A Meta-Scheme for Authentication Using Text Adventures

By

Carson Dylan Brown

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario, Canada

December 2010

Abstract

This thesis proposes that text adventures be studied as the basis for alternative text-based authentication. This includes creating a text-based virtual environment where user authentication involves the user entering commands that include interacting with objects and navigating between rooms as their "password." Three exemplar proposals are implemented and analyzed. The first two examples provide theoretical password spaces comparable to PIN or text passwords, and are designed against given threat models. The third example is designed to offer resistance to keyloggers due to a randomly varying user password based upon a selected authentication portfolio. By leveraging the ability for the user to interact with a text-based virtual world, we show that it is possible to design a scheme that maintains a minimum level of security comparable to text passwords with this novel approach.

Acknowledgements

Firstly, I wish acknowledge the outstanding support from my supervisors Anil Somayaji and Paul C. van Oorschot. Anil's guidance in framing and at looking at the big picture and Paul's detailed critiques, insight and guidance have been invaluable over the course of completing this thesis. I am deeply grateful for their assistance and advice. I thank Carlisle Adams and Robert Biddle as thesis defence committee members for their detailed critique and discussion during the defence. As well, I thank David Mould for chairing the defence.

I sincerely thank Güneş Kayacık for his priceless advice and comments, from early rough draft reviews to the final copy. I also wish to thank my fellow members of the Carleton Computer Security Lab who gave helpful feedback, but especially Mansour Alsaleh, David Barrera, Saran Neti and Terri Oda for help with fine-tuning.

I wish to dedicate this thesis to my late grandfather, Louis J. Thomas. His unique outlook on life; incredible attention to detail; respect and devotion to academic study; and astounding ability to capture a listener's attention with his magnificent storytelling are sorely missed but earnestly remembered.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Text passwords are an extremely popular authentication method, even though they have significant usability issues and are subject to a variety of attacks including replay and dictionary attacks. Current authentication schemes are confined to a process of asking very specific questions, and the user or her authentication tool answers in an equally specific manner. The answers can be computations performed by auxiliary devices (e.g., one-time password generators) or committed to the memory of the user or her memory aid (e.g., password manager). This thesis explores an option for authentication that makes use of text-based input different from passwords, borrowing concepts and tools originally used in works of interactive fiction in order to provide a novel authentication system. The answers given by a user in this case are interpreted, yielding for a system that accepts varying input as equivalent input. Exemplar prototypes are created in order to showcase the attainable security properties, including theoretical password spaces comparable to text passwords and resistance to replay attacks.

1.1 Motivation

Humans have historically communicated with one another through story (parable). Story as a form of communication contains redundancy allowing error correction, structure and repetition, all features that aid in memorability. In this thesis, we argue that the concepts found in normal human communication and storytelling can be adapted for use in authentication. A primary application for text is storytelling, but in regards to authentication, it has only been adapted primarily for use in passwords and passphrases, both of which must generally be precisely correct. We cannot build an authentication system that understands stories, because to fully understand any possible story is a Turing complete problem. Therefore, if we wish to use text in a storytelling-like way for authentication, we must explore a more restricted text-based medium.

Interactive fiction is a genre of computer programs where the user enters text to interact and change a story as it progresses. The programs are not completely open-ended. Authors provide a literary narrative of possible directions the story can go. Text adventures, a type of interactive fiction, are computer games that present a player with text output describing a virtual environment, using a natural language parser for input to interact with the virtual world. Each game tells an elaborate story where the user acts as the main character, playing the game by solving puzzles and exploring. The user interface behaves in a conversation-like manner, where responses and descriptions of the changes to the environment are given based on short commands the user enters. Tools that create these games are highly customizable and require none of the overhead of creating the graphical environments such as those common in modern video games. Furthermore, today's text adventure tools allow for these games to be created in a language similar to English. A user familiar with text adventure

games is able to create her own games with modest amounts of training.

Our work proposes text adventures as an authentication scheme in which a *human*-computable operation be performed: an operation easy for the registered user to do, but difficult for an attacker to capture, replay and understand. Using concepts from text adventures for authentication is a non-trivial idea, as there is a gap between a story-line driven narrative with a small number of possible plots and the necessary large password spaces needed for authentication. Though not originally intended as such, traditional text password authentication systems are a strict and degenerate form of dialogue between entities. That is, the same question (asking the user to enter her secret) and answer (the user entering her password) are used each time. Thus, our new approach can be considered a superset of password-based authentication, as a password system can be implemented within a text adventure framework.

The approach also benefits from the memory advantages of cued recall: rather than having to enter all information in a single step, small chunks of information are requested in steps over the greater authentication event (incrementally in response to context-sensitive prompts). There exists the possibility to accept equivalent input entered in a different manner: allowing users to enter commands in various ways (by syntactically different, but semantically equivalent text input) can aid in memorability—users only have to remember a concept, rather than the exact wording for a concept.

In order to consider a system adapting text adventures for authentication, we must first consider three main challenges:

Entropy: The system requires sufficiently high entropy password spaces to deter brute-force attacks over the space of all authentication attempts, and the minimization of weak password subspaces [45] to preclude statistically-based guessing or dic-

tionary attacks. The sophistication of language, storytelling and discussion in simple human dialogue can provide the basis for these goals.

Structure: The system needs to be presented in a way that is understood by both parties, i.e., rules are concise and memorable for users, and straight-forward in implementation on the server. Despite the artistry sometimes required for an engaging discussion, much can be represented with symbolic logic and natural language systems.

Usability: Related to the above requirement, the system must also be a task that is within the grasp of users of the system. Users ought to be able to accurately authenticate themselves with relative ease.

Before a scheme can be considered for usability testing, we feel that first, it must be determined if it can be implemented at all; second, if it can plausibly meet security requirements; and then third, if it can simultaneously meet security and usability requirements. This thesis explores text adventures as an authentication system that can perform similarly to existing text password-based systems. Text adventures were originally intended as games, and sold well during their heyday at the advent of personal computers. There still exist communities who create and play text adventures. From these facts, we conjecture that authentication systems based on text adventure concepts can be constructed to be usable by some portion of the population. Testing this conjecture is beyond the scope of this thesis.

Our motivation is to leverage the advanced skills humans have in conversation and discussion, rather than rely on our relatively poor ability to memorize precise patterns of text. This work is concerned with introducing the concept of authentication as dialogue, and we focus the majority of our efforts on overcoming the first two chal-

lenges. We have designed and implemented three example systems for exploration in this space of authentication "adventures," corresponding to two types of security in common use: PINs and passwords. These exemplars are designed to be comparable to the traditional authentication systems in size of theoretical password space, and are described in Chapters 5, 6 and 7.

Our approach involves more than training users to answer a given challenge encoded or modified by a daily rule (known as a pass-algorithm [22]). Instead, a user can convince the system through dialogue, a familiar exercise in daily human conversation. The goal here is to the leverage the memorable associations users make, allowing for authentication techniques instead of forcing strict memorization of user-generated, text passwords. The three proposals make use of common behaviour found in text adventures: object collection and room navigation. These are extrapolated into puzzles, where the user completes the correct tasks in the correct order. The PIN- and password-level proposals have theoretical password spaces equivalent to their respective systems, and along with the third proposal involve creating sets of objects that are used for authentication purposes. The differences, creation and analysis are described in subsequent chapters of this thesis.

Keyloggers recording entered usernames and passwords typed into a compromised machine are naturally a concern when using passwords, but making use of a one-time password generator can address this type of attack. The time and cost involved to supply and train users with this hardware, as well as the added strain of protecting such hardware can outweigh the benefits for small or poorly funded organizations. Loss of the device results in downtime for the user and recovering account access can cost involved parties time and money.

The third proposal of this family of systems is designed for use in insecure locations where attackers may have installed keyloggers. This is designed to complement

a convenient standard authentication that can be used in locations where the risk of keyloggers and shoulder-surfing is low. This third proposal offers increased protection against attackers who conduct surveillance (e.g., using techniques such as keyloggers and shoulder-surfing) in order to capture user authentication login instances for analysis.

1.2 Thesis Statement

Our hypothesis is that it is possible to acquire theoretical password spaces equivalent to PIN and text passwords by adapting text adventure tools to create virtual environments for authentication purposes. In order to achieve this, the set of text adventure actions and elements are possible activities to include in authentication systems built this way. Using text adventures for authentication can also lead to systems that provide more security functionality than what's possible with standard text passwords, such as variation that provides resistance to replay attacks due to limited degrees of keylogging or shoulder surfing.

1.3 Contributions

Our contributions include implementing and examining three proposal authentication schemes, as examples to show that text adventure systems with sufficiently large input spaces can be built. We cannot predict the types of users who can reliably make use of this sort of authentication system; thus, we examine these proposed schemes based on the theoretical password space of each, in order to calibrate our comparison to existing systems. Making and testing any claims of usability is beyond the scope of this thesis beyond basic prototype functionality testing by an expert user (the

author).

Our first two proposal schemes act as examples for providing a given password space (PIN and text passwords) and the third yields a text password-like theoretical password space combined with a resistance to replay attacks. The schemes have been built in order to show the security properties that are possible with authentication schemes using text adventures, but no claims of usability are made; we consider these schemes to be designed by and for expert-users. Each scheme is designed to fit within an intended threat model, and a theoretical password analysis has been made for each authentication system. The security properties of the schemes are compared to related work.

1.4 Organization of Thesis

The rest of this thesis proceeds as follows. Work relating to recent research into the security of passwords and novel alternative authentication techniques, separated into groupings as subsections, forms the first half of Chapter 2. A concentrated review of the core threats for authentication systems is given after related and influencing work.

Background information on the research into simulating human dialogue is given subsequently, at the end of Chapter 2. This historical look provides the backdrop upon which Chapter 3 expands. The concept of text adventures is introduced, and its growth and history as a medium is given. Information on the tools and technology used in this thesis are discussed in the second half of Chapter 3.

Chapter 4 provides the introduction and explanation of the three proposals of this thesis, as well as key vocabulary used to describe concepts in this work. Example implementation suggestions for each of these proposals have been prepared, which

mirror the theoretical password spaces of of traditional PIN-based and text password-based approaches in a theme more fitting to the concept of authentication as dialogue with human-computable operations. The third proposal adds resistance to replay attacks as a form of challenge-response, which is common in cryptographic protocols, but not so common for text-based authentication schemes. Each is examined in detail in Chapters 5, 6 and 7, respectively. We examine the threat model each approach is designed to defend against in Chapter 4. Source code for each proposal's text adventure is given in Appendix A.

An analysis of each proposal's password space is presented as a latter section in each proposal chapter, exploring the theoretical space of each given system, as well as discussion of defence for anticipated attack vectors. Strategies for enhancing the security of each proposal are presented as well.

Comparisons of the sample proposals to related research in a security context are given in Chapter 8. Discussion of user choice issues is also given in the chapter. Closing remarks are given in the final section of Chapter 8.

Chapter 2

Background

Authentication research is a busy space of solutions. We group particularly related work into three major categories: text-based, cued response and meta schemes of authentication. Our proposed meta scheme fits under all three of these categories.

Text-based systems that are related to the approach taken in this thesis are described in Section 2.1. These are alternatives to the standard text password systems in use by trying to fix problems with the security of user-selected passwords, or by allowing a different sort of text-based input.

Systems that work to improve the memorability and usability of authentication by cueing users are described in Section 2.2. These cued response systems relieve the user in different ways from having to remember their password completely from memory.

Traditional types of text authentication make use of a one-way hash of the user's password for server-side storage, and this hash is compared to the hash of the value a user enters on an authentication attempt. Meta schemes described in this chapter work instead to augment the server to be more expressive, enhancing what a user can send it. These related schemes are explained in Section 2.3.

Key background material on password security concerns is given in Section 2.4, citing an important list of key threats to consider when analyzing an authentication system. As our work is compared to the traditional password scheme and its replacements, we give this brief background on influential password work as a benchmark.

The history of research into simulating human dialogue, both through creating and testing such, is given in Section 2.5. This background provides the foundation upon which sophisticated systems described in Chapter 3 are built, which in turn are used as tools for our work.

2.1 Text

Text passwords provide an easily understood method of authentication. Originally intended for single users on single machines, the adoption of text passwords has grown to become the default authentication procedure for nearly anything digital [13]. Many users are aware that they require a "strong" password (without necessarily having a good grasp on what "strong" means) and many websites, upon registration, grade passwords as they are typed based on length and selection of characters. Password reuse is prevalent, as users have so many accounts that require passwords, averaging 25 each on the Web alone [17, 18]. Password reuse is a security concern because compromising a user's credentials on one site can cascade across many other sites that share the same or similar password.

The problem of password reuse and attackers observing the user during the authentication step are issues inherent to password-based authentication schemes. While theoretical analysis may consider text passwords that make use of the entire available character set (such as all ASCII printable characters), traditionally users select passwords from a much smaller subset, one containing far less entropy than machine-

generated alternatives [14] [47], as a coping strategy to create memorable passwords, as well as reusing passwords across multiple sites. Human memory is very good at forming associative links, but poor at strict memorization of text [3]. Associative links formed are the relationships between objects or memories, and it is conjectured that associative memory stores concepts such as the colour green meaning go and red meaning stop [3].

Research into trying to improve passwords created by users through the process of persuasion has been explored by Forget et al. [19]. In their study, users personally selected a password, and were then offered improvements by the system. These improvements were random characters inserted at random positions within the user-selected character string. Users had the option to "shuffle" to receive new random offerings by the system. The end result was a user able to still select a personally memorable, albeit modified password, and an administrator who could consider her security system to have a minimum level of entropy, as the resulting user passwords are more resilient to both online and offline[1] guessing attacks.

The closest work to ours in the space of text-based authentication methods is *Pass-sentences* by Spector and Ginzberg [41]. Their system allowed a user to select a complete sentence instead of a password string. Users attempting to authenticate were to type the sentence as best as they could remember, and the system would score it based on how similar the attempt was to the original sentence.

Two thresholds were selected by administrators of the system: acceptance and rejection thresholds. If the sentence did not meet the acceptance threshold, but was above the rejection threshold, the user was given the opportunity to answer

[1]Online attacks denote the type of attacks that may only make attempts involving interaction with an active system, being forced to obey the restrictions and delays made by the system. Offline attacks may verify the success of attempts without involving the authentication system, for example testing correctness of candidate passwords by using "verifiable text" [21].

a request for additional detail. These opportunities to improve the accuracy were scored, and the user could respond to further requests provided they still met the range requirement above. Users below the rejection threshold were rejected outright, but the system would continue to ask for additional detail for different parts of the sentence, ignoring each answer given, until the user gave up.

The sentence choices available to users in the testing of pass-sentences in 1994 only involved one of the sentence types described by Roger Schank's Semantic Primitives [39], which are based on ways of expressing actions and thoughts and his work in conceptual dependency processing. The implementation of Pass-sentences used only of one of the semantic primitives: `Atrans`, the act of a subject buying or selling objects, optionally including additional information such as location, price and details about the second party involved. The major attributes included in the scoring of a sentence are the buyer, seller, purchased object, cost, and location of purchase. In addition, Spector and Ginzberg include further detailing of any of the five attributes (called "attributes of attributes") as ways of narrowing the effective password down to more precise terms.

An example of an pass-sentence might be "Jim bought a pack of sugar-free gum from the corner store for two dollars." The seller attribute is omitted, but the object is given extended attributes, and all other attributes are defined.

Pass-sentences map nicely down to the password space, as each noun, adjective and verb represents a dimension similar to characters in a password. Non-imperative words used to connect the parts of a sentence were ignored. The difference from passwords, however, is that instead of memorizing an exact string, users had the option to make use of potentially more memorable events. The system had the key property of allowing dialogue between the user and system after entering the sentence for clarification of details. This expands the possible ways of authenticating

from merely memorizing a long string. Instead, the *meaning* of the sentence was paramount, and the user had the opportunity for a form of error correction.

The study conducted by Spector and Ginzberg provided evidence of high retention with users months after the initial registration of the pass-sentence. The study tested only a single pass-sentence, and did not consider multiple passwords over multiple accounts, nor password interference. The prototype discussed in the seminal paper had been limited in the type of accepted sentences, but by making use of a natural language processor and additional rules implemented for other semantic primitives, a more expressive set of sentence types could be used in an improved version. There has been little research on using concepts such as dialogue for authentication, especially using techniques similar to Pass-sentences.

The security of system-assigned passwords can be measured using the standard formulae adapted from Shannon's entropy [40], as the characters are all randomly generated. Using entropy as a measurement of password security is well known [13], but system assigned passwords can be hard for users to memorize. The use of pass-phrases has been considered an alternative to passwords, where instead of characters, entire words are used. Bard's scheme [5] assigns users pass-phrases yielding password spaces of cardinality 2^{52} to 2^{112}. The words contained in the dictionary used to create a pass-phrase are guaranteed to be 5 units of distance away from any other word, using the Damerau-Levenshtein string-edit distance metric, whereby two words are distance 1 away from one another if the application of one operation converts one string into the other. These operations are the insertion, deletion or substitution of a character, and the transposition of two adjacent characters. Two strings are considered distance 5 away from one another it is possible to convert one word into the other in at least 5 operations. Bard's dictionaries had 2^{13} to 2^{16} entries, and passphrases were made from a collection of several dictionary words.

In Bard's scheme, the user is presented with some number (4-8) of pass-phrases generated by random selection of 4-9 words from one of the above created dictionaries. The user is required to select one of those pass-phrases for use. When authenticating, the user is allowed to type the words in any order, with up to two spelling errors (two of the above operations) per word, and the words can be typed in any case (upper, lower or mixed). Password interference is not considered in Bard's work, as the system is designed to be used as the authentication for a secure user password manager, such that the user need only recall the pass-phrase, while all other passwords are machine generated passwords stored in the manager.

2.2 Cued Response

Though work to improve password-based authentication has mitigated some of the common threats of authentication systems, such as increasing the strength of passwords against dictionary and special knowledge attacks on the user, this has introduced overhead on a user's memory load already saturated by the myriad of passwords and techniques suggested or required for different sites. Cued response, where some information reminds the user of the correct response, is particularly leveraged by many graphical password schemes. The schemes described in this section can be considered exploration of the hypothesis that users can better remember their authentication secrets if cues that trigger correct answers are given—as opposed to no assistance, as in text passwords.

Hopper and Blum [23] made a theoretical analysis of what it is to provide secure protocols that humans can compute in a given amount of time, unaided by companion devices. Their work involved more than the user uttering the correct shared secret, the scheme had users respond in a way that verifies identity but does not disclose

the secret to any other listeners. This is related to the concept of a zero-knowledge proof [16]. Hopper and Blum defend against attack avenues of shoulder surfing and man-in-the-middle between client and server.

Although not technically a cued-response scheme, Weinshall's [46] Cognitive Authentication approach has the user give answers based on the outcome of a path through a grid of images, some of which involve images previous selected by the user as her authentication portfolio. The user begins at the top left corner, and moves across to the next image in the grid if the image is not part of the portfolio, or down if the image is in the portfolio. The user's path will end at the boundary of the grid, which contains a number, which is the answer for the particular round. Several rounds are played with different panels of images before access is granted.

Access is granted based on the probability that the user's answers were not accidentally correct, based on a configured threshold. Users have a portfolio of roughly 100 images, and must train to memorize it, but aren't required to answer correctly 100% of the time, instead authenticating by completing a number of rounds. The scheme was designed to be resistant to shoulder surfing attacks, as it was claimed that an attacker, given a number of witnessed authentication rounds, cannot deduce the user's portfolio sufficiently to force a login himself. Golle and Wagner [20] later proved that the system can be broken by an attacker with a SAT-solver and 60 witnessed authentication rounds (approximately 6 user logins), with the attacker able to reconstruct the user's secrets due to the limited number of ways a user can arrive at particular end-points of the panel. Weinshall's scheme still has various properties of pedagogical interest [6], such as the inability for users to verbalize the password to an attacker conducting a social engineering attack, and the mitigation of dictionary attacks instead of exhaustive attacks.

Cued Click-Points (CCP) [10] works similarly to other click-based graphical pass-

word systems by allowing users to specify points on images as their password. PassPoints [49] made use of five click-points on a single image as a user's password. CCP differs from PassPoints in that each mouse click in the user's password is on a different image, which is selected based on where the previous click was made. A goal of CCP was that each user click-point is *cued* by the image presented. The intention is that the image provides *implicit feedback* [10] useful to the legitimate users, but not to attackers unfamiliar with the user's password. The prototype system involved 330 images and users created passwords of five click-points.

In order to improve the strength of user-selected passwords, Chiasson et al. created Persuasive Cued Click-Points (PCCP) [9], which persuaded users to select click-points within a given randomly selected viewport of the images presented. This was done to minimize the "hotspots" [44] on images where users most often clicked and to reduce the patterns created in user-selected click-points. The password space and usage of created passwords works the same way as CCP, only the registration stage is modified in PCCP. Users clicked within the given viewport or shuffled the viewport's position to somewhere else in the same image (which can be done an unlimited number of times). The intention was to make creating a less secure task more cumbersome for users and to make more random passwords the path of least resistance. The user study conducted showed that such a goal was reached.

2.3 Meta schemes

A third grouping of systems we consider to be related work are meta schemes: schemes which define families of schemes. This can require that something more powerful is used on the server-side of the authentication system, in contrast to the simpler approach of a hash-matching algorithm traditionally used in password authentication.

This section describes some of the work in this area.

A rarely considered type of authentication makes use of obfuscation. Although it may not be a strong form of authentication with strict security properties, the main idea is to hide the authentication information in plain sight such that only legitimate users and the server are able to deduce the true message. Cheswick [8] offers the use of human-creatable obfuscation tricks to augment the already well-understood challenge/response authentication techniques such that the authentication itself can be done without an assisting device. He suggests that signs used by pitchers, catchers and coaches in baseball be adapted to work in computer security: authentication might be a series of steps, where some steps are only involved to obfuscate the truly relevant steps. Cheswick uses the term "pass-algorithm" to name this authentication system. This is related to the secret algorithms for user validation in Haskett's work [22], which involved users memorizing the daily algorithm for providing a response to the cue given by the server. A partial listing of Cheswick's signals adapted from baseball are given in Section 5.3.

The open questions posed by Cheswick invite the reader to consider applications of human-designable obfuscation attempts for authentication. We believe that if users created their own pass-algorithms, and if adoption reached a certain threshold, attackers might have no other alternative but individual inspection of authentication attempts, thus greatly hampering the rate at which accounts might be compromised. Defeating the automated, online attack is a difficult goal to attain, however.

The above alternative authentication schemes can be considered to be limited forms of dialogue, designed for a specific context, between user and system, where the system action is more elaborate than simply hashing and comparing to a stored value.

Alsulaiman and El Saddik [2] created an experimental 3D graphical password

authentication system which allows users to specify a set of actions in the environment to be their passwords. The system involves a virtual art gallery, allowing users to view art and virtually type on the available computers. The theoretical password space is composed of the ordered permutations of interactions with a subset of the available objects.

Much of the work in creating the experimental system is comprised of creating the graphical environment and its objects, but the research was part of the overarching goal to create a vast 3D world a user can authenticate with. This is particularly related to this thesis work, but differs in that the users' password is the ordered interaction of objects only; our work in Proposal II makes use of the directions taken in the text adventure in addition to the interaction with objects.

Instead of using an authentication system other than passwords, Leversund [29] proposes that the mandatory policies that restrict acceptable passwords be scrutinized. Assuming that users create passwords that are the minimum amount of effort, but still acceptable according to the password policy, Leversund postulates that if the policies were randomly generated for each user, the weak password subspaces [45] of user-selected passwords can be mitigated somewhat. A weak password subspace is a set of passwords in a given theoretical password space that are more likely to be selected by users, and thus more likely to be guessed by an attacker. If a weak password subspace can be enumerated by an attacker, he can create ordered dictionaries of these passwords in order to minimize the effort he requires in order to gain entry to a system. The size and quantity of weak password subspaces can be mitigated with meta password policies because not all users are able to create passwords that fit the same subspace due to changing policy rules. Sample rules include minimum character length, character case restrictions, required inclusion or omission of special characters or digits, and absence or requirement of words from a given dictionary.

2.4 Benchmark

Florêncio et al. [18] present a list of principal threats that passwords strive to be resilient against. We list them here for later reference.

1. Phishing
2. Keylogging
3. A brute-force attack on the user's account (i.e. an attacker knows the userID and tries to guess the password)
4. A bulk guessing attack on all accounts at the institution
5. Special knowledge or access attacks:
   (a) guessing based on information about the user
   (b) shoulder surfing
   (c) console access to a machine where password auto-fill is enabled or a password manager is in use [18]

Florêncio et al. consider brute-force attacks against a single user and against many users of a single institution to be two separate attacks, we will refer to their "bulk guessing attack on all accounts at the institution" as multi-account guessing attacks herein. An attacker may have access to the user's password manager by compromising the integrity of the user's local machine, by rootkit, trojan software or an otherwise malicious program. An attacker could also acquire access to an online password manager. Once access is obtained, the attacker may already be capable of discovering user passwords, as most users do not protect the password store by encrypting with a master password [42]. If the password store is encrypted, an attacker may be able to procure passwords by launching an offline guessing or dictionary attack.

We consider these principal threats by Florêncio et al. for context in evaluating our proposals. Reference to this list is made in subsequent sections analyzing our proposed schemes.

Florêncio et al. [18] argue that users need not be required to have high-entropy, randomly generated, long passwords—provided that the system in use ensures online only authentication attempts and has a "three strikes" rule in place. In fact, they argue that a password of approximately 20 bits of entropy provides enough security for an organization of sufficient size (millions of users) when considering guessing attacks. For smaller systems, the requirements for password strength can be reduced further, as the space of used passwords is even more sparsely populated. Florêncio et al. elaborate that a three strikes rule may simply be an IP address-based lock-out period after three unsuccessful login attempts, but can be extrapolated into a geometrically increasing delay before additional attempts are permitted.

The claim that 20 bits of entropy is sufficient for most security systems may still be difficult to reach for average users, due to the fact that on average users are notoriously poor at selecting strong passwords [14] [33] [27] [47]. Based on the NIST estimations for user-selected password entropy, users left to their own decisions create eight character passwords that yield approximately 18 bits [7], however the validity of the measurement of the estimated entropy of a user-created password has been refuted by Weir et al. in recent work [47]. It is thus unclear how much Shannon entropy user-selected passwords truly yield, and since we have no user studies to evaluate guessing entropy of our prototypes, we consider the coarse estimate of the upper bounds of the theoretical password spaces instead. In this thesis, we argue that creating a human-computable authentication system yielding a theoretical password space comparable to text password authentication is achievable, as illustrated by the proposals shown in Chapters 5-7.

2.5 Simulation of Dialogue

The Turing Test [43] is a classic example of a philosophical question of computer science: can computers be programmed or created to think? Rather than trying to argue the possibilities of proving artificial intelligence when cognitive science cannot fully answer biological intelligence, Turing proposed an analogue to the question that instead asks if a judge can reliably differentiate between a human and machine through text-based communication. Answering such a question was a goal of the field of natural language processing, reaching the forefront of computer science research in the 1960s and '70s. A computer program has *passed* the Turing Test when the judge cannot correctly complete the task above.

Some researchers attempted the goal of passing the Turing Test by simulating dialogue in order to create the illusion of human-to-human conversation. Joseph Weizenbaum created ELIZA [48], a simple natural language parser in the mid-1960s. Rather than creating software which feigns artificial intelligence, Weizenbaum simply added a list of cue words and phrases that corresponded to prepared canned responses (known as scripts). One popular script was known as DOCTOR, which behaved like a simple-minded psychotherapist. The rest of the ELIZA system simply side-steps the input by questioning questions or using other canned avoidance measures. Such a system might be peculiar and artificial to a user, but not if the context of conversing with ELIZA was that of a conversation between psychotherapist and patient. The ruse hides the limited capability of ELIZA, as the conversation merely appears as an, albeit parodied, client-centred psychotherapy interview.

ELIZA can be viewed as a system that capitalizes on a weakness in the Turing Test: rather than attempt to replicate knowledge and intelligence, ELIZA merely

allows for the conversing party to anthropomorphize an otherwise simple chatterbot.[2] A complete implementation of natural language processing is unnecessary to provide the illusion of a realistic conversation in a given context, as the human party in the conversation can accept reasonably non-intelligent dialogue as human-like. Humans consistently project human behaviour onto non-human creatures and objects. This is leveraged by unsophisticated text parsing algorithms to approximate what a human can construe as intelligent conversation.

Chatterbots, with the help of ELIZA and other early natural language processing bots, have grown to have more complex features, making use of real-time learning and evolutionary algorithms [30]. These programs are used as helpful knowledge-base assistants, providing users with a first point of contact before being directed to the correct human authority. Chatterbots have other uses, as amusing members of chat rooms, malicious purposes of phishing and spam and as entries in the Loebner Prize competition. This competition is designed to award prizes to the authors of programs that can perform "well" in a Turing Test, potentially using tricks stemming from ELIZA to persuade the judges of such an outcome. Given the correct context, users can readily accept that a chatterbot is realistic, and if the chatterbot can respond in a reasonable manner, maintaining the illusion of knowledge in context, users can continue holding such beliefs.

The sophisticated natural language parsers created over the past few decades of computer research have been used as tools to further the abilities of many types of systems. Some work, which at the time involved serious effort, was not intended to be taken entirely seriously: one of the first computer adventure games, William Crowther's *Colossal Cave*,[3] was a text-based exploration game created in 1975 [1].

[2] A chatterbot is a program designed to simulate intelligent conversation with a human counterpart. The term was first coined by Mauldin [30], while attempting to create a chatterbot that fools the human conversing with it.

[3] This game has been reimplemented and improved by many, but most famously by Don Woods.

Colossal Cave was one of the first of its kind, and contained a simple natural language parser in order to allow players to enter commands in simple English in order to operate the game. This entertaining program has inspired many people, and was still being modified and improved thirty years later. Work in natural language processing has become the foundation for text adventures and other more recently introduced gaming genres.

It has been known by many names, such as *Colossal Cave Adventure*, *ADVENT* or simply *Adventure*. For consistency, we refer to the game as *Colossal Cave*.

Chapter 3

Text Adventures

Our work involves creating an authentication system that is based on the use of a dialogue between user and system. In order for users to customize and understand how to interact with such a system, we must rely on alternatives to creating an AI-complete system, as passing the Turing Test is arguably impossible.

Interactive fiction is a form of computer gaming dating back to the mid 1970s, reaching a peak in popularity in the early 1980s.[1] Most such interactive fiction games can be referred to as text adventures, and are played by invoking text commands to move a character and interact with objects in the environment. The core concept throughout all such interactive fiction games is that the story becomes an elaborate dialogue between the player and the game.

Though interactive fiction was a popular computer gaming platform in the past, it now has seen a recent resurgence in popularity; tools have advanced to point where creating a new game is feasible by non-programmers. This type of game creation can be used to create proof-of-concept or rapidly developed prototypes for more complex graphical games, as puzzle logic and storyline can be created before graphical assets

[1] A short review of the history of interactive fiction can be found at `http://www.adventureclassicgaming.com/index.php/site/features/503/`.

and game engines are completed.

The history of interactive fiction as a genre is described in Section 3.1, by describing the most historically significant company that produced interactive fiction titles: Infocom. Their platform, known as the Z-Machine, is described therein, as it is used as the target platform in the implementations for each proposal of this thesis.

Frotz, a popular open-source interpreter for games designed for the Z-Machine is introduced in Section 3.2. Inform is a popular, community-produced programming language for creating modern interactive fiction, and is described in detail in Section 3.3. These two products are components in the toolkit used in the execution of the prototype implementations.

3.1 Infocom & The Z-Machine

Text adventures were first written for mainframes and were later ported to PCs. The founding members of Infocom, the most famous interactive fiction game company, were heavily influenced by William Crowther's *Colossal Cave*, one of the first text adventures.

Unimpressed by the weak parser and limited functionality of the virtual experience of *Colossal Cave*, Infocom's founders, then students in the Dynamic Modelling group at MIT's Laboratory for Computer Science, set to work to create their own text adventure game. *Zork* [28] was created with MIT's Lisp-like MDL programming language for DEC's PDP-10, and included an improved parser compared to *Colossal Cave*'s. When Infocom was founded, *Zork* was decided to be the first release.

Significant engineering was required in order to bring the large *Zork* codebase to the PC hardware platforms, which were incompatible with the original mainframe code. This constraint brought about the creation of the Z-Machine, as in 1979 it would

have been an arduous task to rewrite each text adventure anew for each platform. This Z-Machine was a smaller, streamlined version of MDL capable of interpreting only the code related to Infocom's text adventures. Each game was written once in ZIL, the Zork Implementation Language, and was compiled into a binary that worked on any platform for which a ZIP, the Z-Machine Interpreter Program, was written, such as the Apple II or the Tandy TRS-80. While *Zork* fit reasonably on a mainframe's memory, it was too large to fit on the smaller PC RAM. It was broken down into three contained chapters: *Zork I*, *II* and *III*. This was due to the large storyline of *Zork*, not the interpreter itself. The famous beginning of *Zork I* is shown in Figure 3.1 with a few commands already entered. The history of Infocom, and its legacy, can be viewed on its website [24].

The Z-Machine was used as the platform for all of the Infocom text adventures, modifed and expanded as the games required more features. Cornerstone, the business database software that eventually contributed to Infocom's undoing, was written for the Z-Machine software stack in order to maintain one code base for many platforms. The Z-Machine is the lasting legacy of Infocom, influencing other text adventure platforms and other related gaming genres, such as graphical adventure games [26], which received commands from the user as text (like in text adventures), but displayed the results in a graphical form. Graphical adventures had their own portable interpreters as well. For example, LucasArts' SCUMM engine was designed originally for their first graphical adventure game *Maniac Mansion* in 1987, but was later developed as a platform for nearly all of their games until 1998. It lives on as a community-developed ScummVM capable of playing the old LucasArts titles on modern hardware and operating systems. Graphical adventure games led to the rise of point-and-click adventure games, operated entirely by the player with a mouse. These are similar to the original graphical adventures, but lack direct text entry.

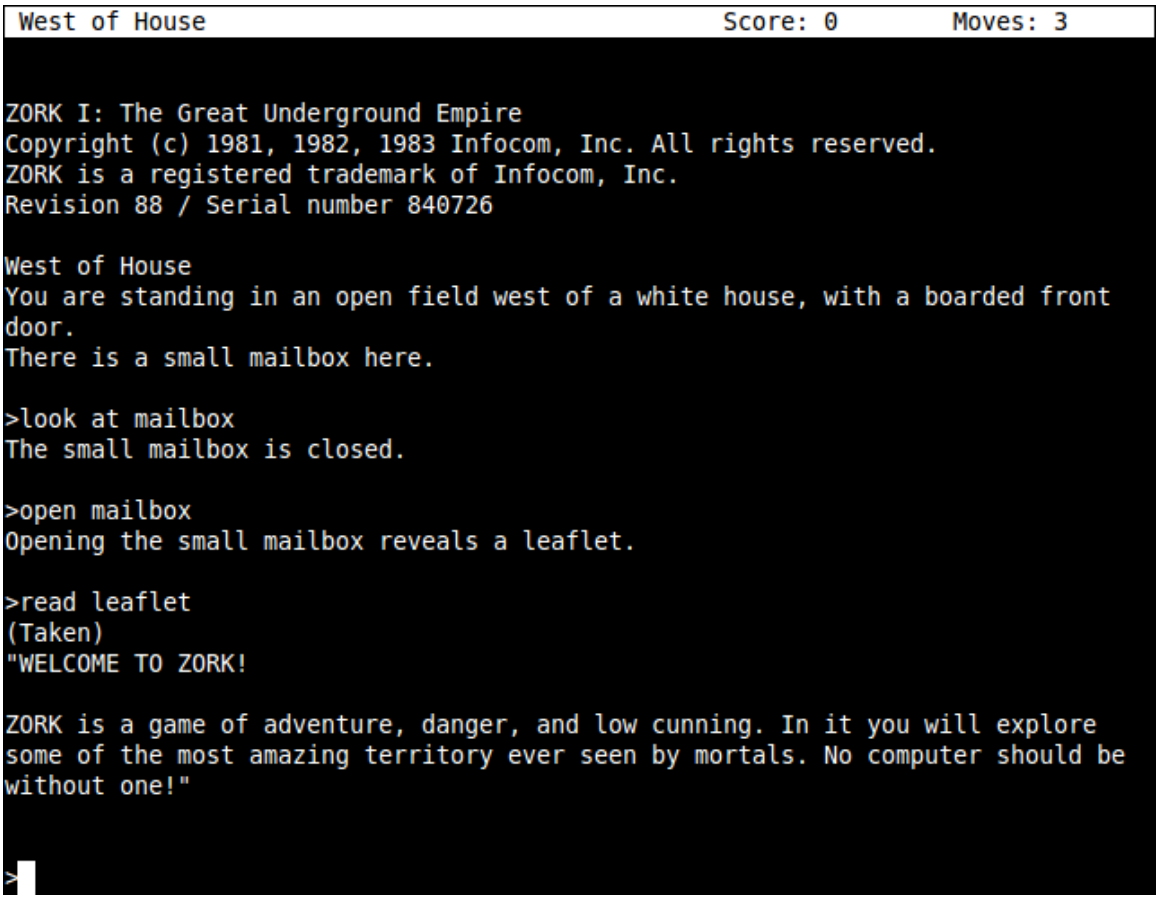

Figure 3.1. The introductory text of *Zork I*, the first chapter of the *Zork* trilogy.

The niche of these text and graphical adventure game genres has lived on, as online communities develop new games. Interactive fiction is still a popular medium, leading to interactive novels and hypertext fiction: digital or traditional books with storylines that are changed by multiple choice decisions made by readers that lead to alternate endings. A popular example of this type of book are the *Choose Your Own Adventure* children's book series [11]. In these books the reader is asked to determine the next course of action at key parts of the story, which require the reader to flip to a specific page based on the choice made. These books are frequently written in the second-person narrative, putting the reader in the shoes of the book's protagonist.

3.2 Frotz

The Z-Machine has lived beyond Infocom, as authors created new interpreters for its code. The open-source Frotz[2] program interprets Z-Machine games in compliance with the official standard [34], and is available on many types of platforms, from desktop operating systems to the Nintendo Game Boy and Apple iPhone. It is capable of handling Z-Machine binaries as well as other text adventure platforms of the past and present by providing an emulated environment.

The techniques used in text adventure games are adapted to create authentication systems in our prototypes. Navigating rooms and collecting objects are abilities common to nearly all text adventures, and are particularly relevant to this prototypical work. Some modification to the software implementation of text adventures was required, however.

The source code of Frotz is released with a less capable version known as Dumb Frotz, which is compiled and run as `dfrotz`. It lacks advanced features added to the

[2]Official website: `http://frotz.sourceforge.net`

```
"Exploration" by "Carson Brown".

The release number is 1. The story creation year is 2010. The story
    headline is "An Exploratory Adventure in Interactive Fiction." The
    story description is "This is a simple story that involves
    interacting with a world for testing purposes."

The living room is a room. "You find yourself in the living room of a
    small, single-story bungalow."

The bedroom is west of the living room. "A fair sized bedroom contains a
     few pieces of matching furniture: a bed, dresser and wardrobe. The
    window curtains are closed."

The dresser is a supporter in the bedroom. The description is "A simple
    dresser made of oak is against the near wall. It has top, middle and
     bottom drawers."

...
```

Figure 3.2. Example code snippet of an interactive fiction written in Inform 7.

Z-Machine in later updates such as illustrations and sound, relying on the terminal environment it is launched in. Designed as a bare-bones version of Frotz for less capable operating environments, this software is particularly suitable to our prototypes because it was easily modified to work within our Python middleware. Dumb Frotz lacks the more complex portions of Frotz which involve the `curses` library, relying only a C compiler and shell. Some editing to the Dumb Frotz implementation was necessary to adjust the output.

3.3 Inform

Inform is an elegant and sophisticated language and IDE for creating interactive fiction. Its current iteration, Inform 7, provides an environment for writing stories in

English sentences, rather than in a more strict programming language.[3] Figure 3.2 gives a small code snippet of the beginning of an interactive fiction written in Inform 7. The output of interactive fiction titles created with Inform very closely resembles *Zork*. A major design feature of Inform is that the interactive world is composed of rooms, just like Infocom games. Each room may house objects, and each object may in turn support other objects. The "support" attribute is Inform's way of allowing objects to be placed on one other, e.g., a table can be a supporter in order to hold a lamp. Objects can be extended to support certain actions and contain some amount of state. Default actions provided by Inform include movements according to the compass rose (e.g., north, south, east and west movement between connected rooms), an inventory system, and the ability to examine objects. Another key concept of Inform is actions, which are carried out by players or non-player characters (NPCs). Some actions are included by the default Inform environment, such as picking up objects, opening containers, moving through rooms. These are actions that can be typed out by the player or scripted to happen by the author of the fiction. A major feature of Inform is the highly extensible nature for adding new types of objects and actions to build more complex objects.

The Inform creators have written two manuals for authors: an overview of the language, *Writing with Inform*, and the goal-specific *Recipe Book*,[4] which provide extensive examples for complex operations in Inform. An extensive plugin library has been developed by the community of Inform, allowing authors of interactive fiction to augment their stories with complex objects, logic puzzles, non-player characters (NPCs) and actions.

[3]Full descriptions and examples of Inform are available at the official website: `http://inform7.com/`.

[4]Both titles are available digitally from the official Inform 7 website: `http://inform7.com/learn/`.

Inform is capable of releasing a completed fiction as a Z-machine compatible binary. It is particularly compatible with Frotz, and offers graphical integrated development environments for Windows, Mac OS X and Linux. A command-line variant is offered that simply compiles Inform stories into binary files designed for the various interactive fiction virtual machines.

3.3.1 Writing in Inform

Inform allows for creating highly detailed works of fiction, complete with realistic objects working as a user might expect them to. It also allows authors to create somewhat complex functionality with minimal effort by extending the program with new action types. A completed work written in Inform is commonly called a *story*. This section examines the work involved in making a simple story related to the famous opening of the film *Indiana Jones and the Raiders of the Lost Ark*. The full Inform source code is listed in Figure 3.3. Using the source code as reference, the major portions of writing with Inform are explained below.

```
"Jones" by "Carson Brown"

Use no scoring. The Abandoned Temple is a room. "You are in a dusty
    temple room deep in a jungle in South America. A glistening golden
    idol—the one you've been searching for—is before you, resting on a
     rough, stone pillar. Based on the past few hours of exploring this
    ruin, you know it simply must be booby trapped."

South of the Abandoned Temple is the Jungle.

The torch is carried by the player. The description is "The torch burns
    brightly, illuminating as much of the temple (and the snakes) more
    than you'd prefer."

The whip is carried by the player. The description is "Your leather whip
     has saved your life too many times to remember. You always carry it
     on adventures like these."

The bag of sand is carried by the player. The description is "The bag of
     sand was brought for the very purpose of disarming any sort of
```

```
    weight-balanced booby traps. Perhaps it will be useful here."
    Understand "sandbag" as the bag of sand.

The fedora is worn by the player. The description is "Your trusty brown
    fedora. What a great hat!" Understand "hat" as the fedora.

The stone pillar is a supporter in the abandoned temple. The description
     is "The pillar seems to be a roughly hewn slab of hard rock, with a
     raised portion acting as an altar to hold the golden idol. Yeah,
    definitely booby trapped."

The golden idol is on the stone pillar. The description is "The idol is
    exactly what you came here for, sparkling in the light of your
    torch. Belonging to the ancient Chachapoyan tribe, it was a golden
    representation of the Aztec goddess Tlazolteotl of fertility and
    childbirth, said to be secreted in the heart of the Temple of
    Warriors. That would explain all those spikes and traps from
    earlier!"

Instead of taking the golden idol, say "Just taking the idol will surely
     release the weighted booby trap underneath it. Maybe you can swap
    it for something of similar weight...."

Before listing nondescript items:
        Change the stone pillar to not marked for listing

Before going from the Abandoned Temple:
        If the player is not carrying the golden idol:
                say "You can't leave just yet, that idol won't capture
                    itself." instead.
                [ We say "instead" in order to do some action instead of
                     the default; in this case, say some text instead of
                     leaving the room. ]

Swapping is an action applying to nothing. Understand "swap" as
    swapping.
Check swapping:
        if the player carries the bag of sand:
                say "You'll need to be more specific about what you want
                     to swap with.";
        otherwise:
                say "You don't have anything good to swap with."

Swap-making is an action applying to one thing. Understand "swap
    [something]" as swap-making.

Check swap-making:
        if the noun is the golden idol:
                if the golden idol is carried by the player:
                        say "You can't switch it back, you'd risk
                            setting off the booby trap.";
```

```
                otherwise if the player carries the bag of sand:
                        try swapping the golden idol with the bag of
                            sand instead;
                otherwise:
                        say "You should try swapping with the bag of
                            sand." instead;
        otherwise:
                say "[The noun] look[if the noun is not
                    plural-named]s[end if] fine the way [if the noun is
                    plural-named]they are[otherwise]it is[end if].";

Swapping it with is an action applying to two things. Understand "swap
    [something]
with [something]", and "swap [something] for [something]" as swapping it
    with.

Check swapping it with:
        if the second noun is not the bag of sand:
                say "[The second noun] can't be used for swapping
                    things." instead;
        otherwise if the player does not carry the bag of sand:
                say "You'd need to find a bag of sand if you want to do
                    that." instead;
        otherwise if the noun is not the golden idol:
                say "[The noun] look[if the noun is not
                    plural-named]s[end if] fine the way [if the noun is
                    plural-named]they are[otherwise]it is[end if]."
                    instead;
        otherwise if the golden idol is carried by the player:
                say "You already did that." instead.

Instead of swapping the bag of sand with something:
        try swapping the noun with the bag of sand.

Carry out swapping it with:
        say "You walk gingerly across the room, avoiding the centre of
            the room's rough stone tiles, for fear of setting off more
            booby traps. Cautious, you remove some sand to reduce the
            weight of the bag to better approximate the golden idol.";
        now the player carries the golden idol;
        now the stone pillar supports the bag of sand;

Report swapping it with:
        say "You switch the bag of sand for the golden idol."

Every turn:
        if the player carries the golden idol:
                say "Though your bag of sand swap seemed an elegant
                    solution, the small difference in weight has
                    depressed the centre of the stone pillar enough to
                    trigger the booby trap, causing the entire temple to
```

```
                begin to crumble before you! Fleeing across the
                temple floor, you dodge deadly arrows, only to
                quicken your pace to outrun a giant rolling ball!";
        end the game in victory;
```

Figure 3.3. A complete Inform story file. The story resembles a famous scene of the film *Indiana Jones and the Raiders of the Lost Ark*.

Every Inform story project begins with a title and author, as well as any preamble necessary, such as included plugins for advanced features. Inform works by navigating through maps of rooms; no story can take place without a location as backdrop.

Objects are added to the story and given descriptions to assist and entertain the player. Objects can be added to or removed from the user's inventory, as well as worn by the user. Synonyms for objects can be given by using the syntax like `Understand "hat" as the fedora` to add another term that players can use. By default, in specifying that the player wears the hat, it is known to Inform that the article is wearable.

Objects can be made as *supporters* and *containers* in order to hold objects on top or inside, respectively. By default, supporters are considered to be pieces of furniture, and are thus not collectable by the player and fixed in place. Containers can be considered openable and lockable. Openable containers, unless instructed to be *transparent* hide the objects contained therein to the player until they have been opened. Objects can be made to be both supporters and containers if necessary.

The majority of the rest of the source code of the story involves overriding the ability for the player to simply remove the golden idol, and instead introduce a new action type for the player to use: the ability to swap the idol for the bag of sand. The code is designed to guide the player into trying to swap for it with the correct object.

At every turn of the game, the player's inventory is checked to contain the idol, upon which a closing message is printed, and the game is ended in victory. Many

popular text adventures feature alternative endings, thus Inform allows for authors to craft games that have multiple story paths, and set conditions in order to end the game differently. Our example, *Jones*, has only one victory condition implemented for brevity—the player picking up the idol.

Chapter 4

Authentication Dialogues: Overview

Authentication making use of a text adventure operated by a user provides broadly scoped possibilities, as the types of virtual environments created is limited only by creativity. This thesis describes three plausible schemes that are implemented as text adventures designed for authentication, and are examples of what users could create and make use of under different threat models. These can be thought of as populating a sliding scale of authentication systems: those offering PIN-level security, those comparable in security to text passwords, and those roughly comparable to token-based authentication, when considering the security properties. This can also be considered a spectrum with usability and security as goals at either end.

We consider a system to be similar to text passwords if it is susceptible to similar types of attacks, e.g., attackers recording the authentication step via keyloggers. PIN-like authentication mechanisms work much like passwords but have a much smaller theoretical password space. We consider a system to contain protection against key-logging attacks if it provides protection against attackers replaying information ob-

tained by from recording the user's input, e.g., when the system makes use of one-time passwords. An example of such a system is RSA's SecurID hardware tokens, which use built-in, factory-set random keys.

In this thesis we give three proposals, by giving example schemes, and each is a representative scheme suitable for a specific class of threat model. Each example scheme is a text adventure designed with authentication in mind. Users of this type of authentication system, familiar with text adventures, can personalize their own adventure in order to specify the objects, puzzles, rooms or tasks desired. We suggest these three text adventures as examples of schemes that users might build.

Chapters 5, 6 and 7 each examine one of these classes, providing a system description, sample use case and security analysis. The threat models and intended usage are given in Section 4.2.

4.1 Key Terms

Some key terms are shared among all proposals:

Environment: The story and items involved in each instance of the authentication system; also known as a *scheme*. Our prototypes make use of Inform story code, released as Z-Machine binaries and run inside Frotz in a Python wrapper.

Instance: One individually generated use of a given environment, i.e., the compiled text adventure used for an authentication attempt. Instances may contain a randomized set of rooms, objects, or tasks, and, depending on the system, may have more than one solution.

Pool: A set of objects used in a proposal that share a common purpose. Objects that are used for authentication purposes are said to be members of the "au-

thentication pool." Graphical password approaches using collections of objects refer to these sets as *portfolios* [6].

The use of multiple grades of authentication has been explored in TwoKind by Bailey et al. [4]. In that research, users were tested on how well they understood the principle of least privilege [38] for using multiple authentication inputs for granting high and low privilege access to their simulated accounts, losing points when high access was granted and abused by an attacker. Users were given more points when using the lowest privilege possible for completing different tasks requiring different privileges. Proposal III can be used in an authentication system much like TwoKind, as it has been designed specifically for higher-threat environments as described further in the threat model below.

Our prototypes use story code (source code) written in the Inform 7 language (see Chapter 3), which is compiled into a binary that meets the ZIL specification. A command-line version of Frotz is used to interpret the text adventure. The current prototype is standalone; in order to integrate it with an existing authentication system (e.g., a website or UNIX login), our prototype would require some integration software to transform adventure output into a form suitable for the authentication system. We have implemented a small Python front-end script that currently expands short-form commands. For example, to integrate the prototype with standard UNIX login, this script would need to be modified in order to fit with the Pluggable Authentication Modules (PAM) specification [32]. Note that existing alternative authentication mechanisms such as one-time passwords and key-based authentication have straightforward PAM implementations.

4.2 Threat Models

The meta-scheme presented in this thesis makes use of an entire text adventure as the shared secret between user and system. The security qualities of any text adventure are likely poor as standard text adventures are designed to be solvable by anyone willing to play them. We expect that user-created text adventures will be similarly solvable without some constraints. In order to explore what secure authentication is possible with text adventures, we have created three distinct text adventure scenarios that have useful security properties. The schemes involve selections of objects, where a user customizes her text adventure by "filling in" the scheme with objects. The prototypes require adding the objects manually into the story code, instead of in an automatic, user-assisted manner. These schemes potentially provide security at the cost of memorability and login time. While not shown in this thesis, we assert that there exist schemes that have both security and usability requirements met.

The proposals made in the next three chapters are examples of families of systems designed to replace or complement existing password-based schemes. In situations where the user has relative trust in the security of her computer, Proposals I and II can be used, replacing a password-based authentication. Proposal III has been designed for use in situations where using a text password is potentially dangerous, due to the risk of a compromised machine, a location under attacker surveillance for password entry or other situations where password entry has a risk for compromising the account.

Each proposal is designed for use in authentication to an online account, that is, verification of authentication with a service provider is required (e.g., banking websites, single-sign on in a distributed computing laboratory, etc). Each proposal is built with the same tools; the difference is the amount of protection each proposal

yields, at the cost of more user involvement and effort.

Systems that allow such online authentication attempts must properly address both targeted and system-wide brute-force attacks (recall the list of principal threats given in Section 2.4) or declare the problems out of the scope of their threat models. Consider a computer laboratory containing multiple machines and users. Users must be able to log in on a machine, from either a remote connection or local access. The authentication scheme must prevent attackers from gaining remote access. Our three proposals fit different use cases, depending on how a user accesses laboratory resources. As with most portable authentication schemes (usable on different systems without special hardware/software), we do not consider offline-based attacks in this thesis. The advent of rainbow tables [35] removes some of the benefit of hashing text passwords instead of storing the plaintext password itself.

Phishing attacks are within our threat model. They would require a man-in-the-middle attack, as the information from the server to the user, and vice-versa, must be relayed by the attacker in order to intercept the user's secret. This type of attack is discussed in Chapter 8. A summary of the key threats in each threat model is given in Table 4.1, where a checkmark is used if the scheme addresses the given threat. Partially addressed threats are denoted by $\checkmark^{1/2}$.

Proposal I Threat Model

Offline attacks such as access to the user's password manager are not considered in Proposal I's threat model, nor is the ability for attackers to compromise the end user's connection or host in order to facilitate keylogging and shoulder-surfing: we consider only the online case described above. Educated guesses based on known user information are within our threat model. The exemplar prototype described in Chapter 5 yields a theoretical password space the size of PIN security, though due

to the constraint of requiring full text entry, cannot be considered a replacement to PIN entry in all locations (e.g., bank ATMs which have only a numeric keypad). In the computer laboratory setting described above, we suggest that Proposal I best suits users whose online access is through local machines. It provides the simplest authentication mechanism of the three, but cannot thwart brute-force attacks: the theoretical password space is too small.

Proposal II Threat Model

Our threat model here includes brute-force attacks as well as educated guesses employing special knowledge gathered on the user. Proposal II is intended as a mid-level security solution, comparable in theoretical password space size more to passwords than PINs but short of higher-level mechanisms that require the user to carry a private key file, one-time-password generator, or other authentication token, which yield further security but have added costs and requirements. We suggest Proposal II for users accessing the computer laboratory via remote connections. Potential benefits compared to ordinary passwords are described in Chapter 6.

Proposal III Threat Model

Our threat model for Proposal III includes hardware and software keystroke logging and shoulder surfing (which defeats traditional passwords). Protection against capture-and-replay attacks requires some form of one-time or challenge-response authentication. In the computer laboratory scenario, users who wish to access laboratory resources from unsafe remote machines should use Proposal III. Other attacks in scope for our threat model here are collecting information on the user and conducting brute-force attacks on one user, or a multi-account guessing attack at one institution: these correspond to the principle threats in Section 2.4 numbered $2, 3, 4, 5a$, and $5b$.

Threat	Proposal I	Proposal II	Proposal III
1. Phishing	$\checkmark^{1/2}$	$\checkmark^{1/2}$	$\checkmark^{1/2}$
2. Keylogging			✓
3. Brute-force attack on one user		✓	✓
4. Multi-account guessing attack		✓	✓
5*a*. Guessing based on user information		✓	✓
5*b*. Shoulder surfing			✓

Table 4.1. Summary of proposal scheme threat models. Each row corresponds to a principle threat listed in Section 2.4.

Some protection is given against threat 1, phishing, as explained in Chapter 8. We offer Proposal III for use in locations that cannot be trusted, and suggest that a less involved authentication scheme be used in trustworthy locales.

Chapter 5

Proposal I: PIN-Level Authentication

Traditional PIN authentication has generally involved digits 0 to 9 and PIN lengths between 4 and 12 digits, mostly settling between 4 and 6 digits [25]. If verification of PIN-based authentication attempts can be forced to require communication with the authentication server (i.e., online authentication), PIN-level security can be considered adequate, provided that attempts are rate-limited or a three-strikes rule is enforced. It is also often accompanied by the requirement of a physical card or other token, such as a banking ATM card. While institutions that use PIN-based security may implement some form of these security constraints, problems arise due to the types of PINs selected by users—memorable to both users, and attackers. These PINs are not equi-probable, and informed attackers can create dictionaries sorted by likelihood of success.

Acquiring PIN-level security from an authentication system using interactive fiction is relatively straightforward if we consider PINs to be an ordered number of selections from a given set. If we look at the most common length of user-selected

PINs, 4, the space of passwords is considerably smaller than the space of standard text passwords. Making our authentication system thus involves a password space of at least 10^4, which can be achieved by a design similar to the task of entering four numbers (albeit requiring more input time): performing four tasks in the virtual environment, where each task is selected from ten possible choices.

With the environments that can be created with Inform's interpreter, users are limited only by imagination when given reasonably well described and furnished virtual worlds. This chapter examines one particular possibility, through setup and execution.

5.1 Representative Example of Proposal I

In this scheme, the authentication task consists of ordered stacking of the correct objects on top of one another. Stacking has a useful side effect of reinforcing order, e.g., it is clear that the third object is on top of the second object. The need to make order clear to the user is important because we make order relevant, as is the case for PINs. In reality, or a game relying on a realistic physics engine, stacking arbitrary objects on one another relies on physical characteristics in order to be possible: flat objects are much easier to stack than round ones. Such actions are possible with Inform regardless of what an object's physical state might be, as objects can be made to be stackable by adding the ability as an object attribute. This means an attacker trying to guess a user's password cannot rule out an object based on its real-world physical properties, though a user might carry this inherent bias of expecting objects to only behave like their real-world counterparts.

Upon registration the user selects a set of objects that can be used in an instance. Currently, we require users to create this list themselves, i.e, we have no sample

object list they may select from. These are the objects that a given user will use for her password. This is akin to a user's keypad still containing the digit '7', though she never selects it for PIN entry; the concept here is that the user can decide what the objects are, instead of working with the set of integers $\{0, 1, \cdots, 9\}$. Previous work has shown that users can select poor passwords [47], however. We suggest that instead sets of objects be presented to the user, one of which can be altered by the user for use in her personalized authentication scheme. Below is our suggestion for registration, followed by the example as implemented.

The user selects a set of ten objects, then performs an ordered selection with replacement (e.g., 4-6 objects) for authentication purposes. These objects are automatically inserted in an Inform story which is coded to end the story in victory if the correct objects are stacked in the correct order. The selection of objects in a bin potentially represents real-world objects that act as reminders of memories. The reason for selecting 10 objects to be used in the authentication story is to have the same theoretical password space as 4 digit PINs.

The Inform instance has all the objects in all four bins, and allows the user to take only one object from each bin, and place it on the stack of selected objects. The bins each contain the complete set of objects users selected in the setup phase. This Inform story is then compiled and run inside Dumb Frotz, which is wrapped in a custom Python script.

In this example, the user has selected 10 objects (e.g., Banana, Fork, Coffeepot, Stapler, Headphones, Pencil, Keyboard, Jar, Mug, Ruler) as elements to be used in the authentication story. Four of these objects (e.g., Coffeepot, Fork, Headphones, Banana) are selected in order to be used as the password. Once this registration phase is over, the user can authenticate using this password.

5.1.1 Authentication Phase

The user is able to stack items by selecting one from each bin in the room. Once the user has selected and stacked an item from each bin, she is free to leave the room, and the system makes a pass/fail decision on the authentication "credentials" provided by the stack of objects.

Sequences of three incorrect authentication attempts trigger the consequence of a three-strike rule. This can be implemented, for example, as a time-based lockout, forcing a CAPTCHA [36] to be solved on subsequent attempts until attempts succeed or stop for a sufficient period of time, or permanent lockout until out-of-band resolution. An example of such a lockout might be to retain the card in an ABM, requiring a physical visit to the bank branch to rectify.

An example of a successful solution to the sample scheme is given in Figure 5.1. The correct order of stacking the objects is $\{\text{hammer}, \text{paintbrush}, \text{soap}, \text{cup}\}$. First, the user enters the command `inventory`, showing all of the carried items. Next, the objects are placed on the table in a stacked order by using the custom `stack` command. Once the stack contains the correct items in the correct order, the user is successful. If the incorrect object is placed on the stack, the user must quit and restart in order to login successfully.

```
  Stacker

  Stage
  You can see a table here.

>> inventory

  You are carrying:
    a bowl
    a hammer
    a ruler
    a paintbrush
    a soap
```

```
    a telephone
    a pair of scissors
    a cup
    a towel
    a camera

>> stack hammer

  You put the hammer on the table.

>> stack paintbrush

  You put the paintbrush on the hammer.

>> stack soap

  You put the soap on the paintbrush.

>> stack cup

  You put the cup on the soap.

  As you drop the last item down, you notice a small crease in the
  wall.   Pushing on it, you find a small doorway, and walk through...

      *** You have won ***
```

Figure 5.1. An example of a successful authentication attempt for a specific instance of the sample scheme scenario of Proposal I.

Informal testing of the time taken to correctly authenticate with this prototype revealed that the author took on average 11.1 seconds, while a non-expert third party unfamiliar with this thesis (but trained how to use the system) took on average 14.2 seconds for the same four-object stack. Instructing this user on how to log in correctly took approximately $5-10$ minutes. Creating the text adventure (by the author) took approximately 1 hour, including the programming of the scheme and creation of the ten objects. We attribute some of the creation time in each proposal to learning the syntax and technique of writing in Inform. These text adventure passwords, while sharing the same theoretical password size of four digit PINs, clearly involve significantly more time to authenticate and register than PINs.

5.2 Theoretical Password Space Analysis

Traditional PINs use 4-6 digits, where the entropy of each digit of a *random* PIN is $\log_2(10) = 3.322$ bits. Our scheme uses the same theoretical password space, but allows for varied vocabulary. Users may enter a given action in different ways that are acceptable to the interpreter, allowing users to remember the essence of the action to be typed rather than a strict string of characters.

Each selection bin used for object stacking works like the set of digits available to a PIN. Each bin contains the entire object set, and the user is permitted to take only one to add to their object stack. This results in a password space identical in size to PINs of the same length: 10^n, where n is the length of the PIN or number of bins. In addition, though the theoretical space may be identical, passwords may be more memorable than PINs, due to the personalization of the objects used. This conjecture, however, has not been tested.

Just like PIN-based mechanisms, users are subject to phishing, brute-force and special knowledge attacks. Direct guessing attacks require more effort to conduct guesses based on the user information collected, as they must identify the correct objects to stack with amongst the other objects the user has selected to be in the room. Potentially all objects in the authentication system have memorability to the user. This differs from attacks on PINs, as the standard attacks based on user knowledge involve sifting out the PINs that involve using special numbers such as important dates—here, attackers need to discern objects (e.g., those the user selected from her desk) instead of more publicly known dates of milestones (e.g., anniversaries and birthdays).

Brute-force attacks are just as effective as with traditional PINs, though in order to automate such an online attack, the commands a user would type would be terminated

by carriage-returns, and queued to be sent after each response from the server (i.e., ready to be sent at each prompt a user would get). An attack done this way might be more awkward, but still possible. We recommend the use of a three-strike rule. The order in which objects are listed in the Inform instance can be considered a way of mapping back to standard PINs—that is, given our example, the banana could be considered digit 0, the fork digit 1, the coffeepot digit 2, and so on. Thus, the order of the objects when presented to the user should be randomized for each authentication attempt. This means that the user does not merely select, for example, the first, third, fourth and eight objects from the list of objects each time. This does not change the order in which the correct semantic objects must be selected by the user, merely the order in which they are presented to the user by the interface. The same semantic choice must be made each login instance, but the order or "indexing" of items might result in a different coding sent back to the system.

5.3 Adding Complexity and Obfuscation

To provide protection against attacks made by shoulder surfing or keylogging (where the attacker merely replays whatever the recorded user had entered in order to break in to the account), the system can make use of obfuscation. These techniques can either involve some one-time pattern, or be changed based on some quantity: the date or some information given to the user by the authentication system in the virtual environment. Obfuscation can be used to protect the user's correct pattern from a small number of these surveillance attempts. The modifications to Proposal I for both of these obfuscation techniques are given below.

Other approaches to adding vocabulary, such as Clark and Hengartner's work on Panic Passwords [12], are restricted to the confined entry methods of PIN and pass-

word systems. Our Proposal I is limited to work within a theoretical password space similar to PINs. We use a signalling system much like that proposed by Cheswick [8]. This allows users to create chains of messages and indirection between these messages. Objects already in use in the authentication system chosen by a user can be made to represent the following signals. We have not pursued any testing of this variation.

No-op: A signal that means nothing

Ignoring/skipping next *n* objects in stack: Can be used to ignore further authentication objects, or signals that occur afterwards.

Reversing the meaning of next object: The object representing this signal means that the next object signal is to be negated in meaning.

Compound objects: Meaning is based on multiple objects in a series, rather than individual items. These objects may have meaning individually, but as a series take on a separate signal.

"Pump system:" The number of items is important, not the objects themselves—this can be used with a prior object to initiate the pump system and can be used with compound objects for denoting numbers.

Wipe off: An object that cancels all past messages, clearing the state completely.

These types of actions can be used to add a one-time password use case to this proposal, as the user can make use of signals in addition to the selected static authentication objects to create new authentication attempts each time. This can allow for obfuscated messages to be sent to the authentication system, potentially notifying the system of some emergency state. A one-time obfuscation operation can be generated for the user while she is logged in. The user can write the operation down (i.e., which

objects to insert into her normal object stack, and where), in order to use it later when logging in from an insecure location suspected of keyloggers. The obfuscation technique will not be accepted more than once. We feel that creating such a "travel password" is a useful addition to this authentication system. The second technique, using a changing value, requires the user to create some obfuscation algorithm which involves a changing quantity (either a known value such as the date, or some changing values inside the environment descriptions). This gives the benefit of a PIN-like authentication sequence with added security while at untrustworthy machines, but at the deficit of a much more complex authentication attempt.

This proposal can be adapted to provide panic passwords [12] with a complete separate second password, but also with obfuscation of the true authentication pattern, strengthening resilience to surveillance-based attacks. These techniques expand the use of Proposal I to places not covered by the original threat model. While these obfuscation techniques may require more objects to be selected for the environment, they do not , however, add new software components to the implementation of the system; the authentication story written in Inform is expanded to include these techniques by the user or developer. In this modified version of Proposal I, the system is always "obfuscation enabled."

5.4 Variations

The key component to this proposal is the ordered selection of objects from a set defined by the user, who may be locally or remotely participating in the text adventure that is running on the server. That means that the objects (even the whole text adventure) used for login are stored on the server for each user. The sample scheme given works similarly to PINs, and thus makes use of a set of ten objects, but this can

be made to be any arbitrary value. This act of stacking the objects is a reasonably well understood type of action that requires an ordered selection. Any alternative scheme designed to fit within the same threat model and theoretical password space requires the creation of a scheme where each choice has k outcomes, resulting in a space of k^n.

Chapter 6

Proposal II: Password-Level Authentication

Password-based authentication schemes have the portability of not requiring a user to carry a memory aid, and offer a potentially higher level of security than PIN-based systems. This proposal aims to provide a similar level of security as passwords, with the representative example affording a theoretical password space on par with passwords with lengths of eight characters. This proposal offers a richer language for communication, as well as leveraging the the ability for the user to make potentially memorable selections. The rich dialogue possible can allow users to create passwords with greater personalization than alphanumeric *user-chosen* passwords. The claim of improved usability, however, is not tested in this thesis.

Our strategy is to create a password space involving the same degree of choice as passwords based on the ASCII character set. As discussed in Section 3.3.1, Inform allows for actions and scenarios not originally anticipated by the platform developers through an extensible framework. This provides the basis for creating the actions required to create a password space such as the example system described below.

6.1 Representative Example of Proposal II

Proposal II is designed to mirror the types of actions found in most text adventure games: navigating through rooms and collecting objects. A given password scheme can be considered to have a certain theoretical password size, which can be measured as the number of bits required to count all password combinations. The binary representation describing choices made in a text adventure-based authentication scheme—the path and objects acquired by a user in a given instance—can be the same number of bits in length, as it encodes the selections of many choice points. Consider the following simplified example:

A user always begins in the same room of a grid-based world. One way to record the path a user selects between two rooms is to store each choice of north, south, east or west on such a path as a two-bit value, and concatenate these selections into one binary string. An example of this process is given in Figure 6.1. If the starting point is fixed, we can know where the user ends up after a walk, as well as all the rooms visited, in order from this binary string.

By elaborating on the description of each room into a simple one-sentence description, users can begin to grow accustomed to seeing the same sort of rooms on each login attempt. An example of such a description is given in Figure 6.2. When the wrong direction is taken, the room descriptions are new and foreign to a user, providing implicit feedback [10] instead of simply notifying a user that the login attempt has failed. Moreover, users can be encouraged to customize the scheme to contain mementos to assist room navigation. An example of this is a user inserting objects found in her house in the authentication path's rooms used. An attacker will be unable to differentiate these objects from the unrelated ones, unless this attacker has been to the user's house (in which case he may already have physical access to her

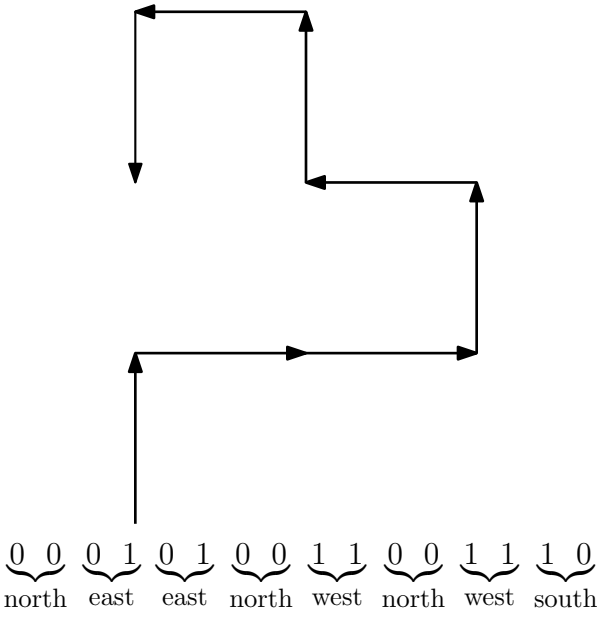

Figure 6.1. Visual and binary representation of a walk along the four cardinal directions.

machine, which provides different avenues of attack). A single sentence of description also keeps the necessary dialogue to a minimum, allowing for quicker authentication attempts by more technically proficient users who make use of the short form commands. Discussion of the available short form commands enabled by default is contained in Section 6.1.2.

The possible options a user can take can be increased, which requires a larger number of bits needed to store the direction chosen. Including the options to move up and down, as well as recording objects removed from the room before exiting extend the encoding to more than two bits per room.

The added task of selecting an object from each room by the user on each login attempt increases the password space of the system, as the attempts must tour the correct subset of rooms in the correct order, retrieve the correct items from each room and deposit the moved items in the correct locations. A simplified subset of problems

fitting such a description has users select one object from each room in the tour, and depositing all these items in one location of the last room. The user can be permitted to drop the items at once or individually, as the order is ignored—unlike the order in which the objects were acquired. This is the example we will examine in the rest of this section, as well in Section 6.2, where a security analysis is detailed.

6.1.1 Registration Phase

Setup occurs during registration, and is comprised of selecting a path as well as the objects for use in the system. The user selects either her own rooms, or selects from a given panel of rooms already written, arranging them into a path, similar to what is found in Figure 6.1. Rooms not part of the authentication walk can be randomly generated and inserted by the system to provide the surrounding virtual three dimensional grid of some fixed size. For systems with lower security requirements, the walk can be restricted to two-dimensional movements only.

The objects in use are in two sets: those chosen by user in a correct password, and those included for periphery and greater security. Users select authentication objects from previously created pools of objects (a list of objects), or create their own using the language of Inform. We intend for this set of authentication objects to be approximately fifteen, depending on the system security requirements. The reason for this is explained in Section 6.2.

All objects and rooms, including the ones not used for authentication purposes, are static—i.e., the system is not randomly generated for each instance. The constraint of forcing the user to invent or select objects not used for authentication is arguably too taxing and risks exhausting user memory capacity on non-imperative tasks, so we offer the alternative that the peripheral objects be randomly inserted by the system. Care must be taken to ensure that objects are not easily found to be similar to those

```
Kitchen
You step into a sunny kitchen, painted a bright yellow and filled
with oak cabinetry as well as a large counter top made of granite.

You can see a lemon, a bowl, a cheese grater, a dishcloth, a green
pepper, a quarter, a vase, an oak cabinet (empty), a ginger, a phone,
a melon baller, some keys, a frying pan, a toaster, a pestle and a
granite counter top here.
```

Figure 6.2. An example room description for Proposal II

selected by the user: confusion and distraction issues may present themselves in the system. An example might be a user including a glass vase as an object as part of her password set, and the system including a ceramic vase as periphery: the objects, while different in the Inform code, can be too similar for a user. This approach of random object insertion is beyond the scope of the thesis.

In order to improve the typing accuracy for selecting the correct object, items in a given room can be selected to each begin with a unique letter of the alphabet. Consider the room description in Figure 6.2: each object is listed in alphabetical order, and no objects start with the same letter. Users can grow accustomed to the positions of their authentication objects in the listings of room items. The use of shortcuts for rapid entry is discussed below.

6.1.2 Authentication Phase

A user authenticating with an instance of the system begins in the first room of the walk previously defined during password registration. The prompt available to her is a full-featured Frotz environment, running the compiled Inform story that forms the basis for the authentication. As an example of this environment, Figure 6.3 gives the opening of a complete implementation of the sample scheme of Proposal II. Every room not on the boundaries of the environment has doorways or staircases permitting

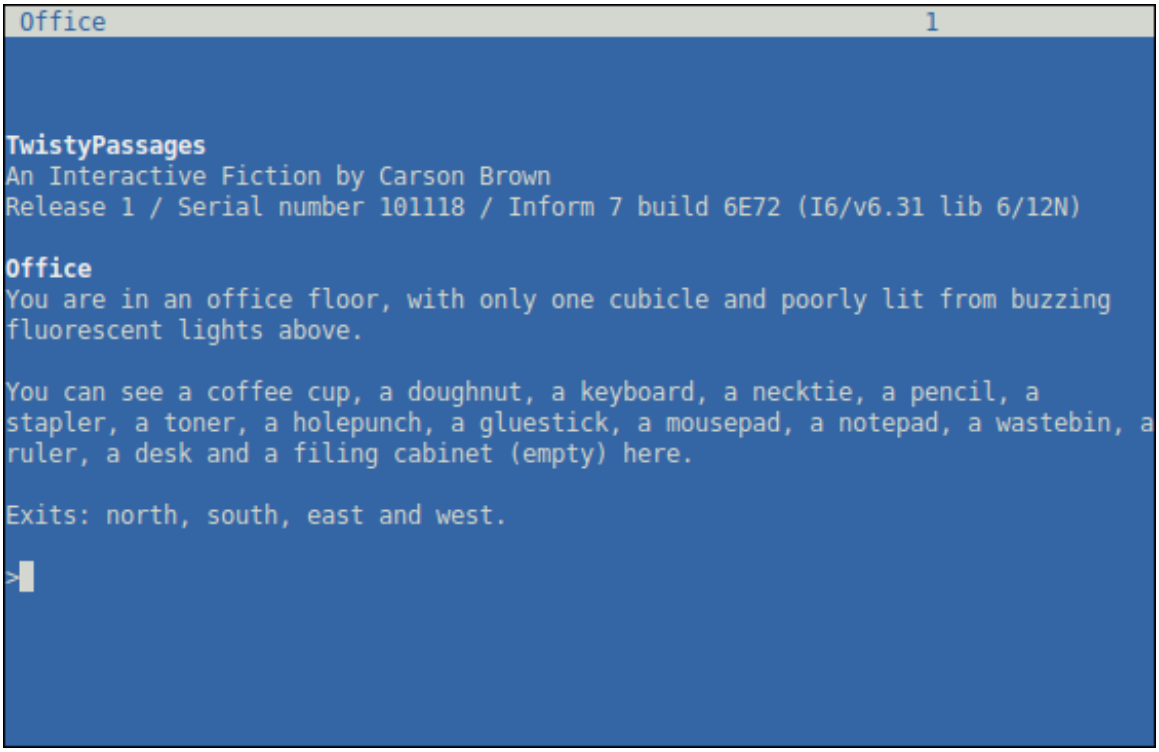

Figure 6.3. The opening text of a running sample scheme implementation of Proposal II in the Frotz interpreter.

movement in each of the directions discussed above.

The user navigates through the rooms, collecting the correct objects from each room as necessary. In this implementation, the user drops all the items in one location in the final room of the walk, and the authentication session terminates. The system verifies the correctness of the tour and the container used to hold the dropped objects is checked to contain only the correct set of objects. The user is granted access only if these criteria are correct, otherwise the user is invited to try again with a new login attempt, assuming no three-strikes rule has been triggered.

Another variant allows deviation from the correct walk provided that the user returns to the correct path. This is an approach we have described for consideration, but is not used in our implemented sample scheme. The added benefit is that users who correctly return to the correct path after immediately selecting the wrong direction can still log in successfully without the penalty of starting over again. This differs from diverging from the correct path but ending up in the correct rooms with the correct objects in a different order.

Short form commands permit expert users to authenticate faster. Assuming care was taken to avoid the case of multiple objects in a room to start with the same letter, a shortcut command can abbreviate the object by its first letter, instead of typing `take apple`, a user can simply enter `t a`, and the Python wrapper can convert this to meaningful Inform command. An example of the user's side of an authentication attempt is given in Figure 6.4. Figure 6.4a gives the full text; Figure 6.4b contains short-form commands, reducing the total number of characters typed.

On average, our testing of the time to authenticate by typing in the full text took 51.5 seconds, while testing by a non-expert user took 62.4 seconds. Instructing the user on how to authenticate with the prototype took $10 - 15$ minutes. Creating the authentication system took approximately $2 - 3$ hours, along with another hour to

```
go north
take apple
go east
go east
take cereal
go north
take bread
go west
take pear
go north
take granola
go west
go south
empty inventory on table
```

(a) Full text

```
n
t a
e
e
t c
n
t b
w
t p
n
t g
w
s
e t
```

(b) Short-form commands

Figure 6.4. A sample transcript of a user authenticating in an instance of Proposal II. Full and short-form versions are given.

invent all of the necessary objects.

Comparison to Proposal I: PIN-based authentication can be considered a simplified version of a full text password system. For this reason Proposals I and II may be viewed as somewhat similar. The difference is that for Proposal II, the path selected by the user, and the objects taken, are relevant, including object order. Proposal I involves only the order in which objects are selected for stacking.

Comparison to Eight-Character passwords: Each round of tasks the user completes—picking an object and a direction to leave the room—yields approximately the same number of possibilities (translating into an equivalent number of bits of theoretical choice space of each character) as the ASCII character set. The goal of the added complexity and registration effort for Proposal II is to limit the size of weak passwords the user can create. A user creating a standard text password often makes use of words that might be found in an attacker's dictionary, but a user creating her

own text adventure carrying personal significance raises the bar for an attacker to create a custom text adventure "dictionary". It may be difficult for an attacker to create a dictionary that reduces his search space if there are no common user choices. We discuss the feasibility of such dictionary attacks in Chapter 8.

6.2 Theoretical Password Space Analysis

Many websites require passwords of at least eight characters and enforce certain restrictions on the content of these passwords. ASCII characters have an upper bound of 6.570 bits of entropy, which amounts to approximately 52 bits of theoretical password space for a password of eight characters. In reality, users choice results in far less entropy, e.g., previously estimated by NIST to have 18 bits of entropy [7], which more recently has been shown to be an over-estimate [47]. A level of 6.570 bits of theoretical password space per decision is replicated in Proposal II, and can be constrained by password policy and extended to a desired password length by changing the number of rounds. The cost, however, is in usability, memorability and time involved.

To compare this with the password space of text passwords, consider how the user's authentication attempt can be stored as a bit string. One can store the object taken from a room as the binary encoding of 1 to 15, and include 0 as the encoding of no object removed. The direction taken in leaving the room can be stored as a three-bit integer, allowing for a possible eight different directions. This provides 7 bits of password space for each room, vs. 6.57 for one character in an ASCII password.

Our current implementation only uses six possible values for direction—north, south, east, west, up and down, the remaining values are unused—technically only providing 2.25 bits of password space rather than the full 3 bits reserved for each

direction choice, and thus provides only 6.25 bits of choice for each room's turn, compared to the 6.570 bits per character in ASCII.

The placement of items in the final room is the last step of authentication, and can be stored in as many bits as necessary to store the number of Inform container-type objects in the room. The prototype implementation has 3 different locations where items can be dropped: the floor of the room and two supporter or container objects. This yields 1.5 bits of choice and every room has this number of locations, not just the user's final room.

6.3 Variations

Proposal II builds complexity upon the first proposal's approach by adding another level of ordered selections. The sample scheme of Section 6.1 required correct room navigation in order, and correct items selected from each room. The final room's selection defines where the collected items are to be deposited, but this can be considered to be no different from selecting one more item from a possible set. These decisions are to create an understood framework for users to authenticate. Variations of comparable security need only create a system that has a similar degree of choice per step. Adding steps is akin to requiring longer passwords, and adding more choices within each step is akin to using allowing a richer character set. Thus, in general, each decision (or group of decisions that work as a "turn" during authentication) should involve $\approx 2^6$ theoretical outcomes, and such a system can have k decisions in order to provide a password space comparable to a password of length k with the ASCII character set.

Chapter 7

Proposal III: Token-Level Authentication

A major hurdle to overcome when considering authentication as dialogue is the need to enforce novel paths and events of conversation. Otherwise, the dialogue could merely be interpreted as a lengthy, drawn out passphrase, subject to the same replay attacks of password-based schemes. In fact, the concept of authentication as dialogue is a superset of password techniques, as password-based authentication can be presented in a dialogue format. This constraint can be enforced by the system randomly generating portions of the authentication puzzle, as explained below, thus providing the opportunity for users to behave differently in each instance.

This proposal combines some security benefits of one-time password authentication while maintaining password portability and convenience. A user makes use of eight coloured labels from the user inventory, attaching them to the correct authentication objects available in the instance—one object to each label. In addition, a number of objects from the non-authentication pool are present to fill in the environment as periphery to the user and to prevent simple brute-force attacks. By having

the server vary which objects are selected for the user to attach labels on, the attacker is unable to leverage a small set of observed authentication attempts, forcing longer term, more time-expensive surveillance attempts.

7.1 Representative Example of Proposal III

7.1.1 Registration Phase

The user must first set up a story and environment for authentication. The prototype implementation uses two pools of objects: an authentication pool and a non-authentication pool. A subset of both pools (randomly selected within constraints) appear in any single authentication instance. That is, one object is randomly selected from each pool of five objects for each label, then a set of non-authentication objects are selected, bringing the total to 95 objects.

Users may create and describe objects for insertion into the system, or select from a toolbox of already created objects. In the future, such a toolbox might be a social collaboration of all users of a similar given authentication system. Another alternative creation strategy allows the user to select packages of objects, where each contains a number of objects for both pools. In this way, objects designed for the authentication pool blend better into the backdrop of non-authentication objects, and don't appear strangely out of place or easy to spot when viewed by an attacker. The intention here is to encourage users to form non-intuitive mappings to objects in the authentication system to real-world mementos to enhance memory recognition, and also to enhance the security of the system against guessing attacks based on knowledge of user-specific information. Once the object pools are created, random instances of the scenario can be created for use, described below.

7.1.2 Creation of an Instance by System

Our prototype scheme is capable of selecting objects from the pools, inserting them into the Inform story source code, and compiling the story to be run. The user then interacts with the system like any other interactive fiction (see Chapter 3). The goal here, though, is to interact with the story in the "correct" order to be granted access to whatever the authentication system is guarding.

One object is selected by the system to correspond to each label and a number of non-authentication objects are added. The total number of objects in a given instance is 95. The selected objects differ between instances, changing as each pool contains many objects. Once the user has completed an authentication attempt (by finishing the tasks, failing to complete within a maximum time period, or abandoning the session), the system decides whether or not to grant access.

7.1.3 Authentication Phase

The objects selected for a given instance of the scheme are placed in a room, which contains the type of objects one would expect in a small house or apartment. The user is permitted to explore and move objects in the text adventure environment. To complete the puzzle, the user attaches the labels in her inventory to the correct objects in the given instance using the action `attach`. Once all labels are attached, she is permitted leave the house through the front door. The system then checks to see if the labels are correct, and ends the game in one of two ways: if she was correct, the game ends in victory (access granted), if incorrect, the game ends in death[1].

An example of a successful solution to an instance of the sample scheme is given in Figure 7.1. The set of items selected for a given instance are placed in various locations

[1] An homage to the classic text adventures.

throughout the environment, and placed within the static fixtures standard in every instance. Because the text adventure is wrapped inside a Python script, enhancements can be provided, such as allowing short forms of longer commands. Instead of typing `stick the red label to the dresser`, one can type `r dresser` and the prototpe will convert to a format the interactive fiction interpreter understands. The shortened text is certainly more cryptic, but allows, "expert" users to log in faster. The full interpreter is always available if short forms are not preferred. Some common text adventure short forms are made available by Inform as well, such as `i` as the short form for `inventory`, which displays all the items the user is carrying.

Recorded authentication times in our informal testing were on average 68.7 seconds, and our non-expert user took on average 87.6 seconds. Both tests made use of the full login text, and used no shortcut commands. Instructing the non-expert user on how to login correctly took approximately $15 - 30$ minutes. Creating the authentication system took approximately $3 - 5$ hours, along with an additional 2 hours to create the necessary objects (both pools).

```
  AuthAdventure

  Mysterious Room
  You find yourself in a large room, containing items that might be
  normally found in a house. There is only one exit, a large door on
  the south wall.

  You can see a bookcase, a dresser, a bed, a picture frame, a sink
  (empty), a toilet, a bath (empty), a medicine cabinet (closed), a
  counter, a refrigerator (closed), a sink (empty) a stove, an oven
  (closed) and a door here.

>> i

  You are carrying:
    a red sticky label
    a blue sticky label
    a green sticky label
    a yellow sticky label
    a purple sticky label
    an orange sticky label
```

```
    a black sticky label
    a white sticky label

> > x dresser

  The plain wooden dresser has top, middle and bottom drawers.

> > r dresser

  You stick the red sticky label to the dresser.

> > x dresser

  The plain wooden dresser has top, middle and bottom drawers. A bright
  red sticky label is attached to the dresser!

> > open top drawer

  You open the top drawer, revealing a pair of socks.

> > open middle drawer

  You open the middle drawer, revealing a t-shirt.

> > b t-shirt

  You stick the blue sticky label to the t-shirt.

> > open bottom drawer

  You open the bottom drawer, revealing a pair of jeans.

> > g jeans

  You stick the green sticky label to the pair of jeans.

> > y picture frame

  You stick the yellow sticky label to the picture frame.

> > x bath

  The bath is empty.

> > p bath

  You stick the purple sticky label to the bath.

> > o toilet

  You stick the orange sticky label to the toilet.
```

```
> > open cabinet

  You open the medicine cabinet, revealing mouthwash, toothpaste and a
  toothbrush.

> > open fridge

  You open the refrigerator. It is empty.

> > bk fridge

  You stick the black sticky label to the refrigerator.

> > open oven

  You open the oven, revealing a cake.

> > w oven

  You stick the white sticky label to the oven.

> > i

  You are carrying nothing.

> > s

  You try to open the large door, and find it easily opens. Stepping
  through the doorway you feel a strange sensation....

      *** You have won ***
```

Figure 7.1. An example of a successful authentication attempt for a specific instance of the sample scheme scenario of Proposal III.

7.2 Theoretical Password Space Analysis

This proposal is intended to bring some of the benefits of token-based authentication—such as resilience to brute-force attacks—while maintaining the memorability and portability of password-based authentication. Consequently, we compare its security to traditional password authentication.

The size of the space of all ordinary 8-character text passwords using any ASCII printable characters is 95^8 ($2^{52.44}$). The effective password space is much smaller—users do not make use of the entire set of permutations [47]. The upper bound of the size of the space of all possible login attempts *in one authentication instance* of Proposal III is set, in our prototype implementation, to be comparable to this value 95^8, to calibrate comparison based on security level. Our system can be expanded to a larger size with more items in the object pools and by allowing the user to select more items for the authentication step if necessary. Allowing 95 items with repetition, the theoretical password space size matches 95^8. Disallowing repeated object selection, this drops to $95 \times 94 \times \cdots \times 89 \times 88$ ($2^{52.12}$), still the same order of magnitude as 95^8.

We consider the authentication object pool to be a superset of the sets of objects reserved for each coloured label. In our prototype example, any one of five objects can be selected for one label, and these eight label sets may have an optional overlap; a user must be able to recognize ≤ 40 authentication objects across all of her authentication instances. It is at most 40 instead of exactly 40 because the user may have selected the same object to be used for more than one label. There exist only 5^8 ($2^{18.58}$) correct authentication possibilities—exponentially smaller than the full theoretical password space. At least 87 non-authentication objects[2] are present in a given instance to give comparability to the set of 8-character ASCII passwords, more if selection with replacement of authentication objects occurs. One must note that if the sets of objects designated for each label are not distinct, there may exist more than one solution for a given instance. The implication of this is the user may not be aware that an attacker attempting to break in has higher chance of guessing the correct value (e.g., 2 in 95^8 instead of 1 in 95^8).

[2] 95 objects exist in a given instance of the sample scheme and ≤ 8 are used for authentication, depending on the number of shared objects between multiple labels.

An attacker who has recorded and discovered each of the authentication objects in a given session must still record further authentication attempts in order to successfully authenticate as the user. This is because subsequent instances are expected to contain different objects randomly selected from the object pools.

In order to simplify the calculation of the resistance to attacker surveillance through keyloggers and screen capturing software installed on the machine used, we consider the case where authentication pools are independent of one another, i.e., no object can be selected to satisfy two labels in any instance. Table 7.1 lists two probabilities given the number of successful authentication attempts witnessed: the probability that the attacker can correctly select an object to satisfy one label, and the probability that the attacker can correctly select all the objects to satisfy one label. These are the probabilities that the attacker has sufficient knowledge to solve an instance, given that a discrete number of past successful user authentication attempts were witnessed. This is known as an "intersection attack" by Dunphy et al. [15], where the aggregate of multiple witnessed authentication allows attackers to narrow down a list of candidate passwords.

An attacker who has never witnessed a user's authentication can't know any of the 5 objects the user selected for a given label (assuming no guessing), but an attacker who has seen a single login certainly knows one of the objects. Table 7.1 reflects this in the probability that the attacker knows the object to select in a later authentication instance; there is a $\frac{1}{5}$ chance that the only object the attacker is familiar with for that label was randomly selected again. Assuming that each label is an independent random selection, the probability that the attacker knows the solution for the whole instance (all eight labels) is $\frac{1}{5}^8 = 2.56 \times 10^{-6}$.

Consider the case where the attacker does *not* know the object for a given label based on n previous witnessed logins by the user. The probability that the object

Number of user authentications witnessed	Probability of knowing object for one label	Probability of knowing solution for instance
1	$\frac{1}{5} = 0.2$	2.56×10^{-6}
2	$\frac{9}{25} = 0.36$	2.82×10^{-4}
3	$\frac{61}{125} = 0.488$	3.22×10^{-3}
4	$\frac{369}{625} = 0.5904$	1.48×10^{-2}

Table 7.1. Probability breakdown of an attacker's gain in knowledge of the authentication pool of objects, by surveillance without guessing.

selected in a previous login is not the object selected is $\frac{4}{5}$. Each instance is an independent event, thus the probability that the object in the instance an attacker is trying to solve was not in the n previous user logins the attacker has witnessed is $(\frac{4}{5})^n$. In the inverse case, the probability that the object was in any of the n witnessed user logins is $1 - (\frac{4}{5})^n$. The probabilities in Table 7.1 are calculated in the above manner.

Though the attacker has a reasonably high probability—significantly better than a random guess—of knowing the object for a given label in an instance by witnessing multiple authentications, the probability of a successful attempt across all labels remains low with only a handful of captured logins. Multiple failed authentication attempts can result in some form of account locking, e.g., a three-strike rule.

The memorization of 40 objects appears to be a high number. These items are intended to have personal meaning, i.e., the items used in morning routines, types of clothing, etc. Such collections for most people are a similar order of magnitude in size. We have attempted to preserve Miller's Law of working memory [31], that the average person can hold 7 ± 2 objects in working memory, as each coloured label is a smaller chunk of memory (corresponding to at most 5 objects), and there exist

eight labels. Work beyond the scope of this thesis is testing this hypothesis with user studies that compare the the memorability of different scenario types.

7.2.1 Analysis Remarks

Once successful attacks require analysis on a per-user basis, the attack cost precludes large-scale break-in attempts, as witnessing enough successful changing and obfuscated logins is infeasible over a reasonable duration of months or more.

A goal of Proposal III is to equip users with the ability to authenticate safely in locations where security is compromised. This is an ability beyond the grasp of standard text passwords, and has been hinted at by Cheswick's obfuscation proposal [8] as a useful feature. Although the effectiveness of a non-expert users' strategies remain untested, the requirements of learning a programming language (cf. [8]), understanding cryptography (cf. [23]) and carrying password list memory aids (cf. [13]) have been lifted.

Chapter 8

Comparison & Discussion

Using dialogue for authentication purposes can have flexible security benefits—more user interactions with a more complex, dynamic world gives a larger password space. This dialling up of security is shown in the three example proposals: as the security constraints tighten, the user involvement is increased.

The extra user involvement, especially in Proposal III, is designed to be used to strengthen traditional authentication schemes that are appropriate in low-risk environments (i.e., a trusted computer with little risk of malware), but are not suitable in high-risk situations (i.e., an untrusted computer that may have malware installed). Resistance to keyloggers can effectively prevent an attacker from gaining access, as attackers have a small number of authentication attempts to work from before the user leaves the compromised location or discontinues use of that given scheme.

The proposals themselves provide plausible example solutions for addressing given threat model scenarios. These exemplars make use of the common characteristics found in most text adventures: collecting objects, solving navigation puzzles by moving through rooms, and performing actions on objects. Though not a conventional conversation, this type of authentication offers a rich dialogue for communication

between user and authentication system, where additional information aside from traditional credentials may be used. This information can be embedded inside text adventures adapted from user memories or personal mementos.

The measured login times for each proposal can be improved by allowing short forms of the user commands, or by designing a scheme that requires less user-entered text. Short form commands reduce the number of keystrokes, but require the user to remember them. A scheme that by design requires less text to be entered must still make the correct set of actions difficult for an attacker to guess, i.e., increasing the set of all possible actions at any round to be much larger than the set of all ASCII characters. Such alternatives are beyond the scope of this thesis.

User choice is an issue if users make use of the same scheme, that is, weak password subspaces become apparent if users select the same sorts of interactions with the text adventure. This allows attackers to construct dictionaries in order to shrink the attack space down from an exhaustive search. Our intention of using modern text adventure tools like Inform is that users can modify and create text adventures themselves, hopefully having the property that the resulting passwords avoid the typical skewed distributions which make weak subspaces exploitable by attackers. We believe that this is a better alternative than assigning text adventures and their solutions to users in order to guarantee a certain level of entropy, as such a process would have user memorability issues.

Password interference exists for a single user making use of similar or exactly the same text adventures across multiple accounts. Thus, for security reasons, we recommend (albeit at the expense of usability, potentially) the modification and customization of text adventures in use. The effort required by a user to create a text adventure can be mitigated somewhat by the hundreds of available modern text adventures: users can take a portion or a single puzzle—a contained section of a game

that requires actions to happen in the correct manner—from one of these works and use it as their authentication text adventure, which translates to thousands of text adventure passwords. Interactive fiction databases containing thousands of titles, such as the IFDB (`http://ifdb.tads.org/`) are available online. While this may save users from inventing text adventures, it doesn't necessarily reduce the amount of effort or time involved to create their password.

Phishing a user who has a text adventure as her authentication system requires one of two strategies: (i) in the case of a system that has a single static solution, the text adventure must be accurately emulated by the attacker in order to fool the user into believing she is interacting with the true system, or (ii) initiating a complete interception attack between the user and the authentication system (man-in-the-middle) and relaying messages to both sides of the conversation. Though both types of attacks are possible, there is more effort required than traditional password phishing schemes of simply faking the appearance of the trusted authentication system.

A summary comparison between our example schemes and related work is given in Table 8.1. The size of theoretical password spaces are ranked according to three categories: PIN-level (○), text password-level (◑) and cryptographic-level (●), along with the size of the theoretical password space (*i.d.* designates that insufficient detail is available in published work). It is important to note that the *effective* password space of each scheme of this thesis is unknown. The second row indicates if schemes have varying passwords across each authentication instance. Systems that make use of implicit feedback [10] are indicated in the third row. A scheme's resistance to phishing falls into one of three possible values: systems that can be phished without the attacker having to replicate any correct server cues or responses (◇), systems where an attacker acquires these cues in one probe (◈), and systems that require an attacker to acquire these cues over multiple probes or conduct a man-in-the-middle

Property \ Scheme	Proposal I	Proposal II	Proposal III	Text Passwords (8 characters)	PCCP [9]	3D Passwords [2]	Bard's Pass-phrases [5]
Theoretical Pswd Space	○ $2^{13.29}$	◑ 2^{50}	◑ $(2^{52.12})$ per instance	◑ $2^{52.12}$	◑ 2^{43}	◑ *i.d.*	● 2^{52} to 2^{112}
Varying Pswd			✓				
Uses Implicit Feedback		✓			✓		
Addressing Phishing	⬗	⬗	◆	◇	◆	⬗	◇
Addressing Keylogging and Shoulder Surfing	◇	◇	⬗	◇	◇	◇	◇

Table 8.1. Comparison of authentication schemes against the example proposal schemes.

interception attack (◆). Last, keylogging and shoulder-surfing attacks are grouped into the next row of the table, where systems do not provide protection against these attacks (◇), provide protection against more than one fully observed or recorded session (⬗), or provide complete protection against keylogging and shoulder-surfing (◆). Proposal III provides partial protection from attacker surveillance such as keylogging and shoulder-surfing because each instance is a random selection of objects from the user portfolio, as discussed in Chapter 7.

8.1 Discussion

Evaluation of the overall effective usability and user password choice of each of the proposals, beyond the minimal reporting of timings of the prototypes, is beyond the scope of this thesis. Nonetheless, we list some relevant questions here. User studies that measure these aspects must cover several areas.

Resilience to user choice issues: How resilient to attacks are user-created authentication scheme instances? While a given scenario may have a reasonably sized theoretical password space, we are interested in identifying the expected or effective password space, and to identify potential weak password subspaces due to users making common selections ("user choice issues"). For example, exploration is needed to measure protection against special knowledge attacks such as guessing based on user information found online [37].

Generation: Can users be persuaded into creating and using objects for secure authentication purposes? Given that a background in text adventures may be required in order for users to better understand how to use the authentication system, this question asks if users can be persuaded, much like the persuasion Forget et al. [19] accomplished, to accept system suggestions at registration time to improve text adventures used for authentication. Forget et al.'s study was designed to improve user-chosen text passwords by inserting or replacing characters with randomly selected ones; a study here must define how weak password subspaces can be minimized by modifying objects and actions in users' text adventures. An example of this might be to suggest a user of a system like Proposal I make use of a glass bottle instead of a wooden crate, because crates are found to be a more common user-selected object.

Customization: How memorable are prefabricated scenarios modified by users for personal authentication? Related to the testing of user-generated schemes above, this question focuses on users taking existing schemes and modifying them according to personal taste. The real-world scenario in mind is that these schemes can be created by experts and given as available options for users to modify. The memorability of these customized authentication schemes can be compared to the memorability of user-generated schemes.

Interference: Can a user correctly memorize solutions for text adventures protecting multiple personal accounts? This question explores the password interference between authentication schemes. We intend for users to make use of separate memories or scenarios for different accounts; user studies are require to explore the usability impact related to interference, and also usability impacts related to choice of distinct passwords on different systems.

8.2 Conclusions

Where traditional authentication schemes that require no assisting devices or memory aids offer memorized, fixed dialogue, in this thesis we have presented a meta system which offers an authentication scheme that provides a sort of dialogue between user and site. This different method of authentication has been shown to have similar security properties as text passwords by analysis of three plausible sample schemes corresponding to three common uses types of authentication. Proposal III offers a method for authenticating in a surveillance-resistant way.

The key component of the authentication step is an operation that is designed to be straight-forward for the user to complete, but difficult for the attacker to guess,

in some cases even with recorded successful attempts. We anticipate that this work has negative impacts on usability, in terms of time to create passwords and log in, but the actions required to log in may themselves be memorable. User choice issues are not explored, and may well be comparable to existing problems in text password systems for the proposals as currently presented.

The text-based virtual worlds for authentication are built with the tools used to create text adventures, a popular gaming platform of decades past with an ongoing online community of players and game writers. The core component of text adventures is the act of storytelling, which is the historical way of communicating with one another. Users who can create their own text adventure for authentication can leverage a rich password space an attacker would be unable to understand, instead of being restricted to inventing a text password that's hard for attackers to guess—but potentially also hard for the user to remember. Ease of login requires formal user study; however, based on the fact that text adventure games have communities of users, we feel that using text adventure concepts for authentication can yield systems that are at least usable for some subset of the population.

The three proposed examples of text adventures for authentication show a range of threat models this type of authentication can be designed to protect against, enriching the type of communication possible with the authentication server. Extensions to this work can, for example, provide more expressive obfuscation similar to the goals set out by Cheswick [8], add the ability for secure authentication while under duress [12] [5], or add another type of secret channel for communication between user and server.

Sophisticated communication in authentication schemes such as this thesis work require usability testing in order to find adequate calibration between user effort and the expected real-world security strength. This work can be considered to propose a new space of authentication schemes, where three particular points in the space are

exemplified. This space still has many more possibilities to explore.

Due to the syntactic flexibility accepted by the interpreters of interactive fiction, authentication schemes that create rich authentication spaces might be possible to create (albeit by experts). By creating rich dialogue, we increase the bandwidth between user and system, allowing for additional information. This bandwidth can be used to create systems that reduce the amount of effort requried by a user (less to type) in a scheme designed for rapid login. Other uses of the wider bandwidth involve creating rich spaces, allowing for creating secret or optional side-channels. These side-channels can convey information to a user without necessarily allowing a complete login, because the user authenticates in pieces or turns over the greater authentication event. Alternatively, the side-channel can convey information from the user to the system, such as panic situations [12] or an intended privilege the user wishes to use. Extensions such as side-channels to this thesis work can increase the utility of using text adventure concepts for authentication.

Another future direction in this line of research is to consider genres related to text adventures as authentication platforms. Graphical adventure games, such as the genre of point-and-click games mentioned in Section 3.1, better make use of human visual recall. Future research could measure the memorability of authentication schemes using graphics instead of only text. The hurdle of creation—involving the graphical assets and design of the 2D or 3D environment—is a challenge for such future work unless reasonable methods of automatic generation or simple creation tools for users are proposed.

References

[1] R. Adams. The Colossal Cave Adventure page. `http://www.rickadams.org/adventure` (accessed 18 August 2010).

[2] F. Alsulaiman and A. El Saddik. Three-Dimensional Password for More Secure Authentication. *IEEE Transactions on Instrumentation and Measurement*, 57(9):1929 –1938, 2008.

[3] J. Anderson and G. Bower. *Human Associative Memory: A Brief Edition*. Lawrence Erlbaum, 1980.

[4] K. Bailey, A. Kapadia, L. Vongsathorn, and S. W. Smith. TwoKind Authentication: Protecting Private Information in Untrustworthy Environments. In *WPES '08: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*, pages 39–44, New York, NY, USA, 2008. ACM.

[5] G. V. Bard. Spelling-Error Tolerant, Order-Independent Pass-Phrases via the Damerau-Levenshtein String-Edit Distance Metric. In *ACSW '07: Proceedings of the Fifth Australasian Symposium on ACSW Frontiers*, pages 117–124, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[6] R. Biddle, S. Chiasson, and P. van Oorschot. Graphical Passwords: Learning from the First Generation. Technical Report TR-09-09, Carleton University, 2009.

[7] W. E. Burr, D. F. Dodson, R. Perlner, W. T. Polk, S. Gupta, and E. Nabbus. NIST Special Publication 800-63-1 Electronic Authentication Guideline. Computer Security Division, Information Technology Laboratory, U.S. National Institute of Standards and Technology, April 2006. `http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf` (accessed 3 August 2010).

[8] W. Cheswick. Johnny Can Obfuscate; Beyond Mother's Maiden Name. In *HotSec'06: Proceedings of the 1st USENIX Workshop on Hot Topics in Security*, pages 31–36, August 2006.

[9] S. Chiasson, A. Forget, R. Biddle, and P. C. van Oorschot. Influencing Users Towards Better Passwords: Persuasive Cued Click-Points. In *BCS-HCI '08: Proceedings of the 22nd British HCI Group Annual Conference on People and Computers*, pages 121–130. British Computer Society, 2008.

[10] S. Chiasson, P. C. van Oorschot, and R. Biddle. Graphical Password Authentication Using Cued Click Points. In *Computer Security ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*, pages 359–374. Springer Berlin / Heidelberg, 2007.

[11] Chooseco. Choose Your Own Adventure - Publisher of Children's Books. `http://www.cyoa.com/` (accessed 21 September 2010).

[12] J. Clark and U. Hengartner. Panic Passwords: Authenticating under Duress. In *HotSec'08: Proceedings of the 3rd USENIX workshop on Hot Topics in Security*, page 8. USENIX Association, 2008.

[13] A. Conklin, G. Dietrich, and D. Walz. Password-Based Authentication: A System Perspective. *Hawaii International Conference on System Sciences*, 7:70170b, 2004.

[14] D. Davis, F. Monrose, and M. Reiter. On user choice in graphical password schemes. In *Proceedings of the 13th conference on USENIX Security Symposium*, page 11, 2004.

[15] P. Dunphy, A. P. Heiner, and N. Asokan. A Closer Look at Recognition-Based Graphical Passwords on Mobile Devices. In *SOUPS '10: Proceedings of the Sixth Symposium on Usable Privacy and Security*, pages 1–12, 2010.

[16] U. Feige, A. Fiat, and A. Shamir. Zero Knowledge Proofs of Identity. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 210–217, May 1987.

[17] D. Florêncio and C. Herley. A Large-Scale Study of Web Password Habits. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 657–666, 2007.

[18] D. Florêncio, C. Herley, and B. Coskun. Do Strong Web Passwords Accomplish Anything? In *HotSec'07: Proceedings of the 2nd USENIX workshop on Hot Topics in Security*, page 6. USENIX Association, 2007.

[19] A. Forget, S. Chiasson, P. C. van Oorschot, and R. Biddle. Improving Text Passwords Through Persuasion. In *SOUPS '08: Proceedings of the 4th Symposium On Usable Privacy and Security*, pages 1–12, 2008.

[20] P. Golle and D. Wagner. Cryptanalysis of a cognitive authentication scheme (extended abstract). In *IEEE Symposium on Security and Privacy (SP '07)*, pages 66 –70, May 2007.

[21] L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets From Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648 –656, June 1993.

[22] J. A. Haskett. Pass-Algorithms: a User Validation Scheme Based on Knowledge of Secret Algorithms. *Communications of the ACM*, 27(8):777–781, 1984.

[23] N. Hopper and M. Blum. Secure Human Identification Protocols. *Advances in Cryptology (ASIACRYPT 2001)*, pages 52–66, 2001.

[24] Infocom. History of Infocom. `http://www.infocom-if.org/company/company.html` (accessed 24 August 2010).

[25] ISO 9564-1:2002. *Banking – Personal Identification Number (PIN) Management and Security – Part 1: Basic Principles and Requirements for Online PIN Handling in ATM and POS Systems*. ISO, Geneva, Switzerland, 2002.

[26] E. Ju and C. Wagner. Personal Computer Adventure Games: Their Structure, Principles, and Applicability for Training. *SIGMIS Database*, 28:78–92, April 1997.

[27] D. Klein. Foiling the Cracker: A Survey of, and Improvements to, Password Security. In *Proceedings of the 2nd USENIX Security Workshop*, pages 5–14, 1990.

[28] P. Lebling, M. Blank, and T. Anderson. Special Feature Zork: A Computerized Fantasy Simulation Game. *Computer*, 12(4):51 –59, April 1979.

[29] J. F. Leversund. The Password Meta Policy. `http://securitynirvana.blogspot.com/2010/02/password-meta-policy.html` (accessed 19 October 2010).

[30] M. Mauldin. ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (vol. 1)*, pages 16–21, 1994.

[31] G. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63:81–97, 1956.

[32] A. G. Morgan. A Linux-PAM Page, 2008. `http://www.kernel.org/pub/linux/libs/pam/` (accessed 3 January 2011).

[33] R. Morris and K. Thompson. Password Security: A Case History. *Commun. ACM*, 22(11):594–597, 1979.

[34] G. Nelson. The Z-Machine Standards Document Version 1.0, 1997. `http://www.inform-fiction.org/zmachine/standards/z1point0/` (accessed 2 November 2010).

[35] P. Oechslin. Making A Faster Cryptanalytic Time-Memory Trade-Off. *Advances in Cryptology (CRYPTO 2003)*, pages 617–630, 2003.

[36] B. Pinkas and T. Sander. Securing Passwords Against Dictionary Attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 161–170, 2002.

[37] A. Rabkin. Personal Knowledge Questions for Fallback Authentication: Security Questions in the Era of Facebook. In *SOUPS '08: Proceedings of the 4th*

symposium on Usable privacy and security, pages 13–23, New York, NY, USA, 2008. ACM.

[38] J. Saltzer and M. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278 – 1308, September 1975.

[39] R. C. Schank. *Conceptual Information Processing*. Elsevier Science Inc., 1975.

[40] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, July 1948.

[41] Y. Spector and J. Ginzberg. Pass-sentence—a new approach to computer code. *Computer Security*, 13(2):145–160, 1994.

[42] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pages 635–647, 2009.

[43] A. M. Turing. Computing Machinery and Intelligence. *Mind*, 59(236):433–460, 1950.

[44] P. van Oorschot and J. Thorpe. Exploiting Predictability in Click-Based Graphical Passwords. *Journal of Computer Security*, 2010 (to appear).

[45] P. C. van Oorschot and J. Thorpe. On Predictive Models and User-Drawn Graphical Passwords. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1–33, 2008.

[46] D. Weinshall. Cognitive Authentication Schemes Safe Against Spyware. pages 295–300, May 2006.

[47] M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, pages 162–175, 2010.

[48] J. Weizenbaum. ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Commun. ACM*, 9(1):36–45, 1966.

[49] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon. PassPoints: Design and Longitudinal Evaluation of a Graphical Password System. *International Journal of Human-Computer Studies*, 63(1-2):102 – 127, 2005.

Appendix A

Inform Source Code

A.1 Proposal I

```
"Stacker" by "Carson Brown"

The Stage is a room.

Use no scoring.

The bowl is a supporter carried by the player. The bowl is not fixed in
    place.
The hammer is a supporter carried by the player. The hammer is not fixed
     in place.
The ruler is a supporter carried by the player. The ruler is not fixed
    in place.
The paintbrush is a supporter carried by the player. The paintbrush is
    not fixed in place.
The soap is a supporter carried by the player. The soap is not fixed in
    place.
The telephone is a supporter carried by the player. The telephone is not
     fixed in place.
A pair of scissors is a supporter carried by the player. The pair of
    scissors is not fixed in place.
The cup is a supporter carried by the player. The cup is not fixed in
    place.
The towel is a supporter carried by the player. The towel is not fixed
    in place.
The camera is a supporter carried by the player. The camera is not fixed
     in place.
```

```
The table is a supporter in the stage.

The Proper Stack is a list of objects that varies. The Proper Stack is
    usually {table, hammer, paintbrush, soap, cup}.

The Chosen Stack is a list of objects that varies. The Chosen Stack is
    usually {table}. [The exact stack the player makes.]

The Last Object is an object that varies. The Last Object is usually the
     table.

After putting a supporter on the Last Object:
        Add the noun to the Chosen Stack;
        Now the Last Object is the noun;
        Continue the action.

Understand "stack [something]" as stacking. Stacking is an action
    applying to one thing.

Check stacking:
        if the noun is carried by the player:
                try putting the noun on the Last Object.

Every turn:
        [say "Your Chosen Stack is [Chosen Stack].";]
        [say "The Proper Stack is [Proper Stack].";]
        If Chosen Stack is the Proper Stack:
                say "As you drop the last item down, you notice a small
                    crease in the wall. Pushing on it, you find a small
                    doorway, and walk through...";
                end the game in victory.
```

Figure A.1. A complete Inform story file used in Proposal I. Some functionality is not shown.

A.2 Proposal II

```
"TwistyPassages" by "Carson Brown"

Include Plurality by Emily Short.

Include Exit Descriptions by Matthew Fletcher.

A shelf is a kind of supporter.
```

```
A cabinet is a kind of container. A cabinet is fixed in place.

A table is a kind of supporter.

The carrying capacity of the player is 8. Use no scoring.

The Office is a room. "You are in an office floor, with only one cubicle
    and poorly lit from buzzing fluorescent lights above."
The coffee cup is in the office.
The doughnut is in the office. Understand "donut" as the doughnut.
The keyboard is in the office.
The necktie is in the office. Understand "tie" as the necktie.
The pencil is in the office.
The stapler is in the office.
The toner is in the office.
The inkwell is in the office
The yellow highlighter is in the office. Understand "hilighter" as the
   yellow highlighter.
The holepunch is in the office.
The gluestick is in the office.
The mousepad is in the office.
The notepad is in the office.
The wastebin is in the office. Understand "bin" as the wastebin.
The ruler is in the office.
The desk is a supporter in the office.
The filing cabinet is a cabinet in the office.

The Bathroom is a room. "The bathroom is a friendly blue, complete with
   white tile floor and counter." The bathroom is north of the office.
The bathmat is in the bathroom.
The lotion is in the bathroom.
The mouthwash is in the bathroom.
The Reader's Digest is in the bathroom.
The soap is in the bathroom.
The toothbrush is in the bathroom.
The compact mirror is in the bathroom.
The floss is in the bathroom.
The hairspray is in the bathroom.
The loofa is in the bathroom.
The nailpolish is in the bathroom.
The plunger is in the bathroom.
The eyeliner is in the bathroom.
The vitamins are in the bathroom.
The astringent is in the bathroom.
The tile counter is a supporter in the bathroom. Understand "counter" as
    the tile counter.
The medicine cabinet is a cabinet in the bathroom.

The Den is a room. "The dimly lit family den has simple wallpaper, a
   coffee table and couch." The den is east of the bathroom.
```

```
The blanket is in the den.
The DVD is in the den.
The rug is in the den.
The telephone is in the den.
The Sudoku book is in the den.
The remote control is in the den. Understand "clicker" as the remote
    control.
The VCR is in the den.
The ashtray is in the den.
The lightbulbs are in the den.
The matches are in the den.
The lamp is in the den.
The yarn is in the den.
The knitting needles are in the den. Understand "needles" as knitting
    needles.
The cables are in the den.
The nesting dolls are in the den. Understand "dolls" as nesting dolls.
The couch is a supporter in the den.
The coffee table is a table in the den.

The Galleria is a room. "Large, bright windows flood the galleria with
    sunny, warm light." The galleria is east of the den.
The bouquet of roses are in the galleria. Understand "flowers" as the
    bouquet of roses.
Some crayons are in the galleria.
The fountain pen is in the galleria.
The paintbrush is in the galleria.
The marble statue is in the galleria.
The oil painting is in the galleria.
The sketchbook is in the galleria.
The thimble is in the galleria.
The dreamcatcher is in the galleria.
The violin is in the galleria.
The compass is in the galleria.
The headphones are in the galleria.
The umbrella is in the galleria.
The grand piano is in the galleria.
The reading glasses are in the galleria.
The glass table is a supporter in the galleria.
The bookcase is a supporter in the galleria.

The Bedroom is a room. "Soft lavender walls surround the bedroom with
    shades drawn."
The bedroom is north of the galleria.
The earrings are in the bedroom. The earrings are plural-named.
The hairbrush is in the bedroom.
The glass of water is in the bedroom. Understand "water glass" as the
    glass of water.
The necklace is in the bedroom.
The pillow is in the bedroom.
The make-up mirror is in the bedroom.
```

```
The slippers are in the bedroom. The slippers are plural-named.
The receipt is in the bedroom.
The coin collection is in the bedroom.
The trophy is in the bedroom.
The embroidery is in the bedroom.
The hangers are in the bedroom. The hangers are plural-named.
The brooch is in the bedroom.
The figurine is in the bedroom.
The novel is in the bedroom.
The bed is a supporter in the bedroom.
The nightstand is a supporter in the bedroom.

The Classroom is a room. "The classroom has white walls and linoleum
    floors, with small desks all facing a larger teacher's desk." The
    classroom is west of the bedroom. The classroom is north of the den.
     The classroom is east of the kitchen.
The binder is in the classroom.
The chalk is in the classroom.
The easel is in the classroom.
The globe is in the classroom.
The protractor is in the classroom.
The fishbowl is in the classroom.
The textbook is in the classroom.
The dictionary is in the classroom.
The posterboard is in the classroom.
The gum is in the classroom.
The abacus is in the classroom.
The calculater is in the classroom.
The nametags are in the classroom. The nametages are plural-named.
The yardstick is in the classroom.
The apple is in the classroom.
The teacher's desk is a supporter in the classroom.
The cupboard is a cabinet in the classroom.

The Pub is a room. "A traditional British-looking pub surrounds you,
    wood panelling throughout." The pub is north of the classroom.
Some darts are in the pub.
The glass mug is in the pub.
Some french fries are in the pub. Understand "fry" as the french fries.
The mounted deer head is in the pub.
The newspaper is in the pub.
Some peanuts are in the pub.
The salt shaker is in the pub.
The toupee is in the pub.
The barstool is in the pub.
The vintage rifle is in the pub. Understand "rifle" as vintage rifle.
The lime wedges are in the pub. Understand "lime" as lime wedges.
The cellphone is in the pub.
The wanted poster is in the pub.
The register is in the pub.
The gumball machine is in the pub.
```

```
The bar counter is a supporter in the pub.

The Greenhouse is a room. "A humid greenhouse filled with plants
    surprises you!" The greenhouse is west of the pub. The greenhouse is
     north of the kitchen. The greenhouse is south of the office.
The fertilizer is in the greenhouse.
The gardening glove is in the greenhouse.
The hose is in the greenhouse.
The radio is in the greenhouse.
The soil is in the greenhouse.
The trowel is in the greenhouse.
The watering can is in the greenhouse.
The birdseed is in the greenhouse.
The rock salt is in the greenhouse.
The crocs are in the greenhouse. The crocs are plural-named.
The lawnchairs are in the greenhouse.
The bug repellant is in the greenhouse.
The nets are in the greenhouse.
The pellets are in the greenhouse.
The owl statue is in the greenhouse.
One shelf is in the greenhouse.

The Kitchen is a room. "You step into a sunny kitchen, painted a bright
    yellow and filled with oak cabinetry as well as a large counter top
    made of granite." The kitchen is north of the bathroom.
The lemon is in the kitchen.
The bowl is in the kitchen.
The cheese grater is in the kitchen.
The dishcloth is in the kitchen.
The green pepper is in the kitchen.
The quarter is in the kitchen.
The vase is in the kitchen.
The oak cabinet is a cabinet in the kitchen.
The ginger is in the kitchen.
The phone is in the kitchen.
The melon baller is in the kitchen.
The keys are in the kitchen. The keys are plural-named.
The frying pan is in the kitchen.
The toaster is in the kitchen.
The pestle is in the kitchen.
The granite counter top is a supporter in the kitchen. Understand
    "counter", "countertop" and "counter top" as the granite counter
    top.

The Cellar is a room. "The cellar is dimly light and is colder than the
    other rooms." The cellar is north of the bedroom. The cellar is east
     of the pub.
The firewood are in the cellar. Understand "wood" as the firewood.
The garlic is in the cellar.
The onion is in the cellar.
The potatoes are in the cellar.
```

```
The vegegatble seeds are in the cellar.
The tomato sauce is in the cellar.
The wine is in the cellar.
The axe is in the cellar.
The jam is in the cellar.
The wagon is in the cellar.
The beer is in the cellar.
Some lightbulbs are in the cellar.
Some cardboard boxes are in the cellar
The packing tape is in the cellar.
The soda pop is in the cellar.
One shelf is in the cellar.

The Garage is a room. "The garage is painted an industrial grey. A
    rusted car sits in the middle of the room." The garage is east of
    the galleria. The garage is west of the bathroom.
Some batteries are in the garage.
The carjack is in the garage.
Some gas is in the garage.
The hammer is in the garage.
The rag is in the garage.
The sandpaper is in the garage.
The monkey wrench is in the garage.
The oil can is in the garage.
The paint is in the garage.
The chisel is in the garage.
The recycling is in the garage.
Some pliers are in the garage.
Some nails are in the garage.
The thick apron is in the garage.
The drill is in the garage.
The workbench is a supporter in the garage.
The toolbox is a closed, openable container in the garage. The toolbox
    is fixed in place.

The Darkroom is a room. "At first the room is pitch black, slowly your
    eyes adjust and you find yourself surround by photo processing
    equipment." The Darkroom is east of the cellar. The darkroom is west
     of the greenhouse.
The pin is in the darkroom.
The developing tray is in the darkroom.
The negatives are in the darkroom.
The reel is in the darkroom.
The scissors are in the darkroom.
The tongs are in the darkroom.
The unlabelled bottle is in the darkroom.
The film is in the darkroom.
The cleanser is in the darkroom.
The basin is in the darkroom.
The old camera is in the darkroom.
The sink is in the darkroom.
```

```
The yellowed paper is in the darkroom.
The waterbottle is in the darkroom.
The light is in the darkroom.
The storage closet is a closed, openable container in the darkroom. The
    storage closet is fixed in place.
One table is in the darkroom.

The Library is a room. "The library is covered in dust and all the
    bookcases are curiously empty. Luckily there is no body in the
    library." The library is east of the office. The library is south of
     the den. The library is north of the pub.
The brass key is in the library.
The dictaphone is in the library.
The ink is in the library.
The library card is in the library. Understand "card" as library card.
The microfilm is in the library.
The paperclips are in the library.
The stamps are in the library.
The thesaurus is in the library.
The kleenex is in the library.
The fax is in the library.
The candlestick is in the library.
The daggar is in the library.
The pipe is in the library.
The wrench is in the library.
The gold trophy is in the library.
The bookshelf is a shelf in the library.
The wooden desk is a closed, openable container in the library. The
    wooden desk is fixed in place.

The Barn is a room. "The floor of the barn is covered in hay. There are
    rows of empty pens but the barn still smells strongly of animals."
    The Barn is south of the galleria. The barn is east of the library.
    The barn is north of the cellar.
The x-acto knife is in the barn. Understand "boxcutter" and "utility
    knife" as the x-acto knife.
The flashlight is in the barn.
The harness is in the barn.
The trap is in the barn.
The mouse is in the barn.
The pellet gun is in the barn.
Some twine is in the barn.
The wire is in the barn.
The oats are in the barn.
The rope is in the barn.
The nail clippers are in the barn.
The shears are in the barn.
The dirty hay is in the barn.
The green paint is in the barn.
The cat is in the barn.
The unused trough is a supporter in the barn.
```

```
The Pool is a room. "You enter a white tiled room, the floor is wet and
    slippery. There is a olympic sized swimming pool in the middle of
    the room." The pool is east of the bedroom. The pool is south of the
     darkroom. The pool is north of the garage. The pool is west of the
    kitchen.
Some chlorine is in the pool.
Some flippers are in the pool.
The kickboard is in the pool.
the noseplug is in the pool.
The snorkle is in the pool.
The towel is in the pool.
The whistle is in the pool.
The oars are in the pool.
The bathing suit is in the pool.
The robe is in the pool.
The lifejacket is in the pool.
The padlock is in the pool.
The mouth guard is in the pool.
The dehumidifier is in the pool.
Some jacuzzi cleaning supplies are in the pool.
The locker is a closed, openable container in the pool. The locker is
    fixed in place.

The Coffee Shop is a room. "You find yourself in an expensive coffee
    shop. There are no baristas to make you a coffee." The coffee shop
    is south of the garage. The coffee shop is east of the barn. The
    coffee shop is north of the darkroom. The coffee shop is west of the
     office.
The apron is in the coffee shop.
The biscotti is in the coffee shop.
Some coffee grounds are in the coffee shop.
The lids are in the coffee shop.
The shotglass is in the coffee shop. Understand "glass" as shotglass.
The tea bags are in the coffee shop.
The vanilla is in the coffee shop.
The cinnamon is in the coffee shop.
The paper menu is in the coffee shop.
The french press is in the coffee shop.
The kettle is in the coffee shop.
The music CD is in the coffee shop.
The glass sugar container is in the coffee shop.
The spoon is in the coffee shop.
Some espresso beans are in the coffee shop.
The dirty counter is a supporter in the coffee shop.
The Proper Path is a list of directions that varies. The Proper Path is
    usually {north, east, east, north, west, north, west, south}.

The Chosen Path is a list of directions that varies. [The exact path the
     player makes.]
```

```
After going:
        Add the noun to the Chosen Path;
        Continue the action.

Every turn:
        [say "Your chosen path is [Chosen Path].";]
        [say "The Proper path is [Proper Path].";]
        [check if the player got to the kitchen by the right path]
        let numItems be the number of things carried by the player;
        If numItems is 0 and the player is in the kitchen and Chosen
           Path is the Proper Path:
                [check if the right objects are in the kitchen]
                if the necktie is in the kitchen and the toothbrush is
                   in the kitchen and the remote control is in the
                   kitchen and the oil painting is in the kitchen and
                   the pillow is in the kitchen and the chalk is in the
                    kitchen and the french fries are in the kitchen and
                    the gardening glove is in the kitchen:
                        say "As you drop the last item down, you notice
                           a small crease in the wall. Pushing on it,
                           you find a small doorway, and walk
                           through....";
                        end the game in victory.
```

Figure A.2. A complete Inform story file used in Proposal II.

A.3 Proposal III

```
"AuthAdventure" by "Carson Brown".

Include Modern Conveniences by Emily Short.

Include Exit Lister by Gavin Lambert.

The story genre is "Authentication". The story headline is "A old way
    for a new
authentication system".

The carrying capacity of the player is 8. Use no scoring.

A label is a kind of thing. The description of a label is "The label has
     an adhesive back that seems to stick to nearly anything[if the
    label is part of something (called the parent)]. It is stuck to [the
     parent][end if]."
```

```
The red sticky label is a label carried by the player. Understand "red
    label" as red sticky label.

Instead of tying the red sticky label to something:
        now the red sticky label is part of the second noun;
        say "You stick [the red sticky label] to [the second noun]."

After examining something when the red sticky label is part of the noun:
        say "A bright red sticky label is attached to [the noun]!"

Before tying the red sticky label to something when the red sticky label
     is part of something:
        if the red sticky label is part of the second noun:
                say "[The red sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the red sticky label;
                if the red sticky label is part of something, stop the
                    action.

Instead of tying the red sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the red sticky label when the label is part of
    something:
        now the player carries the red sticky label;
        say "You peel the label off again."

Instead of tying something to the red sticky label:
        try tying the red sticky label to the noun.

Instead of putting the red sticky label on something:
        try tying the red sticky label to the second noun.

Instead of inserting the red sticky label into something:
        try tying the red sticky label to the second noun.

The blue sticky label is a label carried by the player. Understand "blue
     label" as blue sticky label.

Instead of tying the blue sticky label to something:
        now the blue sticky label is part of the second noun;
        say "You stick [the blue sticky label] to [the second noun]."

After examining something when the blue sticky label is part of the
    noun:
        say "A bright blue sticky label is attached to [the noun]!"

Before tying the blue sticky label to something when the blue sticky
    label is part of something:
```

```
        if the blue sticky label is part of the second noun:
                say "[The blue sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the blue sticky label;
                if the blue sticky label is part of something, stop the
                    action.

Instead of tying the blue sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the blue sticky label when the label is part of
    something:
        now the player carries the blue sticky label;
        say "You peel the label off again."

Instead of tying something to the blue sticky label:
        try tying the blue sticky label to the noun.

Instead of putting the blue sticky label on something:
        try tying the blue sticky label to the second noun.

Instead of inserting the blue sticky label into something:
        try tying the blue sticky label to the second noun.

The green sticky label is a label carried by the player. Understand
    "green label" as green sticky label.

Instead of tying the green sticky label to something:
        now the green sticky label is part of the second noun;
        say "You stick [the green sticky label] to [the second noun]."

After examining something when the green sticky label is part of the
    noun:
        say "A bright green sticky label is attached to [the noun]!"

Before tying the green sticky label to something when the green sticky
    label is part of something:
        if the green sticky label is part of the second noun:
                say "[The green sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the green sticky label;
                if the green sticky label is part of something, stop the
                    action.

Instead of tying the green sticky label to a label:
        say "That would ruin the label entirely."
```

```
Instead of taking the green sticky label when the label is part of
   something:
        now the player carries the green sticky label;
        say "You peel the label off again."

Instead of tying something to the green sticky label:
        try tying the green sticky label to the noun.

Instead of putting the green sticky label on something:
        try tying the green sticky label to the second noun.

Instead of inserting the green sticky label into something:
        try tying the green sticky label to the second noun.

The yellow sticky label is a label carried by the player. Understand
   "yellow label" as yellow sticky label.

Instead of tying the yellow sticky label to something:
        now the yellow sticky label is part of the second noun;
        say "You stick [the yellow sticky label] to [the second noun]."

After examining something when the yellow sticky label is part of the
   noun:
        say "A bright yellow sticky label is attached to [the noun]!"

Before tying the yellow sticky label to something when the yellow sticky
    label is part of something:
        if the yellow sticky label is part of the second noun:
                say "[The yellow sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the yellow sticky label;
                if the yellow sticky label is part of something, stop
                    the action.

Instead of tying the yellow sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the yellow sticky label when the label is part of
   something:
        now the player carries the yellow sticky label;
        say "You peel the label off again."

Instead of tying something to the yellow sticky label:
        try tying the yellow sticky label to the noun.

Instead of putting the yellow sticky label on something:
        try tying the yellow sticky label to the second noun.

Instead of inserting the yellow sticky label into something:
```

```
        try tying the yellow sticky label to the second noun.

The purple sticky label is a label carried by the player. Understand
   "purple label" as purple sticky label.

Instead of tying the purple sticky label to something:
        now the purple sticky label is part of the second noun;
        say "You stick [the purple sticky label] to [the second noun]."

After examining something when the purple sticky label is part of the
   noun:
        say "A bright purple sticky label is attached to [the noun]!"

Before tying the purple sticky label to something when the purple sticky
    label is part of something:
        if the purple sticky label is part of the second noun:
                say "[The purple sticky label] is already stuck to [the
                   second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the purple sticky label;
                if the purple sticky label is part of something, stop
                   the action.

Instead of tying the purple sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the purple sticky label when the label is part of
   something:
        now the player carries the purple sticky label;
        say "You peel the label off again."

Instead of tying something to the purple sticky label:
        try tying the purple sticky label to the noun.

Instead of putting the purple sticky label on something:
        try tying the purple sticky label to the second noun.

Instead of inserting the purple sticky label into something:
        try tying the purple sticky label to the second noun.

The orange sticky label is a label carried by the player. Understand
   "orange label" as orange sticky label.

Instead of tying the orange sticky label to something:
        now the orange sticky label is part of the second noun;
        say "You stick [the orange sticky label] to [the second noun]."

After examining something when the orange sticky label is part of the
   noun:
        say "A bright orange sticky label is attached to [the noun]!"
```

```
Before tying the orange sticky label to something when the orange sticky
    label is part of something:
        if the orange sticky label is part of the second noun:
                say "[The orange sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the orange sticky label;
                if the orange sticky label is part of something, stop
                    the action.

Instead of tying the orange sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the orange sticky label when the label is part of
   something:
        now the player carries the orange sticky label;
        say "You peel the label off again."

Instead of tying something to the orange sticky label:
        try tying the orange sticky label to the noun.

Instead of putting the orange sticky label on something:
        try tying the orange sticky label to the second noun.

Instead of inserting the orange sticky label into something:
        try tying the orange sticky label to the second noun.

The black sticky label is a label carried by the player. Understand
   "black label" as black sticky label.

Instead of tying the black sticky label to something:
        now the black sticky label is part of the second noun;
        say "You stick [the black sticky label] to [the second noun]."

After examining something when the black sticky label is part of the
   noun:
        say "A bright black sticky label is attached to [the noun]!"

Before tying the black sticky label to something when the black sticky
   label is part of something:
        if the black sticky label is part of the second noun:
                say "[The black sticky label] is already stuck to [the
                    second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the black sticky label;
                if the black sticky label is part of something, stop the
                     action.
```

```
Instead of tying the black sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the black sticky label when the label is part of
   something:
        now the player carries the black sticky label;
        say "You peel the label off again."

Instead of tying something to the black sticky label:
        try tying the black sticky label to the noun.

Instead of putting the black sticky label on something:
        try tying the black sticky label to the second noun.

Instead of inserting the black sticky label into something:
        try tying the black sticky label to the second noun.

The white sticky label is a label carried by the player. Understand
   "white label" as white sticky label.

Instead of tying the white sticky label to something:
        now the white sticky label is part of the second noun;
        say "You stick [the white sticky label] to [the second noun]."

After examining something when the white sticky label is part of the
   noun:
        say "A bright white sticky label is attached to [the noun]!"

Before tying the white sticky label to something when the white sticky
   label is part of something:
        if the white sticky label is part of the second noun:
                say "[The white sticky label] is already stuck to [the
                   second noun]." instead;
        otherwise:
                say "(first freeing the label)[line break]";
                silently try taking the white sticky label;
                if the white sticky label is part of something, stop the
                    action.

Instead of tying the white sticky label to a label:
        say "That would ruin the label entirely."

Instead of taking the white sticky label when the label is part of
   something:
        now the player carries the white sticky label;
        say "You peel the label off again."

Instead of tying something to the white sticky label:
        try tying the white sticky label to the noun.

Instead of putting the white sticky label on something:
```

```
        try tying the white sticky label to the second noun.

Instead of inserting the white sticky label into something:
        try tying the white sticky label to the second noun.

Understand the commands "stick" or "apply" as "tie".

Understand the commands "peel" as "take".

[!!begin room code]

The living room is a room. "You find yourself in the living room of a
    small, single-story bungalow."

The bookcase is in the living room. The description is "The bookcase is
    of simple construction and has three shelves."

A shelf is a kind of supporter. The description of a shelf is "a simple
    wood shelf."

The top shelf is a shelf. The middle shelf is a shelf. The bottom shelf
    is a shelf. The top shelf, the middle shelf and the bottom shelf are
     part of the bookcase.

The wicker box is a container. The description of the wicker box is "A
    wicker box designed to hold DVDs. It has grooves to hold about
    thirty of them." Understand "box" as the wicker box. The wicker box
    is on the top shelf.


The candlestick is on the middle shelf. The description is "A burgundy
    candle, recently put out." Understand "candle" as the candlestick.

The dictionary is on the middle shelf. "The large English dictionary has
     a blue linen cover."

The hallway is east of the living room. "Description for hallway."

The picture frame is in the hallway.

The small kitchen is a kitchen. The small kitchen is east of the
    hallway. "Description for kitchen." Understand "kitchen" as the
    small kitchen.

The bedroom is west of the living room. "The bedroom is fairly small,
    provisioned with only the bare essentials."

The dresser is a supporter in the bedroom. The description is "The plain
     wooden dresser has top, middle and bottom drawers."
```

```
A drawer is a kind of container. A drawer is always openable and closed.
    The description of a drawer is "The usual drawer of wood,
    inadequately smoothed for ease of use."

The top drawer is a drawer. The middle drawer is a drawer. The bottom
    drawer is a drawer. The top drawer, the middle drawer, and the
    bottom drawer are part of the dresser. Instead of searching a closed
    drawer, try opening the noun.

The pair of socks is in the top drawer. Understand "socks" as the pair
    of socks. The description of the pair of socks is "A pair of white
    tube socks. Clean and simple."

The t-shirt is in the middle drawer. Understand "t-shirt", "shirt",
    "black shirt", "tshirt" as the t-shirt. The description of the
    t-shirt is "A simple cotton black t-shirt lies folded neatly in the
    drawer."

The pair of jeans is in the bottom drawer. Understand "jeans",
    "bluejeans", "blue jeans" as the pair of jeans. The description of
    the pair of jeans is "A pair of boot-cut blue jeans. Nothing
    special."

The bed is a supporter in the bedroom. The description is "A queen size
    bed is placed with its headboard against the middle of the far wall
    of the bedroom. It looks comfy."

The big bathroom is a bathroom. The big bathroom is north of the
    hallway. "The bathroom is larger than expected, it's a spacious
    arrangement of the essentials: a sink, toilet, bath and a medicine
    cabinet." Understand "bathroom" as the big bathroom.

The toothbrush is in the big bathroom. The toothpaste is in the big
    bathroom. The mouthwash is in the big bathroom.

When play begins:
        Let target be a random cabinet in the big bathroom;
        Move the toothbrush to the target;
        Move the toothpaste to the target;
        Move the mouthwash to the target.

South of the hallway is the Outside World.

The victory condition is a truth state that varies. The victory
    condition is usually true.

Instead of going south from the hallway:
        if the player is carrying a label:
                say "Sorry, you need to stick all the labels.";
        otherwise:
                [check if all the labels are on the right objects]
```

```
if the red sticky label is not part of the dresser,
    change the victory condition to false;
if the blue sticky label is not part of the t-shirt,
    change the victory condition to false;
if the green sticky label is not part of the pair of
    jeans, change the victory condition to false;
if the yellow sticky label is not part of the picture
    frame, change the victory condition to false;
if the purple sticky label is not part of the bath,
    change the victory condition to false;
if the orange sticky label is not part of the toilet,
    change the victory condition to false;
if the black sticky label is not part of the
    refrigerator, change the victory condition to false;
if the white sticky label is not part of the oven,
    change the victory condition to false;
[just check victory now]
if the victory condition is true:
        end the game in victory;
otherwise:
        end the game in death;
```

Figure A.3. A complete Inform story file used in Proposal III. Some objects are not included in the source code for brevity.