StoryPass: A System & Study for Memorable Secure Passphrases

by

Christopher Lockett Bonk

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

Masters of Science

in

Computer Science

University of Ontario Institute of Technology

Supervisor: Dr. Julie Thorpe

October 2014

Abstract

The goal of this thesis is to study the implementation of a passphrase system that implements new creation policies, called StoryPass. We are motivated to do this research as current text-based authentication methods, such as the password, fail to provide adequate security and usability. We call our system StoryPass because we were inspired by previous research which states that information created with stories can be more memorable. The problem we address is the lack of research on secure and usable passphrase creation guidelines. Our main contributions include a theoretical security analysis, a controlled 39-day user study and an estimate of the security that the resulting passphrases provide. Our security estimates are mainly performed through an algorithm that uses n-grams to estimate the number of attempts required to successfully guess passphrases created in StoryPass. We were able to successfully guess 64% of the passphrases collected during our 39-day user study, but with only a very large number of attempts. The passphrases which were not guessed generally contained slang and "non-words" which are words that are not found in standard dictionaries. Using a sentence-like structure in passphrases greatly improved usability. Memory errors were the leading cause of failed login; error correction techniques were used to prevent login failures from typographical errors. This thesis discusses how results from our user study can be used to help guide future passphrase creation policies.

Contents

List of Figures

List of Tables

Listings

1 Introduction

The password has been around since ancient times but in a computing context, it was introduced in the 1960's. Since then it has, for the most part, remained very much the same. This is because the formula for a password is based on a simple yet highly effective formula; the user knows something that supposedly only that user knows or a certain group of people know. Passwords play a critical role in today's world of information technology. Due to the rapid rise of the Internet and globalization, the demand for information to be instantly available at all times has never been greater. This demand has placed a heavy responsibility on computers to be able to securely store our information. On one end, we demand instant access to our information, while we also expect privacy and security to protect our information from malicious users. The vast majority of our information is held behind a gate which is guarded by a password.

1.1 Motivation

Time and time again, passwords have been labelled a relic of the past [6]. However, alternate methods of authentication have failed to gain the same traction and adoption that passwords have achieved. Biometrics, hardware tokens, and personal phone authenticators are all examples of authentication mechanisms which have failed to address the real issue of replacing passwords. Usability is the single most important factor when it comes to user authentication [46]. While one might think security is important, users will often quickly sacrifice security for usability, consciously or unconsciously. The primary issue with passwords is that users tend to implement and use them poorly. Users tend to pick weak passwords and generally reuse them.

These issues when coupled with the fact that modern password crackers can generate millions of guesses per second [19], mean that traditional passwords are becoming increasingly insecure. In recent years, policy makers have begun to require longer passwords in an attempt to improve security. However, this caused users to make equally weak passwords or make horribly unusable passwords [23]. In an attempt to improve security provided by passwords, users have begun to make longer passwords [29].

As discussed in Chapter 2, users either tend to make weak passwords or reuse the same password in order to compensate for the sheer amount of effort it requires to remember all their passwords. Other methods such as password managers like LastPass and KeePass are great tools which can help aid users in managing all of their passwords; however this creates another problem. It opens a single point of attack for malicious users who wish to steal a user's password. It is in a users best interest to use a highly secure and memorable password to secure their password manager. Passphrases are an evolution and improvement upon the password in the sense that it provides users a way to utilize secure and usable text-based authentication.

Passphrases are similar to passwords, except they generally are longer and a series of words. Passphrases tend to be a sentence or a phrase. While being longer, passphrases tend to be easier to remember due to their language syntactical structure [12]. Unlike a single word password, the phrase that is used as a passphrase can communicate or contain a special meaning. With this in mind, a user-chosen passphrase that a person chooses can have a personal meaning to the individual. This meaning can potentially be a mnemonic that can help improve the memorability and therefore usability of the passphrase.

If we accept the reality that text-based authentication is going to continue being the dominant method of user authentication [6], it is our imperative as researchers to provide the best practices when using passphrases. Current research lacks conclusive facts on how to make usable and secure passphrases. Although previous research has shown that many variations of passphrases have great promise for user authentication [5–9, 24, 26, 27, 30]. Specific research from other disciplines outside user authentication in mnemonics, memory recall, and stories have provided new directions for infusing and improving passphrases [3, 12, 13, 19, 22, 31, 34, 49].

1.2 Thesis Statement

We hypothesize that passphrases created using strong creation policies can be used as a secure and memorable form of end-user text-based authentication credentials. In this thesis we propose a new set policies and recommendations surrounding the use of passphrases to achieve stronger security and usability. We implement and test this through a system we call StoryPass. We were motivated to perform this research by the lack of research and understanding of passphrases. While some may think text-based authentication is a fading technology, the ease of usability and deployability are the main reasons why passwords have remained the dominant form of user authentication [6]. Since text-based authentication is unlikely to be completely replaced by another method any time soon, improving its security is an important task. We are motivated to do this research because attempts to replace passwords have failed to gain significant traction.

1.3 Main Contributions

Our contributions include:

1. Novel security analyses and attacks models for long passphrases, leading to a set of recommendations for creating strong passphrases.

2. The pilot testing of these recommendations, along with recommendations from other passphrase research [30] [26] [27] [48] [5] [8], leading to a final set implemented in the StoryPass passphrase system.

3. A security and usability analysis of the StoryPass system from a 39 day user study.

We provide a novel method of estimating the security that user-created passphrases provide. We use n-grams which are modelled and observed from natural language corpora to give security estimates that are based on chaining n-grams together. We also use a custom built dictionary of slang terms from UrbanDictionary.com to identify known slang terms in passphrases. Our user study provides valuable insight into user behaviour with passphrases to ensure reasonable usability and security in a passphrase system.

2 Related Work

Ever since the advent of computers and the rise information of technology and more precisely, the free flowing of massive amounts of information, we have sought to achieve a level of privacy with our digital existence. Whether this is in the form of safeguarding a Facebook profile or protecting national secrets, user authentication has never been more important in human history. The rise of information technology, and more importantly the Internet, has allowed access to nearly the entire collection of human information in mere seconds. This is a double edged sword however, because not all information is public knowledge and we like to think we have a level of privacy when we use computers. For example, I keep this thesis document in a Dropbox account; I like to believe that I am the only person who can access it since I set a password on the account that only I know. Using current technology or malicious techniques it might be possible for someone else to access my personal files. User authentication gives me assurance (within reason) that even though my information is potentially accessible to anyone with an Internet connection, that only I can access my files.

In this chapter, we will discuss the current literature around the password and a variation of text-based authentication called a passphrase. A passphrase is similar to password except passphrases tend to be a full phrase or sentence where as a password is a string of characters, typically 8-16 characters long, which often ends up being a single word in practice.

We will begin with looking at why passwords persist as the dominant form of electronic user authentication in Section 2.1. Next we will look at password policies which guide the creation of passwords in Section 2.2. Continuing on from the blunders of password policies, we will delve in to passphrases in Section 2.3. Following passphrases, we will

look at the research which guided us in creating our StoryPass passphrase creation policy in Section 2.4.

2.1 Persistence of Passwords

Passwords are by far the most common end user authentication method used today. Yet they are considered very weak from a security stance [35] and users have a hard time remembering them [6]. Passwords are far past their expiration date. Computers have become exponentially faster for a fraction of the cost over the past decade. This makes it very computationally and financially inexpensive for malicious users to guess passwords. Passwords have been studied for decades and are known to be vulnerable to a number of malicious attacks [35]. Password policies are usually implemented poorly which makes it even easier for malicious users to crack passwords [18]. Policies tend to forget that people are required to remember multiple passwords for multiple websites [17], which leads to people having memory limitations and therefore preventing them from constructing strong passwords. Passwords are easy for developers and organizations to deploy, but require users to make sacrifices in terms of security [6].

Bonneau et al. [6] looks at a wide range of studies on password implementations that set out to replace the text-based password schemes on the web. Two decades of proposals are studied and ranked using twenty-five usability, deployability and security benefits that the proposed might provide. The study looks at systems such as:

- Password Management Software
- Single Sign On
- Graphical Passwords

- Cognitive Passwords
- One-time Passwords
- Hardware Tokens
- Phone-aided Authentication
- Biometrics

None of the password replacements came close to providing all the desirable benefits. Common text-based web passwords in the study receive a perfect score in the Deployability Section. This is the main reason why passwords remain popular to this day. No other method of authentication received a score even close to perfect in this Section.

However, passwords were scored quite low in usability benefits. They are considered not "memory effortless" (they need to be remembered by the user). They do not scale very well; users tend to create either a few passwords and reuse them or they make many easily guessable passwords to compensate.

While there has been much research done that concludes we are far overdue for switching from a text-based authentication method, it seems though text-based authentication is here to stay. Bonneau et al. [6] and Renaud et al. [41] both came to the same conclusion that due to the deployability of text-based authentication and the lack of additional hardware, that text-based authentication will be here until a better, yet just as deployable solution is available.

2.2 Password Policies

Password policies have proven again and again that they fail to address real issues [18]. They generally blame the user when in fact the real issue is the system. A truly usable

system is one that adapts to human behaviour, not humans adapting to a system. Policies tend to do the latter; they set out a list of rules or standards such as passwords must be a certain length, contain certain numbers or symbols, and be changed frequently. In this Section, we will look into research around current password policies practices and why they tend to do the exact opposite of their intention.

Policies tend to force users to pick weak passwords by asking them to create unnecessarily strong passwords [18]. Users truly do want to help maintain the security of their organization but policies make it difficult to comply. Productivity is hurt because users spend too much time complying with the policies [23]. Policies also do little to prevent against phishing or key logging which are a major cause of password theft [21]. Policies tend to be inversely correlated with security and size of the user community [18]. This is because the largest sites tend to jockey for traffic and user interaction, which lead to larger profits from advertising and sales. Users in general are concerned about security and privacy of their data however, users do not care when nothing of significance to them is being protected such as an account for online shopping, or when they are willingly or unwillingly ignorant of insecure practices [21]. It would be foolish to implement strong password policies which hinder usability and therefore possible revenue [18].

Inglesant et al. [23] research password security policies in organizations and the impact they have on the members. They asked 32 staff members to keep a password diary for one week. The participants wrote down all the passwords they used. They were then interviewed about the passwords in their diary. In general, the participants do want to help maintain security at their organization but the security policies make it difficult to do so. Users care about security when they feel as though they have something worth while to protect. The policies place too great a demand on the users. This impacts their productivity negatively. Forcing a user to change their password

or meet a minimum strength does nothing against phishing or key logging, which is a major vector of attack for malicious users. Requiring excessively strong passwords or too many passwords forces users to cope by choosing weak passwords or using the same one for all accounts. They make a point that brute-forcing passwords is a very real threat. With this in mind, they suggest to leave using truly strong passwords (greater than 12 lower case letters or 10 alphanumeric characters) to situations where a malicious user would actually want to access. Determining when a malicious user would want to steal login credentials is situational, usually when value can be gained. Requiring excessively strong passwords in all situations usually just causes users to implement bad passwords in all situations, even if the situation is critical. This is a concious decision on the part of many users as any time spent without access to a service because they forgot their password is a waste of time and therefore money. Users do not care about security and more specifically, passwords. They just want a password that works and allows them to complete the task behind the password [21]. Password policies tend to force people pick unusable or unmemorable passwords due to the fact they must meet a certain requirement. Komanduri et al. [29] study password policies and the resulting strength of passwords that are composed. Multiple password policies were implemented in a 6,000 person user study over 48 days. The policies were:

1. comprehensive8 which required at least 8 characters and include one digit, one symbol, one upper case and one lower case.

2. basic8Survey which only required 8 characters.

3. basic8 which required 8 characters but also informed users that this was an email password; this was to measure the effect of whether or whether not users

cared about what the password was used for.

4. basic16 which only required 16 characters.

5. basic8Dictionary which must be least 8 characters and not contain a dictionary word.

The results showed that the basic16 policy provided the best resistance to password cracking programs and not being found in password dictionaries besides comprehensive8. While comprehensive8 had the best security results, it was not very usable. 57.6% users failed to create the comprehensive8 password due to failing to meet the policy requirements, compared to 39% for basic16. Basic16 also had reasonable usability ratings with only on average taking 1.66 attempts to create a passphrase, compared to 3.33 for comprehensive8. Security, however was calculated using an outdated method called entropy [43]. While it is an outdated method, it can be be considered a crude method of calculating possible security provided by English text [10]. The calculation of entropy is debatable [43] [45] [10]. Entropy is defined as the average amount of information that is produced by each character in a piece of text [43]. Entropy in [29] is calculated by summing the entropy values of each element which composes a password. The elements are password length, number of each character class (lowercase, uppercase, numbers and symbols) and the content of each character. The main takeaway was that by doubling the character requirement of basic8 to basic16, almost no user reported usability issues were noted and it greatly thwarted guessing attempts. Dictionary checks of passwords can help rule out easily guessed passwords using basic password cracking heuristics but caused immense user frustration during creation. It would be ideal to use full password cracking dictionaries to "blacklist" weak passwords, but this can greatly decrease user satisfaction. It is possible to increase security and maintain usability but it is a fine line. Generally,

the higher the security requirement, the more likely that user frustration will occur. Similar to Komanduri et al. [29], Weir et al. [50] found that using entropy as a metric in password creation policies is a poor idea. Entropy creates a meaningless metric because high entropy can be achieved but the password will be very guessable. The reason most password creation policies fail is even if they end up forcing a user to create a password with more entropy, they will use easy to guess methods to meet the requirements; such as adding 123 at the end of a password. Also similar to the study by Komanduri et al. is the recommended use of password blacklists. Blacklists are lists of the most common passwords, usually made from leaked user credentials databases. This further reinforces the point that humans cannot pick good random passwords that are easy to remember. Another flaw with password policies is the fact that they usually fail to address a much overlooked aspect of user authentication; a high entropy password is not required for every single situation. Requiring high security passwords as a blanket policy only causes users to place overly strong passwords in needless situations and use weak passwords in situations where a strong password should be used. Users should be addressed or informed of when it is necessary to use the effort in creating a strong password [50].

Simply put, users are not to blame for poor password creation [42]. It is poor policies that leads them to create a weak password. Many policies just tend to get in the way of the user's real work. We must design our password policies in such a way that works for humans.

The ability of the human mind to remember events had been studied to show that with proper practice and spacing, we can remember larger amounts of information for a significant duration [4]. It can also astonish you at how quickly it can forget

things too [51]. The mind is very associative in the sense that it remembers ideas and knowledge well when they relate to one another [15]. This is why we educate ourselves in an organized manner. You don't just start memorizing random facts, we start with the basics and work our way up. We learn, understand, and comprehend ideas to better retain knowledge. In the same context, when creating a password, we don't just bash our keyboard randomly. We think of a favourite colour, book, or some kind of idea and use that for passwords. This is why humans are inherently bad at making strong passwords; we are not random. Everything we do is an action that has a reason behind it. We communicate in the language we were taught, we drive on the right side of the road because that's how we drive in North America. It is no different for creating passwords; there are patterns in the way we create a password. This is why password cracking dictionaries exist. We are easily predictable because we lack the memory and motivation to create multiple strong passwords [17]. We must find a way to work with our memory to create strong passwords and passphrases [42].

The human mind can remember chunks of knowledge. Miller et al [34]. proposed the idea that the human mind can retain up to 7 plus or minus 2 chunks of information in short term memory. If we assume that in standard random 8-char passwords that a chunk is a character, it would conform to the standard password policy that many authentication schemes use. However, we can increase the amount of information that is retained by simply increasing the size of chunks. Hence the use of words instead of characters can exploit this characteristic of the human mind. However, Baddeley and Cowan [3, 14] revisits the 1956 article by Miller that proposes the idea that the human mind can retain up to 7 plus or minus 2 "chunks" of information. It is rephrased by Baddeley and Cowan through their independent research that the number is closer to five. They insist though that the number can greatly increase or decrease depending on how integrated the ideas are. This reinforces the idea that

the human mind is extremely associative. It is now clear as to why the standard 8 character password is not a good choice for many users. 8 random characters is too difficult for many people to remember so they opt to choose an easily guessable password such as "princess123" (they tack 123 on the end of an extremely common word). With the average English word at around 5 letters, a passphrase composed of four, 5 letter words would make up a 20 letter passphrase. This might prove to be a much better solution than a standard 8 character password.

2.3 Passphrases

Passphrases, while unlikely to fix all the issues with user authentication, have the potential to help improve the usability and security of text-based authentication [46]. Passphrases take advantage of mnemonics because they are comprised of a phrase or sentence, which is more familiar than a series of characters, numbers, and symbols [30]. Passphrases are superior for our memory; they can be approached like a story and written with a natural flow like a regular sentence or phrase. This in turn can make passphrases more memorable. Memorability is important but so is usability. Usability is closely related to memorability, as a passphrase must be memorable to be usable. Usability is defined in terms of memorability and the associated overheads with using the passphrase or password [1]. Passphrases and generally, password policies that require more length instead of special characters have been shown to have an equal or greater level of user reported usability and a higher level of security [46].

2.3.1 Usability

Usability can be summed up by its ease of use, learnability, and its relation to memorability [1]. Keith et al. [26] delved into the issue of how well users can remember longer passphrases, the strength of the passphrase against attacks, and the satis-

faction of users using passphrases. The main finding was that passphrases lead to more typographical errors. Typographical errors are errors that occur when typing in the password or passphrase, such as missed key or extra key stroke. Users perceived passphrases as more difficult but study results proved that they were no more difficult to remember than other password methods. A significant learning curve was experienced and witnessed in the research with typographical errors by users. However by week 6 of the experiment the difference in typos between passwords and passphrases had disappeared entirely. By week 10, users who authenticated with passphrases rated ease of use lower than password. This is likely because of the typographical errors that occurred for the users in the passphrase group. This is likely due to the fact that passphrases of equivalent security required more concentration and dexterity than passwords. The three different creation policies were:

- Freeform password - Zero requirements
- Stringent passwords - At least 7 characters with one upper case letter, one lower case letter, and one non letter character
- Passphrase - At least 15 characters with one upper case letter, one lower case letter, and one non letter character.

It is important to note that the passphrase creation policy in this study is more of a long password, than a passphrase because it does not encourage words or a sentence like structure. While users may have had trouble typing in their passphrase, it was because of typographical errors, not memory errors. Before accounting for typographical errors, the login rates for the freeform password, stringent password, and passphrase were 85.61%, 80.38%, and 71.58% respectively. After accounting for typographical errors, they were 87.50%, 84.21% and 85.86% respectively. The typographical errors were adjusted based on allowing an edit distance of .125 per character,

or 1 in 8. An edit distance is the number of inserts, deletions, and substitutions of characters that would be required to change one string to another. This means that the word "cat" and "hat" have an edit distance of one and that "cat" and "hatt" have an edit distance of two. The passphrase group had a slightly better login rate than the stringent password group when accounting for typographical errors. This study highlights that users can learn to use longer passwords, even if they include a complex creation policy such as the passphrase group; although users require more time to learn and adapt to the increased typing length of the passphrase. As shown, it can be alleviated by using error correction techniques. The researchers state that this is one of the first passphrase studies and that the users were likely not familiar with passphrase creation policies; with time, users may be able to become more comfortable with using longer passwords and passphrases.

Shay et al. [46] conducted an online study with over 8,000 participants. They looked at multiple password and passphrase creation policies to determine the usability. They reinforced the idea that there is a trade off between security and usability. The creation policies were

- Comp8 - at least 8 characters, including one upper case, one lower case, a symbol and a digit
- Basic12, Basic16, Basic20 - At least 12, 16 or 20 characters
- 2word12, 2word16 - At least 12 or 16 characters and at least 2 words separated by a non character
- 3class12, 3class16 - At least 12, 16 characters. At least 3 of the 4 character classes from the Comp8 policy

The usability results varied across a number of metrics. The highest dropout rate was in the comp8, 3class12, and 3class16 groups. The storage rate was also highest in these three groups as well, with 3class16 being significantly higher than the other two. Comp8 participants required the most creation attempts by users. Users rated the basic12 and 2word12 as the easiest methods during creation. Failure to create a password or passphrase was primarily caused by failing to meet the character class requirements. Participants in the basic12 had the highest successful login rate on first attempt with 96.4% and 3class16 being the worst. Analysing these results, it can be seen that requiring more character length in a password can have more usability than equal counterparts requiring more character classes. It is likely that the additional character classes introduce an additional memory load on the participant. Memory is a flexible form of storage, thought to be limitless [41]. Although some say that people cannot remember strong password [7].

Bonneau and Schechter [7] performed a user study over two weeks to study the usability of system-assigned 56-bit codes. They aimed to debunk the myth that users cannot remember random system-assigned secrets. The remote research users had to login over 90 times with a password of their choosing; next they entered in the system-assigned code. After they entered their password, they had to type in the system-assigned code which was 4 letters or 2 words, 18.8 bits. After the first login, a short delay of 6.9 seconds was added to showing the system-assigned code. The delay increased over time, but users could skip the delay by entering in the system-assigned code from memory. After users learned the code, additional words or letters were added to a total of 56.1 bits. In the end, 94% of participants eventually typed their entire 56-bit code from memory. The median number of logins it took to learn the code was 36, over an average of 10 days. 88% could remember it after just 3 days. In the end, one participant reported that "The words are branded into my brain." This

study is preliminary but it starts the narrative that remembering random system-assigned secrets is possible for the vast majority of people. Although the memory for the 56-bit code seemed to age significantly over time. Users were invited back for 2 follow-up sessions to recall their 56-bit code. By 2 weeks after the end of the study, only 59% of the participants could recall their code. There were two follow up However, memory errors are not the only concern for passphrases.

Typographical errors are a new problem that arise when using passphrases over passwords. This is because the policies that direct their creation tend to suggestion or require longer character requirements. Users may make more typographical errors in their passphrase but users that use passwords may make more memory errors [26].

Memory errors are failed login attempts because of a lapse in recalling the password. A typographical error is caused by a slip of the fingers when typing their passphrase on a keyboard.

Keith et al. [27] designs and performs a user study where passphrases are used for authentication. It is thought that passphrases increase typographical errors [26], and failed login attempts due to their longer character length than password. This is thought to cause user frustration, and therefore poor usability. He presents results that show when passphrases are well designed, they do not increase login failures. Well designed passphrases should be made with the idea of "word processing mode" which means that the passphrase should be similar to a regular phrase you would type in a word text document, i.e., a sentence with spaces between words and no letter substitutions or out of place special characters. The study used three groups of participants to compare standard user generated passwords, standard random passwords and user generated passphrases. The passwords were at least 8 characters long and the passphrase was at least 16 characters long. When users were told to

generate passphrases, they were instructed to use 3 to 5 words in the passphrase. Users made significantly less memory errors in entering passphrases. Users of the passphrase group rated passphrases better than passwords in usefulness. The cause of decreased memory based errors for passphrases is probably because of the "something you know" aspect of passphrases. The reason passphrases are referred to as "something you know" is because the user sets the passphrase with no restraints besides a minimum character length. Passphrases do not need numbers or symbols to meet minimum security requirements. It allows users to construct a phrase or a sentence. It could be 5 favourite words, or based around the names of the user's favourite pet animal. Passphrases are more human friendly; they fit the way people naturally think rather than trying to force humans to think and act like a computer.

One way to combat typographical errors and improve usability in passphrases is to allow error tolerance during authentication. Allowing users to have slight variation during login of passphrases allows for minor typographical errors to occur. Error tolerance can greatly improve login rates and reduce user frustration during authentication. This is compounded by the fact that passphrases are generally longer than passwords; if a user needs to authenticate multiple times before logging in, it can add a great deal of frustration due to the increased typing time during login. Bard et al. [5] propose a system where users can use spelling tolerant order independent system assigned passphrases. The system which is put forth allows users to pick from a selection of passphrases which can be re-ordered during login attempts and have up to Damerau-Levenshtein distance of 2 per word. A Damerau-Levenshtein edit distance metric is a way of determining how many changes to a string would be required to convert one string to another string. It is normally used to compare the similarity of two strings together with context regarding how a user may incorrectly spell or type

a string. The system that Bard et al. [5] put forth, builds a dictionary which consists of words that are at a minimum an edit distance of 2 apart from each other. This is to ensure that even if a user makes errors of 2 or less during an authentication attempt, there can be no mistake as to which word is being used to authenticate. This allows the users to make spelling errors and the server can automatically correct as long as it's within the error tolerance and deduce what words are being used to authenticate with. This allows the server to use hash methods and employ error correction in it's passphrase scheme. The dictionary method of generating error tolerant words requires that the dictionary be precomputed to ensure that two words cannot be the mistaken as the same word when accounting for error tolerance. The passphrase space that is created by using the dictionary words in a passphrase is determined by how many words you require in the passphrase. Given that we know most humans can remember reliably about 5 related ideas [3], the authors compute the total number of unique passphrases that could be generated with the dictionary. If a dictionary of 2^{16} is generated, it gives each 5 word passphrase a guess effort of 2^{80} or 80 bits of guess effort to completely exhaust the passphrase space if re-ordering of the words in the passphrase are not allowed. If we want to allow re-ordering of the words in the passphrase, a 2^{16} dictionary provides 5 word passphrases $2^{73.1}$ or 73.1 bits of guess effort.

Nielsen et al. [36] design a new prototype method to store user selected passphrases securely on Linux based systems, which allow error tolerance during authentication attempts. The difficulty with error tolerant user selected passphrases is that we need to have the plain text passphrase to be able to determine the edit distance between the user-supplied passphrase and the one that the server keeps on record. Bard et al. [5] rely on the fact that the passphrases are not user selected and that it has a

specially built dictionary which makes it impossible to authenticate successfully with an incorrect word in the passphrase. In this paper however, since users can select their own passphrase, a new method of securely storing the passphrases was developed. The system stores and encrypts the user passphrase in the shadow file; with a personal hardware encrypted AES key that is stored in a secure key-storage file. The key-storage file is similar to the shadow file on Linux systems. Each user's AES key is also encrypted by an internal AES cryptocard with a set AES master key. A secure environment is required to perform the AES encryption, decryption and validation of passphrases. The cryptocard provides a trusted hardware platform to perform unobservable validation tasks for the passphrases. This method provides the ability to have user-selected error tolerant passphrases, but has a number of practical issues. It requires special AES encryption and decryption software to work with the native Linux authentication mechanism, and hardware to perform the secure tasks in an unobservable manner.

Various studies have shown that participants find passphrases easy to use [26, 27, 44, 46]. In multiple studies, participants rated passphrases as easy to use or easier than passwords. In one study, Shay et al. [44] conducted an online user study with Mechanical Turk on system assigned passwords and passphrases. Passphrases were generated based on 8 different conditions. The authors conclude that since the passwords and passphrases are system-assigned secrets, that they are all equal probability. Both the passphrase and password groups are given entropy ratings. Previous research has shown different patterns and constructs in passwords and passphrases, that lead to highly variable amounts of security depending on the policy and threat model [40] [8].

- pp-small (29.3 bits) : assigned four random words from a 181 word dictionary.

- pp-med-unorder (29.3 bits) : assigned four words, randomly selected with replacement from a 401-word dictionary. Unlike all other conditions, participants could enter the words in their passphrase in any order.

- pp-large-3word (29.3 bits) : assigned three words, randomly selected with replacement from a 1,024-word dictionary.

- pp-medium (33.9 bits) : Assigned four words, randomly selected with replacement from the 401-word dictionary used in the pp-med-unorder condition.

- pp-large (39.2 bits) : Assigned four words, randomly selected with replacement from the 1,024-word dictionary used in the pp-large-3word condition.

- pp-sentence (30 bits) : Assigned passphrases of the form "noun verb adjective noun" where nouns, verbs, and adjectives are chosen from separate 181-word dictionaries.

- pp-noun (30 bits) : Assigned four nouns, randomly sampled with replacement from a dictionary containing the 181 most common nouns.

- pp-nouns-instr (30 bits) : The condition is identical to pp-nouns, except that they gave the participant specific instructions for memorizing the passphrase. The instructions asked participants to "try to imagine a scene that includes all of the words in your password phrase".

and passwords were generated based on 3 different conditions:

- pw-length5 (30 bits) : Assigned a password of length 5 from a dictionary of 64 characters including lower case, upper case, digits and symbols.

- pw-pronounce (30 bits) : Assigned an 8 character password which is easily pronounceable in English. All passwords were generated before using a method

- pw-length6 (30 bits) : Same as pw-length5, except one extra character for an increased level of security.

All dictionaries which were used for both the password and passphrase generation was done before creation. The passphrase dictionaries were generated using the most frequent words and part of speech tags from the Corpus of Contemporary American English (COCA). For example, the 401-word dictionary has the top 401 words based on their frequency in COCA. Dictionary size for the generation of passphrases were quite small and the difference in memory recall between a dictionary size of 181 and 1024 was minimal. Users found random 6 character passwords very annoying to use and users found 3 word passphrases "disagreeably" difficult or in other words, not challenging to use. Users found passphrases fun when given instructions on how to construct a mnemonic sentence based on their system assigned password or passphrase. Users did not like being assigned system secrets, and used storage methods such as writing it down to aid in memory. 72% of participants either reported or were observed writing down their secret; storage rates was not significantly different between conditions. This lends to the idea that usability is not negatively correlated with security. There was no significant difference in memory recall rates between the 3 word passphrases and 4 word passphrases, however the number of total characters in a passphrase had an negative effect on all usability metrics. The pp-nouns and pp-sentence groups had slightly longer passphrases than pp-small but performed similarly on usability metrics. The pp-nouns and pp-noun-instr groups performed similarly, the instructions given in the study caused little to no difference in usability metrics. The authors hypothesize that if the instructions had guided users to create a story or a scene, the instructions would have been more effective. Users made more typographical errors when using passphrases, so usability could be improved with an error correction mechanism. The pw-pronounceable group did surprisingly

well, out-performing all other conditions. The authors state that the pronounceable passwords gain their advantage from being all lower case letters and easy to type. The authors state this study did not have user generated passphrases unlike both Keith et al. [26, 27] studies which had user generated passphrases. The study used a crowd sourcing service called "Mechanical Turk" which allows people to be paid small amounts of money for completing tasks; however using Mechanical Turk may not be appropriate for studying human behaviour [2]. It is disputed whether Mechcanical Turk, and other crowd sourcing tools, can be used to gather authenticate data on user password habits [2]. It is highlighted that true password habits can only be observed from real life password environments. Mechanical Turk can be used be used by anyone, and that there is no control on the the type of people which can sign up for Mechanical Turk. However, one study compared results gathered from a real password environment with results gathered from a Mechanical Turk password study. Mazeruk et al. [28] use Mechanical Turk to collect passwords. They juxtapose the results gathered from students that use Carnegie Mellon University's email system with results gathered from a similar password policy in the study. They find similar sentiment and behaviour in password habits, which highlights that regardless of whether real or practice, users behaviour is determined by the password composition policies.

2.3.2 Security

Passphrases are more usable when designed properly because they have memory cognition advantages and therefore memorability. The security benefits of passphrases need to be analyzed too. Traditionally security was measured using entropy. Calculating the randomness of user chosen passwords and passphrases is difficult due to the fact that users do not uniformly pick passwords or passphrases. For the sake of

simplicity, we will assume that every character has an equal chance of being put in a password. It helps us determine a theoretical estimate for the number of guesses or trials it would take to determine the password. For example, if we have an 8 character password that is constrained within the policy of upper and lower case letters and numbers. The formula for calculating the entropy according to NIST, the National Institute of Standards and Technology, of a randomly generated 8-character password is as follows [11].

$$\textit{Shannon's entropy in Bits} = \log_2 64^8 = \textit{48 bits}$$

The entropy calculation assumes there are 8 possible characters in the password and each character can be one of 64 different characters. This is an over estimate as language models can used to determine the probabilities of characters. This is because languages are not random. This means that each of the 64 character possibilites are not even. An example entropy calculation for passphrases is a bit different. A little background information should be covered first though about how users pick their passphrases. Even more so it is important because we know that people are not random and therefore passphrase word choice will not be uniform.

In the first study, Bonneau et al. [8] studied the linguistic properties of multi-word passphrases on the popular online merchant, Amazon. User passphrase selection was not random. Users preferred to choose simple noun bigrams that occur frequently in the English language. Users did not pick passphrases composed of completely random words. Users prefer phrases which represent as a single object (e.g. fascinating book) or single action (e.g. harrowing escape). Almost no passphrases longer than 2 words were found to be registered. Given that users pick passphrases in accordance with

natural language, even 3 or 4 word passphrases can be easily guessed. Passphrases for the Amazon service was also protected with a 4 digit pin that was assigned to the user. This may have reduced the pressure on the user to create a stronger passphrase. Users were also not given any instruction on how to construct a strong passphrase except that it must be a number of words.

In another study, Kuo et al. [30] conducted a user survey of creating mnemonic phrase based passwords. The survey was to see if users created passwords based on easily found/guessed mnemonic phrases. The mnemonic dictionary used in the attacks was gathered from:

- Advertising slogans
- Children's nursery rhymes and songs,
- Movie quotes
- Famous quotations
- Song lyrics
- Television theme songs

129000 phrases were gathered that were used to generate passwords of eight characters or more. The majority of mnemonic phrases that were used by subjects were based on external sources. 65% of mnemonic phrases used by users were easily found with Google searches. It should be noted that in this study, passphrases were not used but rather mnemonic phrases were used to generate a password. An example is if I had the mnemonic phrase "Soft kitty, warm kitty, little ball of fur" my password could be "Sk,Wk,Lbof". While this study is not directly about passphrases, its research into how users pick mnemonic phrases is significant. It shows that users tend

to pick phrases from their favourite pop culture references.

This leads us to why people use known phrases for passphrases. A mnemonic is a form of a memory aid, used to help improve the memorability of a memory. A phrase is inherently mnemonic. It is a chain of words that share a common theme or idea. That theme can help cue user's memory to the words that make up the passphrase. Mnemonics can help users create stronger passwords that are harder to crack [52]. In one study on mnemonic based system-assigned passwords conducted by Juang et al. [25], users were put in one of three groups.

1. No guide on how to use mnemonics.
2. User generated mnemonics – users were instructed how to create a mnemonic.
3. System generated mnemonics – users were randomly assigned a mnemonic, which was constructed by taking each letter from the assigned password and finding a word that started with that letter from a pre-built word list.

The users were asked to create three accounts using the method corresponding to the group. The mnemonic-based groups also were required to create a visual representation of their mnemonic phrase using either hand drawn images or images from a Google image search. There was no limit to how much they could draw. The users were shown the image upon a login attempt to help cue their mnemonic based password. In the first session, users create their password and then did five minutes of basic arithmetic. After the arithmetic they were asked to recall the password to one of their three accounts. The task was repeated for all three accounts. They had five attempts to login for each account. One week later they were invited back for session two to repeat the same login procedure as session one.

The results of the study were a great success for the mnemonic groups, especially

the system assigned mnemonic groups. The first-session login success rate was 87% for system generated, 53% for user generated and 38% for none. The recall success rate was better in session one and two for system assigned mnemonics than the user assigned or none. Users also rated the usability of system-assigned mnemonics higher than user-assigned or none. Although the recall and creation time for system-assigned mnemonics was longer, the initial time investment to create is a one time event; whereas forgetting a password can create even greater amounts of lost time in the future.

One research idea was put forth by Jeyaraman et al. [24] to generate mnemonic sentences from randomly generated passwords to improve memorability and in turn, increase the usability of randomly generated passwords. Reuter's corpus was used to compile sentences to be used for mnemonic sentences. This follows in line with the idea that distinct sentences are more "memorable" than other sentences [12] which is discussed further in Section 2.4.2. Using a corpus of 1000 sentences, they were successful in finding a mnemonic sentence for 80.2% passwords with 6 lower case letters. For passwords with 7 lower case letters, 62.7% could be paired with a mnemonic sentence. It should be noted that the matching of mnemonics is quoted to be as computationally expensive as brute-force attacking passwords. While this is only a research idea and not a study, the work done is based around the idea that mnemonics improve memorability and therefore the usability of passwords. The idea also focuses solely on passwords, not passphrases which we are interested in. It might be even better if the mnemonic phrase itself was the text used for authentication.

Now that we know how users select passphrases given the freedom to select their own passphrase, we can estimate the theoretical security in terms of the number of guess attempts. We calculate the number of guess attempts an attacker would need to make in order to have an measurable chance of successfully guessing correctly;

to know how well passphrases compare to passwords. Guess attempts are used over entropy because entropy does not factor in human selection, or language models, which are not random. Passphrase guess attempts are calculated based on words instead of characters. This is because users are instructed to use multiple words instead of characters for the construction of a passphrase. If we were to construct a dictionary to guess passphrases, we would use permutations of common words in phrases.

If we assume that every word in the English dictionary has an even chance of being picked in a 5 word passphrase, the potential high-end estimate of the theoretical security is:

$$\textit{Theoretical Security in Bits} = \log_2 500000^5 = \textit{94} \quad \textit{bits}$$

For the dictionary size we are using the Corpus of Contemporary American English(COCA) [16]. It is the largest corpus of American English that is available. Assuming all words are equally probable and independent, it means that a 5-word passphrase offers 94 bits of entropy, compared to 48 for the standard 8 character password. However, we know that users do not randomly pick passphrases. It is highly likely that users will only use a fraction of the total amount of words in COCA. Zechmeister et al. [53] discovered that the average junior highschool student has a lexicon of 10,000 to 12,000 words, with college to university students reaching up to 17,000 words. If we assume that users only pick from the top 10,000 most common words we get a theoretical security of

$$\textit{Theoretical Security in Bits} = \log_2 10000^5 = \textit{66} \quad \textit{bits}$$

66 bits of entropy is still much stronger than the 48 bits for passwords. These passphrase calculations are also assuming that people even use words in the first place. There is nothing stopping people from using slang or even gibberish that is unknown to other people but highly memorable for the person who makes the passphrase. This would increase entropy greatly as an attacker would then have to guess random strings in addition to common words.

2.4 Passphrase Creation Strategies

Passphrases are more memorable than passwords while potentially providing more security than a traditional password [8] [48] [52]. Phrases are very common in all aspects of the English language, especially in media. There are quotes, poems, idioms, and other various forms of short phrases which stand on their own. "What makes these phrases memorable" is an interesting question. If we could somehow reverse engineer a formula for creating a memorable phrase, it could be leveraged for creating and guessing memorable passphrases. Passphrases are different from passwords in the sense that they tend to be composed like a phrase or a sentence. In Section 2.4.1 we discuss stories which have been harnessed by humans for centuries as a means to pass down information from generation to generation. In Section 2.4.2 we discuss the characteristics of memorable phrases.

2.4.1 Stories

The human mind has its talents and flaws, strengths and weaknesses. It seems like the natural choice to leverage the advantages that our brains have developed over generations of evolution. Stories have been used for centuries by humans to pass down information from generation to generation. Whether the story is used for entertain-

ment or to teach people about a culture, they are a staple in human communication. Stories are used by humans because they improve memory recall and memorization. In a study by Carminatti et al. [13] they discuss the process of recalling stories and events. They find that in general people are better at recalling memories if the memory is approached like a story. It was even more beneficial if the story was recalled in a group setting with multiple participants. Simply interviewing a participant asking "what happened" was not very effective or accurate at recalling a memory. In the study, he has one group of participants that are individually interviewed about the story and two other groups which as a group, recall the story. It is found that both groups that have to retell the story rather than interview about the story fare much better at accurately retelling the story. Our brains are better at retelling a story than simply answering questions about the story.

The idea that humans are better at recalling stories rather than random facts is useful for passphrases because passphrases can be approached like a story. They have a meaning and a central idea they communicate. Stories can invoke an emotional feeling; whenever they think of that story they may recall the emotion that is associated with it.

Emotions can also help people recall memories [31]. A passphrase can be a story and have an emotion linked to it. In the study by Laird et al. [31], participants were told to read an either funny or sad story. Afterwards, two groups of users were asked to recall the story. The users in one group, the controlled cue, were told to either frown for the funny stories and smile for the sad story. The second group, the self cue, users could either frown or smile for both stories. The recall rate was significantly higher for users who smiled for the funny stories and frowned for the sad stories. The self produced cue group had the most significant results. 30% of the participants

successfully recalled the funny story when smiling versus 20% who frowned. 80% successfully recalled the sad story when frowning and only 60% successfully recalled when smiling for the sad story. This leads us to believe that emotions do have an effect on memorability. Having congruent emotions attached to memories can have a positive role on successfully recalling memories. This can be harnessed by passphrase creation policies. If users are free chose their own passphrases and are given recommendations to use memories as basis for a passphrase, users may be more likely to pick a passphrase which can provoke emotions and therefore potentially improving memorability.

2.4.2 Phrase Memorability

In one study by Danescu-Niculescu-Mizil et al. [12] they looked at the phrasing of certain quotes and studied what makes a quote or phrase memorable. They found a set of characteristics which can increase the memorability of a phrase or quote. They extracted and constructed a corpus from 1000 movies across a variety of genres, decades, and popularity. For each movie, memorable quotes were selected by using IMDb's memorable quotes Section from the movies' page. Memorable quotes that were multiline or multi sentence were not used because generally they included a "build up" and "punch line", so in order to isolate the memorable part, single line memorable quotes were only used. Next, for each memorable quote M, an equivalent non-memorable quote N was required for comparative testing. N was selected by its similarity, except in choice of words, to the memorable quote M. M and N quotes were as close in length as possible, spoken by the same character and close in location within the movie script. After this process, a collection of pairwise quotes M and N is generated. The pairwise quotes are:

1. similar in length

2. spoken by the same character

3. located close to one another

4. N is not in the memorable quotes section on IMDB, where M is

After collecting 2200 pairs of M and N quotes, a small pilot study consisting of 6 native English speakers were presented 11 or 12 pairwise quotes. The quotes were randomized and were from movies that the subjects said they had not seen before. They were asked to select the more memorable quote. On average the subjects selected 75% of the time the more memorable quotes, with some subjects even performing much better than 75%. Test subjects were able to identify the more memorable quote of the two even if they had not seen the movie. This implies that memorable quotes do tend to have distinct characteristics. The characteristics the authors identified are:

- Turn of phrase i.e., how the phrase is worded. A memorable phrase has an artful turn of phrase. An example is

  "To be or not to be, that is the question" - *William Shakespeare*

  In this quote, it has an artful turn of phrase because it is alluding at the idea of being alive or dead. The quote dances around the idea of one's mortality, without specifically mentioning death.

- Generality, meaning the phrase can be taken out of context and applied to other situations. An example is

  "Luke, I am your father" - *Darth Vader*

  This is a memorable quote because it has been used by many other people in real life and in literature.

- Lexical Distinctiveness, which can be summed up as less common word choices but common syntactic phrasing patterns. It is suggested that the lexical distinctiveness of phrases can be a key factor in memorable quotes. An example is

  "Ay, carumba!" - *Bart Simpson*

  It is a famous quote because no one else says this and the character who coined this quote, is on a famous animated television show. No one else except Bart Simpson is notable for using this phrase.

To test the theory, a language model approach was used. In brief, the language model proposed in this paper determined that in 60% of the quote pairs, the memorable quote was more distinctive based on these three principles. Passphrases can the same characteristics of memorable phrases to potentially improve memorability. Advising users to use turn of phrase and generality in a concise and understandable way is difficult. However, using lexical distinctiveness by suggesting uses of unique word choices is possible.

2.5 Summary

We acknowledge with our research that passwords are here to stay but are also motivated by our research that it is possible to improve end user text based authentication. Passwords have been, and will continue to be the dominant method of authentication for virtually all websites. This is primarily because of three factors: usability, deployability and security. User familiarity with passwords, the ability to easily create a service using passwords and a potentially moderate amount of security are what keep passwords as the most used form of authentication.

Passwords at a basic level, are merely a string of characters, being most commonly letters but sometimes including numbers and symbols. Password policies are the instructions which users follow when creating passwords. Most password policies which have the intention of improving usability and security, tend to cause the opposite. This is a failing to understand human behaviour. Users do want to maintain security of their data but policies make it difficult to do so. The policies end up reducing productivity and make it difficult to use the service or data that which they safeguard. This is observed as the larger the user community is, generally the laxer the password policy is, as to not hinder or annoy the user community. The most well known password creation policy is the traditional 8 character policy. This policy commonly includes at least 8 characters including one upper case, one lower case, one number and one symbol. It is a terrible policy as it does not provide ample security; it is unusable as the average person can only remember roughly 5 independent ideas at once, and due to the requirement of symbols and numbers, using some sort of grammatically correct word or phrase is not possible.

Passphrases are not going to fix all the problems with end user text-based authentication although they have the potential to improve usability and security if users can be instructed to use them properly. Passphrases can lead to more typographical errors than passwords but no more memory based errors than passwords. Typographical errors can be minimized by using error correction techniques such as edit distance tolerance and error tolerant dictionaries to generate passphrases. A tolerance of 12.5% for edit distance appears to be the magic number for passing imperfect authentication attempts. Users are unfamiliar with passphrases and require more time to become comfortable with using them for login purposes; although if passphrases become more widely adopted, this may change. Passphrase creation policies should enforce a greater length and be similar to a sentence; one author called it “word pro-

cessing mode" which takes after programs which are used to write formal documents, such as this thesis. Users do not like to be assigned a password or passphrase and will use storage methods such as writing it down to compensate.

Passphrases' level of security is closely related to the policy which guided the user during creation. Passphrases, just like passwords, are not random when the user selects them. Users tend to pick words in a passphrase which represent a single idea or thought. Users also tend to pick passphrases which are known from famous phrases such as movie quotes, titles, song lyrics and other pop culture sources. However, passphrases have the potential to be much more random than passwords. The total space which must be enumerated by an attacker to cover the majority of passphrases is much larger than for passphrases. If we give users good passphrase creation policies, passphrases have the potential to be very secure.

Instructing users to pick passphrases which have a special meaning or tell a message such as a story or joke might prove to be more memorable. Certain characteristics, such as lexical distinctiveness, make phrases more memorable. It is possible to identify these characteristics and include them in passphrase creation guidelines to potentially improve memorability. Our creation guidelines in Section 3, were inspired by these findings discussed in literature review.

3 The StoryPass System

As computing power increases and with the rise of distributed high power computing, the ability to enumerate large amounts of guesses in a short time period is a swelling issue. Designing password policies becomes a cat and mouse game, with other balancing factors at play. Should organizations raise the minimum number of characters or words required in a passphrase to make it more difficult for offline attacks; or require more complex patterns such as more character classes (i.e., symbols, numbers, upper and lower case letters). Unfortunately, factors which cause an increase of security, tend to cause usability issues. Users can not sacrifice usability for security consciously or subconsciously; users will sacrifice security to make it usable [29].

StoryPass is the name of the custom system we designed perform the user study under the set of passphrase creation guidelines we aim to study. We were motivated to build a custom system as there are a number of variables which we want to control and measure to accurately perform our analysis. We needed a system that we could control for over a month to simulate real life passphrase use. We named it StoryPass as our literature review suggested to us that creating story like passphrases may provide innate memorability benefits; By encouraging users to create passphrases with the characteristics and structure of a story we can hopefully create very memorable, usable, and secure passphrases.

In this thesis, we refer to the passphrases created in StoryPass as *passphrases*. However, in our user study, we referred to the passphrase as a StoryPass. (See Figure 1 and 2)

3.1 User Study System

StoryPass, a custom user study system, was created and deployed for the user study. Figure 1 shows a screen shot of the page that the user sees when practicing creating a StoryPass. The practice and real creation pages are exactly the same except for wording which tells the user to use this time to practice creating a passphrase.

3.1.1 StoryPass Creation Policy and Guidelines

Participants were required to adhere to some conditions during creation. (As seen in Figure 1). We discuss how we decided upon this policy in Section 3.2.

We require:

- At least 7 words in length
- Use at least one proper noun (i.e., names, places, things) such as 'McDonalds' or 'ThinkPad'

We recommend:

- Use slang or other words not found in a dictionary (e.g., 'bazinga' or 'wazzup')
- Not including sequences of common three word phrases (e.g., 'it is a' or 'all of it')

Behind the scenes, we also checked users passphrases for any three, four and five words sequences in the top 10,000 3, 4 and 5 n-gram lists. This is done to help provide a minimum level of security. We cover this more in Section 3.3.

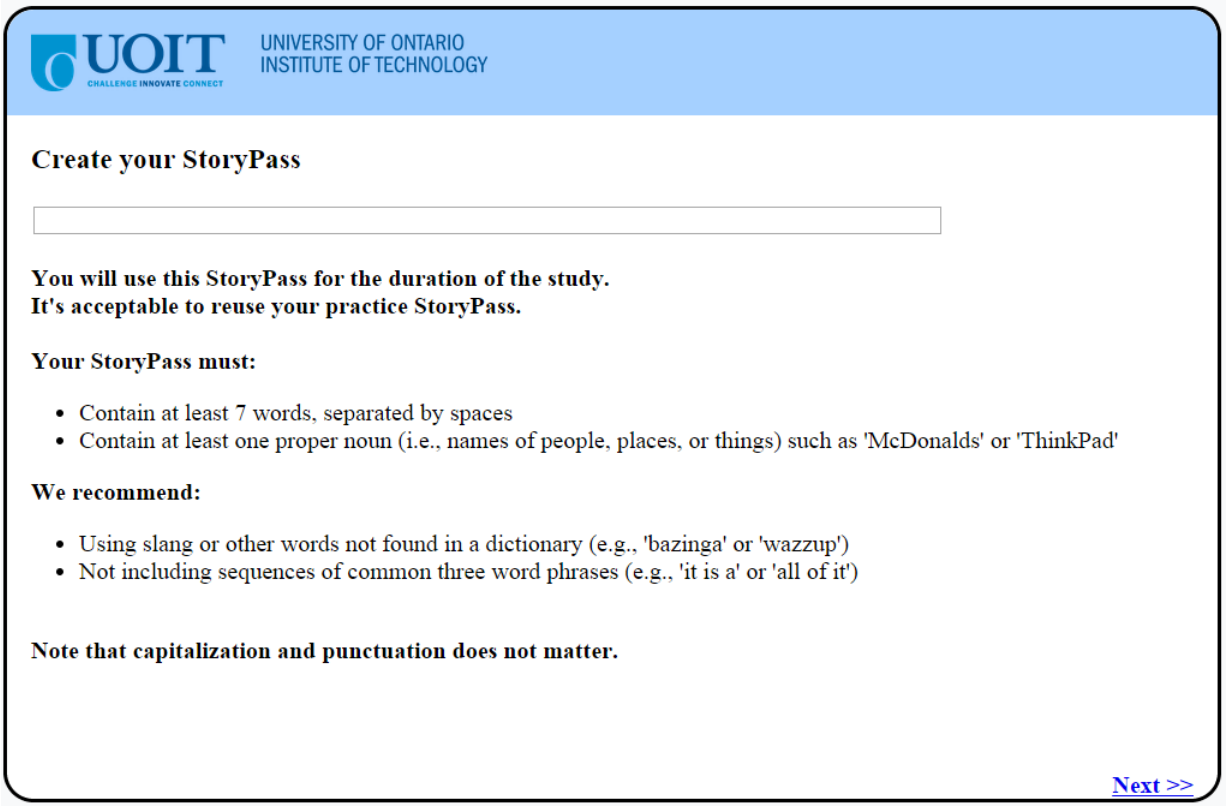

Figure 1: The StoryPass creation screen

The cue creation screen in Figure 2 was immediately after the passphrase creation screen. The purpose of the cue was to provide a mnemonic for the users passphrase. Mnemonics can potentially improve memorability as discussed in 2.3.2.

The cue creation had no required conditions, however we had two related instructions:

- Your cue can be an object, or a memory. Anything really! Examples might be a book, business card or a funny memory

- This cue will serve a memorable reminder of your StoryPass that you created on the previous page

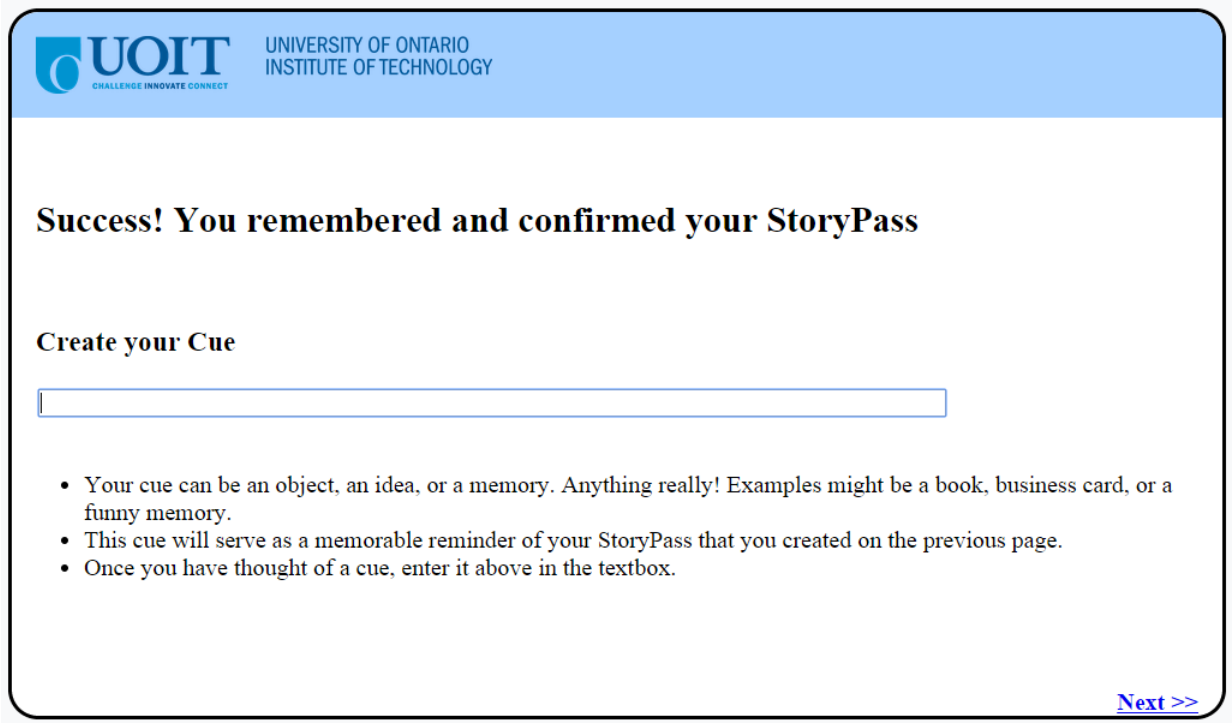

Figure 2: The mnemonic cue creation screen

Figure 2 shows a screen shot of the page that the user sees when practicing to create a cue for their StoryPass.

3.1.2 Error Correction

In our system, we allow for users to make typographical errors during login. This is to help with the usability of passphrases. It is known that the longer the required passphrase, the more typing errors are likely to occur [27]. We can counter this by allowing users to make errors in the confirmation of their StoryPass. Our error correction is enabled by using a function called the Levenshtein Distance. This function compares two strings together and calculates the number of single character edits, e.g., substitutions, deletions, or insertion needed to change one of the strings to another. A substitution is the replace of a character, for example “cat” and “hat”, have a substitution edit distance of 1. A deletion is the removal of a character, for example

"cat" and "at" have a deletion edit distance of 1. An insertion is the addition of an additional character, for example, "cat" and "catt" have an edit distance of 1. Our system allows an edit distance of 0.125 when a users verifies their StoryPass. This equates to a single error per eight characters. This value was determined by a study run by Keith et al. [27]. We discuss further below in Section 3.3.12 the impact this has on security.

3.2 Design of StoryPass and Mnemonic Cue Creation Policy

There are two requirements and two recommendations in the creation guidelines.

The first requirement is the use of at least 7 words, separated by spaces. We require at least 7 words because our theoretical security analysis (see Section 3.3.8), revealed that we would need 7 words to reach a sufficient theoretical security threshold for high-security use cases such as master passwords for password managers.

The second requirement of at least one proper noun is to improve security. By requiring proper nouns, we increase the difficulty of enumerating large amount of possible passphrases by an attacker, and possibly make the passphrase more distinct and lexically diverse. This is ideal as distinct and lexically diverse phrases are more memorable, as covered in Section 2.4.2.

We have two recommendations in our passphrase creation policy. We recommend the use of slang in passphrases as it improves security and usability. Similar to the requirement of proper nouns, many slang terms can be used as a noun. Slang terms are often unique and difficult to enumerate from an attacker's perspective. In Section 3.3.11, we show that by enumerating slang terms found on the UrbanDictionary.com website, we can estimate the total guess effort it would take to exhaust

all 7 word passphrases that include one slang term. We blacklist the top 10,000 3-grams, 4-grams and 5-grams from being included in passphrases. In Section 3.3.9 of our theoretical analysis, we show that not blacklisting the top 10,000 2-grams greatly improves usability. This is why we recommend to users to not include common three word sequences.

After a user creates a passphrase in StoryPass, they must make a mnemonic cue to help aid the memory of their passphrase. Our mnemonic cue creation has only two recommendations. The first recommendation is using the cue as a memorable reminder of their passphrase. The second recommendation is using objects or a memory, as using a objects or memories related to the passphrase may help memorability.

Passphrases and passwords which require longer character requirements over symbols and numbers have been shown to improve security while maintaining usability as highlighted in Section 2.3.1. Rainbow tables are available for MD5 and SHA-1 hashes, which are used for storing a password, to the public for passwords up to 10 lower case alphabetic characters [39]. Using traditional passwords is not a very effective strategy to protect against offline attacks. To counter this, the easy solution is to increase the length of the passwords being made. However when you increase the length of a traditional password, you tend to run into issues with memorability and therefore usability. In our next section, we perform a theoretical security analysis to see if our passphrase creation guidelines are strong enough to prevent different possible threat models.

3.3 Theoretical Security Analysis

There is little research on the creation of passphrases; specifically what the creation policies should be when recommending and guiding users on creating secure and usable long passphrases. Previous work has shown that short, 3 to 4 word passphrases are weak when chosen by users, as covered in Section 2.3.2. There is a need for understanding how users can make long secure and usable passphrases. To better understand, we imagine ourselves as an attacker who would want to guess passphrases that would be created with the StoryPass passphrase creation policy. In our theoretical model and section below, we assume an attacker who has unthrottled login attempts to our passphrase system and that they are motivated to guess a target passphrase. We assume this to ensure that the worst case scenario is covered in setting security-related parameters; we aim to have these parameters provide a minimum level of security to the system's users. In the next section, we first describe our n-gram lists and UrbanDictionary.com list, followed by possible attacker strategies.

3.3.1 N-gram Lists and UrbanDictionary.com List

The main data source that we use in our theoretical analysis are n-grams and a custom list composed of slang terms available from UrbanDictionary.com [37]. N-grams in our case are a sequence of words observed from the Corpus of Contemporary American English (COCA) [16]. The n-grams range in size from 1 word to 5 words. They are ranked based on frequency, which is determined by counting how many times each unique n-gram appears in COCA. COCA is comprised of 450 million words of American English text and is equally derived from spoken, fiction, non fiction, magazines, newspapers and academic research. It has 20 million words from each year,

1990 to 2012. It provides an excellent varied source of written and spoken contemporary American English. There are 492,630 1-grams, 48,759,993 2-grams, 179,113,521 3-grams, 331,925,953 4-grams, and 1,295,695 5-grams. It is important to note that the 5-gram table was artificially limited to 1.2 million n-grams by the authors.
The UrbanDictionary.com list provides a survey of the type of slang and non dictionary terms which are generated by the world wide web. It has 681,981 unique words, phrases and acronyms. These slang terms have a total of 7.5 million definitions [37]. Anyone can make a definition on UrbanDictionary.com, which provides an excellent source for gathering slang which is sparse and non standard in nature. We made this list by using a custom built web scraper script. Unlike the n-gram lists, it is not ranked and contains no part of speech tags.

3.3.2 N-gram Joins

An important technique that we perform with n-gram lists are joins. Joins are used to help enumerate passphrase guess attempts. We join n-grams together that have the same word as the trailing or leading word. This is done is by checking the trailing word or starting word of an n-gram to see if it matches the starting or trailing word of the next n-gram. As an example, consider the follow two n-grams below.

5 Gram: (Reading a science fantasy book)
3 Gram: (book cover page)
Join result:*Reading a science fantasy book cover page*

We assume that the joined n-grams give a possible phrase which could exist in the English language.

3.3.3 Attacker Strategies

Below are the two methods by which an attacker would best approach guessing passphrases created with our StoryPass creation policy.

1. **Frequency/ranking modelling.** The attacker tries to build passphrases based upon n-grams and words, using frequency modelling. The n-grams are from COCA, the words are from the COCA lexicon. We determine the probability of some of the phrases appearing using n-grams ranging in size from 1 to 5 words to generate passphrases based on probability. By generating a passphrase, we are assuming that n-grams can be joined together, by similar words, to model what users will choose for their passphrase. This method provides a fast method of determining an estimated level of guess effort that a passphrase can provide; however, it does have some drawbacks. It assumes that the attacker is building optimal guessing lists made up of n-grams. The guessing lists are trained, which depending on your training data, can have a dramatic effect on the outcome of the guessing algorithm. Our training data is COCA, which is a large sample of written English. While it is an excellent source of written language, it is not composed of passphrases, but English taken from a variety of (non-passphrase) written contexts. We make the assumption that it provides enough context to provide relevant training data for our n-gram guessing lists. We use this method for modelling the potential security that passphrases can provide.

2. **Brute force/dictionary.** This method involves gathering and generating as many phrases as possible. This method, while somewhat crude, could be very effective. This is because the phrases are collected from a limited number of human written sources. However, this method requires collecting of vast amounts of unique phrases, which is very computationally expensive. It has been shown that humans tend to pick popular phrases to use as passphrases [30]. This method is explored in Section 5.1.10, on the passphrases collected in our user study.

3.3.4 Modelling with N-grams

We use n-grams and a formula called Marginal Guesswork to determine how many guesses it would take to guess N passphrases under a known probability distribution (i.e., assuming that the attacker knows the optimal order of passphrases to be guessed, in the order of most probable to least probable). We use n-grams to estimate the probability distribution of passphrases because n-grams are modelled after the English language. This is because in our guidelines, we are encouraging users to write a short story/phrase, which we assume will resemble natural language. Previous research has shown that users tend to pick passphrases that are similar in structure to the English language [40]. If the probability models after the distribution of the passphrases and if the top 5% of the guesses has a summed probability of 50%, the distribution of passphrases make it viable and advantageous to not guess all the passphrases, but only the top 5%. We can employ the blacklisting of certain n-grams to help improve the theoretical security when considering marginal guesswork.

3.3.5 Marginal Guesswork

We use marginal guesswork for determining the amount of effort it will take to guess a certain number of passphrases. It is used because it can determine how much effort required is needed to guess only a certain percentage of the total passphrases. It most cases, you only need to guess a certain percentage, it is not required to guess the entire data set. Marginal guesswork is defined by:

$$w_\alpha(X) = min\{i | \sum_{j=1}^{i} p[j] \geq \alpha\} \tag{1}$$

Marginal guesswork measures the maximum cost of determining the value of X when we wish to be guaranteed an appreciable chance of success (the probability of α) in a brute-force search [38]. i is the rank of the possible non-uniform outcome being searched for if all events are ranked according to probability. It exploits the uneven distribution of passphrase selection since humans are not truly random. It is probable you can guess a disproportionate amount of passphrases with a small amount of guesses. This is because we can use language modelling to determine what words and phrases are most likely to appear in user selected passphrases. An attacker to guess the most passphrases with the lowest amount of guesswork possible. If we look at the marginal guesswork graphs below in Figure 3 and 4 we can see how much effort it will take for an attacker to guess a certain percentage of n-grams. Marginal guesswork is calculated by ordering the n-grams that would be used by the attacker in order of the highest probability to the lowest probability, then summing the probabilities in descending order. This can determine at what point in guessing effort a certain percentage of n-grams would be guessed. This can be useful for looking at the distribution of n-grams. We want to avoid having very easily guessed passphrases, so we wish to prevent users from selecting very easily guessable passphrases. We can

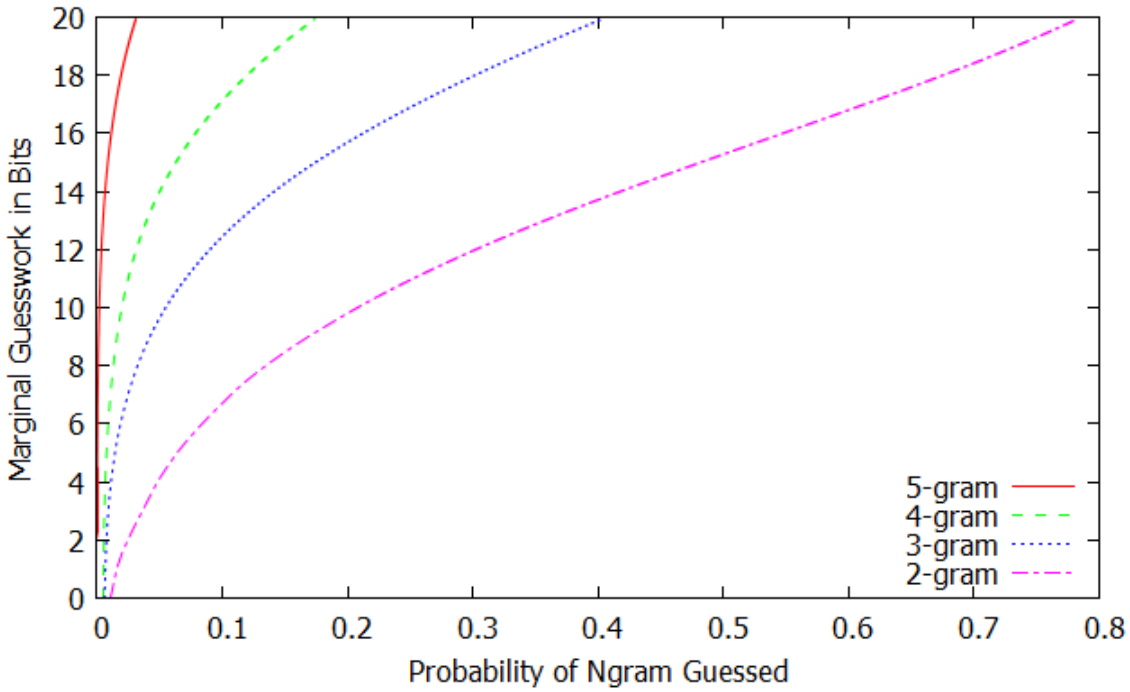

Figure 3: Marginal guesswork of the top 1 million 2-gram, 3-gram, 4-gram, and 5-gram.

vastly improve the distribution of passphrases created in StoryPass by blacklisting even the top 1% of the most common n-grams. In our system we blacklist the top 10,000 n-grams from the 3, 4 and 5 gram tables. We discuss this more in Section 3.3.6 above.

Figure 3 shows the marginal guesswork for the 3, 4, and 5 grams with the top 1 million n-grams. Figure 4 shows the same n-grams except the top 10,000 most common n-grams are removed. Most modern password crackers work by trying to guess the most passwords in the least amount of time. They exploit the fact that users pick poorly chosen passwords. By removing the top 10,000 most common n-grams we remove the ability to guess a large portion of passphrases easily. We are improving the distribution of passphrases by essentially blacklisting the passwords which would be the target of most attacks. When a user is creating a passphrase, StoryPass proac-

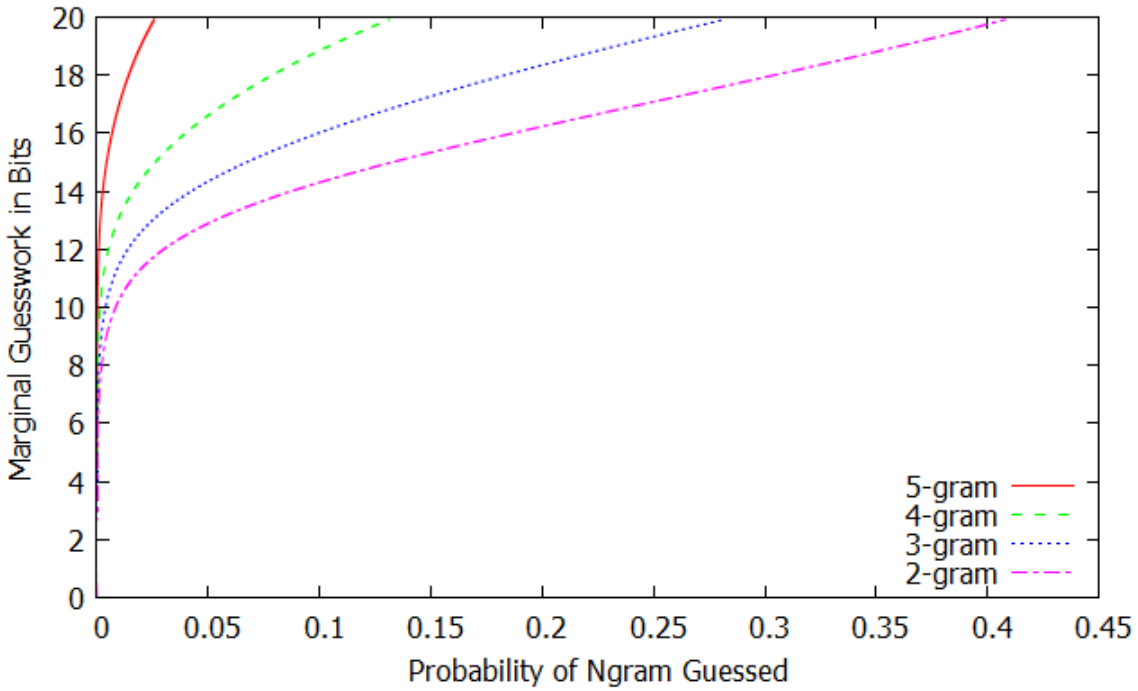

Figure 4: Marginal guesswork of the top 1 million 2-gram, 3-gram, 4-gram, and 5-gram with the top 10,000 removed.

tively checks the passphrase, looking for any sequences of words that are found in the top 10,000 of the 3-gram, 4-gram, and 5-gram n-gram tables.

3.3.6 Blacklisting Improves Security

How does the blacklisting of the top 10,000 n-grams from the 3, 4 and 5 list translate into a gain in security? If we assume that our system enforces a 7 word passphrase, we can determine a worst case scenario to help solve this problem.

Firstly, why do we pick 10,000 as the cut off? We chose the value 10,000 to blacklist because of the probability distribution of n-grams. As shown in Figure 3, at the

10,000 guesses mark or roughly 13 bits of marginal guesswork, the distribution of n-grams starts to flatten out. This implies that the average probability of each n-gram begins to be more equal, which is what we want to achieve. In an ideal passphrase system, every single passphrase would have an equal probability. If every n-gram that is allowed to be used in a passphrase has an equal probability, we have an even probability distribution, implying that no n-gram is more probable than another. An even distribution is advantageous because it gives an attacker who wants to guess our passphase no attack method that can guess passphrases using n-grams faster than guessing randomly. However, achieving an equi-probable distribution is not possible by blacklisting and is very unlikely when user choice is permitted. Additionally, It has been noted that the average high school student knows roughly 10,000 words [53] and with our pilot testing, we determined that blacklisting with a cut off at 10,000 offered the best trade off versus security. A higher value could have been used but the system would start to reject many more passphrases. A lower value could have also been used; however, it would allow weak passphrases to be made and therefore would be easier for an attacker to guess.

Assuming the worst case scenario passphrase, the user picks a passphrase that could be found by joining a 5-gram and 3-gram from the COCA n-gram lists. This is because 5-grams have more words being used in a correct context than the smaller n-grams, which makes it a logical choice to be the first choice when trying to guess a passphrase. We need the other 3-gram to be joined with the 5-gram to make a 7 word passphrase.

If a user is able to pick a passphrase that is composed of n-grams from the top

10,000 n-grams, it allows the attacker to easily guess highly probable passphrases. Let's assume attacker is able to make 1,024 guesses.

In a scenario where a user can pick from the top 10,000 n-grams, an attacker would have a small chance of correctly guessing a passphrase. After 1,024 guesses, an attacker would have roughly a .185 % chance to guess just the 5-gram and after 1,024 guesses, an attacker would have a 0.3822% chance to guess the 3-gram. Even then, an attacker can't guess these independently. It would require 1,024 * 1,024 = 1048576 guesses to have a 0.0707% chance to guess a poorly chosen 7-word passphrase.

Compare the above scenario to a scenario where the top 10,000 n-grams from the 3, 4, and 5 n-gram list are blacklisted, and the probability distribution is much better. After 1,024 guesses, an attacker would have a 0.0225493% chance to guess the 5-gram and a 0.182% chance to correctly the 3-gram. Once again since the n-grams cannot be guessed independently and must be guessed as a whole string, it would require 1048576 guesses to have a 0.00411013306876% chance to correctly guess the whole phrase. By blacklisting the top 10,000 3, 4 and 5 n-grams from our users passphrases, we can greatly improve the security against an attacker's best effort at guessing passphrases, assuming the passphrases created in StoryPass follows the same patterns as regular English.

In Table 1, we have analysed the distribution of n-grams in depth, especially the top 1 million n-grams. For 2-grams after making 1 million guesses, an attacker would have an estimated 75% chance of guessing that n-gram.

With the blacklisting effect shown in Table 1 in mind, we see that it would be ideal for our StoryPass system to reject any passphrases that have 3-grams, 4-grams

n-gram Size	Probability of Guessing n-gram after 1 Million Guesses	Probability of guessing n-gram after 1 Million Guesses (after top 10,000 n-grams blacklisted)
2-gram	75%	42%
3-gram	40%	27%
4-gram	17%	13%
5-gram	3%	2.5%

Table 1: The total probability of the top 1 million n-grams with and without the top 10,000 most frequent n-grams

or 5-grams in the top 10,000 most frequent n-grams. By blacklisting the top 10,000 n-grams from each, we can even out the distribution of user chosen passphrases. In Section 3.3.9, we describe why we do not blacklist the top 10,000 2-grams.

3.3.7 Setting a Guess Threshold

NIST recommends an 80 bit guess threshold for user created passwords as specified in the Electronic Authentication Guideline [11]. Additionally, current computed rainbow tables [39] of all hashes for lower case alphabet only passwords from 1 to 10 characters are available free online. Rainbow tables can be used to launch very fast brute force attacks on cryptographic hashes which are used to store strings such as passwords and passphrases. An appropriate threshold should be equal to or above these standards to avoid offline brute force attacks. Our requirement of 7 words in the passphrase creation policy sets a high threshold for enumerating all possible combinations of guesses. With the average word of 5.1 characters, an average length passphrase will be 35.7 characters. If we calculate all the 35.7 length combinations of the lower case alphabet, numeric, and space characters, we can get a total high end of guess effort(A) of a brute force attack. $A = log2(37^{35.7}) = 185.977 bits$

185 bits greatly exceeds NIST's recommendations but, user selection of passwords

and passphrases is not uniform. Modern password crackers leverage word lists and known dictionary attacks to perform offline attacks which outperform brute force attacks. However, no known precompiled passphrase list or dictionary exists which is congruent with our creation policy. To estimate the guess effort required for human selected passphrases, we will use the COCA database to make combinations of n-grams to resemble human selected phrases.

To ensure that a passphrase created in StoryPass provides a high level of security, we assume an attacker would build a guessing dictionary by generating all 7 word combinations with n-grams, we estimate how many total guesses that would take in Section 3.3.8 .

3.3.8 N-gram Combination Security Estimate

To find an upper bound for the attacker, we enumerate the number of joins that can be made in our n-grams table. We calculate the total number of joins by enumerating how many n-grams end with each word that appears in the data set. We can then multiply the counts for each n-gram together to determine how many joins can be made for 7 word passphrases.

Assuming an attacker wants to guess all n-gram combinations, the best list for an attacker with unlimited resources would be the combination of 2 + 2 + 2 + 2 + 2 + 2 + 2 because all of the combinations that appear in this join also appear in all the

n-gram combination	Guesses (Total count)	Total Guess Space (log2)
5 gram	1,283,537	20$\tilde{}$bits
4+2	9,699,924,863	33$\tilde{}$bits
3+3	14,070,011,809	34$\tilde{}$bits
3+2+2	191,557,413,657,821	47$\tilde{}$bits
2+2+2+2	2,935,122,350,891,525,134	61$\tilde{}$bits

Table 2: The number of 5 word phrases which can be made with different combinations of n-grams. * The 5 gram count was artificially limited by the creators of COCA.

6 Gram Combos of n-grams	Guesses (Total count)	Total Guess Space (log2)
5+2 gram	1,434,334,222	30$\tilde{}$bits
4+3	134,232,243,8443	36$\tilde{}$bits
4+3	565,331,337,984	39$\tilde{}$bits
4+2+2	38,392,884,671,383,263	55$\tilde{}$bits
3+2+2+2	298,720,376,151,321,744	58$\tilde{}$bits
2+2+2+2+2	15,935,382,010,112,724,924	63$\tilde{}$bits

Table 3: The number of 6 word phrases which can be made with different combinations of n-grams.

7 Gram Combos of n-grams	Guesses (Total count)	Total Guess Space (log2)
5+3 gram	1,417,764,398	30$\tilde{}$bits
5+2+2	394,341,239,321	38$\tilde{}$bits
4+4	18,300,329,475	34$\tilde{}$bits
4+3+2	49,203,349,121	35$\tilde{}$bits
4+2+2+2	323,494,001,239	38$\tilde{}$bits
3+3+3	584,254,394,554,391	49$\tilde{}$bits
3+3+2+2	54,954,112,453,203,853	55$\tilde{}$bits
3+2+2+2+2	1,152,921,504,606,846,976	60$\tilde{}$bits
2+2+2+2+2+2	140,115,309,088,333,521,056	66$\tilde{}$bits

Table 4: The number of 7 word phrases which can be made with different combinations of n-grams.

other combinations. This is because the 2-gram, 3-gram and 4-gram were trained on the same set of data. This is likely an overestimate of the maximum number of guesses an attacker would have to make to guess a passphrase from our system. When we are joining 2 word n-grams, we are not filtering for proper grammar rules or syntactic rules. This means some of the guesses may not follow proper English grammar rules. This value of 66 bits of guess effort can be used to show why we recommend at least 7 words for a passphrase. However, we consider this an underestimate for a number of reasons. The 66 bits only included n-grams, it does not consider the possible guess effort required by the inclusion of at least one proper noun and the recommendation of slang. This would require more guess effort by the attacker. Proper nouns and slang can potentially increase security against guess effort attacks as covered in Section 3.3.10 and 3.3.11. This estimate turns out to be much lower than the user study results for some passphrases in Figure 6 for a number of reasons. It also considers 1-grams which are necessary to guess any of the passphrases collected, and greatly increase the guess effort.

3.3.9 Why no Blacklisting of 2-grams

We decided to not blacklist the top 10,000 2-grams during the creation of StoryPasses. During our pilot testing of the system we found that blacklisting 2-grams lead to much frustration in our pilot testers; it was not very usable. We decided to allow common 2-grams in StoryPasses for the sake of usability but it affects the security of StoryPasses. While it may seem from our previous section on the security blacklisting that this is a poor decision, it isn't a big loss in terms of security potentially provided by StoryPasses. This is because even though we allow common 2-grams, we are still blacklisting in the top 10,000 3-grams, 4-grams and 5-grams. The n-grams were all trained on the same data set so if a 2-gram is in the top 10,000, it is likely that the

2-gram exists in the top 10,000 of either the 3, 4 , or 5-gram list.

To estimate the amount of security that is lost by allowing top 10,000 2-grams, we need to make some assumptions in our theoretical analysis. If we assume that an attacker picks the minimum length of a passphrase, 7 words, and includes all words which can be modelled after a 2-gram. This means that we need 6 2-grams to model a 7 word passphrase. The highest rank in the 2-gram table is 48759699, we will assume all 2-grams in our passphrase are half that at, 24379849.5. We will call 2-grams with a rank of 24379849 an average 2-gram. We then assume that one 2-gram is rank 1, or that the attacker correctly guesses the 2-gram on the first try. We need to include 3 rank 1 2-grams in our worst case situation. This is because, 3 2-grams can fit in to a 7 word passphrase that are not overlapping and thus would not be filtered as a probable 3, 4 or 5-gram. For example:

(Reading a) science (fantasy book) (cover page)

The brackets identify a 2-gram found in the top 10,000 2-gram list. The amount of guess effort in a passphrase with a three 2-grams at rank 1 and 2 average 2-grams would be 49 bits. The amount of guess effort in the total 7 word passphrase with 2-grams is 66 bits, as outlined in Table 4. Given our assumptions, we can see that by including three very common 2-grams for usability purposes does hinder security quite significantly however; this is a worst case scenario. It is also very unlikely that someone will pick a passphrase that has three 2-grams not overlapping as the likelihood that the sentence reflects natural language structure is low. We do not consider multiple 2-grams in the top 10,000 because top 2-grams will be in the top 3, 4, and 5-gram lists, which are blacklisted.

3.3.10 Low Estimates Including Proper Nouns and Slang

We suggest the use of proper nouns and slang for a number of reasons. Proper nouns and slang may enhance the security in a passphrase because modern password crackers rely on using dictionaries for guessing a given password. They use lists of pre-built characters, n-grams and phrases to guess the most likely combinations. To decrease the chance of being guessed, it is advantageous to use words that are not well known or are very unlikely. Due to the sheer number of proper nouns, encouraging the use of them can help improve the odds that the dictionary won't be able to guess the passphrase.

In order to account for all guidance that is suggested by the password creation policy, we have to address the guideline that recommends the usage of proper nouns. If we take two databases, one of which has 39336 unique cities from around the world [33]. The second which has 33542 unique baby names from 2012 [47]. The names were gathered by the Social Security Administration of the USA. We can use cities and names because they are always proper nouns. we can enumerate how many 7 word passphrases could be made with the guidelines of four words with a proper noun.

If we use 7 words using two 4-grams , with only one 4-gram having a proper noun. We get a total of 4089759832798345 7 word combinations that have at one proper noun. If we multiply this value by the cities and names database to make potential passphrase guesses we arrive at:

4089759832798345 * 33542 names = 66 bits

4089759832798345 * 39336 cities = 67 bits

This brings us up closer to the recommendation of 80 bits set by NIST [11]. This can be considered a low estimate since we require one proper noun in a passphrase, we assume that users will pick either a name or proper noun and we are only considering the combination of two 4-grams. There are plenty other proper nouns which we do not consider or enumerate that make the passphrase space very large.

3.3.11 Slang

In order to determine the level of guess effort that can be provided by encouraging the use of slang, we need to first determine how many slang words there are. It is difficult to quantify the number of slang words that exist. This is because slang is generally not formally accepted but is known to a group of people. It usually has a vernacular meaning and significance. Knowing this, it is very difficult to determine exactly how to enumerate the number of slang words that exist.

UrbanDictionary.com is a free online web resource that contains more than 7.5 million definitions as of March 2nd 2013 [37]. It specializes in collecting and containing slang or cultural words and phrases. If we take our value of 4089759832798345 which is the total number of 4-gram plus 4-gram joins with at least one proper noun, we can substitute in every UrbanDictionary.com definition to get a rough estimate of how

many 7 word StoryPasses could be generated that contain a single slang term.

4089759832798345 *7,500,000 = 74 bits of guess effort

By encouraging our users to use slang during the creation of StoryPasses, we are greatly increasing the passphrase space that must be enumerated by a potential malicious password cracker.

While our calculation can be considered an overestimate, it conveys the potential that StoryPass has for making strong passphrases. Slang is not always a noun, it can be considered a verb or adjective depending on the definition, this limitation of our calculation can been seen that we are greatly undercutting the potential strength of our StoryPass creation guideline. It should also be noted that not every slang word or phrase can be substituted in to every 7 word combination with a proper noun. Users tend to pick passphrases which are based on proper English structure. That being said, our estimate is liberal in the potential security it offers to recommend the inclusion of slang words.

3.3.12 Error Correction

Error correction will improve usability by allowing typographical errors to occur when a user confirms their StoryPass; however, it will impact security. One thing to consider with error correction and Levenshtein distance is the ability for attackers to improve their guessing rate. With Levenshtein distance tolerance, attackers can in essence guess multiple words at the same time when making a single guess. This

happens when multiple words are only minimally different (cat, hat, heat, beat). An attacker could make categories of words which are only one or two edits different and guess just one of each category to guess all them. Since an attacker will know our Levenshtein distance tolerance is 12.5% or 1 in 8 characters, they could strategically formulate their guessing order to take advantage of this usability feature. While the full impact of this is difficult to quantify, we look at this in Section 5.1.6 and do a qualitative analysis to determine if the error correction would have any impact on the effect of our Passphrase Ranking Algorithm.

3.3.13 Limitations

There are a few limitations to consider with the theoretical analysis. When we blacklist the top 10,000 from 3, 4 and 5 n-gram tables, we still acknowledge that some passphrases will be easier to guess than others. This is because users do not pick random StoryPasses. There will always be a probabilistic way to guess passphrases, which can be clearly seen and is acknowledged in other research. Marginal Guesswork is the closest theoretical model we have to identifying this phenomenon.

There are also limitations in our slang and proper noun estimate numbers. We have access to limited data on names and proper locations. We also make the bold assumption that all proper nouns that are in the passphrases that users create with StoryPass are a name or a city or a slang term. We also do not consider that users might use more than one of these terms, or multiple of each, or terms which do not embody any of these concepts.

Our use of the terms gathered from UrbanDictionary.com is quite assumptious and limited in determining the amount of increased security they can provide. There is no way of exactly identifying a slang term as a slang term. Some words in the UrbanDictionary.com list may not be slang, they may be words found in a proper dictionary but can be used out of context as a slang term.

Finally, our estimates do not consider natural language patterns in different languages. Users created primarily English based passphrases in our StoryPass user study but 3 were written in other languages. Without us, the researchers, knowing these other languages, it is impossible to make an educated estimate of their theoretical security.

4 User Study

To test the usability and effective security of StoryPass passphrases, we run a controlled user study. We hope to demonstrate that long passphrases created with StoryPass can be more secure and just as memorable as other forms of text-based authentication.

4.1 Methodology

We designed and executed a controlled user study that took place over 4 Sessions that spanned 39 days, to simulate real life user authentication conditions. Session 2 was held approximately one day after Session 1, this is to simulate user self-reported frequency of logging into email account [20]. Session 3 was held approximately 7 days after Session 2. This was to simulate self-reported user login frequency to financial websites such as banking [20]. Session 4 was 30 days after Session 3 to simulate self-reported user login frequency to e-commerce and financial websites as well [20]. Users were recruited via an email that was sent out to all students at the University of Ontario Institute of Technology and by posters on campus. Users were compensated as part of our responsibility as researchers to provide ethical considerations for our participants. The compensation model was structured so that participants received more as the study elapsed. They were given $2, $3, $5, and $5 after Session 1, 2, 3 and 4 respectively. We collected data during these four sessions on the login attempts, and the user's perception.

4.2 Data Collected

We observed many pieces of data during the study using an array of techniques. A list of the data we collected:

- The passphrase
- Passphrase usability metrics such as creation time, login rates, and typing time
- The mnemonic cue
- Demographic information
- Pre-study questionnaire in Session 1
- Post-study questionnaire in Session 3
- Memorability ratings in all sessions

The questions asked in the pre-study questionnaire were about demographics, computer skills and habits, and password habits. In the post-study questionnaire, we asked questions mostly about their experience and opinion using cues and passphrases in StoryPass, memorability of their passphrase, and future password or passphrase plans.

4.3 Study Schedule

The full schedule of the user study as follows

Session 1 (Day 1, in-lab)

1. User creates a practice passphrase and memory cue, until they are able to successfully confirm.
2. User creates and confirms passphrase and memory cue.
3. User completes pre-study questionnaire.
4. User logs in with passphrase.

Session 2 (Day 2, online)

1. User enters their perceived rating of memorability of their passphrase.
2. User logs in with passphrase.
3. User enters their perceived rating of memorability of their passphrase.

Session 2 Attempt 2 (Day 3, online)

If a user fails to successfully authenticate their passphrase 10 times during Session 2, they must reset their passphrase. They are also given the option to reset after 3 failed login attempts. If they reset, they must return one day later for this Session.

1. User enters their perceived rating of memorability of their StoryPass.
2. User logs in with passphrase.
3. User enters their perceived rating of memorability of their StoryPass.

Session 3 (Day 9 or 10, in-lab)

If a user reset their passphrase in Session 2, and then did Session 2 attempt 2, they will do Session 3 on day 10. This is to ensure 7 days between Session 2 and Session 3.

1. User enters their perceived rating of memorability of their passphrase.
2. User logs in with passphrase.
3. User enters their perceived rating of memorability of their passphrase.
4. User completes the post-study questionnaire.
5. User enters their perceived rating of memorability of their passphrase.

Session 4 (Day 39 or 40, online)

This session is scheduled for 30 days after Session 3. This will be either day 39 or 40.

1. User enters their perceived rating of memorability of their StoryPass.
2. User logs in with passphrase.
3. User enters their perceived rating of memorability of their StoryPass.

4.4 Participation

In total, we had 40 participants start the study and complete Session 1 and 2. We had one participant whose results could not be included due to a technical glitch in Session 1, although they completed the entire study. One participant completed Session 1 and 2 but explicitly dropped out for unknown reasons. They stated that we had permission to use the results gathered up to that point. 3 participants did not return for Session 3 after Session 2, no reason was given. 35 participants completed Session 1, 2 and 3. 31 Participants completed the entire study up to Session 4. There was no correlation between users who did not return for sessions and usability factors such as failed logins or passphrase resets.

4.5 Demographics

Our study recruited 40 participants from the campus of the University of Ontario Institute of Technology. We had 21 males and 19 females in our study. The responses given for ages ranged from 18 to 62 with the average being 34.2 years old. Our participants response to "current level of education completed to date" was varied as well, ranging from high-school to masters, with the mode being high school diploma. Participants current field of employment ranged highly as well from student to part-time seamstress. Our participants were asked what their mother tongue was, or what

language was spoken at home. The most common response was English(48%) but we had a total of 12 unique languages with Urdu(13%), Chinese(10%), Spanish(8%) and Persian Farsi(5%) being other notable languages.

5 Results

Our results were gathered over a 39 day user study that was performed at the University of Ontario Institute of Technology. In this section, we will first perform a security analysis on the collected passphrases and then a usability analysis of the results. Lastly, we look at a small subset of participants that had a significantly higher number of failed login attempts than average.

5.1 Security Analysis

In this subsection, we introduce our Passphrase Ranking Algorithm. We explain how we use n-grams to perform a guessing attack against user generated passphrases within StoryPass. Next we discuss the effect of error correction on the security of the passphrases. We then categorize words and n-grams found in the passphrases based on COCA and "slang". We use our UrbanDictionary.com list to search for words which are not found in COCA. Since we recommend the use of slang and non-dictionary words in our creation policy, we evaluate if users are actually picking slang and non-dictionary words.

5.1.1 Determining Security

All passwords traditionally are given a strength level that is measured with entropy levels. Our first task was to create a theoretical analysis to determine what our requirements and recommendations would be to users when creating passphrases in the study. After completion of the study, we performed an analysis of the effective security using the passphrases that were created during the study.

In an effort to make user passwords vary from the most common forms of natu-

ral language models, we require at least 7 words, a proper noun, and blacklist the top 10,000 3 to 5-grams. This policy was informed by a theoretical security analysis of passphrases under a guessing model in Section 3.3.5. Our theoretical security analysis was done mainly using n-grams [16]. The n-grams are used by chaining them together to create passphrase guesses. These guesses are used to estimate the total search space size of potential passphrases that are made by our StoryPass user study participants; however, it does not tell us where in this search space the passphrases are most likely to exist.

To determine the security of passphrases collected in the study, we employed the use of n-grams from COCA and dictionary words from the COCA lexicon. We built the Passphrase Ranker Algorithm, a guess-number calculator, in order to map the passphrases to a guess number. Kelly et al. [28] introduced the concept of a guess-number calculator. A guess number is the number of guesses that are required to successfully guess the passphrase. We base our attack on the premise that an attacker would know what the parameters are for creating passphrases. We use the COCA n-grams in our guess-number calculator to simulate an exhaustive brute force search by matching a user's passphrase to multiple n-grams found in COCA based on the expected frequency of the phrase in natural language (according to COCA). The algorithm used is described in Section 5.1.2.

5.1.2 Passphrase Ranking Algorithm

Our security analyses take advantage of n-grams to generate our guesses against passphrases created by participants during the study. Our approach for this analysis uses n-grams, ranging in size of 5 to 1, which have been sorted and ranked based on their frequency. By sorting our n-grams by rank and combining them according

to the Passphrase Ranking Algorithm, we estimate the number of guesses it would take to crack a user's passphrase under the assumption that the adversary uses the n-grams to generate an ordered set of guesses. Our post analysis focuses on determining where our user-created passphrases fit in to the attacker's ordered set of passphrase guesses. The frequency is number of the times that a particular n-gram appeared in the corpus (COCA) that the n-grams were observed in. When searching for n-grams, we use overlapping to ensure that the n-grams have proper context and relationship between each other. This means that a 7-word passphrase would need at least two 4-grams (or a 5-gram and 3-gram) to cover all the words. The following example below depicts two 4-grams being joined. Consider the following passphrase:

UOIT deploys Lenovo ThinkPads for all students

We would need to find the following two 4-grams

(UOIT deploys Lenovo ThinkPads)
and
(ThinkPads for all students)

If either 4-grams could not be found, the method will fall back to use smaller n-grams. E.g., a sequence of 3-grams and 2-grams would look like this:

(UOIT deploys)

(deploys Lenovo ThinkPads)

(ThinkPads for)

(for all students)

Our algorithm will prefer to use overlapping as it will provide a method of "chaining" n-grams together.

The n-gram lists are "ranked". We order all the n-grams based on frequency, giving each a rank. The highest frequency n-gram has a rank of 1, the lowest frequency n-grams have the lowest rank of 331925771 in the 4-gram table. When n-grams have the same frequency, we give them an equal rank. However, we treat the next n-grams rank as if each equally ranked n-gram before it was guessed. For example if the top 5 n-grams have the same rank, they are all rank 1 but the 6th top n-gram is given a rank of 6. This will cause our algorithm to underestimate our guess score because we give the attacker the benefit of the doubt that they will pick the correct n-gram with the same rankings on the first try. This makes our guessing scores more conservative albeit hopefully more realistic given the limitations of our security analysis.

Verbally our algorithm, a guess-number calculator, estimates the number of guesses it would take to correctly guess the passphrase. We believe this to be the correct way for an attacker to guess passphrases made in StoryPass. An attacker would start with the largest, most frequent n-grams before moving on to the smaller and less frequent n-grams. This in essence, is a breadth-first search. It will always try to use the largest n-gram size. Even when using 3-grams, if it finds a potential spot for overlapping, it will immediately increase the size to 4-grams. In a worst case scenario, an attacker would know the optimal order of n-grams to guess and the correct n-gram size to use

all situations.

It starts by searching for all possible 5 word sequences within the passphrase in the 5-gram table, and then all 4 word sequences in the 4-gram table and so on until 1-gram. If a match is found at any time, the rank of that n-gram is stored, and those words are marked as being matched to an n-gram. Every time an n-gram search is performed, the maximum rank of that table is kept if no match is found, and stored, for measuring the total guess-effort. In essence, the n-gram high estimate factors in all guess-effort done with the algorithm, while the low estimate only tracks guess-effort that returns a result. We also do a 1-gram permutation estimate. This estimate only uses 1-grams to find a guess-number. It will try to find all words in the passphrase in the 1-gram table. If the passphrase is longer than 7 words, it will factor in guess effort of enumerating all shorter passphrases. This is because we assume an attacker does not know the exact size of the passphrase, but they do know the 7 word minimum. This means if it a passphrase is 9 words, it will include the guess effort of enumerating all 7 word and 8 word passphrases in the final guess-number.

When possible, the algorithm will include an overlap word which is already matched to an n-gram, if it adjacent to either the front or the back; this is to cause chaining of n-grams, to provide a stronger syntactic context. For example, if we have 3 words in a row, we check the word before and after the 3 word sequence, and if either has been found, we add one word to the 3 word sequence and do a 4 word search instead. When an overlap has been found, we now search with wildcards for all words except the joining word. The reasoning behind this is to only return the n-grams which contain the overlapping word. This gives us a chaining effect and greatly reduces guessing effort on the attackers part. This means that when using an overlap, we are only searching a very small subset of the data that contains the already known word. We then *dynamically re-rank* the overlapping subset of n-grams based on frequency. If

an n-gram in the subset matches the all words we are searching, we save the dynamic rank instead of the rank associated with n-gram in the n-gram database. The algorithm will only fall back to using 1-grams when all possible overlapping combinations of 5 to 2-grams have been exhausted.

The algorithm returns a n-gram high and low score, and also a 1-gram permutation score. The difference between the n-gram high and low score, is that the high score includes all the stored ranks of the n-gram searches in the sum, which were done, including those that did not match an n-gram. This is to simulate an exhaustive guessing attack. The low score includes only the product of the ranks from n-grams which were matched by a search, this assumes the attacker would know the exact n-gram table to search and the exact location of n-gram in the passphrase. We multiply all the ranks because the idea is that the n-grams cannot be guessed individually. This simulates that every guess must contain all words in the passphrase being attacked, not one n-gram at a time. For example, if one 4-gram is rank 400 and a 3-gram is 800, the system would guess those two n-grams together at guess number $(400 * 800 = 320000$).

The 1-gram permutation score only uses the 1-gram table to find a guess-number for the passphrases. The score returns the ranks of all the found words in the passphrase and also the guess effort of enumerating shorter passphrase permutations. If it is a 7 word passphrase, it only includes the score of the found words, as 7 words are the minimum requirement and therefore the first passphrase size that an attacker would guess. If it is 8 words, we include the guess effort of enumerating the all 7 word permutations and so on.

To summarize, the algorithm is about finding enough words to do an n-gram

search and handle the overlapping of words for chaining.Next, it handles doing a database look up of n-grams and then marking down the rank of the found or not found sequence of words.

To use n-grams for generating a guess-number, we use Algorithm 1 to find matching n-grams that are in a passphrase.

Algorithm: Passphrase Ranking Algorithm

Data: passphrase to search *pass*

Result: It will return a n-gram high and low estimate, and a 1-gram permutation estimate.

Set *score* to new array, used to store ranks of found n-grams
Set *scorenotfound* to new array, used to store guess effort of not found n-grams

```
begin
    /* The variables below are global variables, all other functions
       are called from here.                                          */
    s = new Array[]
    s = explode(pass, " ")
    /* Tokenizes pass based on spaces.                                */
    found = new Array[]
    found = array_fill(0,0,len(s))
    largestngram = 5 or 1 /* We use 5 for the n-gram estimates.  We set
       it to 1 for the 1-gram permutation estimate.                   */
    /* This fills found with zeros, used to track if a word has been
       mapped to an n-gram.1=found, 0=not found.                      */
    foundngram == false
    tosearch = new Array[] /* Used to hold the words, which are eventually
       passed to the database for search.                             */
    numnotfound = FindWordsToSearch()
    specialcase = false /* Used to track overlapping.  False means no
       overlap.                                                       */
    for n=largestngram, n >=1, n - -
    do
        /* For every n-gram size                                      */
        for  i=0,While i ≤ len(s)−n,i++
        do
            /* We do as many iterations as there are words in the
               passphrase minus the size of the n-gram we are searching.
               */
            searchstartingpoint = i /* Used to keep track of what words we
               are searching as we iterate through the passphrase.  This
               is a global variable.                                  */
            searchgramsize = n/* Used to keep track of the size of the
               current n-gram being searched.                         */
            FindSpecialCase() /* Finds an overlap if possible.        */
            PrepareWords() /* Prepares the words to do a MySQL search.  */
            result = DoNgramSearch() /* Does the MySQL search.        */
            if result >1 then
                SaveScoreMulti() /* Saves the score from the MySQL search if
                   it is multiple n-grams.                            */
            end
            else
                SaveScoreSingle() /* Saves the score from the MySQL search
                   if it is a single n-gram.                          */
            end
            if foundngram == true then
                MarkFoundWords()
            end
        end
    end
    ReturnScore()
end
```

```
Function:FindSpecialCase()
begin
    /* This function determines if we can do an overlap and
       therefore an n-gram chain.                                    */
    if searchgramsize >1, /* if we are not using 1-grams             */
    then
        if numnotfound ≥searchgramsize−1
        then
            /* To see if we can include an overlap in our algorithm,
               ensures we do not set 6 words in tosearch.            */
            if found[searchstartingpoint] == 1 then
                /* If we have found the word at position
                   searchstartingpoint we can use it to overlap      */
                numnotfound++ /* increased by one to accommodate
                   overlap                                           */
                specialcase = true
            end
            if found[searchgramsize+searchstartingpoint]==1 then
                /* If we have found the word at position
                   searchstartingpoint+searchgramsize we can use it to
                   overlap                                           */
                numnotfound++ is increased by one to accommodate overlap
                searchstartingpoint++
                specialcase = true
            end
        end
    end
end
```

```
Function: PrepareWords()
begin
    /* This functions adds the words or wildcards to an array in
       preparation of the MySQL search.                          */
    if numnotfound ≥searchgramsize
    then
        /* This is to ensure that we search with the largest n-gram
           combinations possible.                                */
        for i2=0 , i2≤numnotfound,i2++
        do
            /* We are now filling the tosearch array with words or
               wildcards to perform a MySQL query.               */
            if specialcase == true
            then
                /* If we haven't found this word yet            */
                if found[i2 + searchstartingpoint] == 0 then
                    array_push(tosearch, % )
                    /* tosearch is filled with a % wildcard for the
                       MySQL query.  We only use wildcards in an
                       overlap because we need wildcards for finding
                       which haven't been found.  When we are using a
                       found word, we search with that word and use
                       wildcards as placeholders for the words we want
                       to find.                                  */
                end
                else
                    array_push(tosearch, s[i2 + searchstartingpoint] )
                    /* If the word has been found by a previous search,
                       we include it in our search.              */
                end
            end
            else
                /* If specialCase is false, we just need to fill words
                   from s, no wildcards are used.  We have no previous
                   context, so we just directly search with the words
                   we want to find.                              */
                array_push(tosearch, s[i2 + searchstartingpoint] )
            end
        end
    end
end
```

```
Function: DoNgramSearch()
begin
    /* This function does the n-gram look up with a MySQL query.
       */
    switch count(tosearch) do
        /* Counts the number of words we have to search.          */
        case 1
            Do 1-gram search
        end
        return results /* Results return the rank of the n-gram and
           the words.  It can be either be an array of size one or
           an array of arrays, of n-grams with the associated rank.
           If nothing is found, it is empty.                      */
        case 2
            Do 2-gram search, return results
        end
        case 3
            Do 3-gram search, return results
        end
        case 4
            Do 4-gram search, return results
        end
        case 5
            Do 5-gram search, return results
        end
    endsw
end
```

```
Function:MarkFoundWords
begin
    /* Marks down the words found in the n-gram search.          */
    for iter=0;iter≤searchgramsize−1;iter++
    do
        /* Marking the found words as 1.                         */
        found[searchstartingpoint + iter] = 1
    end
end
```

```
Function: SaveScoreMulti(results) begin
    /* This function determines if we found a matching n-gram and does
       the dynamic ranking for the overlap if applicable.  It then
       saves the rank in score.                                        */
    if results >1 then
        /* If we have multiple n-grams returned from our search.      */
        dynamicrank = 0
        while row = results do
            /* While loop will end when every row in results has been
               iterated over.                                          */
            dynamicrank++ /* This is used to keep track of the dynamic
               ranking guess number.  Since we are searching with an
               already found word, we can limit the number of n-grams
               which would need to be searched.                        */
            if row matches n-gram we are searching for then
                /* For brevities sake, this line is checking if we have
                   found a matching n-gram.                            */
                array_push(score,row[rank]) /* We use the rank from the
                   n-gram list in row if there is no overlap as this
                   n-gram was guessed while searching the entire n-gram
                   list.                                               */
                foundngram = true
                if specialcase = true then
                    array_pop(score)/* We are removing the rank we just
                       pushed to add the dynamic rank since this is an
                       overlap query.                                  */
                    array_push(score,dynamicrank)/* Adding our dynamic rank
                       instead                                         */
                    foundngram = true
                end
            end
        end
        if foundngram == false then
            /* If our search did not return any results               */
            rank = Max rank of n-gram table searched.
            if specialcase == true then
                array_push(scorenotfound,dynamicrank)
            end
            else
                array_push(scorenotfound,rank) /* Save the n-gram list size.
                   This is used for the n-gram high estimate.          */
            end
        end
    end
end
```

```
Function: SaveScoreSingle(results)
begin
    /* This function determines a matching n-gram was found and
       then saves the rank in score.                              */
    if (results == 1) and (results matches n-gram we are searching for) then
        if specialcase == true then
            /* If an overlap was used                             */
            array_push(score,1) /* We found an exact match, so we add a
               rank of 1.  The attacker knows the overlapping word
               and would not guess n-grams that do not match the
               overlap.  We only need a rank of 1 because there was
               only one possible n-gram match.                    */
        end
        else
            array_push(score,results[rank]) /* if no overlap used, use
               the rank from the n-gram table.                    */
        end
        foundngram = true
    end
    else
        /* If our search did not return any results               */
        rank = Max rank of n-gram table searched.
        array_push(scorenotfound,rank) /* Save the n-gram list size.
           This is used for the n-gram high estimate.             */
    end
end
Function: ReturnScore() begin
    for iter=1;iter<= count(s)-7;iter++
    do
        array_push(unigramCombos, pow(493906,(6+iter)))
    end
    n-gram low security estimate = array_product(score)
    n-gram high security estimate = array_product(score) plus
    array_product(scorenotfound)
    1-gram permutation estimate = array_product(score) plus
        array_sum(unigramCombos) /* The 1-gram estimate is executed
        separately from the n-gram estimates.  This is because we
        need to set the largestngram variable to 1-gram.          */
end
```

```
Function:FindWordsToSearch()
begin
    /* Finds the number of sequential not found words in found
       array.  Used to determine if we have enough to do an n-gram
       search.                                                    */
    numnotfound = 0
    for iter=0;iter<= searchgramsize-1; iter++
    do
        /* We look at the current search point and count for the
           size current n-gram search size.                       */
        if found[iter + searchstartingpoint] == 0 then
            numnotfound++ /* If the word has not been mapped to an
               n-gram yet.                                        */
        end
        else
            numnotfound = 0 /* Resets the count if not sequential.
               */
        end
    end
    Return numnotfound
end
```

Algorithm 1: Our custom built guess-calculator.

5.1.3 The Ranking Algorithm's Relationship to Markov Models

In simple terms, a Markov Model is a stochastic system where the process (our Algorithm) and state (the passphrase guess) change depending on the previous state [32]. The state changes based on the current form. Natural language modelling is a stochastic model. We use specifically Markov Chains, which is the simplified expression of Markov Models. An example is if I say the word "The" and then I need to determine what word is most likely to come next, I would consider the fact that I have previously said "The". While our Algorithm does not specifically use a Markov Model, we leverage and employ n-grams. We do this by chaining them together, when we find one n-gram, we then use the last or first word in the n-gram to pick our next n-gram to search for. In a way, the "process", our algorithm, is considering the "state" which are the currently found n-grams to find the next most probable n-gram. We consider the current state (words successfully guessed) when using chaining (overlapping) in our n-gram searches to reduce the total search space.

Our algorithm puts this in to practice by using trailing or preceding already found words to reduce the dataset that it will search. After our algorithm has found one n-gram, it will use the trailing or preceding word of the n-gram to search for another n-gram. If it finds an n-gram with a trailing or preceding word, we dynamically rank that smaller subset of n-grams that turned up with the trailing or preceding word. This dramatically reduces the search space of n-grams if we have a trailing or preceding word to use because the probability of matching. An example is if we find the n-gram "we went swimming" as the first n-gram in the passphrase; we then use "swimming" and search for the largest n-grams which have the word "swimming" as the first word. Assuming we find n-grams with "swimming" we then dynamically rank those n-grams based on how many there are, and use the one that matches the

remaining words in the passphrase.

5.1.4 N-gram Low vs High Estimate and 1-gram Permutation Estimate

In the Passphrase Ranking Algorithm and Section 5.1.2, we briefly describe a N-gram low and high estimate, and 1-gram Permutation. These three estimates vary greatly due to the way that they are calculated. The n-gram low estimate assumes that an attacker knows every n-gram size that will be found in the passphrase and which words in a passphrase will not found. We do not include the effort of exhausting the larger n-gram size lists before moving on the smaller sizes. An example is if a passphrase is made up of 4 2-grams, which makes it a 5 word passphrase. The n-gram low estimate would not consider the fact that an attacker would guess all the 5-grams first, then two 4-grams, then 3 3-grams and finally it would find the passphrase in the 2-gram list. We make this assumption because we want to simulate the strength of a passphrase in an absolute worst case scenario for security under our guessing model. The n-gram high estimate includes all of these earlier guesses. The 1-gram permutation estimate only uses the 1-gram list to guess the passphrases. It assumes that an attacker knows the 7 word requirement. The attacker would start with the most frequent 7 word permutations, then the most frequent 8 word permutations, and so on.

5.1.5 Results of Security Estimates

After the study had taken place, we used the Passphrase Ranking Algorithm to find guess-numbers for the passphrases which provides an estimate of the security of the passphrases created. 25 out of 39 (64%) passphrases were given a guess-number with our algorithm using the n-gram high and low estimates meaning this attack method would guess 64% of the passphrases eventually. Figure 6 shows a graph of the number of passphrases that were successfully guessed vs. the number of guesses (log 2) that would be required using our n-gram low estimate. For clarity, we do not include passphrases which were not fully found using our guessing algorithm. Only passphrases which were fully matched with n-grams from our ranked n-gram lists would ever be successfully guessed using our method. In the n-gram low estimate, the first passphrase was found at $2^{49.5}$ guesses. In the high estimate, the first passphrase is found at $2^{128.5}$ guesses although the guess effort required in the n-gram high estimate, grows dramatically after the 60% guessed.

23 out of 39 (59%) passphrases were given a guess-number with our algorithm using the 1-gram permutations estimate. Figure 7 shows a graph of the number of passphrases successfully guessed vs. the number of guesses (log 2) that would be required using our 1-gram permutation estimate. This would be a good attack model for finding the weakest passphrases made in StoryPass, since it is still feasible to enumerate a lot of low guess-effort attempts with minimal effort. However, it does not guess as many passphrases as the n-gram model, which has more words and n-grams to use in the attack. 1-gram permutations do not guess as many passphrases because certain words are found in the larger n-gram lists which are not in the 1-gram lists.

If we do not allow the use of 1-grams in the Passphrase Ranking Algorithm, we get a dramatically different result. 0% of the passphrases are fully found. We analyse further in Section 6.

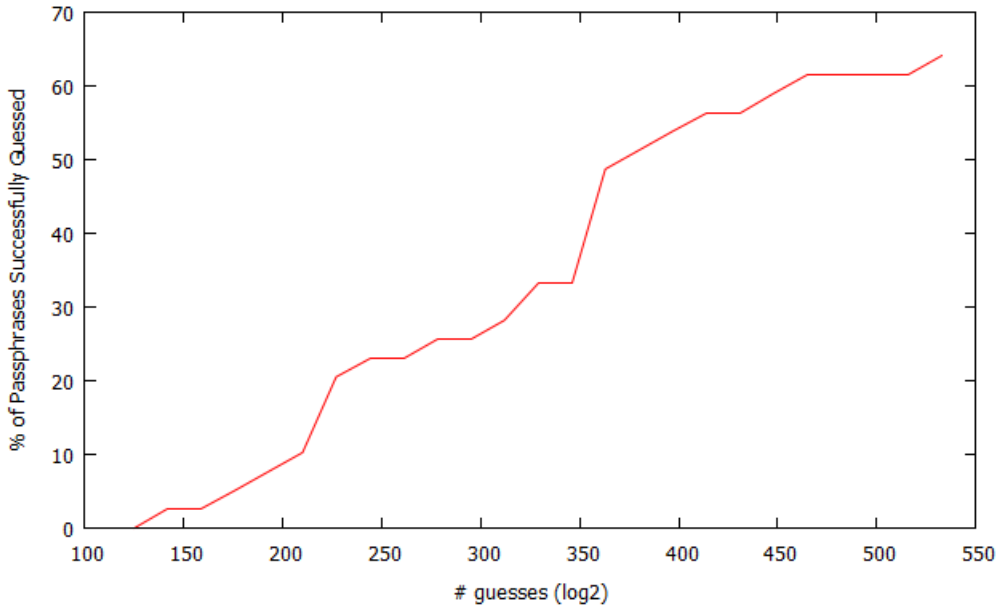

Figure 5: The number of passphrases successfully guessed with the n-gram high security estimate.

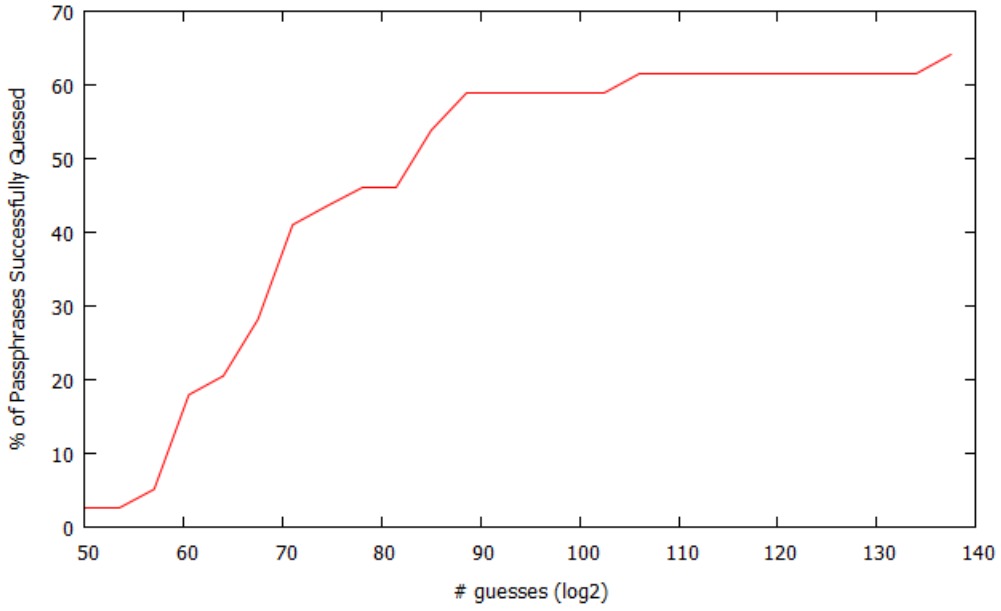

Figure 6: The number of passphrases successfully guessed with the n-gram low security estimate.

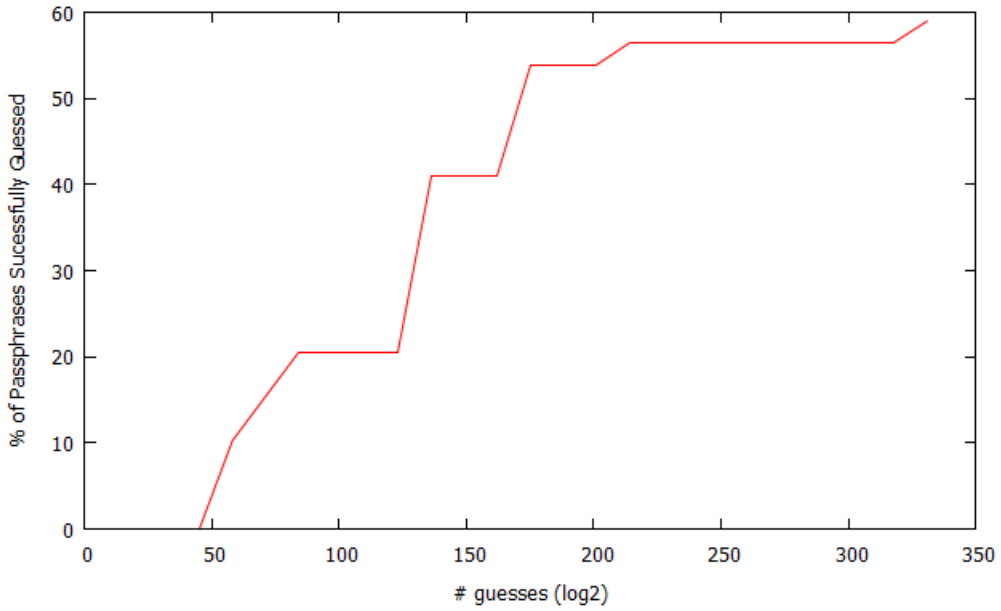

Figure 7: The number of passphrases successfully guessed with the 1-gram permutations security estimate.

5.1.6 Effect of Levenshtein Error Tolerance on Security

One dimension we have considered greatly in our security analysis is the effect that our error tolerance has on security. Our system allows for a 12.5% Levenshtein error tolerance during passphrase login attempts. This equates to roughly a one in eight edit distance tolerance when typing in a passphrase. We set this because we found during our literature review that this is an optimal error tolerance. Pilot testing confirmed that users experienced login failures due to typographic errors when typing their passphrase in blind, as in not being able to see the characters on the screen. Users are reliant on seeing the characters as they type them to make correction for simple typographic errors.

To gain some understanding of the effect that error tolerance has on security, we look at the passphrases that we were not able to obtain a guess number for using

Algorithm 1. (14/39) however, with some qualitative analysis we might be able to determine that an attacker would be able to successfully guess a passphrase with our guesses. To determine this, we looked at all the words, in Table 7, which were not mapped with our Algorithm 1. We then searched for these words manually in the n-gram lists using regular expressions and wild card matches. Regular expressions and wild card matches allow for flexible search parameters, which allowed us to find words which were within the error tolerance limit. We were able to determine that with our error tolerance, only one additional passphrase would have been given a guess number. This is because one participant incorrectly spelt a single word during creation. However, this was not a one-time typo on the behalf of the user but rather they continuously misspelt for the duration of the study. The remaining 13 passphrases would not have been guessed as they contained too many words which were not found in the COCA n-gram lists (even when tolerating typos).

5.1.7 Distribution of N-grams in Passphrases

Table 5 shows the average amount of guess effort that each word and n-gram has from the passphrases collected during the study. We calculated these numbers by summing up the total guess effort for all the passphrases from our low-end security estimate from Figure 6 and then dividing by the total number of n-grams found and words found. This does not include the words in passphrases which were not mapped to an n-gram; however, it does include words and n-grams found in passphrases which were not fully found. Table 6 shows the average number of guesses to correctly identify each n-gram size. N-grams given a dynamic rank in our Passphrase Ranking Algorithm are included in the calculation of Table 6. This includes the 39 passphrases

which were collected over the study. We had a total of 209 n-grams found within the 39 passphrases. 2 were 5-gram, 18 were 4-gram, 26 were 3-gram, 61 were 2-gram and 102 were 1-gram. It is important to note we only had a database of 1.2 million 5-grams, which is why only 2 were mapped to words in passphrases. We had a total of 341 words in the passphrases. 92.9% of the 341 words from the passphrases were found with just the 5, 4, 3, 2 and 1-gram lists. It is important to note that only 25 of 39 passphrases were fully guessed. The majority of the passphrases that were not fully guessed had only one word which were not found in the n-gram lists. The words not found provided the protection from guess-calculator attacks in Algorithm 1; these words are discussed below in Section 5.1.8.

If we do not include 1-grams in our Passphrase Ranking Algorithm, 0% of the passphrases are fully found. 102 of 341 words are not mapped to a 2,3,4 or 5-gram. If we remove the terms which are not found with 1-grams, we have unique 77 words which are only found in the 1-gram table. Using COCA to determine the Part-of-Speech tags of the 77 words only found in the 1-gram table, reveals that 83.2% are considered some variation of a noun. Our attack model in Section 3.3.8 uses only 2-grams as to provide a method of chaining n-grams together to provide context. This shows that our n-gram combination security estimate was low because no passphrase could be modelled 5, 4, 3, and 2-grams.

5.1.8 Words Not Found with N-grams in Passphrases

One major component of our passphrase creation recommendations is to use slang and other non-dictionary words. We recommend this because we assume an attacker

Average number of found words per passphrase	8.2
Average number of $\log_2$ guesses per word in passphrase	5.75
Average number of found n-grams per passphrase	5.2
Average number of $\log_2$ guesses per found n-gram	9.01

Table 5: The average amount of $\log_2$ guess effort per passphrase based on words and n-grams.

N-gram	Found	Average guesses
1-gram	102	28973.656
2-gram	61	1343351.91
3-gram	26	18529826.61
4-gram	18	95264482.44
5-gram	2	736472

Table 6: Average guess effort for n-grams

would use resources such as dictionaries, lists of known passphrases, and words. We want our users to pick words that would be the most difficult to gather and to rank. It is clear that by recommending words that are difficult to gather, it makes it much more difficult for an attacker to guess a passphrase because they likely do not have that particular word in their guessing attacks. By also recommending slang words which are difficult to rank, we are making it difficult for attackers to easily guess a passphrase. Attackers would guess the most probable or highest ranking guesses first, as it is more likely to be correct. We make it more difficult for an attacker to be able to rank guesses, as the non-dictionary words do not have known ranks; also, non-dictionary words may be considered more lexically distinct, and therefore memorable.

An interesting category of words found in the passphrases are the ones that were not found in COCA or in the UrbanDictionary.com list. We separate the words in to three categories: non English words, which are words that are known to be words in other languages besides English; Non words which are English words but not a proper word that would be found in a dictionary or widely recognized; and other, which are

English words that might be found in a dictionary however, these are not found in COCA. An interesting finding in non-words is "Decemeber". It is clearly a misspelling of December, however by misspelling even one letter causes our Passphrase Ranking Algorithm to be ineffective at fully ranking the passphrase. Another interesting word is the word "proffession" which is a close spelling of the word "profession" but looking at the passphrase as a whole, reveals that it was written in french. "Proffession" is close enough for to "profession" with error tolerance that it would be accepted but as a whole, the passphrase would not be guessed.

One interesting finding is that in four passphrases, users chose words which have at least one word which relates to UOIT. The proper noun "UOIT" which is an abbreviation for University of Ontario Institute of Technology was in three passphrases. If an attacker knew that these passphrases were made by users at UOIT, it would be a smart choice to include words that relate to UOIT. Another passphrase had the name "Dr Dincers", which is a close variation of the name of a professor at UOIT. We can clearly see that there is a tendency for users to create passphrases which relate to the people, events and objects around them. In future studies and research, password and passphrase crackers should be studied that have context of the users that create them. If an attacker knows for example, that they have passwords created by users who liked Soccer it could be hypothesized that words, terms and phrases related to Soccer would have a higher frequency in the passwords or passphrases than a sample of collected from a variety of websites. Knowing the circumstances, context, and personality of a user who created a password can greatly aid in cracking the password. We discuss this more in Section 7.2.

Non English Words	Non words	Other
Ptrik Unova Heera Sriragavan Thayalini Robini Intharani Kosalan Luyi jaimerais exercee proffession	Mewmew decemeber	Haaris Dincers UOIT 1935 8 Jimwatts Blizzcon

Table 7: The words which were not found with our Passphrase Ranking Algorithm

5.1.9 The Use of Slang in Passphrases

In our theoretical analysis in Section 3.3.11, we discussed the potential security that requiring a single proper noun in a passphrase can provide. We also recommend in the passphrase creation instructions to include non-dictionary words and slang. We used UrbanDictionary.com as a source of data to calculate a high end estimate of the potential passphrase space that can be achieved. It makes sense that an attacker would want to then include slang in their cracking lists. To include slang in our attacks the best way possible, we decided to gather as many slang terms as possible. We scraped UrbanDictionary.com for all of the slang definitions. In total we have 681,981 slang terms from UrbanDictionary.com.

We decided for the analysis of slang to not use Algorithm 1, as it is unclear how to incorporate the n-grams and slang terms together. This is because unlike the n-grams, we do not have any reliable English context to use with the slang. It is not always clear how to determine if a slang word is a noun, verb, or adjective. There is no accurate way to substitute words in our generated guesses with slang terms. To address the security that slang provides, we settled on using a qualitative approach. We analysed all the created passphrases and searched for slang terms from Urban-Dictionary.com. While this method is not ideal, we realized a number of important facts during the research on slang. Our difficulty in using slang in an automated guessing could be a blessing in disguise. Since one of the largest flaws of passphrases

is that users tend to use language syntax and structure to create a passphrase, and that slang does not follow language syntax and is very difficult to categorize such as, its use may present a security advantage due to its unpredictability. The analysis in Table 8 gives us an idea of how many slang terms can be identified in user-created passphrases, and if an attacker could manage to automate guesses that incorporate slang, how many passphrases would be potentially cracked.

Slang was searched for by breaking up the passphrases on spaces and searched for in the slang corpus which was generated from UrbanDictionary.com.

# of passphrases	39
# of words total between all passphrases	341
# of passphrases with at least one term from the UrbanDictionary corpus	19
# of passphrases with at least one word in the UrbanDictionary corpus and COCA	39
# of passphrases with at least one word not found in COCA or UrbanDictionary	11
% of words from passphrases that were found	96.7%
# of passphrases with one definition from UrbanDictionary.com OR not found in COCA and UrbanDictionary.com	39

Table 8: Slang terms found in passphrases

As we can see, users did not pick many non dictionary terms. Of the 39 passphrases we collected in our user study, 19 contained at least one word from our UrbanDictionary corpus. 11 passphrases had at least one word not found in COCA or the UrbanDictionary.com list. Two passphrases of the 11 that had terms which were not found in all corpora contained words or were written in a language other than English. The passphrases which were not in English were in Basa Jawa (Javanese) and French. This can be seen as a limitation of our security analysis and also a benefit (at least currently) as online resources for non-English based n-grams and frequency are

limited. The last column in Table 8 shows the number of passphrases which contained one term found in the UrbanDictionary.com list or not found in the COCA database. This indicates that users did try to include slang and non-dictionary words in their passphrase.

Requiring users to create passphrases with one proper noun was part of the creation guidelines; however, we only recommended slang or non dictionary terms. It might be effective to require users to pick something that cannot be found in COCA although the challenge will be communicating this to users with passphrase creation guidelines.

5.1.10 Brute Force Estimates

In our theoretical analysis Section 3.3, we described a dictionary-based threat model which could be used to estimate the theoretical security. To quantify if a dictionary attack based on common phrases would be effective for guessing passphrases created by our users in the study, we use a simple method to determine if the passphrase exists as a phrase in a web-scrapable list. We use quoted Google searches on all the passphrases to see if the phrase as a whole has been indexed by Google; this gives us a very easily attainable measure if any of the chosen passphrases could be found if a bruteforce method was pursued. The results of the Google searches came up with 2 exact matches for passphrases and 37 of them not being found.

Only two passphrases were found, one is a well-known tongue twister “peter piper picked a peck of pickled peppers how many pecks of pickled peppers did peter piper

pick" and the other is a phrase "saskatchewan is the land of living skies" which is apparently a well known phrase about Saskatchewan.

Since a dictionary method would rely solely on a pre-compiled list of common passphrases, this result implies that a dictionary guessing attack method would not be very effective at correctly guessing passphrases made in our StoryPass system.

5.2 Usability Analysis

In this subsection, we look at the creation time, login time and login success rate. Next we look at failed login attempts and passphrase resets, and further analyse the data. We look at error correction and the average edit distance of each successful login. We look at the storage rates of passphrases(i.e., the number of participants who recorded their passphrase) and the associated login rates with storage and non-storage participants and other memorability indicators. We quantify the results of the pre-study and post-study questionnaire.

5.2.1 Creation Times and Login Times

To assess the usability of passphrases we collected a number of key statistics. To start, we measured the creation time of passphrases. The creation time can help identify if users are having difficulty meeting the StoryPass creation guidelines. We also gathered user login times to determine how long users are taking to type their passphrase. These two factors can help determine if users are having difficulty using the StoryPass system and satisfying the creation policy.

Figure 8 indicates how long it took users to create their passphrase, correctly confirm the newly created passphrase, and then successfully login during each session. Our

successful login times do not include attempts that failed to log in. This way we exclude attempts where the user may have accidentally hit enter too soon or clicked next before they were done typing.

5.2.2 Login Success Rates

We can measure how effective users are at properly remembering their passphrase by determining how many failed authentication attempts occurred. Figure 9 shows the login success rates during the four different Sessions. These values were determined by counting the total number of successful logins, and then dividing by the total number of successful and failed login attempts. For Session 1 logins, we do not include the practice login attempts in our calculation.

As you can see in Figure 9, users had more difficulty logging in successfully during Session 1 and 2 than Session 3. It is probable that the improved login rate was caused by users becoming more familiar with passphrases by Session 3. There was a slight dip in login success rates for Session 4. We had an excellent turn out for Session 4, with 31 participants returning to the complete the final Session 4. There was a small number of users who had difficulties using passphrases, which lowered the average login success rate. As discussed in Section 2, users are more likely to remember a password the more often they use it. If users had to use the passphrase more often, login rates would probably be better.

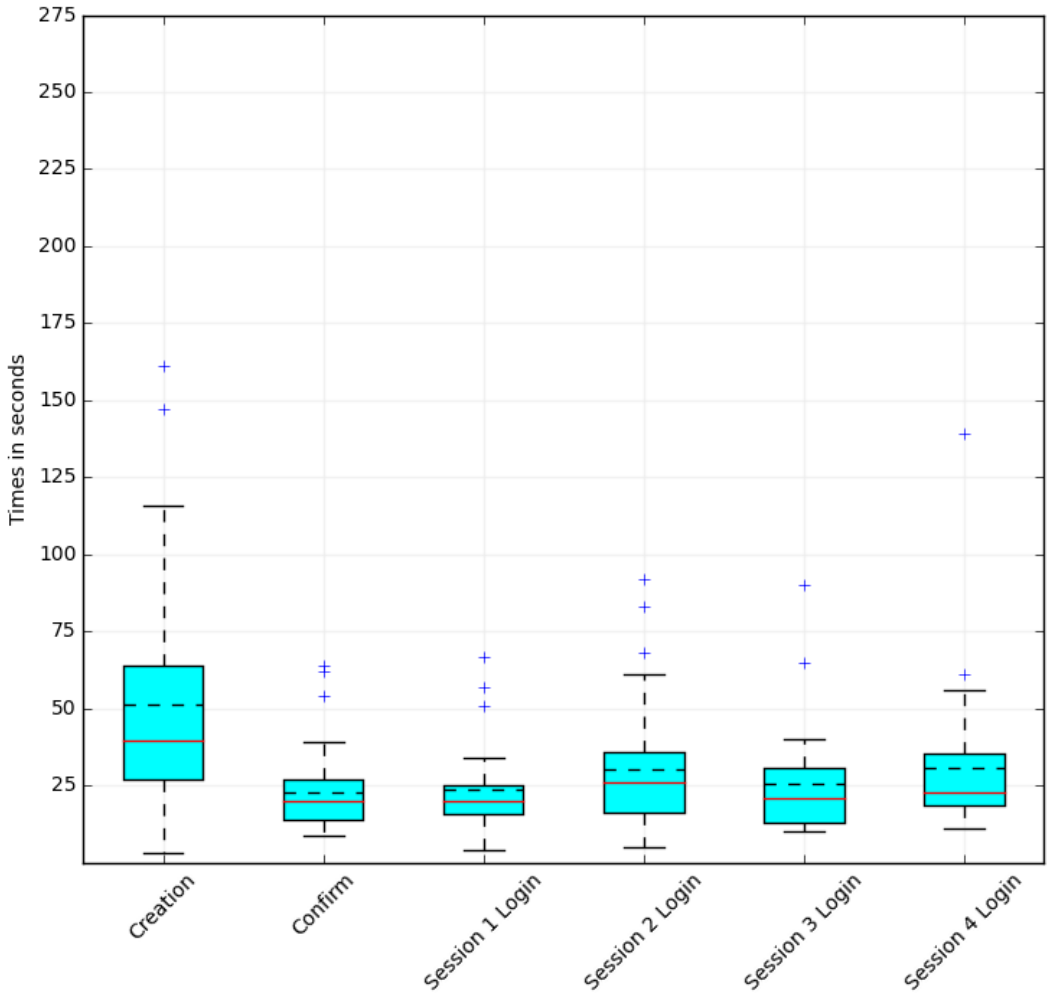

Figure 8: The passphrase creation time, confirm time, and login time for Session 1, 2, 3, and 4.

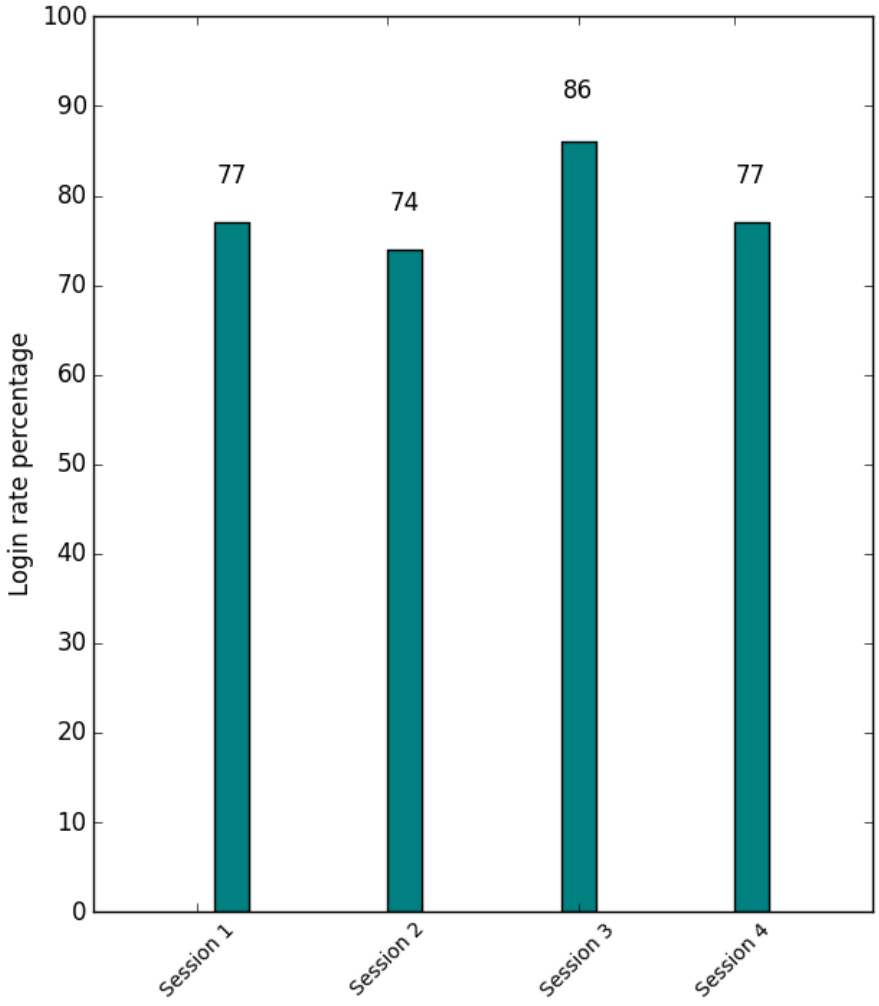

Figure 9: The login success rates for all sessions in the StoryPass study.

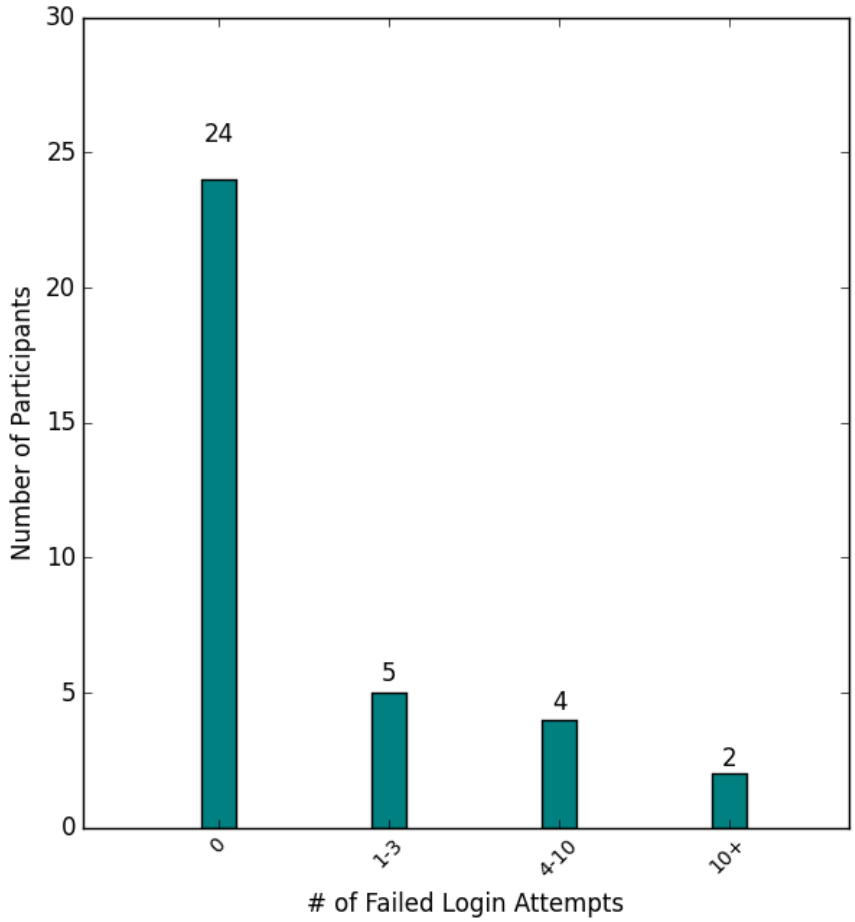

Figure 10: Failed logins from Session 1 (not including practice) until end of Session 3.

5.2.3 Failed Logins

Figure 10 breaks down users into groups based on the total number of failed logins for Session 1, 2, and 3. 35 participants finished Session 3. It is apparent that the majority of users have no difficulty logging with passphrases; however, there is a also a group who have great difficulty logging in, with two participants accumulating over 10 failed logins each.

Figure 11 shows the same as Figure 10 except all users who finished Session 4.

There are 31 users in total who finished Session 4. As we can see, there is still a majority of users, 65%, who have 0 failed login attempts. Qualitative analysis on the data reveals some interesting insight in to failed login attempts. Users who have more failed login attempts tend to have the one of the following pitfalls:

- A passphrase which had too many common words, words which could be easily changed or reordered, such as past tense pronouns and conjunctions. e.g., today my friends went shopping at the mall.
- A passphrase which was not a phrase but rather a series of seemingly random words, i.e., it had no grammatical structure.

It may be possible to check for the pitfalls listed above. For the first one, you would have to be able to determine if a sentence has a unique ordering of words i.e., it cannot be reordered. It might be possible with part-of-speech tagging. The second one is similar, if part-of-speech tagging could be used to determine if a passphrase follows proper syntactic structure, to ensure it makes sense grammatically. Another leading cause of failed logins was not an issue with the passphrase itself, but rather how the user attempted to login. 4 participants, who all completed all Sessions, attempted at least once to login with their passphrase as one long string, with no spaces. This leads us to acknowledge that users need more direction on how to use passphrases. It is possible that previous habits using passwords, which are generally one long string, caused these login failures.

5.2.4 Error Correction and Edit Distances

In our study we used error correction and edit distances to improve usability for our participants. Our level of error tolerance used in the study was to allow the Levenshtein distance to be $\frac{1}{8}$th or less. 1 in 8 means that for every 8 characters in

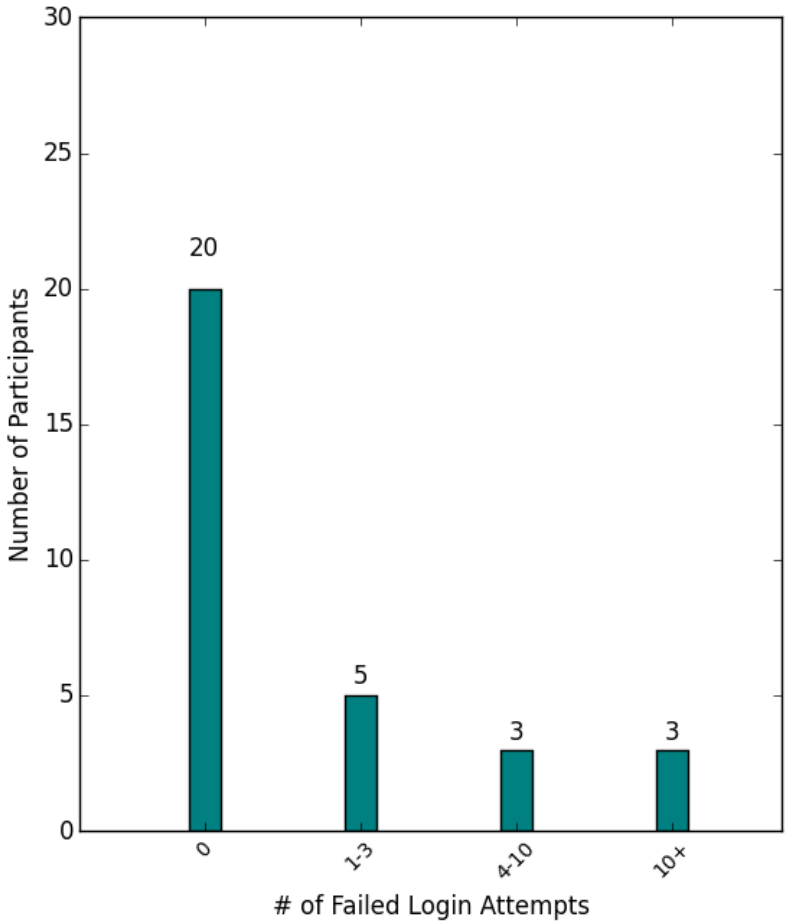

Figure 11: Failed logins from Session 1 (not including practice) until end of Session 4.

a passphrase, you can have 1 edit error. An error being an incorrect character, a missing character or an extra character. The login rates in Figure 12 show the average Levenshtein distance per successful login attempt. The Levenshtein distance was calculated by determining how many insertions, substitutions or deletions it would require to transform the passphrase entered by the participant to the passphrase that was created during the creation phase of Session 1. All of the values are less than 1 in all the Sessions. This means that on successful login attempts, users were most often correctly recalling their passphrase within one edit distance. Using qualitative analysis of failed login attempts, we enumerate how many failed login attempts were because of typographical errors. Not a single failed login attempt was caused by typographical error. There were 55 successful logins from Session 1 (not including practice) to Session 4 which had error correction. The maximum edit distance for successful logins was 11.9%. Only 3 logins had an edit over 10%. 36 of the 55 login attempts had an error distance less than 6.125% which is half of the maximum error tolerance. Given that 0% of failed logins were caused by typographical errors, the error tolerance that was used in StoryPass as suggested by Keith et al. [26] seems to be the ideal threshold for usable passphrases.

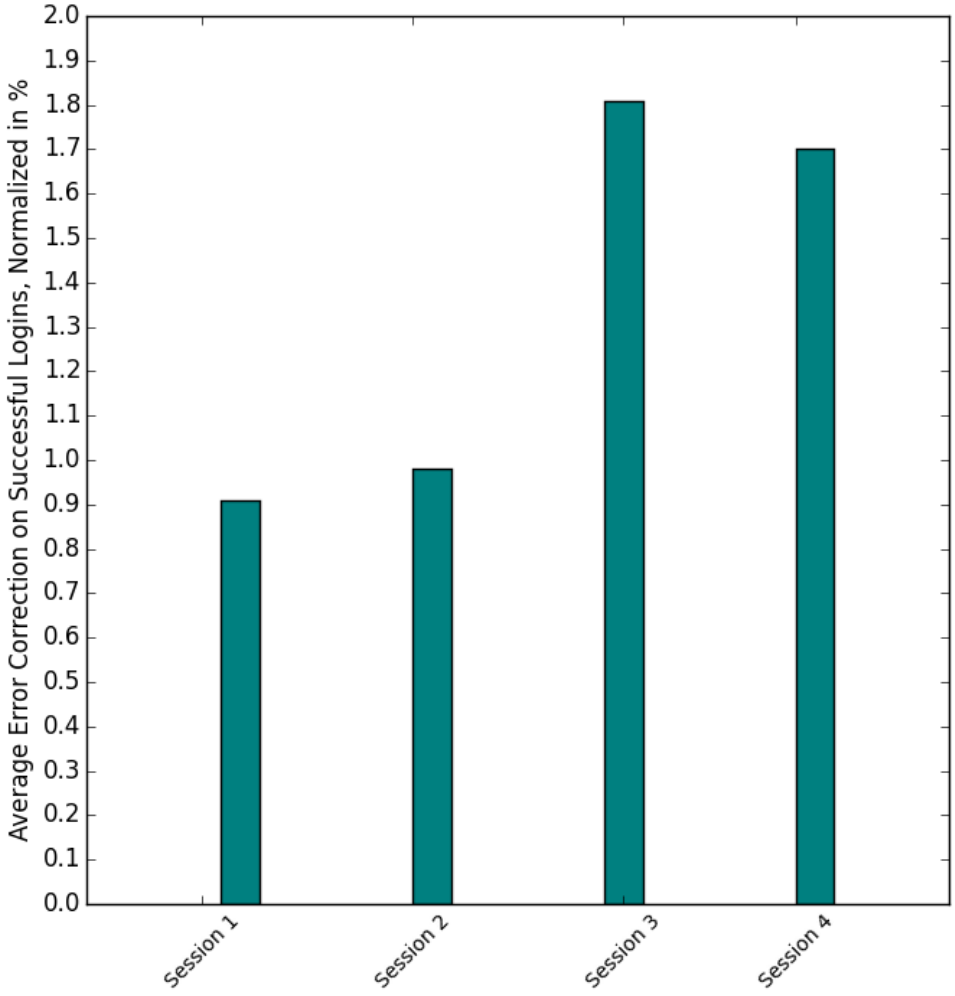

Figure 12: The average Levenshtein distance on successful logins per Session. All values in this graph are less than one because users on average made less than one typographical error per login attempt.

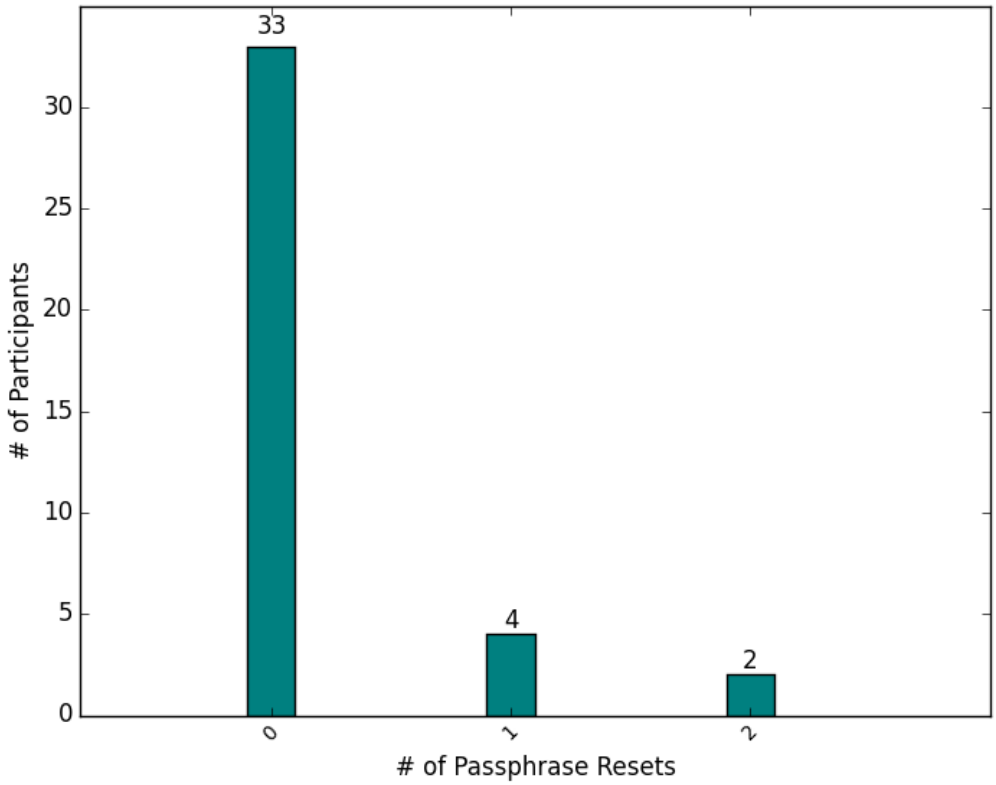

Figure 13: Number of passphrase resets across all Sessions, not including practice.

5.2.5 Passphrase Resets

Participants in the study had the option to reset their passphrase during any of the Sessions. After 3 failed logins, users were given the option to be redirected to the reset pane. After 10, they were required to reset. Only one participant, the one who was excluded from results, waited until 10 failed login attempts to reset. This participants excluded due to a technical glitch during the pre-questionnaire, which caused the Session 1 to take much longer than other participants. Specifically, there was a much longer duration between the creation and login at the end of Session 1.

5.2.6 Storage

In our user study, we observed our participants during creation to see if they wrote down their passphrase. During Session 1 and 3 login recalls, we observed if participants refereed to any form of storage to help remember their passphrase. Our Session 3 questionnaire asked if our participants wrote down their passphrase to help aid memory. In Table 9 we break down the login success rates and compared them between the different categories of participants in our study. Login rates were determined by dividing the total number of successful logins by the total number of login attempts.

	Amount	Login Rates without practice session	Login Rates with practice session
All participants across all sessions	39	77.9%	76.6%
Participants we observed using storage	5	83.3%	58.0%
Participants that reported storage used	10	70.4%	72.3%
Participants who did not use any form of storage	24	76.4%	74.7%

Table 9: The login success rates for the different groups of participants who used storage methods for their passphrase

Our results suggest that users who did not write down their passphrase had less difficultly logging in. Another issue which was predicted and also anecdotally verified with qualitative notes is that some users had difficulty remembering the order of some words in their passphrase sometimes. Users also reported being unable to remember which words in general that they had used in their passphrase. This is because some

common words can be substituted with ease since there is always more than one way to write a sentence and convey the message in a sentence (e.g., 'Some friends went to the ball game' or 'a few buddies went to a baseball game'). An interesting observation is that users who experienced large amounts of failed logins during practice were more likely to be observed using storage for their passphrase.

5.2.7 Cues

During creation we asked our users after creating and confirming their passphrase to create a cue which would help aid in memorability of their passphrase. We never prompted our users for the cue until Session 3 where we asked if users wrote down their cue and we asked our users to recall their cue if they could. Out of the 35 participants who returned for Session 3, 26 participants were successful in remembering their cue. In our post-questionnaire we asked our participants who wrote down their cue. 28 of 35 did not write down their cue. It is an interesting observation that participants were fairly successful in remembering their cue. One idea for this result is that users were actually cued by their passphrase to remember their cue. Qualitatively, when users were asked to recall their cue, many participants asked what the cue was. Although many participants forgot that they had made a cue during Session 1, a majority were successful in remembering it during Session 3 with no reminders in between.

In another question, we asked our participants "Would you use cues as we suggested in this study to help remember future passwords?". 30 of the 35 participants

	Participants	Remembered Cue
All Participants who returned for Session 3	35	26
Used Storage and returned for Session 3	10	8
No Storage and returned for Session 3	25	18

Table 10: The breakdown of participants who used storage and could remember their cue.

said they would use cues again as they did in this user study. One interesting finding was to the question "Would you use the same cue again that you used for the passphrases you created in this study?" 13 out of 35 participants said they would use the same cue. This lends support to previous research that shows that users like to reuse passwords for multiple accounts. Future research should be done in how we can help users create unique memorable cues for all the passwords or passphrases that they use, without having to reuse the same cues. Considering that the majority of users would not use the same cue, it indicates that users do want to use unique passwords for different accounts. 23 of 39 participants in Session 1 responded Yes to "Are you concerned about the security of your passwords?" which indicates and confirms that the majority of users are concerned about the security of their online data.

5.2.8 User Responses to Usability Questions

Perhaps the most straightforward way to determine if passphrases are usable is to ask usability-related questions. In our pre-questionnaire in Session 1, we asked a

couple of questions that related to usability. Figure 14 shows user response to the questions asked in Session 1. 42% of participants prefer to use passwords that are easy to remember and 54% responded strongly disagree to using passwords that are recommended. This falls in line with previous research which shows that system assigned passwords are unusable.

Participants responded favourably to the question "I prefer to use passwords that are difficult to guess" which extends the thought that participants are concerned about the security of the passwords and the data that they secure. However, the two questions "It would be hard for someone who does not know me to guess my average password" and "It would be hard for someone who does know me to guess my average password", which have different responses, show that users are aware that their personal passwords do contain easy-to-guess elements which people who are close to them may know. This can be interpreted that users are already using personal memories or ideas, like the StoryPass creation policy suggests in this study. However, as we know users tend to create weak passwords even if required by a strict policy. Users will generally try to pick a memorable password, so by acknowledging that people who know them can potentially guess their passphrase, it can be deduced that users have been using personal memories and ideas for passwords before they were suggested to do so with passphrases. This reinforces the idea that if we can create a password policy like that used by StoryPass which uses personal memorable elements that users like to use, with characteristics that can provide sufficient security, we can develop a highly secure and usable authentication method.

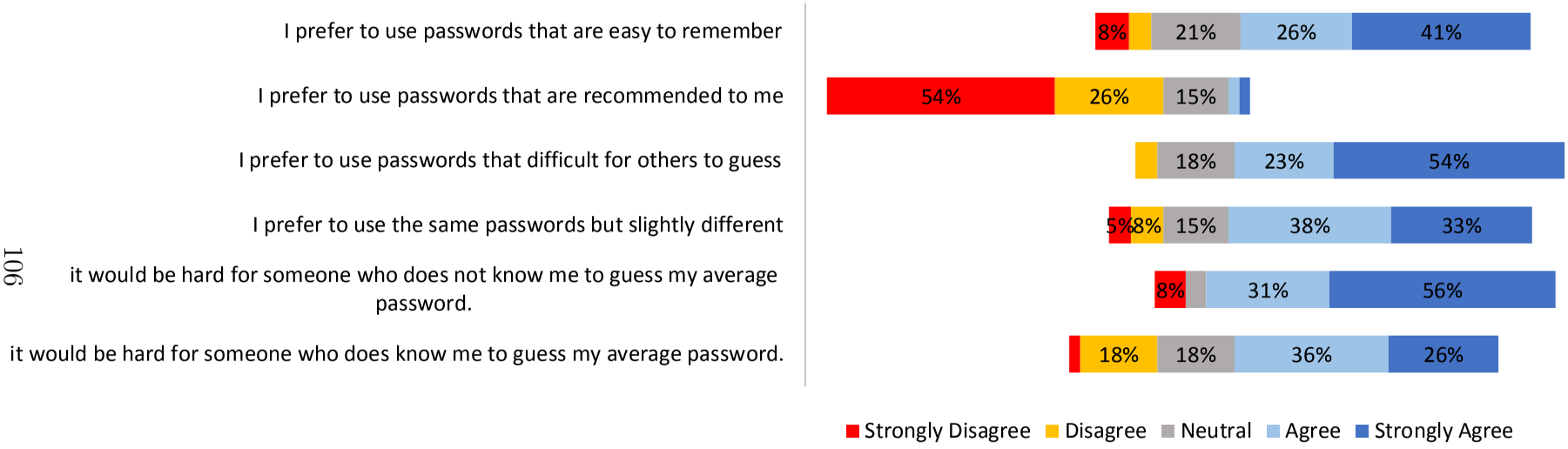

Figure 14: The responses to the usability questions in Session 1

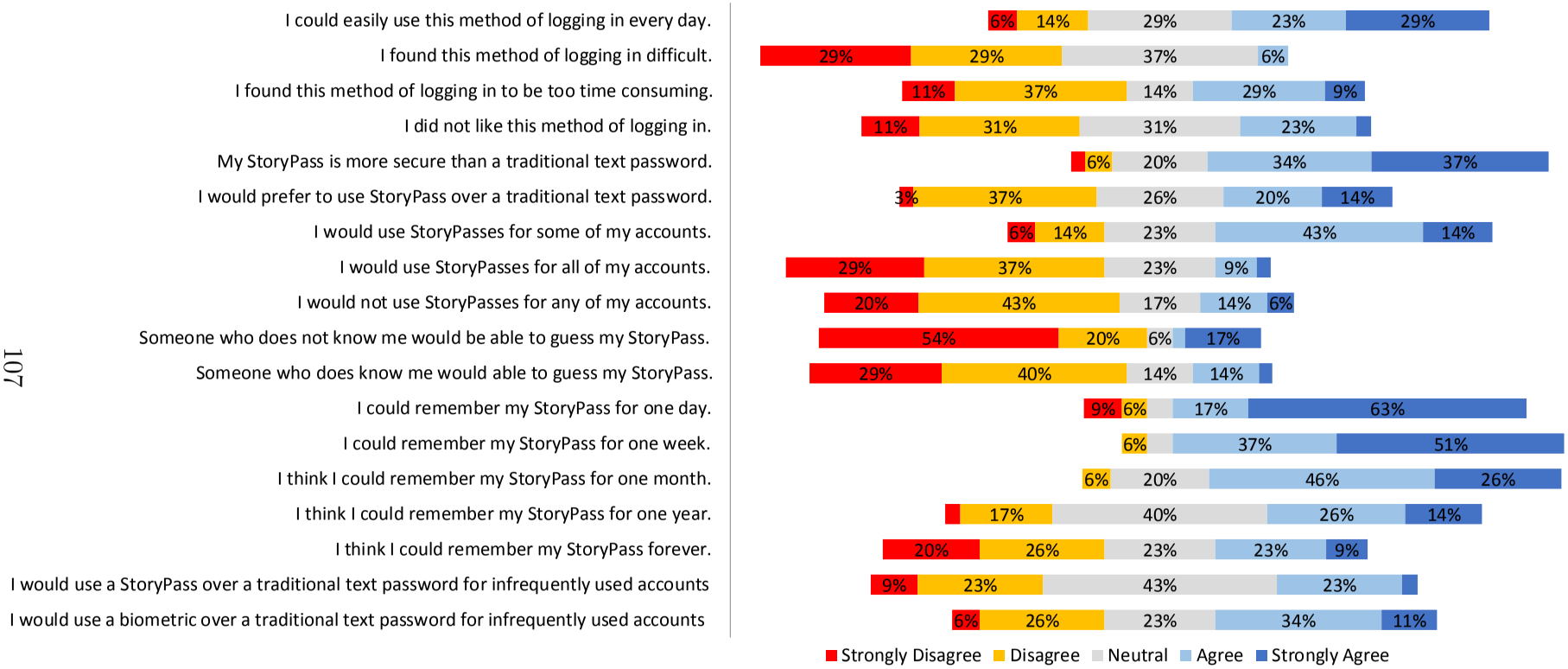

Figure 15: The responses to the usability questions in Session 3

Figure 15 shows the responses to the likert style questions that were asked in Session 3. At this point, users had been in the study for 8 days.

Users in general seemed to like using passphrases when considering all the different responses to the multiple questions. They agreed to the question "I could easily use this method of logging in every day". 12 participants said they liked using passphrases for some accounts but, they would prefer to use methods that are easier to use for other accounts. The length required of a passphrase was too great compared to normal password habits. I hypothesize that if users were used to using longer passwords or passphrases, that users would have responded more favourably to using passphrases for all of their accounts. This is also substantiated by the two questions "I would use StoryPasses for all of my accounts" and "I would not use StoryPass for any of my accounts", both of which were strongly disagreed with.

Another interesting deduction from the post-questionnaire is that users reported their StoryPass passphrase as being more secure than a traditional text password. If we also consider the vastly different responses to the question "I would use my StoryPass for some of my accounts" which was generally agreed with and " I would use StoryPass for all of my accounts" which was generally disagreed with; we can conclude that users think passphrases created in StoryPass is more security than they need for average accounts. The sentiment that users think their StoryPass is very secure is also confirmed by the question "Someone who does not know me would be able to guess my StoryPass" which was strongly disagreed with. The last two questions in the post-questionnaire indicate that users would not use StoryPass passphrases for infrequently used accounts. Although given that users seem to think StoryPass

passphrases provide strong security, they would be ideal for accounts that require strong security.

5.2.9 Memorability Rating

During all four Sessions, our participants were asked this question a number of times:

"On a scale of 1 to 5, with 1 being poor and 5 being excellent, how would you rate the memorability of your StoryPass?"

The average responses per session are displayed in Figure 16 below.

Figure 16 shows that user self perception of the memorability of their passphrase improved from Session 1, to 2 and to 3 but slightly dipped in Session 4. This is probably because as the user recalled their passphrase more, they became more comfortable and familiar with their personal passphrase; however by Session 4 which was 30 days after Session 3, users were more likely to not remember their passphrase. This is confirmed by login rates in Figure 9.

From Figure 15 above in Section 5.2.8, users were polled a series of questions which asked them if they could remember their passphrase for a duration of time ranging from one day to forever. Users seem to agree that they could remember their passphrases for one year; however remembering passphrases forever was perceived to be a stretch for most users in our study. However, many users during the study on Session 3 had reported verbally and in the additional comments section that they

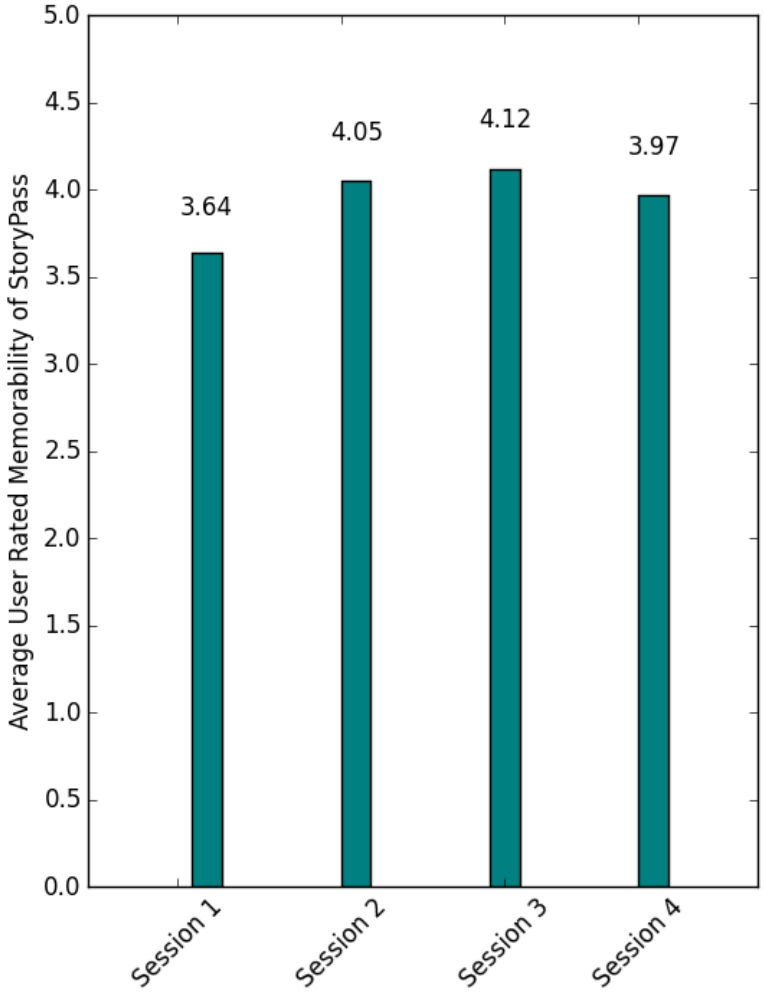

Figure 16: User reported level of passphrase memorability across all 4 sessions

enjoyed using passphrases in our StoryPass system and would use the creation guidelines for future passwords and passphrases.

5.2.10 Usability Impact of Blacklisting 2-grams

To test how much usability would have been lost if 2-grams were blacklisted during passphrase creation, we took the user created passphrases that users used for the duration of the study and searched them for the top 10,000 2-grams. 31 of the 39 passphrases had at least 1 2-gram from the top 10,000. From qualitative analysis during creation and literature review, users do not like having creation attempts rejected or the creation process taking too much time. It can be concluded from this analysis and our pilot testing analysis, that allowing 2-grams from the top 10,000 was necessary for usability.

5.3 Participants Who Experienced Difficulties

After analysing our user study data, we noticed that a small subset of 4 users had great difficulties with passphrases across usability metrics. This group of participants skewed a majority of the data for the worse. To quantify, we decided to re-analyse the data. We call this subset of users *participants who experienced difficulties*.

5.3.1 Login Success Rates

This metric determines what percentage of total login attempts across all four Sessions were successful. The average for successful login rates for the participants who experienced difficulties was 27%. The average for successful login rates for the all others was 88%.

5.3.2 Failed Logins

The average failed login attempts for participants who experienced difficulties was 12.3 failed logins. The average failed login attempts for all other users was 0.7.

5.3.3 Passphrase Usability Pitfalls

As we can see from the usability metrics above, most notably the login success rates and failed logins, the participants who experienced difficulties represented a large percentage of the total failed logins. However, this does not explain why this group of users had difficulties using passphrases. Doing a qualitative analysis of the login attempts is necessary to determine what the underlying issue was.

As discussed in Section 5.2.3, we highlight a number of pitfalls that caused poor usability results. These pitfalls lead to one of two issues, a memory issue where the user forgets either the entire passphrase or more commonly, a partial memory issue where only a few words are forgotten. The second issue is simply a typographical issue where the user repeatedly mistyped the passphrase enough that exceeded the error correction tolerance; after qualitative analysis, this encompassed a very small amount of failed login attempts, as the error correction tolerance helped alleviate the majority of these errors. Memory errors were by far the most common error for users. They either entered in a completely wrong passphrase, or partial passphrase, or entered in the correct words but in an incorrect order. Interestingly, a number of participants who experienced difficulties attempted to login with their passphrase as one long string, with no spaces in between despite having created the passphrase only moments before with spaces between. Another common issue for participants who experienced difficulties was to mix up the order of common nouns, pronouns and conjunctions. While we cannot provide the exact passphrase which users used as they may reveal personal details, we can provide similar examples to show what

happened. Of the 4 users who experienced difficulties, there are 2 participants in each pitfall category. Consider the following passphrase:

Everyone has been bowling at Lucky Strike this week

Now, users would attempt to login such as:

Everyone this week has been bowling at Lucky Strike

This happened frequently for participants who experienced difficulties. Another issue was the lack of a proper sentence.

Pear top montreal green up down middle

Turkey europe secret journey on a bike and moped

It is common sense that we think in the language we speak. We hypothesize that having a passphrase in a non sentence form seems to have impeded the mind's natural ability to help remember the passphrase. This is most likely the cause of issues with password policies which require symbols and numbers, as words never require symbols and numbers. Further more, this is why users tend to substitute numbers and symbols which look or sound like a letter.

One user had a passphrase which did not have either of the issues discussed above. However, they were older than the other participants and rated their computer skills as the lowest possible score. They required much more time to practice the creation

and had many failed login attempts in Session 1. However, for Session 2, 3 and 4, they had a perfect login record. It is possible that users with a low level of computer finesse require more instruction and time to train to use passphrases than users with a higher computer skill ability; further study with such populations should be performed.

5.3.4 Qualitative Observations

During Session 1 and 3, qualitative observations were made. These observations while, not quantifiable like login times, provide valuable insight to human behaviour and factors which have not been prepared for in a formal manner.

Two users during creation had an interesting observation. These two participants when creating the passphrase during practice, had entered in their passphrase as one long string with no spaces. This can be seen as two issues; (1) that it is so ingrained for users to create traditional passwords which are generally one string, (2), that users need more direct creation policies.

16 of the 41 participants who attended Session 1 seemed apprehensive during creation. This could be because of a number of reasons. The user study was performed in a room one on one with the participants and the administrator. It may make users uncomfortable during the session to have an administrator observe them. Users were informed that we were taking notes on the system and not them to help keep users relaxed. It is important to ensure that our participants are relaxed so that their behaviour in the lab is as close to regular computer behaviour as possible. Our

study which uses a controlled lab to allow qualitative observations can be seen as a limitation; however we try to balance this by having two Sessions online.

Many people were surprised during login attempts at the ease of the StoryPass system. 18 of 39 who returned for Session 3 said or were observed being surprised at the system accepting their passphrase as a correct login. This can be attributed to passphrases being significantly longer and more complex than normal or regularly used passwords and users not being aware of the Levenshtein edit distance error correction.

6 Discussion

In this section we discuss overall results, the ecology validity of our study, the ethical implications, and the limitations of our methodology.

6.1 Highlights of Results

Our calculated guess effort in the security analysis has given insight to the potential security that can be provided by long passphrases. We have found that requiring users to create long passphrases does have a negative impact on usability when creation instructions are not followed, especially with regard to memory errors. A group of users experienced significant difficulties due to not following passphrase creation guidelines as discussed in Section 5.3. However, there is hope that with enough practice through deployment and additional instruction, that users should become more comfortable with using longer passphrases. There was no noticeable correlation between passphrase length and usability metrics. The negative impact on usability with long passphrases does improve security. Our qualitative analysis of the passphrases highlighted an interesting aspect of passphrases; due to the increased length of passphrases over passwords, they require additional semantic and lexical structure to help aid memory and typographical load. Participants who used passphrases that had a grammatically correct structure and underlying message, like a story, faired much better at successfully logging in. The use of proper nouns and slang greatly aided security as estimated by our Passphrase Ranking Algorithm (Algorithm 1). This is deduced because of words which were not enumerated in our n-gram lists and UrbanDictionary.com list. Usability comes from the use of memory aiding factors such as sentence-like structure and choosing a distinct idea that is relevant to the user. We believe there is potential for passphrases to be more

usable and secure than passwords with the use of proper grammatical structure for usability and the use of varied proper nouns and slang as the anchor of security. Our qualitative analysis revealed that context of the passphrase creation becomes very relevant when considering security and guess effort. Our findings showed that $\frac{3}{39}(7\%)$ of passphrases created had a strong correlation to our institution, UOIT. Participants included words which had relation to UOIT such as professor names; one included the name of the study administrator, this is discussed in Section 5.1.8. Section 7.2 proposes passphrase creation policies to address this issue.

6.2 Best Attack Strategy

There are two very important factors which are at play in the security estimates which help determine which attack strategy is better.

1. Duplicate guesses are made. This only applies to our high-end estimate.

2. Syntactic structure and proper grammar are not filtered for.

The biggest caveat for the n-gram high security estimate in Figure 5 is that many duplicate guesses are performed. It is likely that a 7-word phrase that is made up of a 5-gram and a 3-gram, also appears as a 4-gram and a 4-gram. This is because we do not track if duplicate guesses are made. It is important to note that the 1-gram estimate and the n-gram high estimate both take in to account effort performed searching for n-grams even if no match is found. Whereas the n-gram low only considers the rank, the effort, of matched n-grams in the final guess-estimate. This means that the 1-gram estimate and the n-gram high estimate are more similar than the n-gram low estimate and the 1-gram estimate.

It is also important to consider the creation policy. An attacker will not know the length of the passphrase they are trying to guess. We assume they start at 7-word

phrases, before moving to 8-word phrases and so on. For the 1-gram estimates, we add in the guess effort of enumerating all 7-word phrases when considering an 8-word phrase; and all 7-word and 8-word phrases for 9-word phrases.

Another consideration is that we blacklist the top 10,000 n-grams in the 3, 4, and 5 lists whereas the 1-grams and 2-grams are not. This means that users can pick the most common and frequently used 2-grams and 1-grams that do not appear in combinations in the 3, 4 or 5-grams list. This means that matched 3,4, and 5 grams will be at a minimum rank of 10,000 whereas 2 and 1 grams could be as low as rank 1. It is important to note that we do not normalize the ranks of the n-gram lists when we blacklist the top 10,000. As in the lowest rank is still 10,000 even after the blacklisting.

The syntactic structure is also another important factor. Some of the passphrases are guessed more easily with the 1-gram permutation whereas some are guessed more easily with the n-gram low estimate. This can be attributed to the syntactic structure that is inherent to the larger n-grams, which are modelled after syntactically-sound spoken and written American English. This can be advantageous but since our largest n-gram is 5 words and the minimum passphrase is 7 words, some of the guesses we make with the n-grams are still syntactically and grammatically unsound, but to a lesser degree than using only 1-grams.

The best attack strategy for guessing the easiest passphrases would be to use the 1-grams to try and find the 7-word passphrases. However, as the passphrase word count increases this becomes more difficult. This is because it becomes computationally expensive to enumerate longer passphrases with the smaller n-grams. The all n-gram attack strategy becomes more appropriate for guessing the longer passphrases. However, if any of the passphrases contain many non-dictionary words or obscure words, it is difficult or impossible to guess these passphrase without significant guess-

effort.

6.3 Validity

In order to measure real-world text-based user authentication behaviour, we took a number of measures to ensure to the best of our ability that the results are as authentic as possible. We ensured that all of our participants did not come from a computer security background. This was to ensure that our users did not have higher than average understanding of passwords and computer security metrics which we measured. At UOIT, we have a large computer and network security study population, so we wanted to be certain that we didn't include security experts. Session 1 and 3 which were in person, were run in a controlled room that was designed for the specific purpose of administering user studies. Only two principle investigators were involved during Session 1 and 3 to ensure that consistent instructions and qualitative observations were taken. The same script was used to instruct participants on how to use the StoryPass user study in both Session 1 and 3.

All of our participants except one, was a student. It may be an issue including only people who are students. Students may have a better understanding of using a computer and usually are younger than the average person. We only had one participant over the age 40, and they typed slower than the average participants, had more failed logins, and required much more instruction during creation and time for Sessions 1 and 3.

One aspect that can affect validity of all research, ours being no exception, is confirmation bias. With this in mind, it would still be naive to say that zero confirmation bias exists but we make our best effort to ensure it is kept to a minimum. We perform our analysis of pre-determined metrics that are used in most text-based user authentication studies, such as login times, failed logins and passphrase resets [1,6,7,44,46,52].

We do not analyze the data in ways to only support our hypothesis. We give the attacker the benefit of the doubt in our security estimates by performing multiple estimates, each with varying assumptions and rationale. Some of the data we put forth does not support our hypothesis. The mnemonic cues do not seem to help aid memory and it seems that many users are still reliant on the use of storage to remember their passphrase. It can be seen in Figure 15 that users are hesitant to use StoryPass passphrases for all of their accounts. This implies that there are still usability factors which are not ideal. No user authentication method is perfect but some work better in certain situations than others. This is why we have a suggested use case for StoryPass passphrases. It is recommended to use StoryPass passphrases in high security situations such as a password manager or a bank account.

6.4 Limitations

There are a number of limitations in our study and research. While our users were compensated for participating in the study, users still did not have to use the passphrase to protect anything that the user considers valuable. In some other studies [26] [8], users were given a realistic password usage scenario or password habits were studied during real world usage for various activities like banking or school. Given that UOIT has a mandatory laptop program for all students, it would be interesting for future studies to utilize these laptops to observe and measure passwords habits, although it would be very challenging to implement such a study ethically.

Our security analysis used n-grams with our Passphrase Ranking Algorithm to estimate the number of guesses it would take an attacker to guess a passphrase created in StoryPass. Our algorithm is quite simple and we do not employ the use of grammatical structure or semantic context of the passphrase such as grammar or part of speech tags. It would be interesting to see if more effective guessing algorithms

could be designed that leverage more information. Another possible enhancement discussed in Section 5.1.8 was incorporating the context in which the password was made.

6.5 Ethical Considerations

Due to the sensitive nature of our study, we took all precautions possible. Our study was approved by the Research Ethics Board at UOIT, all gathered information was stored and backed up securely on site at UOIT. All participants at any point could drop out of the study and if they wished we would delete all data that was given and collected from that user. We only had one drop out in the study. No reason was given for the drop out. They gave consent to use information collected from them up until they dropped out.

All users were compensated for the study as described in Section 4, to ensure that users were given something in return for their time.

7 Conclusions

7.1 Summary Of Results

We present novel research regarding the usability and security of passphrases. No previous studies have been done which analyze both security and usability factors. Generally, previous passphrases studies have focused on one or the other, not both [5, 6, 8, 25–27, 30]. While isolating variables in regards to security and usability is important for performing quantifiable research, it is vital to remember that these two factors are highly dependant on each other [1]. In our opinion, the loss or gain of either generally leads to a change in both. Our study tries to balance both security and usability to study real human habits and interaction concerning creating secure and memorable passphrases.

We have presented a number of interesting results from our analysis and user study. Our main findings are in the area of passphrase security, usability, and passphrase creation policies. It shows that users when given instructions on how to create secure, usable passphrase, are willing and generally successful at doing so. The main value of the study is that it indicates a novel set of policies (e.g., requiring non-dictionary words) and recommendations (e.g., using sentence-like structure). We also confirm previous research such as system settings (e.g., appropriate error tolerance) that help users choose secure and memorable passphrases. Using long passphrases as a master passphrase for password managers such as Keepass and LastPass as the best short term solution for user authentication.

7.1.1 Security Results

Our theoretical analysis in Section 3.3 presented a novel way of using n-grams to estimate the total level of guess effort it would take to enumerate passphrases created using the StoryPass creation policy. We show how blacklisting only a small percentage of n-grams can give a large theoretical security advantage. Similarly to passwords, we show that passphrases modelled with n-grams are susceptible to guessing attacks which help minimize the total effort required to guess sizeable percentages of passphrases. Our estimates of the amount of security provided by the use of slang and non-dictionary words is novel. We show that slang and non-dictionary words have the potential add a lot of security to passphrases. Even with our low end estimates relying on the oxymoron of a dictionary of non-dictionary and slang terms from UrbanDictionary.com, we show that exhaustive guessing using 7 word phrases and these terms provides a very large theoretical search space.

In the results Section 5, we present our custom built Passphrase Ranking Algorithm which is used to successfully find a guess-number for 64% of the passphrases which were collected in our StoryPass user study, but only after a large number of guess attempts. Using just 2,3,4 and 5-grams, we were unable to find a guess-number of any of the passphrases. We show how n-grams can be chained together in Section 5.1.2, to build passphrase guess attempts efficiently. Our analysis of the error correction mechanism on the security of the passphrases shows that error correction does not improve an attacker's chances of correctly guessing a passphrase by a significant margin. We show that by including non-dictionary words and slang terms, users can significantly increase the amount of guess effort required to correctly guess the passphrase. Many of the non dictionary and slang terms found in passphrases were not covered in our guessing lists. This leads to a very high level or an incalculable level of security. Some users during the study used the context of their

situation to help create their passphrase. Five (13%) passphrases used acronyms or terms relating to the environment around them. Some users appear to offload the memory and usability burden of creating a passphrase by leveraging their surrounding and recent memories to select words and themes for their passphrase space. We emulated a dictionary attack against passphrases by assuming that an attacker could gather as many already known phrases and use them as guessing attempts. Only two passphrases were found using quoted Google searches.

7.1.2 Usability Results

Our results on usability revealed some interesting insight and confirmed previous findings on passphrase usability. It was apparent that a subset of the users had much more difficulty logging in that others as observed in Section 5.2.3. These users had similar issues which caused so many failed logins and passphrase resets. Passphrases containing interchangeable common words, passphrases that were not a sentence or a phrase, but rather a string of unassociated words, were generally difficult to recall. Our study had an auxiliary Session 2 set up for the day after Session 2 in case a participant had to make a new passphrase that day. Fortunately not a single participant had to reset their passphrase during Session 2.

Error correction on successful logins was surprising. On average, users made less than .5 typing errors for Session 1 and Session 2 successful login attempts, as can be seen in Section 5.2.4. For Session 3 and 4, users made on average .7 typing errors for each successful login attempt. Error correction and tolerance is reconfirmed to be a requirement for usability when using long passphrases. It seems that being required to enter in a long passphrase blind, to be quite difficult, even for some advanced computer users. Storage was measured using a questionnaire and qualitative analysis

to determine if users were storing their passphrase. Storage seems to help with some memory errors; however, writing it down does not help with login issues arising from typing errors. We asked our users to recall their cue that was created in Session 1. 27 of 35 participants successfully recalled their cue and 28 of 35 reported to not writing down their cue. Participants did not have difficulty recalling their cue. 30 of 35 participants said they would use cues again to help remember future passwords. This is a very positive result, it seems that the instruction to create a mnemonic cue was helpful and also perceived as helpful by users. Overall, 52% of participants agreed that they could use passphrases created in StoryPass for logging in every day, with another 29% being neutral and only 20% disagreed. 29% strongly disagreed that it was a difficult method of logging in and only 6% agreed. 71% agreed that their passphrase created in StoryPass is more secure than a traditional text password. However, only 34% agreed that they prefer to use a passphrase over a traditional text password. Qualitative analysis revealed that users are content with using their own personal passwords due to being familiar with their own password habits. Users reported the memorability of the passphrase created in StoryPass multiple times. The peak memorability was reported during Session 3. This can be correlated with frequency of use, as the memorability drops during Session 4, which was 30 days after Session 3; A much longer time than between Session 1 and 2, and 2 and 3.

7.2 Future Work

For future work, it would be interesting to see what could be done with more advanced natural language tools during a security analysis; tools such as longer n-grams, larger collections of slang terms and common phrases used on the internet. During our security analysis we do not employ the use of syntactic and lexical structure or part of speech tagging. It would interesting to see if structures exists that can be susceptible

to guessing attacks. It is highly likely since patterns exist in natural language since we were able to guess so many passphrases using simple brute force methods, coupled with n-grams observed from English language corpora. More refined passphrase creation policies can be imagined after analysing the results. Such as requiring the use proper grammatical structure in passphrases and to use words that do not exist in dictionaries. More effectively instructing users on using a mnemonic cue and structure of the passphrase would be beneficial during the creation portion of the passphrase.

7.2.1 Guessing Algorithms

Another interesting point of research would be to see if lists for longer than 5 word n-grams can be made. Our main motivator for recommending at least 7 words for the StoryPass creation policy was the fact that generating guesses was very difficult due to the lack of tables for larger n-gram sizes. This is quite difficult as the larger the n-gram size is, the more unique n-grams exist.

A guessing algorithm that can determine if a generated guess has proper grammar would be a step in the right direction. A large number of guesses which our Passphrase Ranking Algorithm makes do not follow proper grammatical structure. If there was a way to determine if a guess follows grammatical structure, it would make the theoretical analysis in Section 3.3 much more accurate, and most likely determine the true low end estimate of total guess effort space. This could be done by using part of speech tagging on large corpora of text, and then doing a frequency analysis of adjacent tags, much similar in concept to generating n-grams.

Incorporating non-dictionary words and slang in more intelligent and sensible man-

ner that takes in to consideration the context and usage of the slang would be another method for improving the guessing algorithm. We consider that all slang from UrbanDictionary.com to be proper nouns, which is very assumptious. The UrbanDictionary.com list was an effective resource for qualitatively evaluating the security of the passphrases collected. It was able to match words found that our COCA n-gram lists could not. The grammar of slang needs to be determined, as like most other words, slang can be used in multiple contexts. Slang and non-dictionary words change very rapidly, with new terms appearing every year, and even more terms that are used but never gaining significant traction. Such terms can be viewed and studied using modern mediums of communication such as social media; Twitter and Facebook are examples of the pioneers and provide a vista of the future of the written word.

7.2.2 Creation Policies

If we consider all the results that we gathered and analysed, some of the most significant findings open up questions in to how password policies are created. One very interesting result is that users were quite successful in remembering their cue. Users responded very favourably to the question "Would you use the same cue again?" and "Would you use StoryPass for some your accounts?". If we could re-run the user study, I would put the cue creation on the same page as the passphrase creation and I would really emphasize the use of it to enhance the memorability of the passphrase. I believe this to be the case as the cue can provide context and a starting point for the user to remember their passphrase. Much like a photo reminds you of the event and context of the photo. By asking our users to create the cue after the passphrase, we didn't emphasize the point of using a single significant memory, idea or object to

inspire the creation of a passphrase.

Emphasizing proper grammar and a sentence-like structure is important for usability and memorability of a passphrase. Future creation policies should reflect this discovery.

An important aspect we did not have issues with in our study, but considered after is that we did not implement any mechanisms for preventing abuse of the creation policy. For example, if someone set their passphrase to " 1 1 1 1 1 1 1 ", it would be accepted because 1 can be considered a proper noun and it is at least 7 words separated by spaces. To counter this, strategies such as enforcing a minimum character length per word or over all character length could be done. It is also possible to check if some of the words exist in dictionaries. This would need to be pilot tested to ensure that the changes to the policy do not introduce any factors which cause usability or security issues.

Language specific rules should be considered when considering creation policies for passphrases. As different languages have different rules and patterns, different rules need to be determined with theoretical analysis and pilot testing.

Determining if 7 words minimum for our creation policy is sufficient is another analysis that we could perform with our data. Determining if meeting our minimum requirements was sufficient or if users needed to go beyond the minimum word length requirements to achieve a secure passphrase. We could simply group the passphrases based on length and number of proper nouns to answer the question "Were the weakest passphrase those that just followed the minimum?". However, this is more complex than at first thought; if we were to raise the minimum to 8 words, an attacker would know this and adjust their attack model accordingly and users would most likely also

adjust their creation habits as well. This analysis could be done at a theoretical level with our current data, but running another user study that groups participants based on word length could help definitely answer this question.

7.2.3 Other Potential Improvements

I firmly believe that using a long passphrase, such as the ones created in StoryPass, as a master password for a password manager such as LastPass to be the best short term goal for improving user authentication. Password managers remove the memory burden and improve usability by offering plug-ins for many web browsers and application to allow for fast or instant password entry. This improves usability vastly by eliminating password entry time and removes the hassle of personally managing a collection of login credentials.

Considering the context that the passphrases in this user study were made in, we can see that users included directly the term "UOIT" in their passphrase. In the same sense that knowing context can help guessing algorithms, it can be interpreted that users can harness their environment to create passphrases. It may be possible to suggest random ideas or nouns to help create unique passphrases. This needs to be done carefully to not bias users, create predicable passphrases, or negatively affect usability or memorability.

We perform error correction based on Levenshtein distance during logins to allow for typographic errors. It would be interesting if semantic error correction could be performed. Considering that one of the passphrase pitfalls outlined in Section 5.3.3 is the re ordering of common word choices during login attempts, semantic error correction could potentially alleviate this.

7.3 Final Thoughts

The literature review, the user study and the analyses have invoked some of the most thought brewing ideas that I have ever experienced. One over arching idea and theme that I cannot shake is this idea that people remember ideas and memories in a very specific manner. We and all us, as researchers, try to design computers to do tasks, automate processes but we tend to neglect the usability and human interface which we design for. We tend to forget that human use is the final destination for all the work that computers perform for us. When considering all the data points I have gathered about passphrases, passphrase length, proper nouns, slang, memorability, mnemonic cues, storage rates and so, it becomes extremely difficult to correlate the results in a meaningful way. Similarly, Bonneau et al. [6] stated that many password studies fail to have any impact due to lack of consistency in the structure of studies; this makes it difficult to correlate results and state the significance of research variables. While this may be the case for some of the results gathered in our user study, one lingering idea remains from my experience of talking with participants, qualitative observations and from my interpretation of all the data. Users like to remember their password, one password, their first password, their first password creation method. The first of anything has a deep significance to human habit. We all celebrate the first day of our existence, the first day of a relationship, the first day of the year. The first of anything tends to carry an innate amount of memorability and familiarity regardless of what it is. I feel like this tendency can be seen in the use of cues and the users sentiment in being unwilling to adopt any method besides passwords for authentication or more humanely, with the nostalgic tendency of human memory. Passwords and passphrase need to reflect this human habit; specifically with something like passwords which is something that an individual keeps secret. The point that I am trying to draw upon is that people do not want to change, they do not want to be told what to do

different or what is uncomfortable. It can be said that the first mechanism of user authentication, most likely, the only mechanism people know are passwords. That's the main underlying point I can draw after reviewing all the literature review and my experience being a researcher. This research suggests that passphrases created with the StoryPass creation policy offer a minimal change to users which allow a significant gain to security and minimal usability losses.

References

[1] Adams, A., Sasse, M., and Lunt, P. Making passwords secure and usable. *People and Computers XII* (1997), 1–15.

[2] Adar, E. Why I hate Mechanical Turk research (and workshops). In *Proceedings of CHI Workshop on Crowdsourcing and Human Computation.* (2011), pp. 12–35.

[3] Baddeley, A. D. The magical number seven: still magic after all these years? *Psychological Review 101*, 2 (1994), 353–256.

[4] Baddeley, A. D. *Human memory: Theory and practice.* Psychology Press, 1997.

[5] Bard, G. Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. *In Proceedings of the Fifth Australasian Symposium* (2007).

[6] Bonneau, J., and Herley, C. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. *IEEE Symposium on Security and Privacy (2012) 2012*, 817 (2012), 553–567.

[7] Bonneau, J., and Schechter, S. Towards reliable storage of 56-bit secrets in human memory. *In Proceedings for USENIX Security* (2014).

[8] Bonneau, J., and Shutova, E. Linguistic properties of multi-word passphrases. *Financial Cryptography and Data Security 43*, 2 (2012), 1–12.

[9] Brown, C. D. MSc thesis, Carleton University, a meta-scheme for authentication using text adventures, 2010.

[10] Brown, P., Pietra, V., and Mercer, R. An estimate of an upper bound for the entropy of English. *Computational Linguistics 18.1 10598* (1992), 31–40.

[11] Burr, W. E., Dodson, D. F., and Polk, W. T. Electionic authentication guidelines. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf, Retrieved August 10, 2014.

[12] C. Danescu-Niculescu-Mizil; J. Cheng; J. Kleinberg; L. Lee. You had me at hello: how phrasing affects memorability. In *Proceedings of the ACL* (2012), vol. abs/1203.6, pp. 892–901.

[13] Carminatti, N., Borges, M., and Gomes, J. Analyzing approaches to collective knowledge recall. *Computing and Informatics 25*, 3 (2012), 547–570.

[14] Cowan, N. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *The Behavioral and Brain Sciences 24*, 1 (Feb. 2001), 87–114; discussion 114–85.

[15] Craik, F., and Lockhart, R. Levels of processing: a framework for memory research. *Journal of Verbal Learning and Verbal Behavior 684* (1972), 671–684.

[16] Davies, M. Corpus of Contemporary American English. http://corpus.byu.edu/coca/, Retrieved December 3, 2012.

[17] Florencio, D., and Herley, C. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* (2007), pp. 657–665.

[18] Florêncio, D., and Herley, C. Where do security policies come from? In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (2010).

[19] Goodin, D. How the bible and youtube are fueling the next frontier of password cracking. http://arstechnica.com/security/2013/10/how-the-bible-and-youtube-are-fueling-the-next-frontier-of-password-cracking/, Retrieved August 2, 2013.

[20] Hayashi, E., and Hong, J. A diary study of password usage in daily life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, New York, USA, 2011), ACM Press, p. 2627.

[21] Herley, C. So long, and no thanks for the externalities: the rational rejection of security advice by users. *In Proceedings of the 2009 Workshop on New Security Paradigms Workshop* (2009).

[22] Hsu, J. The secrets of storytelling. *Scientific American Mind 19*, 4 (2008), 46–51.

[23] Inglesant, P., and Sasse, M. The true cost of unusable password policies: password use in the wild. In *Proceedings of the 28th International Conference on Human factors in Computing Systems* (2010), vol. 1, pp. 383–392.

[24] Jeyaraman, S., and Topkara, U. Have the cake and eat it too-infusing usability into text-password based authentication systems. *Computer Security Applications Conference 21*, ACSAC (2005), 473–482.

[25] Juang, K., Ranganayakulu, S., and Greenstein, J. Using system-generated mnemonics to improve the usability and security of password authentication. *In Proceedings of The Human Factors and Ergonomics Society Annual Meeting 56*, 1 (Oct. 2012), 506–510.

[26] Keith, M., Shao, B., and Steinbart, P. J. The usability of passphrases for authentication: An empirical field study. *International Journal of Human-Computer Studies 65*, 1 (Jan. 2007), 17–28.

[27] Keith, M., Shao, B., and Steinbart, P. J. A behavioral analysis of passphrase design and effectiveness. *Journal of the Association for Information Systems 10*, 2 (2009), 63–89.

[28] Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., and Lopez, J. Guess again (and again and again): measuring password strength by simulating password-cracking algorithms. *2012 IEEE Symposium on Security and Privacy* (May 2012), 523–537.

[29] Komanduri, S., Shay, R., and Kelley, P. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems.* (2011), pp. 2595–2604.

[30] Kuo, C., Romanosky, S., and Cranor, L. F. Human selection of mnemonic phrase-based passwords. In *Proceedings of The Second Symposium on Usable Privacy and Security* (New York, New York, USA, 2006), ACM Press, p. 67.

[31] Laird, J. D., Wagener, J. J., Halal, M., and Szegda, M. Remembering what you feel: effects of emotion on memory. *Journal of Personality and Social Psychology 42*, 4 (1982), 646–657.

[32] Markov, A. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. *Dynamic Probabilistic Systems 1* (1971).

[33] MaxMind. World Cities. http://www.maxmind.com/en/worldcities, Retrieved February 6, 2014.

[34] Miller, G. A. The magical number seven. *The Physological Review 63*, 2 (1956), 81–97.

[35] Morris, R., and Thompson, K. Password security: a case history. *Communications of the ACM 22*, 11 (1979).

[36] Nielsen, G., and Vedel, M. *Improving usability of passphrase authentication.* PhD thesis, Technical University of Denmark, 2009.

[37] Peckham, A. Urban Dictionary. www.urbandictionary.com, Retrieved July 20, 2014.

[38] Pliam, J. On the incomparability of entropy and marginal guesswork in brute-force attacks. *Progress in Cryptology INDOCRYPT* (2000), 67–79.

[39] Project Free Rainbow Tables. Free rainbow tables. https://www.freerainbowtables.com/, Retrieved August 20, 2014.

[40] Rao, A., Jha, B., and Kini, G. Effect of grammar on security of long passwords. *In Proceedings of The Third ACM Conference on Data and Application Security and Privacy* (2013), 317.

[41] Renaud, K. Quantifying the quality of web authentication mechanisms: a usability perspective. *Journal of Web Engineering 3*, 2 (2004), 95–123.

[42] Sasse, M. A., Brostoff, S., and Weirich, D. Transforming the 'weakest link' — a human/computer interaction approach to usable and effective security. *BT Technology Journal 19*, 3 (2001), 122–131.

[43] Shannon, C. Prediction and entropy of printed English. *Bell System Technical Journal 1*, 30 (1951), 50–64.

[44] Shay, R., Kelley, P., and Komanduri, S. Correct horse battery staple: exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security.* (2012), p. 7.

[45] Shay, R., Kelley, P. G., Leon, P. G., Mazurek, M. L., Christin, N., and Cranor, L. F. Encountering stronger password requirements : user attitudes and behaviors categories and subject descriptors. In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (2010).

[46] Shay, R., Komanduri, S., and Durity, A. Can long passwords be secure and usable? In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems* (2014).

[47] SSA. List of baby names for 2012. http://www.ssa.gov/oact/babynames/limits.html, Retrieved June 16, 2014.

[48] Topkara, U., Atallah, M., and Topkara, M. Passwords decay, words endure: secure and re-usable multiple password mnemonics. *Proceedings of the 2007 ACM Symposium on Applied Computing. 3225*, 2 (2007), 292–299.

[49] Tulving, E., and Schacter, D. L. Priming and human memory systems. *Science 247*, 4940 (Jan. 1990), 301–6.

[50] Weir, M., Aggarwal, S., Collins, M., and Stern, H. 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* (2010), ACM Press, p. 162.

[51] Wixted, J. T. The psychology and neuroscience of forgetting. *Annual Review of Psychology 55* (Jan. 2004), 235–69.

[52] Yan, J., and Blackwell, A. Password memorability and security: empirical results. *IEEE Security Privacy Magazine 2*, 5 (2004), 25–31.

[53] Zechmeister, E., Chronis, A., Cull, W., D'Anna, C., and Healy, N. Growth of a functionally important lexicon. *Journal of Literacy Research 27*, 2 (June 1995), 201–212.