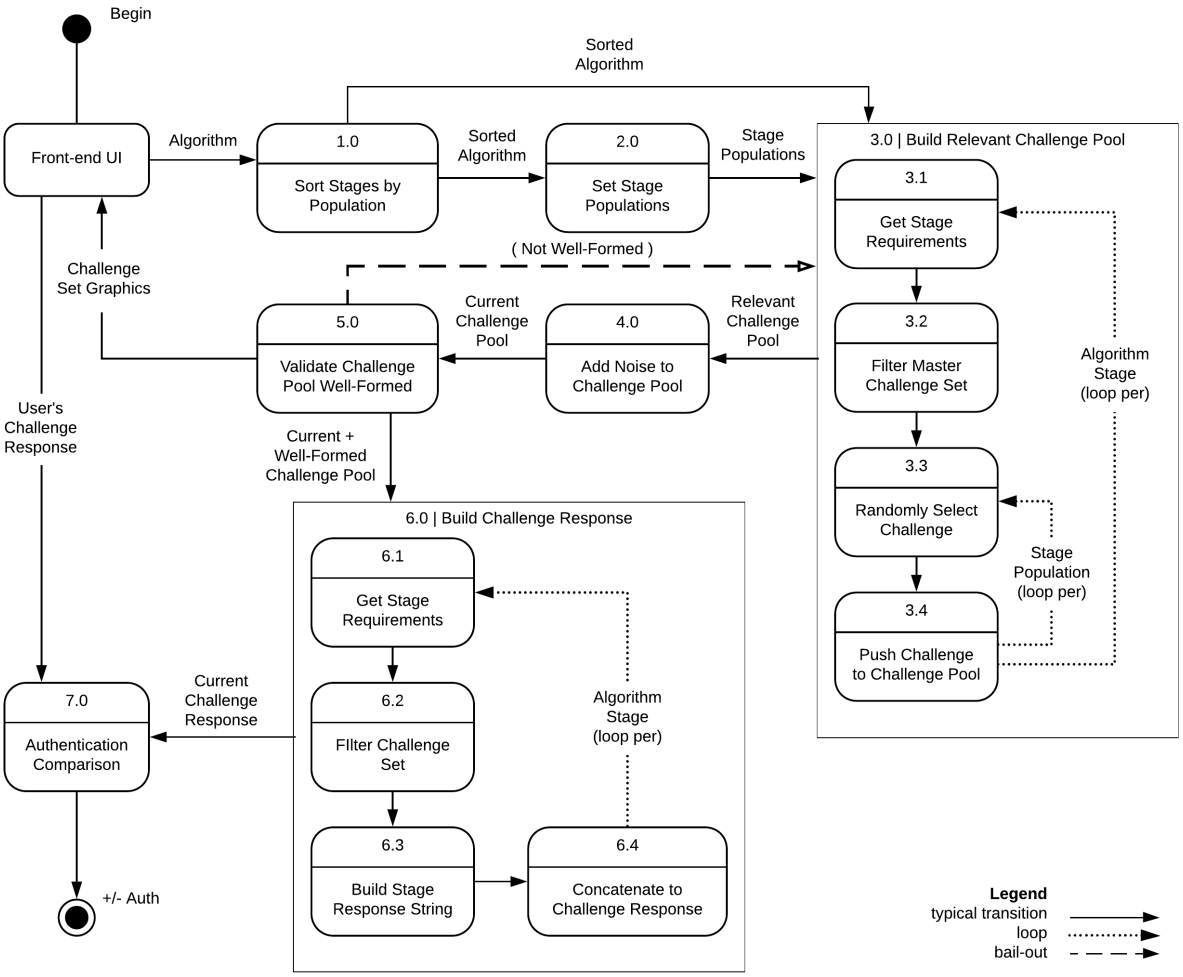

Figure 4.7: ACRAS backend flow diagram

set. The population count which was generated earlier in the process has a maximum valid Challenge threshold of 65% of the final set, or 13 valid Challenges. The remaining Challenge-set membership vacancies are filled with randomly selected Challenges from a subset of the complete set of Challenges which has been filtered by the Algorithm-blacklist which would explicitly break an exclusivity requirement. Once the noise has been added to the Challenge-set, the set is run through a final validation step to ensure that none of the exclusivity requirements have been broken, or the maximum valid threshold exceeded. If either of these conditions has occurred, the ACRAS system discards the entire Challenge-set and returns

to the beginning of the set definition process to try again until a complete valid set has been compiled. It is noteworthy that the prototype validation process has been successful in catching invalid sets although some of the error conditions returned during the course of user-experiments were not well-handled, requiring a manual restart of the Challenge-set generation process.

Once a well-formed Challenge-set has been generated and validated, the set is then sent to both ACRAS frontend for presentation to the user and the Build Challenge Response process. The Build Challenge Response process again parses the user algorithm definition, one Stage at a time, although this time the algorithm is stepped through in its original user-defined order rather than the order sorted by exclusivity. As the system interprets the algorithm selection basis, it filters the Challenge-set for qualifying members and then applies the algorithm's Action and Subject to generate the component portion of the overall OTP, concatenating each Stage's response until reaching the end having compiled a complete response to all Stages of the algorithm. This response is then used to compare against the user-provided response to determine if a match, resulting in ACRAS authentication.

4.8 Design.4 Database Integration

The final portion of implementation for the ACRAS prototype was the integration of the database for collection of data during user-experiments. Results from user-experiments, and discussion of their relevance are to follow in Chapter 5.

4.8.1 Database Integration

Typically, web applications are designed to be stateless and require interface with a data model for scalability. Due in-part to time-constraints but also in response to the single-user nature of the ACRAS prototype, the database integration for the broader system was not implemented, relying instead on a stateful application backend with the database solely serving the user-experiments. As detailed in the Tools + Techniques section, an SQLite database was selected to support user-experiments and interfaced via the SQLAlchemy tool

from within the backend Python code. A series of triggers with associated timers were selectively placed in function-calls throughout the backend to gather duration information for both the Algorithm registration process as well as the challenge/response process for each Challenge-set. Identifiers for both Algorithm and Challenge-set are derived by simply hashing the JSON stringified values for each. During the course of ACRAS prototype user-experiments, the following metrics have been collected for each participant and stored as JSON within the database:

- Unique user-name (database key)
- List of algorithm definitions with corresponding identifier (algo-ID)
- List of Challenge-sets with corresponding algo-ID and Challenge-set ID (set-ID)
- List of expected Challenge responses with set-ID
- List of user-provided Challenge responses with set-ID
- Duration of registration with algo-ID
- Duration of each challenge-response with set-ID

One metric not collected during the course of user-experiments was whether or not a participant referenced the registration interface which remained on the UI during the challenge/response process. This was concluded to be less critical a metric, as in real-world scenarios a user may keep reference materials (written reminders for example), and the corresponding time added to the recorded response duration accounts for this activity.

Chapter 5
RESULTS

Following the design and implementation parameters for the ACRAS system as detailed in the previous chapter, the results of a limited theoretical system-analysis and series of system-usability experiments are presented in this chapter. The analysis findings illustrate the ACRAS system's multiplicity of potentially relevant challenge collections and characteristics as well as the expansive state-space of possible resulting one-time-passwords (OTPs). Next, a description of the conditions and methodology governing system-usability experiments precedes a summary of measured results and subject impressions of the ACRAS prototype. Discussion of these findings follows in Chapter 6, Discussions & Future Work.

5.1 System Analysis

The ACRAS system relies on two fundamental mechanisms to produce unpredictable responses to presented sets of challenge graphics. The first, quantitative mechanism is an overall state-space of possible responses which exceeds 5.5 quintillion responses. The second is a qualitative property of the ACRAS system whereby a single response is capable of being arrived at by a multiplicity of different algorithm definitions.

5.1.1 ACRAS Response State-Space

This figure is based solely off of the variability of responses possible for each individual algorithm stage independent of the means with which the response was derived. Practically, what this means is that multiple algorithm definitions may be capable of producing the same response output given an appropriate set of challenges. For example, in figure 5.1, the algorithm used to generate the "8" component of the response "star82" from this set of

challenges could be *echo:value(#)* of *red:outlined word:number*. Another user's algorithm may contain a different stage definition which also produces the response "8" given an appropriate set of challenges. For example, figure 5.2 illustrates a different set of challenges which when paired with an appropriate algorithm definition, also produce a response with "8" as a component of the complete response "square38". In this case, the "8" is derived by a stage definition of *count:occurrences* of *filled letters*.

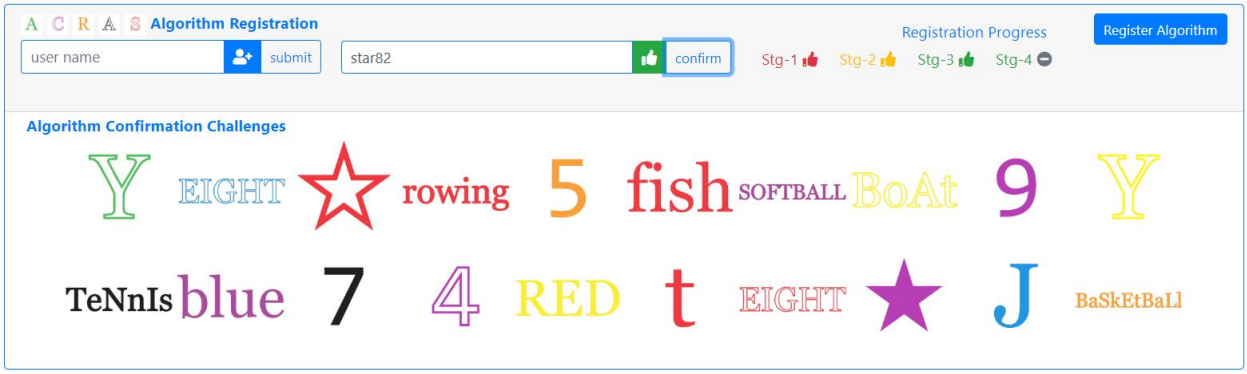

Figure 5.1: ACRAS algorithm and challenge deriving common response component *8*.

Figure 5.2: ACRAS algorithm and challenge deriving common response component *8*.

While this common result derived from independent algorithm and challenge pairings may seem initially unsettling, it is in actuality, an intended effect which provides resistance

to different attacks as will be demonstrated later in this section. As related to the overall variability of possible responses however, the effects of combinatorial explosion provided by multiple algorithm stages, each with an already substantial set of possible responses is quite considerable. The table in figure 5.3 quantifies each of the simple potential responses for various *action:subject:object* pairings. The column worth taking particular note of is the column labeled *unique*; as has been demonstrated with the two previous examples, multiple algorithm paths may lead a user to a common response. As a result, several of the *action:subject:object* pairings will have value overlap with others.

ACRAS Combinatoric Analysis						
simple outputs						
verb	subject	noun	qty	unique	type	examples
count	occurrences	-	9	**9**	number	0-8
echo	color	-	7	**7**	string	black, red, orange, yellow, green, blue, purple
	english	shape	6	**6**	string	circle, triangle, square, heart, star, arrow
		digit	10	**10**	string	zero, one, two, three, four, five, six, seven, eight, nine
		letter	52	**52**	string (letter)	a-z, A-Z
		word	120	**103**	string	full set less colors and numbers (above)
	value	digit	10	**1**	number	0-9
		word (number)	20	**10**	number	0-19
	even/odd	digit	2	**2**	string	even, odd
		word (number)	2	**0**	string	even, odd
	direction	shape	4	**4**	string	up, right, down, left
math	add	digit	36	**17**	number	0-36
		word (number)	76	**40**	number	0-76
	multiply	digit	37	**1**	number	... 81
		word (number)	143	**105**	number	... 361

Figure 5.3: Quantitative summary of ACRAS simple outputs

In addition to simple responses, the ACRAS system has been designed to incorporate another element, *transformation* which introduces further variability into the set of simple responses. Figure 5.4 quantifies the expansion of simple responses provided by *transformations*. Note that not all *transformations* will have an expansive effect, as the system experiences some value overlap in possible responses. For example, again consider figure 5.1, if a user was presented with this set of challenges and their algorithm contained *echo:English* of *yellow word:color*, the corresponding response to that stage would be "RED". At the same

time, yet another user who's algorithm included *echo:color* of *word:vehicle:lowercase* with a *capitalize* transformation would also respond "RED" when presented with the challenges in figure 5.2.

ACRAS Combinatoric Analysis					
transformed outputs					
response	transformation	qty	unique sets	total	remarks
numbers	sort low - high	165	140	**25**	combinations of 2 or 3 members (n!/(r!(n−r)!))
	sort high - low	165	149	**16**	combinations of 2 or 3 members (n!/(r!(n−r)!))
letter	sort a-z	23426	23426	**23426**	combinations of 2 or 3 members (n!/(r!(n−r)!))
	sort z-a	23426	23423	**23423**	combinations of 2 or 3 members (n!/(r!(n−r)!))
	capitalize	1	0	**0**	
	lowercase	1	0	**0**	
	invert case	1	0	**0**	
	double vowel	2	1	**12**	vowels [a, e, i, o, u, y] in both cases
	double consonant	2	1	**40**	consonants in both cases
word	capitalize	1	0	**0**	
	lowercase	1	0	**0**	
	invert case	1	0	**0**	
	double vowel	2	1	**132**	new word
	double consonant	2	1	**132**	new word
	omit vowel	2	1	**132**	new word
	omit consonant	2	1	**132**	new word
	reverse	2	1	**132**	new word
	*rotate clockwise	1	0	**0**	
	*rotate counterclockwise	1	0	**0**	

Figure 5.4: Quantitative summary of the effect of ACRAS *transformations*

The *sort* transformation has a particularly strong expansion effect when applied to *letter* challenges. By providing an ordering directive, *sort* modifies a typically singular *echo:English* for *letters* (*echo:value(#)* for *digits*) into a multi-part response with between one and three values. Using a possible character set of 52 members (lowercase and uppercase), each of the *letter sort* directions has an expansion effect equal to the number of combinations of three characters from the set:

$$ {}^{n}C_{r} = \frac{n!}{r!(n-r)!} $$

The combinatorial explosion results in nearly 47,000 possible sorted sequences of three *letters*.

Note that *letter sort* in the decreasing direction has value overlap with the *word:number* challenges "one", "OnE", and "ONE" resulting in a slightly lower expansion effect. Similarly, *sort digit* experiences considerable overlap with various numerical values resulting from *count, echo:value(#)*, and *math* actions.

ACRAS Combinatoric Analysis						
summary of possible responses						
response	simple	transformed	total		algorithm	possible responses
numbers	183	472	655	48,454^1	single stage	48,452
letter	52	46,953	47,005	48,454^2	two stages	2,347,596,304
words	132	660	792	48,454^3	three stages	113,745,736,121,408
		total	**48,452**	48,454^4	four stages	5,511,208,406,554,460,000
					total	**5,512,232,190,957,410,000 (5.5E+18)**

Figure 5.5: Summary of possible ACRAS response outputs resulting from between one and four algorithm stages including both simple and transformed responses.

The table in figure 5.5 summarizes the possible permutations of ACRAS responses for between one and four algorithm stages. This quantity may also be stated simply as the sum of all possible permutations for each quantity of algorithm stage, or:

$$\sum_{r=1}^{4} n^r$$

Where n is the number of possible stage responses, and r is the number of stages within the algorithm. This results in over 5.5 quintillion possible permutations of multi-stage response. This value represents the abstract limit to possible algorithm responses. As has been demonstrated, this number includes repetition within stage responses. For instance a user-algorithm could, in-theory, be created with the following stage definitions:

Example User-Algorithm Definition

1. *echo:color* of *shape:outline*
2. *echo:color* of *word:animal:lowercase*
3. *echo:English* of *yellow:filled word* in *lowercase*
4. *echo:color* of *filled letter:lowercase:consonant*

If a user with this algorithm definition was presented with the challenges from figure 5.1, the corresponding response would be "redredredred". Obviously, the output "red" is relatively limited with respect to derivation pathways, but one can extend this concept to a far more common response-type such as numerical values and immediately see the potential for repetition within a multi-stage algorithm response.

As an abstract ceiling, no single user-algorithm is capable of 5.5E+18 possible responses (the highest theoretical limit for a single algorithm would require four serial *letter sort* stages and result in approximately 3.0E+17 responses). This metric is useful for understanding the potential state-space for all ACRAS algorithms, with an approximate equivalency to five times that of a traditional 10-character password using a 64-character set (1.15E+18). Such a comparison is not wholly commensurate however, as ACRAS is an OTP system and the password will change with new challenge sets. An analysis of the state-space associated with actual algorithms defined during user-experiments follows later in this Chapter.

5.1.2 Plausible Alternatives

In addition to, and further compounding the extensive set of possible ACRAS algorithm result outputs is the non-singular nature of their derivation. As has been illustrated several times already, identical ACRAS stage responses may be arrived at via many different means. The multi-dimensional aspect to ACRAS challenges intentionally provides a significant volume and diversity of conditions and characteristics within a challenge-set in order to complicate the reverse-engineering of the mechanism by which the response was arrived at.

By employing Almesheka and Spafford's [2] three deception techniques of dazzling, mimicking, and decoying, a quality of uncertainty is introduced to simple analysis of corresponding ACRAS challenge/response pairs. Each of these deception techniques is at-play within a single ACRAS challenge-set, often with challenges representing multiple techniques simultaneously.

Fundamental to the ACRAS system, challenges act as dazzlers. In any one challenge-set up to 65% of the challenges may be valid contributors to the algorithm response, leaving a minimum of 35% (or seven of 20) as mimickers. These mimicking challenges will appear no different from valid challenges to an outside observer unfamiliar with the user's algorithm. In fact, challenges that act in a mimicking capacity for one user's algorithm may be be entirely valid in the context of another user's algorithm. Challenges also exhibit decoying behavior, both as valid challenges which present some irrelevant characteristics along side of relevant traits, and as mimickers which merely act as noise within the challenge-set.

During the course of ACRAS research, one of the intended analysis efforts was a quantitative review of challenge-set and response data to determine the number of plausible alternatives for a representative pair. Unfortunately, scripting this problem using Python proved to be a non-trivial task and was abandoned in-favor of a manual analysis due to project time constraints. As the manual analysis was conducted on data obtained from user experiments, the details of this effort, are discussed further in the user-experiments section of this Chapter.

5.2 User-Experiment Conditions & Methodology

As a part of the ACRAS research, a series of user-experiments were conducted using the ACRAS prototype system to gather useful data-points and usability metrics. The experiment pool is comprised of 30 subjects of varying age, gender, and perceived technical sophistication. Figure 5.6 illustrates the distribution of age and perceived technical sophistication of user-experiment subjects. As one can see from the figure, there is a greater density of subject representation within the 35-45 age range and a higher concentration of perceived technical

ability within the central band of 15-25, 25-35, and 35-45 age ranges.

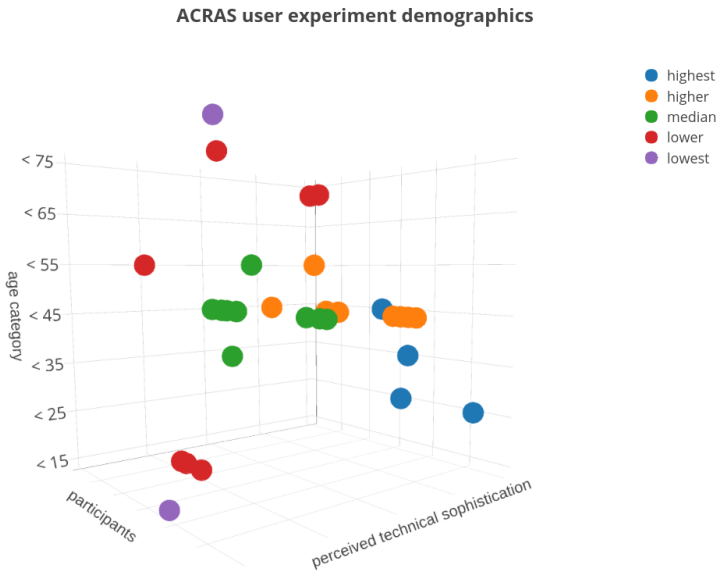

Figure 5.6: ACRAS User-Experiment subject demographics

All of the experiment sessions were scheduled and arranged in advance, and all subjects are familiar to the researcher. Each of the experiments was conducted on the researcher's personal computer system, a Lenovo Yoga 2 Pro 13" ultrabook with a Bluetooth wireless mouse. Following is a description of the user-experiment process typically followed.

The user-experiment sessions typically began with an introduction to the motivation behind the project; some background related to the inherent weaknesses of traditional, static, text-based password systems; and usually some light conversation to ensure the subject was at-ease before commencing with the experiment. Reference was made to some data being collected in the background of the system, but details of what metrics were being collected were not discussed and in-particular, the timed-nature of responses was not shared with subjects so as to not introduce additional stress. An overview of the experiment process

was given with a rough estimate for duration to be sure each subject was aware of the time commitment the experiment typically required.

Following the introduction, a demonstration of the algorithm registration process occurred. The procedure for this demonstration varied between session but largely followed a similar path. Step-by-step narration of what was occurring, why it was happening, and what ramifications it would have downstream were typical in all demonstrations. Demonstration algorithms varied between 3-stage and 4-stages depending on the subject's perceived level of understanding of the system and registration procedure. Typically, at least one of each type of action (*count, echo,* and *math*) were demonstrated, with an emphasis placed on demonstrating additional selection basis options to illustrate the functionality and repercussions of the additional specificity.

After defining the algorithm, a series of challenge/responses was conducted to demonstrate how the algorithm which had just been defined with the subject was applied to the various challenges to generate an OTP. The total number of challenge/responses varied by session depending on how well the subject appeared to understand the process but typically ranged from three to six attempts. For the first few attempts, reference to the registration interface was used to demonstrate that this was an option, and typically at least one response was intentionally input incorrectly to both illustrate that attention to detail would be required, as well as to normalize incorrect responses so as to not set expectations of 100% accuracy. By the last demonstrated attempt, the OTP was entered correctly at least once without using the reference information to demonstrate the feasibility of a correct memory-based response.

At this point, the subject was asked if they had any questions about how the ACRAS prototype performed or about what they were expected to do. After addressing any outstanding questions, the browser was refreshed and the computer turned over to the subject to define their own algorithm. Questions were responded to during their registration if they were brought up by the subject, in a small number of cases additional direction was provided to help with minor UI details, or in one case steer the subject away from a registration which

would have resulted in a circular definition error with the exclusivity requirement that was not anticipated nor explicitly handled programmatically.

After registering the algorithm, the subject was asked to conduct a series of 20 challenge/response attempts. During this time, no coaching was provided, and questions were replied to with minimal direction and an accompanying explanation that the system was intended to stand on its own and the researcher was seeking to avoid steering or influencing the results in any direction by providing guidance during this portion of the experiment. The two exceptions to this rule being for character-recognition questions (further elaboration on this issue to follow in the Discussion & Future Work section), and explanation of incorrect responses if the subject requested them. Subjects were asked to leave incorrect responses without correcting them and proceed with the next challenge/response.

Following the 20 attempts at the challenge/response, the subject was presented with a short verbal survey. The survey consists of questions related to the subject's current password habits, and some questions related to their impressions of the ACRAS prototype system. Light discussion typically accompanied these survey questions and an effort was made to provide distraction from the system for approximately five minutes before the subject was surprised with a request to conduct the challenge/response process two more times without reviewing the registration reference information. The intent behind the surprise two requests was to see how well a subject could recall their own algorithm after taking a break from the serial repetition of the initial 20 responses and having shifted their thinking away from the operation of the prototype system.

5.3 User-Experiment Results

The following section summarizes and and discusses the results of the ACRAS prototype user experiments. Results have been divided into two primary sections. Recorded metrics covers details including algorithm definition metrics, manual analysis of stage data correlation, user response accuracy, registration and response durations. User-survey summarizes the responses to survey questions related to the usability of the ACRAS prototype.

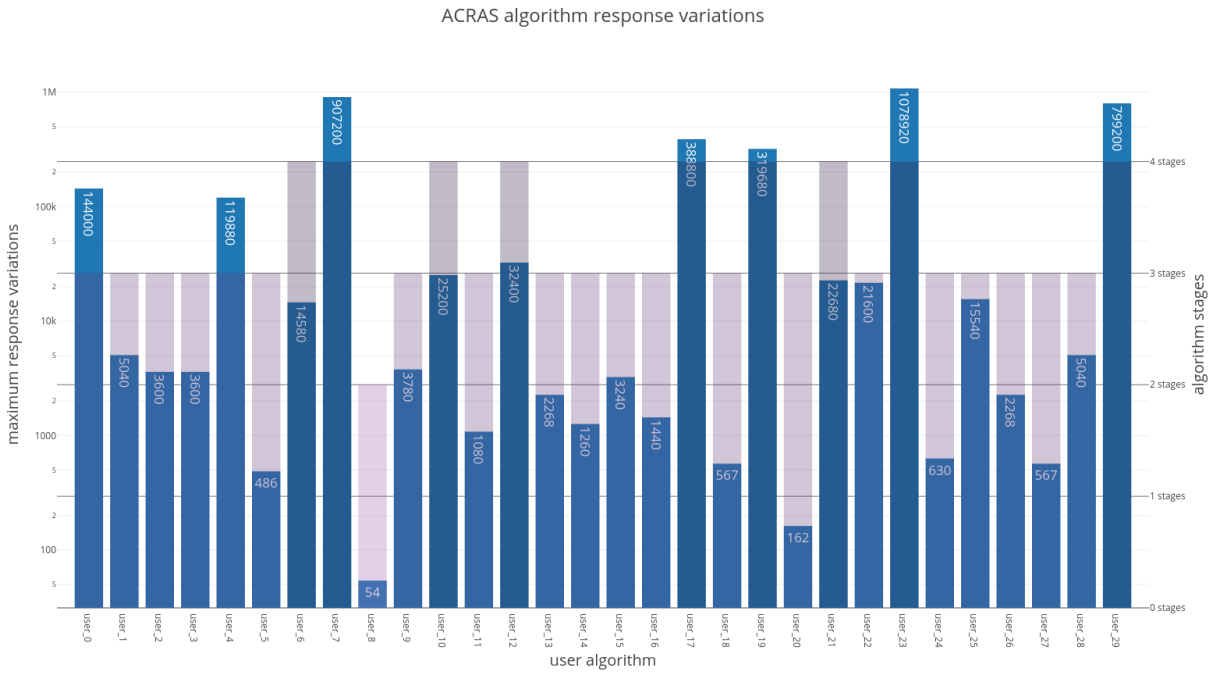

Figure 5.7: ACRAS User-Experiment subject demographics

5.3.1 Recorded Metrics

A description of the abstract ACRAS response state-space was provided earlier in the Chapter. A similar analysis of each subject's user-defined algorithm has been conducted and is presented in figure 5.7. The blue bars in the foreground of the chart and their corresponding tick-marks on the left side depict the number of possible responses each algorithm is capable of producing (note the logarithmic scale to the Y-axis). The purple bars in the background and corresponding tick-marks on the right of the chart depict the stage-count or "depth" of a user's algorithm. Together, these two metrics provide a relative means of judging an algorithm's strength.

As one might expect given the combinatorial mathematics involved in deriving the number of possible responses, there is a noteworthy correlation between an algorithm's stage depth and the quantity of possible responses. In only one case, (user_08) did an experiment subject

elect to include fewere than three stages in their definition. The corresponding possible responses for this algorithm is a comparatively low value of 54 possible responses. User_08's algorithm definition includes: stage 1) *echo:English* of *outlined shape*, with six possible values [circle, triangle, square, heart, star, and arrow]; and stage 2) *count:occurrences* of *blue:filled letters*, with nine possible values [0-8]. In stark contrast, there were seven algorithms defined during the course of user-experiments which exceeded 100,000 possible response each, the highest of which belonging to user_23 with over 1 million possible responses. Each of the top five algorithms defined four stages, with those responsible for the sixth and seventh highest possible responses each defining three stages. These two 3-stage algorithms each made use of an *echo:English* of *word* stage without any further selection-basis or *transformation* narrowing the scope of the *word* which results in 360 possible responses. The choice of this broadly-scoped stage in these two cases explains how these 3-stage algorithms have a higher possible response count than the 4-stage algorithms defined by user_06, user_10, user_12, and user_21.

Despite the previously noted correlation between stage-count and number of possible responses, the range of individual stage-specific populations demonstrates that it is not reasonable to conclude a simple causation between stage-count alone and algorithm strength. In order to have a significant effect on result possibilities, an ACRAS user should endeavor to include some stage definitions with a large range of possibilities. With that stated however, anecdotal evidence suggests the additional complexity associated with an attacker needing to correctly derive component responses for higher stage-counts increases the general resistance to guessing attacks as compared to fewer stages with higher variability. The reasoning behind this is simply that within a challenge-set of only 20 ACRAS challenges, there are a limited number of potential valid challenges that may affect any one current response.

Challenge/Response Decoupling

To this point, the focus of this Chapter has been on the range of possible responses a user's algorithm may produce. This quantifiable aspect has been designed into the ACRAS

system in order to complicate *guessability* attacks as outlined in the earlier section defining the attacker model. A second mechanism, decoupling, has been incorporated specifically to address the risk of *observability* attacks. The concept of decoupling as it relates to the ACRAS system involves the both literal separation of challenge from response, as well as the potential for a variety of interpretations through ACRAS challenge multi-dimensionality.

The first means of decoupling within the ACRAS system, the separation of challenge from response provides a coarse scale mechanism to keep one from clearly identifying itself as the origin the other. By separating the text-input response mechanism from the representation of challenges, it becomes harder to immediately correlate the two. In contrast, many Graphical Password Authentication (GPA) systems utilize a simple pick-select technique for identifying relevant images or challenges and providing a response. The precise pick-select details may vary from a literal clicking of the image, to more novel approaches like that presented by Gao et al. [13] in their proposed CDS system in which a user draws a connecting line across each of the randomly placed challenges. In either case however, the response mechanism either significantly narrows the field of possibly relevant challenges or outright identifies them to observers.

The ACRAS system on the other-hand utilizes a text-based response mechanism which affords it the opportunity to remain separate from on-screen challenges. As an ACRAS user responds to the presented challenge-set, the challenges themselves remain unaffected, no touching, clicking or other means of selection is necessary. A user must simply recognize a challenge's relevance, interpret it according to their algorithm, and respond via entering text. While no physical (or on-screen) interaction with the ACRAS challenge is required, it is worth noting however that during user-experiments it was observed that many subjects (roughly 50%) would either use the mouse-pointer or a finger to either intentionally, or even subconsciously point at challenges as they scanned the set for relevant members. Further, users tended to pause, even if only a slight hesitation as they came across a relevant challenge during their scan. This behaviour appeared to diminish as users became more comfortable with the system, but in some cases remained throughout an entire session with a subject.

In addition to the elimination of pick-select identification, the textual response to graphic challenge-data affords additional system decoupling as well. By switching contexts from graphics to text, a data abstraction occurs which allows singular characteristics of the ACRAS challenge to be either recognized or interpreted in isolation from the remainder of the challenge graphic. This abstraction and characteristic-isolation leverages the multi-dimensional nature of the graphic challenge to further decouple the response from the defining challenge data. Throughout this document, considerable emphasis has been placed on the multiple reads or interpretations one may derive from a set of ACRAS challenges. The importance of this system feature can not be overemphasized, as it is the foundation of the system's resistance to observability attacks. Stated in another way, the ACRAS system relies on an asymmetrical relationship between a deterministic response which results from the interpretation of a seemingly random challenge-set with a known user-defined algorithm, contrasted by the confusion and distraction provided by the compounding of numerous potentially relevant challenge characteristics when one does not know the secret algorithm.

Plausible Alternatives

In order to investigate the effectiveness of this multiplicative challenge environment on observability, and specifically on intersection attacks, a manual analysis of corresponding OTPs and challenge-sets has been conducted. The manual analysis reviewed actual challenge-set data collected from a subject during the ACRAS user-experiments. A series of challenge-sets and their corresponding OTPs were reviewed to see how many successive challenge/response pairs were required to confidently identify the user-algorithm via an intersection attack.

Manual Analysis Round 1

The first recorded round resulted in the OTP "810EIGHT", and the challenge-set is represented in figure 5.8. The concentration of digits towards the beginning of this OTP complicated the task of correctly decomposing the password into clearly defined stages. As a result, multiple parallel cases were initially considered including: three distinct number stages, rep-

Figure 5.8: ACRAS User-Experiment subject user_19 Round 1 Challenge-Set

resented by "8", "1", and "0"; and two possible two-stage scenarios: "8", and "10"; and "81" and "0". The only possible means of generating "81" with ACRAS' is current definition is a *math:multiply* of two values of nine. No challenges with the value of nine are present in this set, so the third decomposition scenario was ignored.

After decomposing the first OTP into these two possibilities, the challenge-set was reviewed for possible means of decomposed component generation. The number of possible generations for each component were significant; as a result, documentation was limited to approximately 10 possibilities with the understanding that if downstream analysis did not afford any intersections that the earlier challenge-set could be revisited to identify additional candidates.

Both potentially valid candidate decompositions for the first challenge/response round identify "8" as the stage-one OTP component. Analysis of the first round challenge-set reveals a substantial number of plausible alternatives for the derivation of "8". Figure 5.9 summarizes the set of possibilities which were considered[1] during manual analysis.

ACRAS Plausible Alternatives Analysis user_19 Stages 1-3: Round 1								
component	action	selection-basis	component	action	selection-basis	component	action	selection-basis
8	value	orange filled digit over 3	1	value	black digit	0	value	red digit
		blue uppercase filled word			black outline			red outline digit
8	count	warm color	1	count	...	0	count	...
		fille shapes & outline digits						
		shapes & outline letters	1	math	add black digits	0	math	add red
		shapes & fill letters			multiply black digits			multiply red
		digits & uppercase letters						multiply warm
		outline digits & letters						multiply primary
		digits>0 & letters						
		digits & outine words	10	value	green number word			
8	math	add primary colored digit	10	count	digits & words			
		add digits exactly 4			letters & shapes			
					primary color			
			10	math	add cool digits			

Figure 5.9: user_19 Plausible Alternatives Analysis Round 1, Stages 1-3

For stage-one's "8", there are plausible alternatives which include *echo:value(#)*, *count*, and *math:add* stages. The next decomposition has the potential for multiple reads, either as "1" in a four-stage algorithm, or "10" if the algorithm is only three stages. As this is undetermined during the first round, both conditions are analyzed. For the former case, the "1" may be representing an *echo:value(#)*, any number of different *count* operations, or a *math* operation. In the latter, the "10" may also be derived via any of the three actions, although clearly involving a different selection basis. Finally, in the case were the second decomposition is "1" (vs. "10"), it follows that there would be a third decomposition for the "0". Similar to the "1" in the previous decomposition, the "0" may be represented via

[1]Note the table in 5.9 indicates that it contains data for stages 1-3, and the table in figure 5.10 represents stage-4.The distinction is made in response to the need to split data into two tables for formatting purposes. At the point of round-1 analysis, it is still undetermined that there are four stages in this user-algorithm.

all three actions, including a vast set of *count* operations where the object is anything not represented within the current challenge-set.

ACRAS Plausible Alternatives Analysis user_19 Stage 4: Round 1		
component	action	selection-basis
EIGHT	english	orange filled digit>3 (capitalized)
		uppercase word
		blue word
		uppercase blue word
		uppercase number word
		upercase filled word
		blue filled word
		blue number word
		uppercase blue filled word

Figure 5.10: user_19 Plausible Alternatives Analysis Round 1, Stage 4

Regardless of whether the "810" represents two stages or three, there is a final-stage to the user-algorithm which derives "EIGHT" from the current challenge set. This final stage is clearly an *echo:English* action, however there are two potential challenges within the current set which may be used to derive "EIGHT". In addition, there are several different selection-bases which may be used to distinguish the relevant challenge within future challenge-sets.

Manual Analysis Round 2

Figure 5.11: ACRAS User-Experiment subject user_19 Round 2 Challenge-Set

Following the first round of analysis, a second round was conducted which also included an unrestricted collection of plausible alternatives. Introduction of a second data set affords one the option to focus solely on intersections, although given the manual nature of the analysis, the additional coverage associated with a second round of full-scope possibilities was opted-for in order to reduce the chance of missing anything relevant in the first challenge-set.

ACRAS Plausible Alternatives Analysis user_19 Stages 1-3: Round 2								
component	action	selection-basis	component	action	selection-basis	component	action	selection-basis
8	value	orange digit	0	value	yellow fill < 6	18	value	-
		purple filled digit						
		purple digit over 7	0	count	...	18	count	triangles, digits, letters, words
								all (digits > 2)
8	count	shapes & outline letters	0	math	multiply yellow fill			all (uppercase letters)
		digits			multiply yellow<6			all (mixed case words)
		filled digits & words			add fill<4			
					multiply warm fill<6	18	math	multiply warm outline digits
8	math	add yellow digits						add outline digits

Figure 5.12: user_19 Plausible Alternatives Analysis Round 2, Stages 1-3

ACRAS Plausible Alternatives Analysis user_19 Stage 4: Round 2		
component	action	selection-basis
HELICOPTER	english	blue word
		uppercase word
		vehicle word
		blue filled word
		blue uppercase word
		blue vehicle word
		filled uppercase word
		filled vehicle word
		uppercase vehicle word
		blue filled uppercase
		blue filled vehicle
		blue uppercase vehicle
		blue filled uppercase vehicle

Figure 5.13: user_19 Plausible Alternatives Analysis Round 2, Stage 4

Figure 5.11 depicts the challenges from user_19's second round of challenge/response. The corresponding OTP for this set is "8018HELICOPTER" which begins to immediately suggest some things about the number of stages within the algorithm. Recall, our first round was either three or four stages, including the final *echo:English*. Again, we are prompted with the possibility that the digits at the beginning of the OTP are representing either two or three stages: "80", and "18"; or "8", "0", and "18" respectively. With the total stage limit for the system currently set at four; at most, these digits may represent three stages. In addition, there are no value responses that employ a leading "0", making "01" an invalid possibility. A brief analysis of the makeup of the first decomposition stage quickly identifies the constituent members necessary to derive "80" (which can only be achieved with *math:multiply*) are not present in the current set. This indicates we are dealing with a four-set algorithm, which may be decomposed into the following components: "8", "0", "18", and "HELICOPTER". Figure 5.12 lists a series of plausible alternatives for the first three stages of the algorithm according to the second round challenge-set. As one can see, already we have established some intersections with the first round.

Figure 5.13 also identifies a number of intersections, in this case for the final algorithm stage between "EIGHT", and "HELICOPTER". It is notable that the number of intersections for this final stage include various combinations of the shared characteristics between the two *word* challenges which are capable of producing the correct output. With limited exposure to only two representative challenges, one still does not know what the precise selection basis is for the final stage.

Manual Analysis Round 3

Figure 5.14: ACRAS User-Experiment subject user_19 Round 3 Challenge-Set

By the third round of analysis, the general outline of the user's ACRAS algorithm have begun to take shape. With complete data from the first two rounds, We have defined that there are four stages to the algorithm, the first three will result in a numerical values, and the fourth is an *echo:English* of a (so far) *word blue:filled uppercase* challenge. The third round of analysis marks the previously referenced shift from alternative collection to intersection and filtering of the collections. This change in technique coupled with analysis of the decomposed stages of the third round OTP, "6210JUICE" narrows the focus on stages one, appears to identify stage three, and loosens the selection-basis specificity on the final-stage to eliminate fill or outline requirements. The second stage of the algorithm continues to enjoy a significant set of plausible alternatives.

ACRAS Plausible Alternatives Analysis user_19 Stages 1-3: Round 3								
component	action	selection-basis	component	action	selection-basis	component	action	selection-basis
6	value	orange digit over 2	2	count	black digit	10	math	multiply warm outline
		orange filled digit over 3			blue digit			
					yellow digit			
					digit exactly 1			
					cool heart shape			
					outline lowercase word			
					green word			
					warm lowercase word			
					lowercase number word			

Figure 5.15: user_19 Plausible Alternatives Analysis Round 3, Stages 1-3

ACRAS Plausible Alternatives Analysis user_19 Stage 4: Round 3		
component	action	selection-basis
JUICE	english	blue word
		blue uppercase word

Figure 5.16: user_19 Plausible Alternatives Analysis Round 3, Stage 4

In three rounds of analysis, the second stage has been presented as "1", "0", and "2". Possible *echo:value(#)* and *math* actions appear to have filtered themselves out, leaving *count* in play. Earlier reference was made to the sheer volume of possible selection-bases which could result in a count of "1" or "0"; "2" is a considerable improvement although there are still several viable possibilities which warrant consideration. Figure 5.15 details these plausible alternatives which still meet the "1" and "0" *count* conditions presented by the first and second stages.

Manual Analysis Round 4

Similar to the third round, the fourth round of manual analysis further narrows the set of plausible alternatives. The fourth round OTP, "8330FUSCHIA" supports third round inferences on stages one, three and four, and additionally narrows the second stage possibilities down to a single plausible alternative. Stage one and stage four each retain two possible *echo* definitions with sufficient respective overlap to make identification of the valid challenges likely, although not certain. Stage three had already been narrowed to a single possible *math* operation with consistent correlation to the OTP.

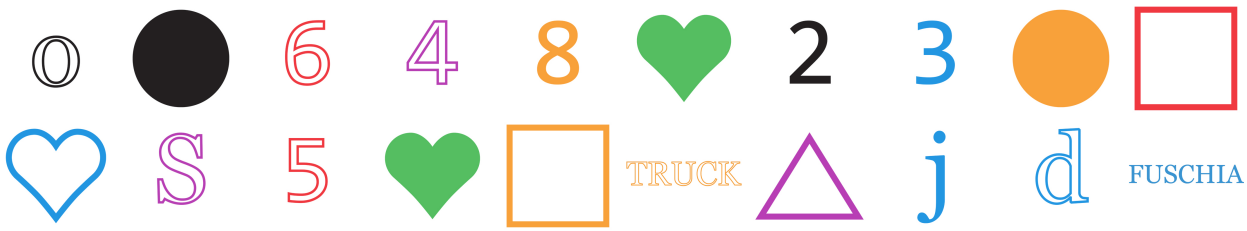

Figure 5.17: ACRAS User-Experiment subject user_19 Round 4 Challenge-Set

ACRAS Plausible Alternatives Analysis user_19 Stages 1-3: Round 4								
component	action	selection-basis	component	action	selection-basis	component	action	selection-basis
8		orange filled digit over 3	3	count	cool heart shapes	30	math	multiply warm outline

Figure 5.18: user_19 Plausible Alternatives Analysis Round 4, Stages 1-3

ACRAS Plausible Alternatives Analysis user_19 Stage 4: Round 4		
component	action	selection-basis
FUSCHIA	english	blue word
		blue uppercase word

Figure 5.19: user_19 Plausible Alternatives Analysis Round 4, Stage 4

Recalling from earlier discussions of attacker-model and related benchmarks for success, for the purposes of this research, the estimated number of shoulder-surfing attacks a user would be expected to withstand is four. This number accounts for session timeout, accidental closure, and general conservativity. We concede that the manual analysis of one single user's challenge/response data can hardly be considered exhaustive or statistically representative of the entire ACRAS system's resistance to intersection attacks. With that stated however, this manual analysis serves to illustrate the complexity and general level of effort that is required to intelligently correlate the data for reverse engineering of ACRAS algorithms.

After four rounds of manual analysis of data collected from a single user-experiment session, the algorithm has been reversed with reasonably close accuracy. There remains a minor degree of uncertainty as to the precise selection-basis for stages one and four, however stages

two and three have been accurately filtered via correlated challenge/response intersection. With the relatively close selection bases defined for each, identification of valid challenges within future challenge-sets would likely not pose much problem, potentially affording the reversing party the opportunity to fine-tune the basis on-the-fly based on variance in presented challenge characteristics. It is noteworthy that this intersection analysis has been conducted with complete data, resulting from the limited set of algorithmic variability implemented into the ACRAS prototype. Common implementation techniques for authentication systems, such as obfuscation of user input entry (onscreen character representation as a dot or asterisk) would likely complicate the collection of such complete data. In addition, there is considerable space for increasing the potential dimensionality of ACRAS challenges which would further expand the set of plausible alternatives.

User-Experiment Usability Metrics

Security in the absence of usability is an abstract metric at best. In addition to the security benefits, one of the underlying principles which has driven the research into the ACRAS system is the significant opportunity for increased usability over traditional static authentication systems. As has been referenced several times already, both the recognition of graphic content, as well as the recall assistance provided by graphic cues provide considerable benefit over the rote recall of a piece of static text. In order to examine the effects of these mechanisms within the context of the ACRAS system, several metrics were recorded during user-experiments. The two primary recorded metrics related to usability of the ACRAS prototype involved the accuracy of subject responses and the duration associated with both the initial algorithm registration phase and that of the challenge/response activity itself. The following section will discuss these results.

Algorithm Registration

One of the acknowledged benefits to the traditional username/password system is the ease and speed with which it is performed. As the system is ubiquitous in modern computing, it has become reflexive in operation, if not in the generation of secure credentials. Some of the ease of use for these traditional authentication systems may be attributed to their familiarity, but there is an undeniable simplicity to the system as well. The ACRAS system on the other hand, trades some of this simplicity for what we hypothesize to be an improvement in both security and memorability. This trade-off is acknowledged and the ACRAS system goals do not include an improvement in the time it takes to initially set-up or register a user's algorithm compared to that required for registering a static text-based password. Further, as the system is unique unto itself, the system does not share commonalities with other common authentication systems which a user can use to help orient themselves. In short, it's unfamiliar territory, and it requires some exploration to grasp and implement the system nuances.

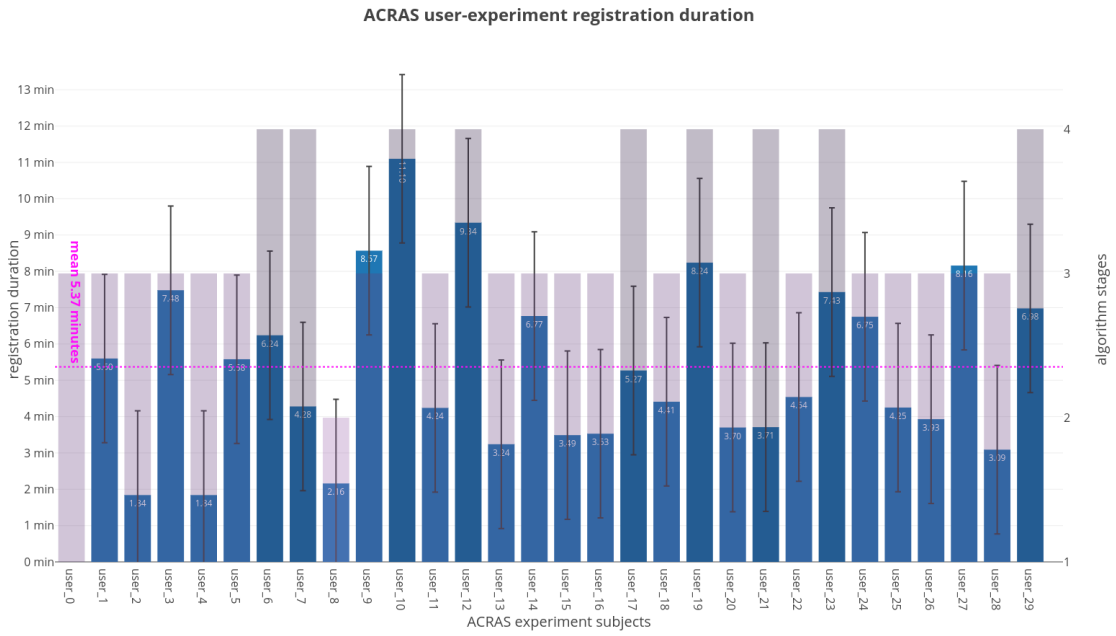

Figure 5.20: ACRAS User-Experiment Algorithm Registration Durations

Figure 5.20 illustrates the range of time subjects spent registering their algorithm. The mean registration duration was just over five and a half minutes. The blue bar represents how long a subject took to review their options, and complete algorithm registration. Several subjects (user_02, user_05, user_10, and user_12) registered multiple algorithm definitions, which accounts for at least part of the two highest durations recorded. The purple bars in the background of the chart represent the same stage-count metric as is represented in figure 5.7 for a relative comparison of duration time with algorithm depth.

Challenge/Response Duration

After completing the registration of their algorithm, subjects were asked to conduct a series of 20 challenge/response to collect several points of data. Similar to the initial registration phase, each successive challenge/response attempt was timed to gauge how easy or difficult the authentication mechanism was for the subject. The time was taken from the moment the challenge graphics were sent to the browser to the time the subject clicked on the "Confirm" button to validate their response against the expected response. The measured time represents how long it takes for a subject to interpret the challenge-set, recall how to apply their algorithm, actually apply their rule-set, and finally, enter the OTP into the text field within the UI. The time duration also includes any time spent referring to the actual algorithm definition interface which remained at the bottom of the ACRAS UI as a convenience for user reference during the experiments.

Figure 5.21 depicts the challenge/response duration data collected from 30 subjects across 20^2 challenge/response attempts. Each subject is represented by a single line within the graph, with the X-axis depicting attempt number, and the Y-axis representing the duration of the attempt. The black line with diamond symbols represents the mean time for all user-data within three standard deviations of the average. The mean curve demonstrates a consistent downtrending of the duration of subject response, as well as a narrowing of the duration

[2]As the first subject, user_00 only conducted 15 challenge/response attempts. The experiment was subsequently modified and the remaining subjects were each asked to perform 20 attempts.

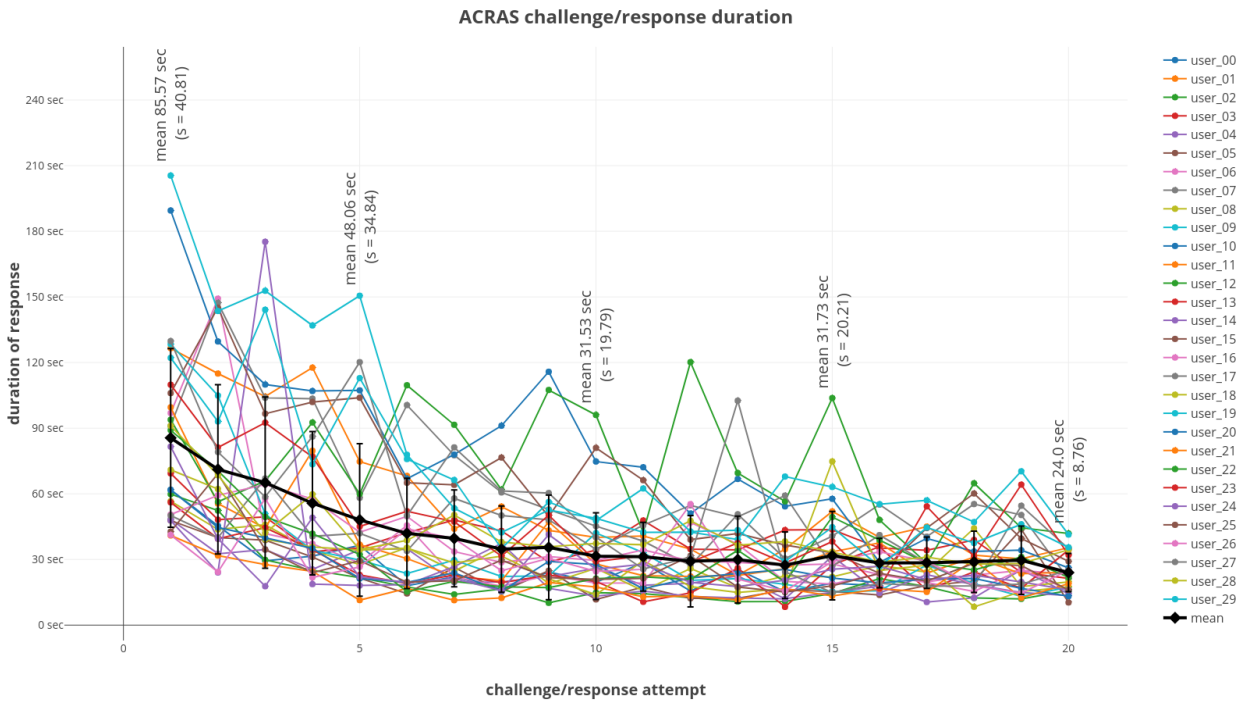

Figure 5.21: ACRAS User-Experiment subject demographics

distribution. The steady decrease in duration from a mean of approximately 85 seconds for the first attempt to a mean of 24 seconds for attempt 20 illustrates a steadily increasing level of user-comfort with ACRAS as they gain in familiarity with the system. Anomalous peaks within the graph may be attributable to any number of things, including external stimulus, particular care and/or subject referencing the registration UI for a reminder following an incorrect attempt.

Anecdotal evidence suggests that there is a certain degree of mental training an ACRAS user undergoes while learning the system. With the obvious advantage of system familiarity afforded by designing and implementing the system, we have observed a considerable improvement with our own performance on the system during the course of user-experiments. It was observed that while subjects frequently needed to refer back to the registration UI (or in several cases their own notes) for reminders on the algorithm definition, particularly at the

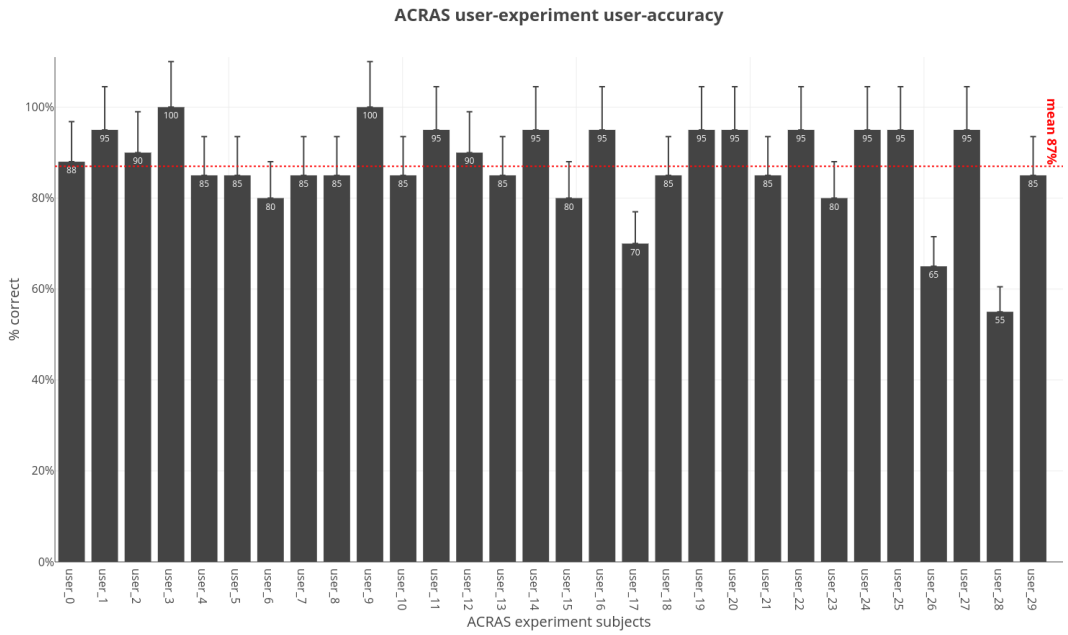

Figure 5.22: ACRAS User-Experiment subject demographics

beginning of the 20 attempts, as the experiment persisted the researcher needed less and less time to memorize the algorithm. An informal data-set from 19 subjects was (inconsistently) collected starting with user_07 where the researcher tracked at which point in the sequence a subject had memorized the algorithm, or at least felt comfortable enough to go-it-alone without reference. This point ranged from attempt number three all the way to the last attempt, but averaged 7.11. By the end of the series of experiments, we found with our own system familiarity that we were able to consistently and accurately recall the algorithm by the second or third attempt with a considerably faster mental response noted. This observation suggests that as a user's exposure to the system grows, the challenge/response duration will continue to fall and the system will continue to become easier.

Duration of challenge/response alone suggests the degree with which a user is comfortable with the system. By itself however, it does not address the broader system usability issue the user-experiments were designed to test. User accuracy is another critical metric for judging

the usability of an authentication system. If a user is unable to consistently accurately respond to the system, it will not serve the intended purpose well. Figure 5.22 depicts the rate of accuracy for each of the subjects in the ACRAS user-experiment. The accuracy rates range from 55% on the low-end of the scale to 100% accuracy on the high-end. The mean accuracy for all ACRAS user-experiment subjects across 20 attempts is 87%.

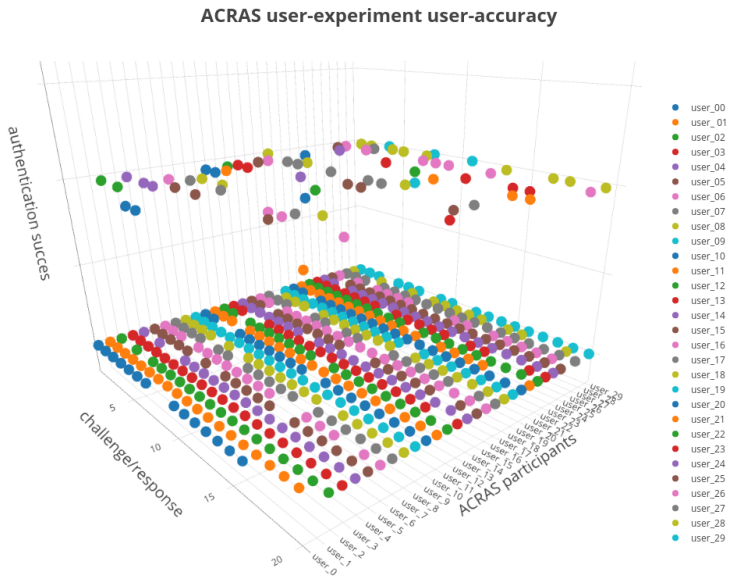

Figure 5.23: ACRAS User-Experiment subject demographics

Figure 5.23 also depicts subject accuracy for the 20 attempts, in this case the distribution of correct and incorrect responses is also displayed. In this graph, a matrix depicts each subject's responses with color-coded spheres. Accurate responses are represented on the lower plane and incorrect responses on the upper plane. By reviewing the clustering of incorrect responses, one can see some trends which correspond to informal observations made during across the greater experiment. Incorrect responses towards the beginning generally corresponding with a user initially learning their own algorithm definition, typically trying to

respond without reference material. A second band of incorrect responses is again observed slightly past the midpoint of the set, this group may be attributed to a common increase in user comfort with the system and a corresponding slip in attention which frequently resulted in a simple missed count or identification of a required challenge. Rarely by this point did a subject incorrectly recall their algorithm definition, or the selection-bases. More often, a rhythm of response had been achieved and as the response becomes more reactive than investigative, a simple miss occurred, typically resolved by paying a little more attention to the challenge-set on subsequent attempts. Finally, a third group can be identified towards the end of the challenge/response attempts. This group can be attributed to a similar issue as the second, with subjects getting a little "sloppy" with the details. Generally by this point subjects were testing the bounds of accelerated responses, either due to an eagerness to complete the experiment, or in some cases a competitive aspect was read from the subject, with more than one closing comment from the subject along the lines of "that's kinda' fun," or "it's like a game".

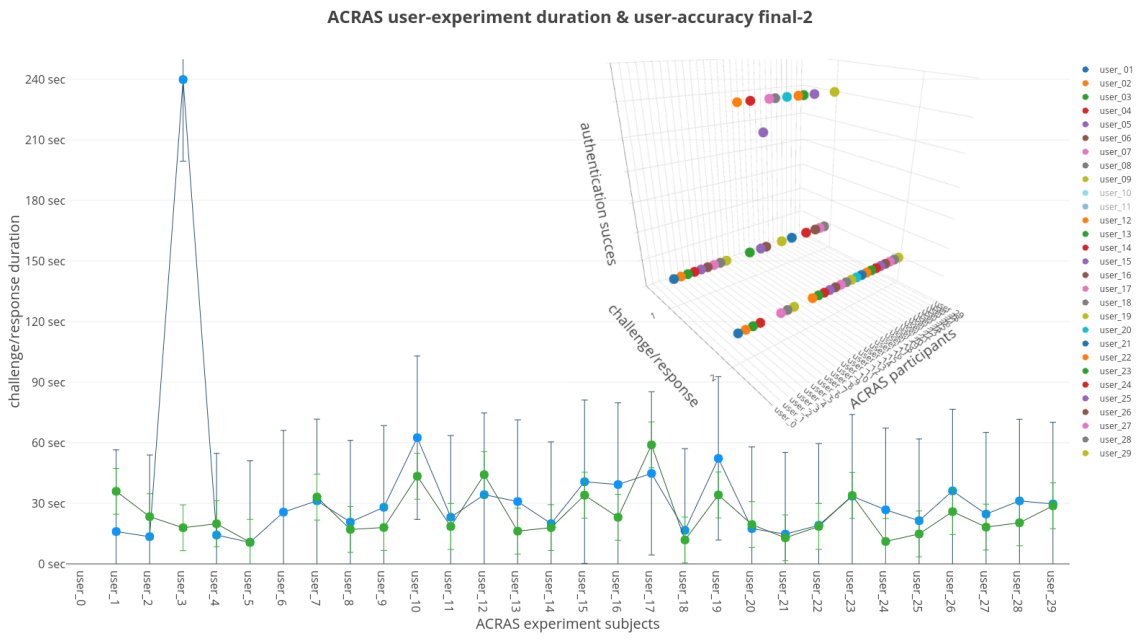

Figure 5.24: ACRAS User-Experiment Final Two Challenge/Responses

Following the 20 challenge/response attempts, each user was presented with a short user-survey. The details of the survey responses will be discussed later in this section however the survey served a second purpose as well - distraction. Unbeknownst to the subjects, a second set of challenge/response attempts would be conducted after taking a few minutes away from the UI. Users were presented with the verbal survey questions, and engaged in some light conversation, frequently expanding on their impressions of the system. After several minutes of conversation had passed, it was explained to the subject that they were expected to perform two more challenge/response attempts without the use of any reference resources as a reminder. Roughly half the time this was met with some degree of surprise or a comment that they'd have made more of an effort to remember their algorithm if they knew they were going to need it again.

Figure 5.24 details the durations and accuracy distribution for these final two challenge/response attempts. Across the bottom are the duration points for each of the users, blue represents the first attempt following the survey and break, and green is the final attempt. Aside from one outlier, durations were nearly the same as where they'd left off after 20 serial attempts. Mean duration for the two attempts (disregarding the one outlying datapoint) were 27.83 seconds and 23.09 seconds respectively. The three-dimensional scatter-plot in figure 5.24 represents the accuracy distribution for the two attempts. The initial attempt in the background, and the final attempt in the foreground, accurate responses represented on the lower plane with incorrect responses documented on the higher plane, similar to figure 5.23. There was a higher rate of incorrect responses with the initial response, generally attributable to missed details similar to late-round misses in the primary experiment set. Only one miss occurred on the final attempt, with the subject having accurately responded on the initial attempt. Accuracy rates for the two attempts are as follows: 68.97% correct for the initial response, 96.43% correct on the final

5.3.2 User-Survey

The ACRAS user survey was designed to accomplish several tasks: first, document some minor statistics related to the subject's current password habits for study context; second, to get the subject's impressions of the prototype system; and finally, as stated previously to provide a buffer between challenge/response sessions. The short verbal survey was divided into two sections according to the first two survey goals. Following are the survey questions.

Current Password Habits

1. How many unique passwords do you currently maintain?
   [1, 2-5, 6-10, 10+] (Refer to figure 5.25 for results.)

2. Do you write your passwords down, email them to yourself, use a password-manager, or use another means of documenting them? (Refer to figure 5.26 for results.)

3. How frequently do you change your passwords?
   [1x per year, 2x per year, Only When Forced, Other] (Refer to figure 5.27 for results.)

The results of survey questions related to subjects' current password habits generally follow anticipated trends with a few surprises. One such surprise was user_12's use of only a single unique password. The same was true of user_19, although this subject was in the youngest age-group of experiment participants and among the lowest perceived technical sophistication. User_12 on the other hand was one of the higher-ranked subjects for perceived technical sophistication. As expected, users of password managers tended to have higher existing password counts than their counterparts. Mildly surprising, although perhaps it should not have been, was the fact that over 50% of the subjects wrote down their passwords to ensure they didn't forget them, and an overwhelming majority of subjects only changed their password when forced, either because of policy requirements or because they could not remember what it was. While best-practices continue to evolve and there is some debate as to the value of rotating passwords if they are strong enough, none of the 27 users who chose "only when forced" alluded to any such justification.

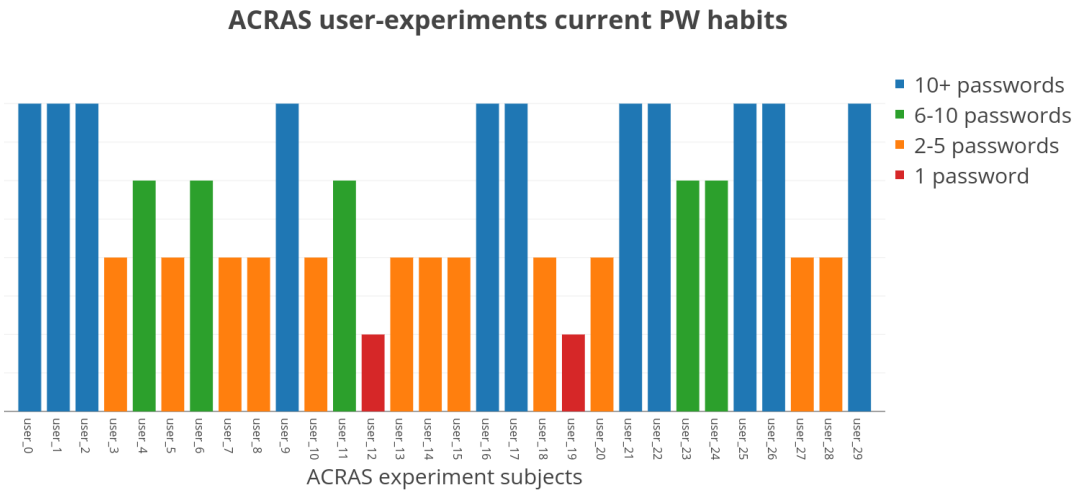

Figure 5.25: ACRAS User-Survey: How Many Unique Passwords

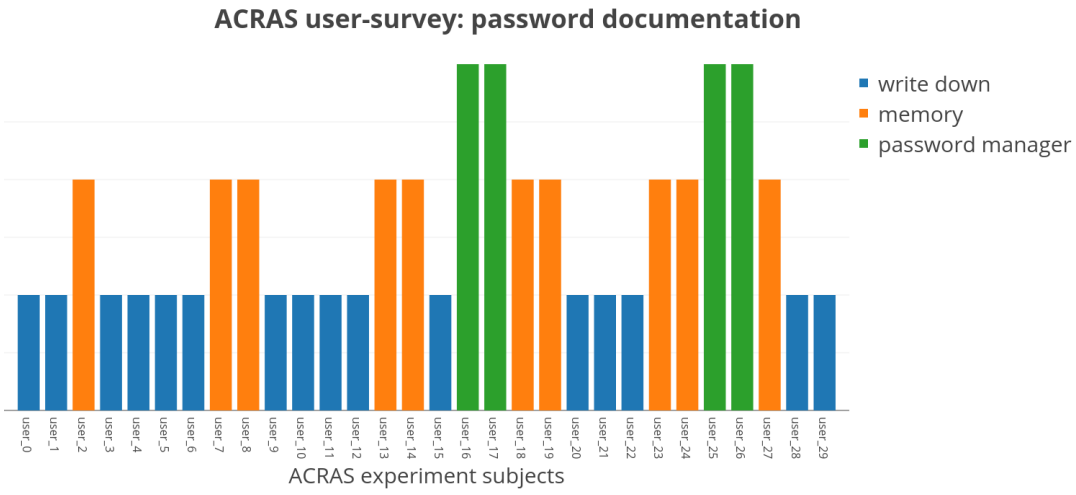

Figure 5.26: ACRAS User-Survey: Password Documentation

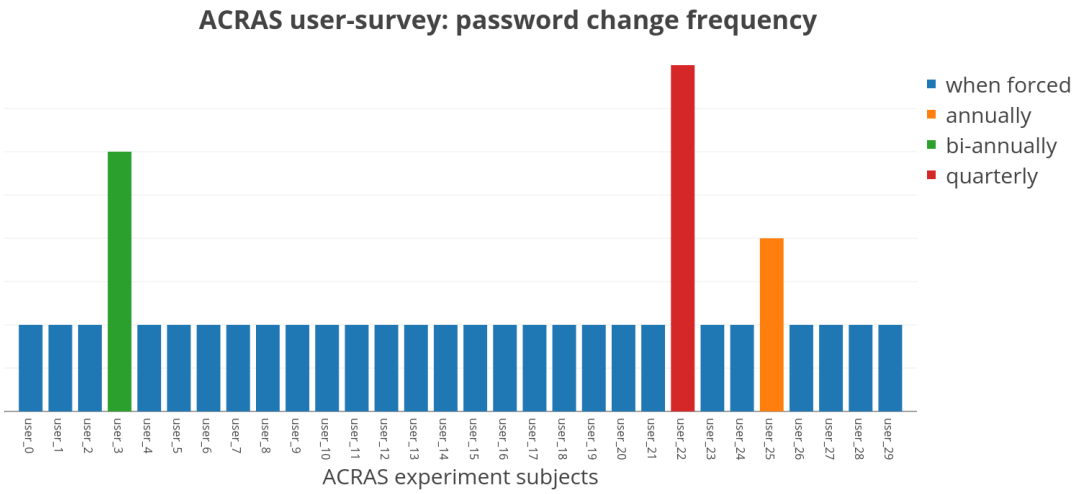

Figure 5.27: ACRAS User-Survey: Password Change Frequency

Impressions of the ACRAS System

1. Given the choice, would you choose to use the ACRAS system over a traditional text-based password authentication system? (Refer to figure 5.28 for results.)

2. On a scale of 1 - 10, with 1 being easiest and 10 being hardest, how easy or difficult was it to initially register your user-algorithm? (Refer to figure 5.29 for results.)

3. On a scale of 1 - 10, with 1 being easiest and 10 being hardest, how easy or difficult was it to recall and apply your user-algorithm against challenge-sets? (Refer to figure 5.30 for results.)

4. Based on your impressions of the ACRAS system from this experiment, does the system feel less-secure, more-secure or the same as a typical text-based password? (Refer to figure 5.31 for results.)

The results of the survey questions related to subject impressions of the ACRAS prototype were encouraging, although one consideration that was not accounted for when designing the survey was the possible effects of verbally conducting the survey as compared to that of a written response. If we are to take the survey answers for what they are, better than half of the subjects (19/30) would choose to use the ACRAS system over the traditional text-based password. Several subjects who indicated that they would not select the ACRAS system over

text passwords offered additional explanation. This explanation included the obvious speed and convenience of traditional passwords, though more than one such user indicated they may be inclined to use the ACRAS system for accounts which they deemed more sensitive, like banking and credit-cards due to the perceived improvements in security. Several of the more technically sophisticated subjects indicated they would be inclined to use the system if it were provably secure (these responses have been accounted for in the no column until such a proof is available). One user suggested they were not likely to use the system in their desktop environment, but they would be very interested in using ACRAS on a mobile platform.

Rankings for ease of registration, and ease of system use were also promising, with peak difficulty for both identified as a 7/10. Mean values for the two metrics sit at 3.5/10 for ease of registration, and 4.1/10 for ease of use. The ease of use metric proved to be trickier for subjects to pin down, and results may be skewed as there was some confusion as to whether the question meant ease of use initially, or at the end of the experiment. Inquiries to this end were responded to with a directive that the question was seeking the "overall" ease of use which is acknowledged to be a bit of a non-answer though in-line with the general approach to provide a consistent experiment for each of the subjects. This effort towards consistency was maintained throughout the experiment despite lessons learned along the way so as to not inadvertently steer results.

Finally, the question regarding the perceived security of the system as compared to traditional text-based passwords demonstrated an overwhelming opinion that the ACRAS system felt more secure than the text password. Subjects were encouraged, though not required to elaborate on their response to this final question. Responses tended to fall into one of several common categories, including: security resulting from the fact that ACRAS OTPs use common terms, or are scrubbed of personalized details; security resulting from the dynamic nature of OTPs continuously changing; lack of familiarity with the system and UI; and the gamification, or "fun-factor" associated with the ACRAS authentication process. The following figures summarize responses to the ACRAS impressions set of survey questions.

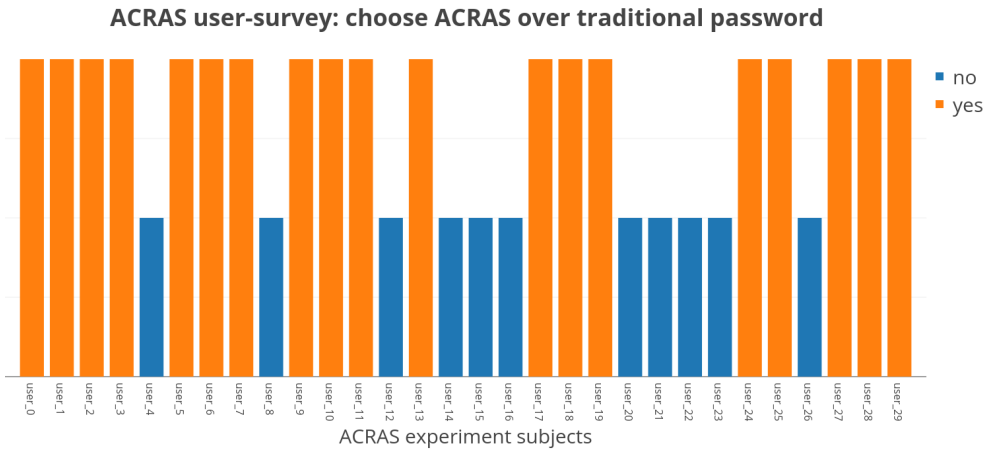

Figure 5.28: ACRAS User-Survey: Choose to Use ACRAS Over Text Password

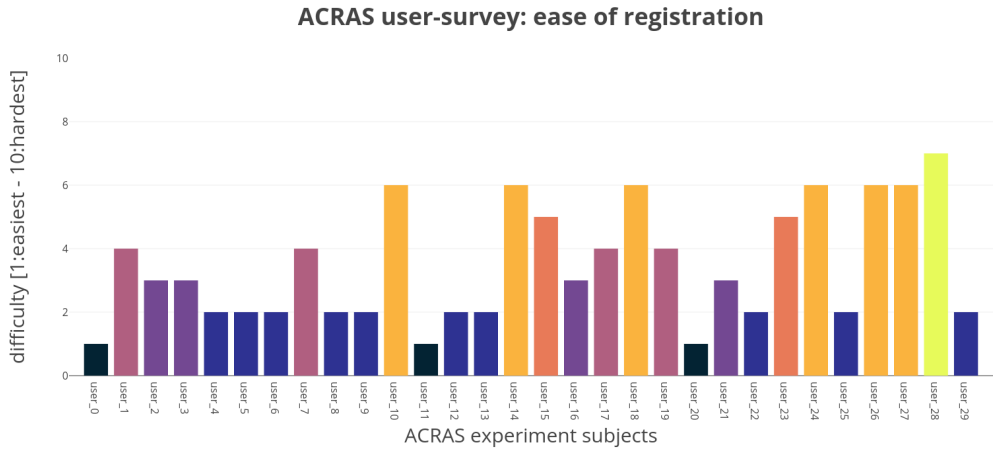

Figure 5.29: ACRAS User-Survey: Ease of Registration Process

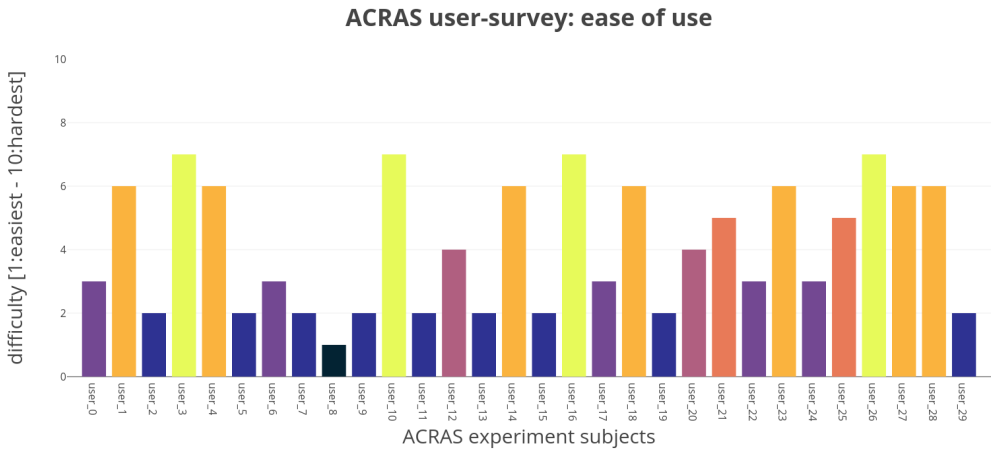

Figure 5.30: ACRAS User-Survey: Ease of System Use

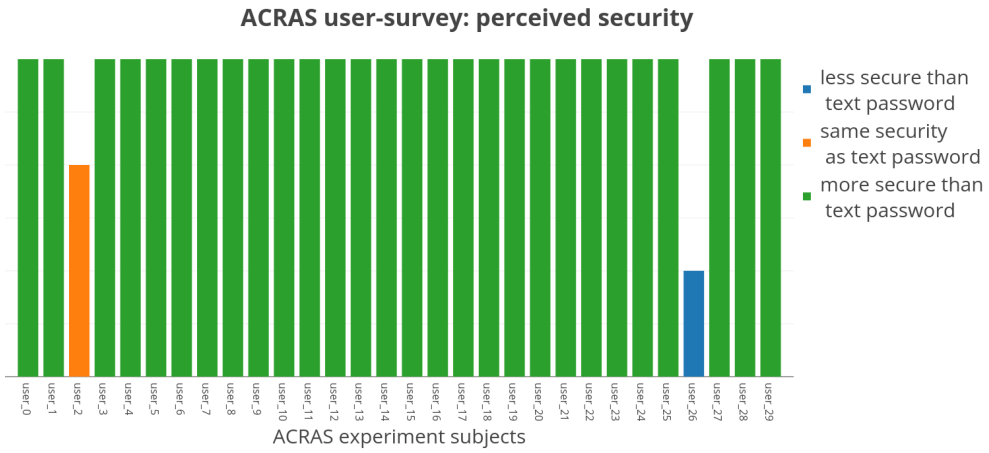

Figure 5.31: ACRAS User-Survey: Perceived System Security Compared to Text Password

Chapter 6
DISCUSSION & FUTURE WORK

The user authentication space within modern computing has significant room for improvement. Today's users are increasingly putting their faith and their assets - be they data or financial, into seemingly more and more technologically accessible platforms. While the increase in value and distribution of user investment in these platforms has grown, comparatively little development has occurred in the mechanisms that are used to protect these resources. The ACRAS system has been designed with the underlying goal of addressing this imbalance. The following section contains discussion of the successes and limitations of the ACRAS system, as well some areas for future work.

6.1 Discussion

6.1.1 ACRAS Security

In Chapter 3, Attacker Model & Benchmarks for Success, the ACRAS system was framed by its susceptibility to three classifications of attack common to user authentication systems: guessability, observability, and recordability. As is frequently the case, the ACRAS system is subject to a greater degree of vulnerability to some classes of attack, and less to others.

Guessability Attacks

Guessability attacks include brute-force efforts and dictionary, or familiarity-based attacks. As a one-time-password (OTP) system, ACRAS is inherently resistant to traditional brute-force and dictionary attacks where an adversary cycles through all variants, or a more refined preset list (dictionary) of possibilities. For example, an adversary may try password A on one attempt, upon authentication failure, the adversary may then try password B, should

that fail then try password C, and so-on. In an OTP, the actual password may not have been password A on the first attempt, although with sufficient state-space in the system there is nothing preventing the actual password from being password A on the second, or third attempts which the adversary would not be checking if they were iteratively cycling through each of the possibilities. OTPs are however subject to a reverse brute-force attack where instead of attempting many variations of password, the attacker continuously attempts the same password (potentially even replaying a legitimate OTP somehow obtained by the attacker) hoping for the conditions which make the password valid. This type of attack is simple to recognize though, and may be defeated with a simple fail-to-ban policy within system implementation. Such a policy with the ACRAS system might freeze an account after 5 tries with the same password. What all of this means is that in the ACRAS context, guessability attacks come down to just that - a guess.

State-space analysis efforts have been conducted as a part of the ACRAS research for both the broader system-wide limit of possible OTP responses, as well as with each of the user-experiment subject's own algorithm definitions to determine the extents of the set an attacker would need to guess from. It is worth noting that these analyses have been conducted in the context of the current ACRAS prototype. Additional extension or retraction of any of the algorithm options would have a corresponding positive or negative effect on the total possibilities. The overall system state-space for the current ACRAS prototype affords in excess of 5.5 quintillion (5.5E+18) unique possibilities. Practically, this means that barring any specifics of algorithm definition, or knowing the precise details of the challenge-set an attacker would have a 1:5,500,000,000,000,000,000 chance at guessing the OTP for a session. Knowing an ACRAS system user's algorithm definition narrows the field considerably, although if an adversary knew the definition they would likely not be guessing the password. The range of user-experiment subjects' unique possibilities varied from 54 on the low-end to in excess of 1 million on the high-end with a mean value for the set of experiment subjects of 130,825 unique possibilities.

Observability Attacks

Similar to the state-space analysis of user algorithms, significant work was put into the analysis of the ACRAS system in the context of observability attacks. These types of attacks primarily include shoulder-surfing and spyware attacks capable of capturing both OTP and challenge-set data for intersection analysis. In contrast to the algorithm-oriented state-space analyses, analysis for observation-based attacks comes from the challenge-set side of things.

Initial observability analysis included efforts to write a script to analyze challenge-sets for correlations. This proved to be a non-trivial task however. Parsing a challenge set with complete data[1] is in and of itself not a particularly difficult task. The subsequent correlation of the numerous data-points which can potentially interact with each other according to specific rules proved to be a significant task which did not fit within the constraints of the ACRAS thesis research. Instead, a manual analysis of a single randomly-selected subject's OTPs and corresponding challenge-sets was conducted as a minimally representative indication of the system's resistance to shoulder-surfing.

In Chapter 3, it was reasoned that for various reasons a typical user might be required to authenticate into a system up to four times in a single-session. This number was in-turn used as a metric for successful resistance to shoulder-surfing attacks. The manual analysis of user_19's complete OTP and challenge-set data resulted a series of intersection-based reductions of plausible alternatives for generating the OTPs from their corresponding challenge-sets. In the end, user_19's algorithm survived three rounds of analysis with the fourth round of analysis resulting in a sufficient narrowing of the possibilities to a small enough set with only limited variance that the algorithm could be reasonably applied to future challenge-sets.

Several valuable observations were made during the course of this algorithm reversing ef-

[1]Complete data is simply available from the ACRAS backend. Front-end only access to complete data is complicated by the image-based presentation of challenge-graphics. An adversary would need to create a tool incorporating Optical Character Recognition technology to parse through a challenge-set and correctly characterize the many dimensions of each challenge graphic.

fort. First, numerical outputs including *count*, *echo:value(#)*, and *math:add* operations have significant generative overlap, and are relatively difficult to isolate as compared to other character-oriented outputs. The late introduction of zero as a possible numerical output for *count*, and *math* operations was a particularly effective mechanism for decoupling challenges from the resulting output. In addition, ganging numerical output together rather than spreading it across the algorithm stages creates some additional uncertainty with respect to stage delineation. Second, borrowing from conventional wisdom related to traditional passwords, and the abstract combinatorial effect of a large set of potential challenges, subjects were encouraged to include an *echo:English* or *echo:Color* into their algorithm definition. As it turns out however, with complete data, an *echo:English* of a *word* easily identifies the relevant member and quickly focuses the analysis on the limited subset of characteristics available to that *word* challenge. Finally, an algorithm does not need to be reversed to absolute precision in order to be defeated. Sufficiently narrowing the possibilities to a manageable subset allows an attacker to interpret the approximate algorithm in the context of the current challenge-set and not only recognize the intended relevant set but potentially even fine-tune the selection-bases as a result of singular near-miss instances.

Spyware related observability attacks have been discussed earlier but as the potential variability of such attacks is so broad, and the potential impact of inadvertently running malware on one's computer is so great, these attacks have been considered out of scope for the ACRAS research. In the end, as demonstrated with only four rounds of manual analysis in the shoulder-surfing resistance study, intersection attacks with serial examples of complete data will eventually result in a complete and accurate algorithm definition. It is assumed that spyware attacks are capable of recording and exfiltrating complete data for numerous examples. As such, the ACRAS system does not provide an effective resistance to spyware.

Related to both shoulder-surfing and spyware observability attacks, it is important to recognize the broader context within which these attacks on the ACRAS system lie. Despite the successful reversing of the algorithm via manual analysis and intersection in four rounds, the ACRAS system remains comparatively resistant to shoulder-surfing when compared with

the ubiquitous static text-based credentials that are vulnerable to only a single exposure to either shoulder-surfing, or spyware. Additionally, the shoulder-surfing intersection analysis was conducted with complete data. Incomplete, or incorrect data within analyzed data-sets would greatly affect the efficiency and accuracy of similar analysis.

Recordability Attacks

The applicability of recordability attacks within the context of the ACRAS research has been previously addressed in Chapter 3, Attacker Model & Benchmarks for Success. These types of attacks are related to the theft of user documentation of their algorithm definition, and phishing attacks where a malicious site attempts to trick users into thinking it is legitimate and entering their credentials. The ACRAS system does not provide any direct means for addressing these types of attacks as they are largely outside of the scope of the authentication system. ACRAS does however provide some indirect resistance to recordability attacks.

In regards to theft of user documentation, there is no way the ACRAS system can keep a user from writing down their algorithm definition. The ACRAS system does however indirectly affect this type of attack through the underlying graphical recognition and cued-recall techniques associated with graphic and pattern memory. By relying on these memory mechanisms, the ACRAS system is arguably more memorable than a static textual password, potentially indirectly reducing or altogether eliminating the need to write one's password down. As for phishing resistance, the ACRAS system is fundamentally resistant to spoofing without prior knowledge of a user's algorithm and the corresponding challenge-set requirements. A phishing site attempting to mimic a legitimate ACRAS implementation would still need to present a user with a viable challenge-set based on their algorithmic requirements. A phony ACRAS challenge-set would be immediately recognizable (and avoidable) as a randomly generated challenge-set would likely not fill all of the algorithmic requirements a user needed to derive their OTP.

6.1.2 ACRAS Usability

As has been discussed in Chapter 5, Results, security alone does not make an effective authentication system without a corresponding degree of usability. Usability metrics were analyzed for the ACRAS system by way of the ACRAS prototype user-experiments. A collection of 30 subjects were put through a registration, and subsequent series of 20 challenge/response attempts with the ACRAS prototype. The subject pool varied in age-group, gender, and perceived technical sophistication. Due to the time-commitment for the experiment, all sessions were scheduled ahead of time and all subjects are personally known by the researcher. Following the bulk of the user-experiment, a short verbal survey was conducted followed by two unexpected rounds of challenge/response to see how well subjects were able to recall and perform the authentication process after a short period of distraction.

Registration

Timed durations for both the registration and authentication phases suggest that, as expected, the ACRAS system as a whole takes more time than that of authentication via traditional username and password. This very fact was raised several times by experiment subjects, although typically with an acknowledgement that there is a fundamental trade-off occurring with the additional time it takes to complete the more complex ACRAS process delivering a perceived increase in security. The mean time spent by experiment subjects on algorithm registration was 5.37 minutes, with the experience ranging from just under two minutes for some users to over 11 for one user who spent more time exploring the available options than most. Interestingly, after concluding the experiment, nearly half of the subjects expressed the sentiment that if they were to repeat the process for a real account that they would define their algorithm quite differently. We attribute this to the familiarity of the system and available algorithm options gained from interpreting an algorithm within the context of numerous challenge-sets.

Challenge/Response

The durations of challenge/response attempts demonstrate a consistent downward trend, beginning with a mean duration of over 85 seconds for the initial attempt and steadily decreasing to a mean of 24 seconds for the twentieth. The duration trend suggests that a learning curve is experienced by ACRAS subjects but that with increased exposure to the system the overall ease of use increases. In the end, there is a floor to the duration of challenge/responses simply associated with the time it takes to review the complete set of challenge graphics, interpret and process them and finally enter the OTP into the input field. The same duration for security trade-off described previously applies to the ACRAS authentication phase. Subject accuracy during the experiment suggests that the system is appropriately usable, without being too easy. A mean accuracy of 87% was achieved by all users for the initial set of 20 attempts.

Following the user-survey, subjects were surprised with a second set of challenge/response attempts after a short duration away from the system and some intentional distraction. This second set of authentication attempts was designed to gauge short-term ability to recall the algorithm definition and how to apply it. The intent for this portion of the study was not to determine the efficacy of long-term recall, rather to test how well the ACRAS system remained memorable beyond the serial repetition of the first 20 attempts while still remaining within a reasonable time-frame for a single session for the experiment. Durations for the final two attempts remain in-line with those towards the end of the initial set supporting the consistent downward trend. Accuracy took a hit in the in first attempt following the break, falling to 69% and generally corresponding with a slip in user attention. The group finished strong however with a 96% accuracy rate on the second and final attempt.

User Survey

The responses to the short verbal survey are discussed in more detail in the preceding Chapter, but several salient points are worth noting in the larger context. First, the current pass-

word habits represented by the subject pool largely exhibit poor security practices. These practices include considerable password re-use and documentation of passwords. Second, nearly two-thirds of subjects indicated they would choose to use the ACRAS system over a traditional text password. Those who said they'd prefer to stay with text-based passwords gave reasons including the speed and convenience of traditional systems, and the as-of-yet proven security of the ACRAS system, although 90% of subjects indicated that the ACRAS system felt more secure to them than text-passwords. Finally, the mean values for perceived ease of registration and ease of use on a scale of one to ten was a promising 3.5 and 4.1 respectively. This, coupled with the increase in performance as subject exposure and familiarity built suggests that the system meets the broadly scoped usability requirement.

6.2 Future Work

One of the benefits realized via working with a system prototype and perhaps amplified with the addition of user-experiments is the continuous realization that the system is not complete. While this is the very nature of a prototype, this realization is frequently accompanied by the temptation to start modifications and feature development as the opportunity is discovered. As expected, during the course of ACRAS user-experiments numerous opportunities for system improvement were recognized, frequently discussed with and even suggested by user-experiment subjects themselves. While the temptation to get straight to work on many of these was strong, restraint was the ultimate victor, relegating these opportunities to the "to-do" list. Following are several noteworthy opportunities for future work within the ACRAS system, organized according to system level.

6.2.1 ACRAS Future Work

There are several areas for future work within the larger ACRAS system as a whole. One such area is porting the ACRAS system to a mobile platform. As mobile systems rely on a different set of input mechanisms, namely touch-screens vs. keyboard and mouse, the graphic to text decoupling warrants investigation within the mobile context.

Another area of future work within the broader system is the secure storage of user algorithm definitions. Best practices for traditional text passwords include maintaining a hash of the password on the backend for comparison so that user passwords do not need to persist in the clear. Similar considerations should be made for the ACRAS system to ensure that user algorithm definitions are not compromised by backend breach.

As the ACRAS system provides a considerable variety of algorithm definition options, it is entirely feasible to create a weak algorithm, in the same way one can create a weak traditional password. An opportunity exists within the ACRAS system to have the algorithm registration phase begin with a randomly selected and validated strong initialization algorithm that the user may modify or replace altogether. The creation of a strong initialization algorithm would help to ensure that ACRAS users are presented with a minimally strong foundation for creation of their own algorithm, in the same way that password managers afford users the chance to create random strings for text-based passwords.

6.2.2 Interface Future Work

During the course of user-experiments, several subjects made reference to the different colors chosen for the stage registration UI. These colors, red, yellow, and green were selected to suggest a relative strength of algorithm associated with use of increasing numbers of stage. Subjects were associating the color of the stage registration interface with their stage definition, for instance looking for *red* challenges for the first stage (colored red in the UI), despite the fact that *red* had nothing to do with their stage definition. The colors of the UI should be eliminated or reworked so that they do not cause confusion with stage definitions.

6.2.3 Algorithm Future Work

Several opportunities exist for future work at the ACRAS algorithm level. The first of which is incorporating additional relativity into algorithm definitions. Examples could include value-based or positional relativity. Value based relativity would extend the existing comparison against a user-defined set-point to include a lowest value or highest value into

selection-bases. Positional relativity would include selection-bases based on the location of a challenge within the UI. Examples include challenge position nearest each of the four corners, or even challenges pointed to by a *shape:arrow* challenge.

Compound algorithm definitions are another space for future work in the ACRAS system. Consider a stage definition that was variable and dependent upon the random value of another challenge within the set. For instance, if a *letter red:filled vowel* is uppercase, respond to an *echo:English* action in uppercase, otherwise default to lowercase. Compound definitions would drastically increase challenge decoupling from response.

Another area for ACRAS algorithm development has to do with output of numerical values. *echo* actions have the option to output a *value(#)*, or an *English* which changes the representation of *digits*. A similar option could be extended to *count* and *math* actions in order to again increase challenge/response decoupling.

Finally, algorithm transformations should be extended for digits to provide additional options. Possible digit transformations include add, and round transformations which again, would provide an opportunity to increase the decoupling effect.

6.2.4 Challenge Future Work

Areas for future work related to the design of ACRAS challenges include expansion of the *shape:primitive* set, expansion of the *word:category* set, and integration of languages in addition to English. Expansion of these sets will necessarily involve the review of proposed vs. existing members in order to maintain the critical clarity principle. Similar review of all members within these sets will need to accompany any additional language integration as different languages will undoubtedly contain a different collection of collisions and potential uncertainties.

While efforts were made to assure proper interpretation of letters and digits, including the relative sizing of uppercase and lowercase letters, the same success was not in the end, realized with the varying casing of some of the word challenges. User experiments revealed at least one instance where the distinction between the lowercase and uppercase word represen-

tations was not apparent. Graphic representation of letters and digits maintains a consistent scale which aids users in making the distinguishing between different casings of some letters. Due to the limited scale of challenge graphic slots, and the variable length of words combined with a desire to maximize the scale of the graphic representation in order to promote legibility and the correct interpretation of challenge characteristics such as *color* and *fill*, word challenge graphics were constructed with variable scale to utilize the available horizontal dimension of the challenge graphic space. This variable sizing paired with the font-selection had the unforeseen effect of making the Word cow nearly indistinguishable from its upper-case counterpart COW. Potential solutions to the confused casing include a consistent scale to all Words, or the inclusion of background datum guidelines similar to those provided to students learning to print. The latter potentially providing a further means of distinction between digits (which would not have the guidelines) and letters.

Figure 6.1: COW and cow Challenges are Difficult to Distinguish

Chapter 7

CONCLUSION

Traditional username and password authentication systems are an outdated solution which is failing to adequately serve contemporary computing needs. Regular advances in computational power, coupled with poor password-management habits, generally resulting from large numbers of parallel user-accounts, are promoting an increasingly insecure authentication environment for end-users. In an effort to address this imbalance, this thesis has explored the viability of a moving-target defense approach; specifically investigating one-time passwords (OTPs) based on a challenge/response mechanism utilizing graphical challenge sets and a user-defined algorithmic procedure to produce a textual response. Graphical challenges have been employed to leverage their inherent dimensional multiplicity. This presents an opportunity for the applicable details within a set of challenges to be masked to potential unauthorized observers by the simultaneous presence of numerous other possibly relevant details. A legitimate user of the system would possess the knowledge to easily filter out the irrelevant from the relevant challenge data and use the applicable details to construct an OTP. We have applied intentionally deceptive techniques including dazzling, mimicking and decoying to produce plausible alternatives and misdirection within challenge-set presentation to further complicate efforts to subvert the system; specifically aiming to address a perfect shoulder-surfing attack and increase the iterative requirements to mount a successful intersection analysis.

In addition to an objective, abstract statistical analysis of the strength of the system, a prototype algorithmic OTP system has been implemented using readily available open-source software to conduct user-experiments to gauge the usability of the approach. As an OTP, the system is inherently resistant to brute-force as each subsequent response-attempt would

be measured against a new challenge eliciting a new successful response. The analysis of an iterative series of 20 challenge sets from 30 experiment subjects resulted in an average of over 130,000 possible responses. A manual analysis involving complete challenge/response data required four rounds of analysis to successfully correlate the challenge data with responses and reverse-engineer the user-algorithm. Durations were captured from the series of user-experiments resulting in a mean time of 5.37 minutes for user-registration and a range of durations for challenge/response negotiation beginning with a mean time of 85 seconds and concluding at less than 24 seconds. Average durations indicate the algorithmic OTP system takes considerably more time than traditional entry of text-based username/password; however, a survey of experiment participants suggests the increased feeling of security is worth the increased time to perform an authentication.

During the course of the ACRAS research project some of the lessons learned include the need to test iteratively and narrowly bound functionality extents within prototype systems. As a prototype, it is important to illustrate effect and extrapolate the impact of additional breadth, while efficiently utilizing implementation resources. Results to both the objective analysis and subjective participant survey suggest the algorithmic OTP authentication system does provide for a viable alternative to traditional text-based username/password user-authentication systems. The algorithmic OTP system does however take more time to complete an authorization than these traditional systems as the process of analyzing a challenge set and applying a user's algorithm involve more steps than a simple static password entry. The ACRAS system affords a means of generating OTPs without reliance upon a second-factor device or service, which while effective, provide an arguably over-restrictive condition for authentication.

BIBLIOGRAPHY

[1] M. R. Albayati and A. H. Lashkari. A New Graphical Password Based on Decoy Image Portions (GP-DIP). In *2014 International Conference on Mathematics and Computers in Sciences and in Industry*, pages 295–298, September 2014.

[2] Mohammed H. Almeshekah and Eugene H. Spafford. Planning and Integrating Deception into Computer Security Defenses. In *NSPW '14 Proceedings of the 2014 New Security Paradigms Workshop*, pages 127–138. ACM Press, 2014.

[3] H. Alsaiari, M. Papadaki, P. Dowland, and S. Furnell. Secure Graphical One Time Password (GOTPass): An Empirical Study. *Information Security Journal: A Global Perspective*, 24(4-6):207–220, 2015.

[4] Goh Wen Bin, Sohail Safdar, Rehan Akbar, and Suresh Subramanian. Graphical authentication based on anti-shoulder surfing mechanism. In *Proceedings of the 2Nd International Conference on Future Networks and Distributed Systems*, ICFNDS '18, pages 20:1–20:6, New York, NY, USA, 2018. ACM.

[5] Raluca Budiu. Memory Recognition and Recall in User Interfaces.

[6] CAPTCHA. The Official CAPTCHA Site.

[7] David S. C. Chen, Richard Tung, Boyi Tzen, and Der-Joung Wang. Method and computer system for dynamically providing multi-dimensional based password/challenge authentication, January 2017.

[8] Y. L. Chen, W. C. Ku, Y. C. Yeh, and D. M. Liao. A simple text-based shoulder surfing resistant graphical password scheme. In *2013 International Symposium on Next-Generation Electronics*, pages 161–164, February 2013.

[9] E. Darbanian and G. Dastghaiby fard. A graphical password against spyware and shoulder-surfing attacks. In *2015 International Symposium on Computer Science and Software Engineering (CSSE)*, pages 1–6, August 2015.

[10] Stina Ehrensvard. Launching The 4th Generation YubiKey, November 2015.

[11] R. English and R. Poet. The Effectiveness of Intersection Attack Countermeasures for Graphical Passwords. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1–8, June 2012.

[12] W. Tecumseh Fitch, Angela D. Friederici, and Peter Hagoort. Introduction: Pattern perception and computational complexity: introduction to the special issue. *Philosophical Transactions: Biological Sciences*, 367(1598):1925–1932, 2012.

[13] H. Gao, Z. Ren, X. Chang, X. Liu, and U. Aickelin. A New Graphical Password Scheme Resistant to Shoulder-Surfing. In *2010 International Conference on Cyberworlds*, pages 194–199, October 2010.

[14] Jacob Hoffman-Andrews and Gennie Gebhart. A Guide to Common Types of Two-Factor Authentication on the Web, September 2017.

[15] Google. google-authenticator: Open source version of Google Authenticator (except the Android app), March 2018. original-date: 2014-10-08T17:54:27Z.

[16] Wazir Zada Khan, Yang Xiang, Mohammed Y. Aalsalem, and Quratulain Arshad. A Hybrid Graphical Password Based System. In *Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part II*, ICA3PP'11, pages 153–164, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] Shushuang Man, Dawei Hong, and Manton Matthews. A shoulder-surfing resistant graphical password scheme - wiw. In *Proc. International Conference on Security and Management*, volume 3, pages 105–111, 01 2003.

[18] S Murugavalli, S.A.K. Jainulabudeen, G Senthil Kumar, and D Anuradha. Enhancing security against hard AI problems in user authentication using CAPTCHA as graphical passwords. *International Journal of Advanced Computer Research*, 6:93–99, 2016.

[19] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. Finding Focus in the Blur of Moving-Target Techniques. *IEEE Security Privacy*, 12(2):16–26, March 2014.

[20] Allan Paivio, T. B. Rogers, and Padric C. Smythe. Why are pictures easier to recall than words? *Psychonomic Science*, 11(4):137–138, Apr 1968.

[21] K. Renaud, P. Mayer, M. Volkamer, and J. Maguire. Are graphical authentication mechanisms as strong as passwords? In *2013 Federated Conference on Computer Science and Information Systems*, pages 837–844, September 2013.

[22] Lawless Research. TeleSign Consumer Account Security Report 2016. Technical report, Fort Collins, CO, 2016.

[23] S. S. Shen, T. H. Kang, S. H. Lin, and W. Chien. Random graphic user password authentication scheme in mobile devices. In *2017 International Conference on Applied System Innovation (ICASI)*, pages 1251–1254, May 2017.

[24] Kristian Skrai, Predrag Pale, and Zvonko Kostanjar. Authentication approach using one-time challenge generation based on user behavior patterns captured in transactional data sets. *Computers & Security*, 67:107 – 121, 2017.

[25] Elizabeth Stobert and Robert Biddle. The password life cycle: User behaviour in managing passwords. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 243–255, Menlo Park, CA, 2014. USENIX Association.

[26] H. M. Sun, S. T. Chen, J. H. Yeh, and C. Y. Cheng. A Shoulder Surfing Resistant Graphical Authentication System. *IEEE Transactions on Dependable and Secure Computing*, 15(2):180–193, March 2018.

[27] Huiping Sun, Ke Wang, Xu Li, Nan Qin, and Zhong Chen. PassApp: My App is My Password! In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 306–315, New York, NY, USA, 2015. ACM.

[28] Toomaj Zangooei, Masood Mansoori, and Ian Welch. A hybrid recognition and recall based approach in graphical passwords. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 665–673, New York, NY, USA, 2012. ACM.