Bachelor Degree Project

UNIVERSITY OF SKÖVDE

EXPLORING THE NUMBER OF TRIES RELATED TO CRACKING PASSWORDS GENERATED WITH DIFFERENT STRATEGIES

Bachelor Degree Project in Information Technology
IT610G, G2E, 22.5HP
Spring term 2019
Date of examination: 2019-06-07

Marcus Birath
b16marbi@student.his.se

Supervisor: Joakim Kävrestad
Examiner: Marcus Nohlberg

Table of content

Abstract

As more services and workflows are moved into computerized systems the number of accounts a person has to be aware of is on steady increase. Today the average user is likely to have more than 25 accounts for different services used on a daily basis that all need authentication. The dominant authentication mechanism used today is still password authentication. In an attempt to satisfy the requirements of different password creation policies and to recall all passwords when needed users tend to rely on different strategies for password creation. These strategies may all seem to provide adequate security, and they may do, but the reality is that they differ tremendously in terms of how time consuming it is to crack passwords generated with the different strategies. By conducting interviews with domain experts different password creation strategies are discussed and pseudo algorithms for cracking passwords are constructed. Based on mutual definitions of the classes and a predefined word list the cost for cracking passwords generated by different strategies are explored. Major findings indicate that strategies based on phrases are at the top of the list. Using a strategy to create a seemingly random password based on a logical phrase, where only the first letter from each word is used, tends in some cases to be the best of choice. An example is to turn the phrase "this password is the greatest of them all" into "tpitgota" instead of using the phrase "goodword" to create an 8 character long password. However, if the phrase contains words not usually found in common dictionaries the best strategy seems instead to be utilizing character substitution as in turning the phrase "my dog Krillex is cool" into "myDoGkriLLExiscooL".

Acknowledgments

First I would like to thank all the interview participants for taking time off their busy schedule to help me conduct this study. Next, I would like to thank my supervisor Joakim Kävrestad for introducing me to this study topic and with that, offering me a chance to participate in a bigger research project. He also deserves my deepest gratitude for giving me guidance and feedback throughout the whole study process. I also want to thank the course examiner Marcus Nohlberg for providing me with valuable feedback regarding how to improve the report structure and content. Last, but by no means least, I would like to thank my study peers Markus Lennartsson and Ramus Kullman for providing additional feedback and support throughout my studies.

Marcus Birath

1 Introduction

Authentication is used in systems to ensure user identity before giving access to the system's functionality where also personal or sensitive information may be present (Pfleeger, Pfleeger and Margulies, 2015). Sensitive information may include organizational secrets or customer information; personal information such as addresses, bank account numbers or credit card numbers. According to a study by Florencio and Herley (2007) the average user has approximately 25 accounts where authentication is needed before access to its information or functionality is granted. Although that study was conducted years ago it is not likely that the number of accounts has decreased. Contrary, with the advent of workplace systems, social media, cloud computing, and other online services it is possible that the average user now has more than 25 accounts (Yıldırım and Mackie, 2019).

Since the introduction of GDPR (General Data Protection Regulation), which regulates how information is stored, handled, and accessed, the importance of restrictive access to sensitive information is higher than ever. Identity Force (2019) lists some of the major data breaches over the last years. Several of which include the theft of password hashes which an attacker later can use in an offline attack to retrieve the password.

Passwords remain to be the primary authentication mechanism used by system administrators for systems and networking equipment. It is also the predominant method for user authentication. However, passwords are, according to Yıldırım and Mackie (2019), one of the biggest weaknesses when it comes to system security as they are susceptible to attacks and thus passwords need to be hard to guess. Ur et al. (2016) provides comprehensive conclusions about users' misconceptions of what constitutes a strong password as well as how attacks are executed, and implies that the most promising evaluation help for users is targeted feedback during password creation.

According to Yıldırım and Mackie (2019) the current guidelines generally provided to users regarding password creation are not enough to motivate users to choose better passwords. Users are not aware of the implications of not following the guidelines and try to circumvent the complexity by using evasive strategies.

This study aims to explore the cost of cracking passwords generated by different semantic strategy classes for user password creation presented by Kävrestad, Eriksson and Nohlberg (2018). This paper will use the term strategy and class interchangeably to refer to the semantic strategies used to create passwords. In this context, the notion of "cost" is the number of iterations that in worst case are needed to crack a password within each class (i.e. how many tries are needed to test all possible password combinations that can be created by utilizing a creation strategy together with a given word list). Together with future research, on the memorability of passwords created within these classes, this study hopes, by exploring the cost of cracking passwords within each class, to provide a foundation for instant usability feedback to users upon password creation as well as educating users and administrators in what is a secure, easy to remember password.

2 Background

The main topics that are of importance for this study are: authentication, password policies, password cracking, and the strategies used to circumvent the complexity of the policies. To better understand the study, these topics will be presented in more detail.

2.1 Authentication

According to Pfleeger, Pfleeger and Margulies (2015) authentication is the process of proving ones identity (i.e. that a person is who she says she is). Authentication is used everywhere from systems at work, cloud and online services, to social media and bank accounts. The authentication can be based on something you know (e.g. password or passphrase), something you are (e.g. fingerprint or voice) or something you have (e.g. key or phone). Stavrou (2017) states that passwords still are the leading method for authentication and that weak passwords are the primary vulnerability exploited by attackers to gain unauthorized access.

2.2 Password Policies

System administrators are aware of this fact and in order to protect sensitive data organizations create password policies stating how users should manage their passwords regarding for example composition requirements, reuse, and password storage constraints (e.g. the user can not write down the password). These password policies are often based on enforcing technical difficulties (e.g. long passwords) and show less concern regarding human usability (Choong & Theofanos, 2015). A study by Komanduri et al. (2011) explores how different password creation restrictions affect users during password creation. Findings include changes in memorability and frustration but the most relevant conclusion is that, regardless of restriction, users tend to just barely exceed the minimum requirements. Patterns were found where users try to overcome the complexity by using one strategy after the other. One example presented is where the rejected password “cheese” (regular dictionary word) turned into “1cheese1” which was rejected again to eventually turn into “12#$qwER” (a pattern of adjacent characters). They also found that the participants who successfully created a password with more complex restrictions tended to store their password in some way which opens the discussion of security versus usability.

2.3 Password Security

When discussing passwords and password security there are several things that will affect whether a password is considered to be more or less good. A common topic for discussion is the security versus usability of passwords. From a theoretical point of view, as an attempt to make password guessing harder many services and organizations enforce password policies that should be followed when creating a password. However, if a password becomes too complex to remember a user may write it down and be less willing to change password regularly (Komanduri et al., 2011). Writing down the password on a piece of paper and put it on the computer monitor might be harmless, unless it is the cleaning personnel alone at night that poses a threat. From a technical viewpoint passwords are often stored as an encrypted hash to protect it from being viewed by unauthorized persons. To further secure the passwords they may be run through the hashing algorithm several times or mixed with a salt (e.g. a username) making it more difficult and time consuming for an attacker to crack it. Balagani et al. (2018) present how attackers may take advantage of shoulder surfing or analyzing the in- and output devices to retrieve a password consequently bypassing other security enforcement procedures. In a process called phishing, an attacker abuses the human factor and tricks users into revealing sensitive information such as password protected information or maybe the password itself. Education and awareness of potential threats are also considered to be security procedures (Thakur & Verma, 2014).

Security involves consideration of many different aspects like policies, technical difficulties, human factors, education etc. It all depends on what to protect and who to be protected from.

2.4 Cracking

Passwords are generally not stored as plain text in systems. Instead, they are stored as a scrambled mess called a hash. This hash is the result of putting a plain text password through a non-reversable function which makes it generally difficult to retrieve the plain text password by reverse-engineering the hash. The attacker's only way to find out the plain text password is by putting possible passwords through the hashing algorithm and then compare the hashes. If the computed hash matches the stored hash on a system the plain text password has been found (Arends et al., 2018). These guessing procedures are often categorized as dictionary attacks or brute force attacks. In a brute force attack, as the name implies, the attacker aims to test every possible combination of a chosen character set until the right password is found. A brute force attack will eventually always succeed but is exceedingly time consuming (Pfleeger et al., 2015). In a dictionary attack, to save time, the attacker uses a predefined word list to see if any word in the list matches the password. This list can be either a straight up language dictionary or a modified version in which the words have been changed in accordance with different semantic strategies that users are believed to utilize. As users tend to use semantic modifications of regular words, for easier memorability, a dictionary attack is often the better choice and less time consuming for an attacker. However, this method requires the dictionary to include the base password used for modification (Weir, Aggarwal, Medeiros & Glodek, 2009).

2.5 Strategies

As presented by Komanduri et al. (2011), Ur et al. (2016) and Stavrou (2017) there is no secret that most users turn to strategies when creating passwords. The strategies help users to comply with password creation policies and to remember more complex passwords. Kävrestad et al. (2018) presents a model that classifies commonly used strategies used for password creation. By conducting interviews with forensics experts from the Swedish police discussing how users tend to create their passwords and by validating their model against lists of leaked passwords they present the strategies shown in Figure 1 below.

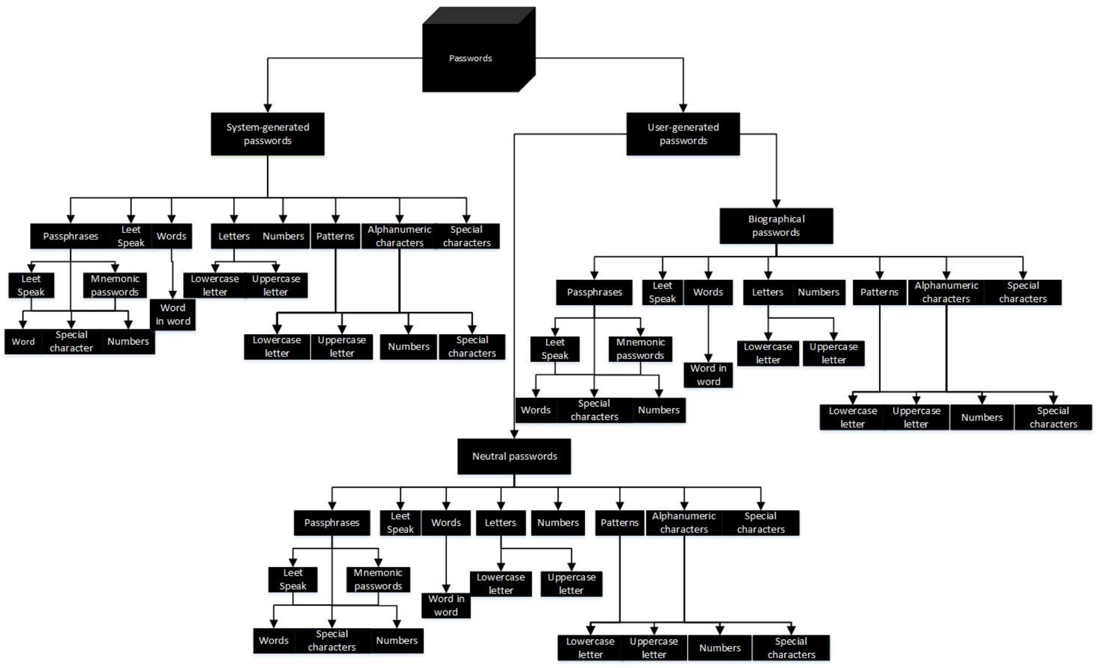


Figure 1. Password classification model (Kävrestad et al., 2018, p. 11)

The two main categories are “System generated passwords” which is generated without human intervention, and “User generated passwords”. The latter is further divided into “Biographical passwords” which includes words of interest to a person (hobbies, family, fictional characters etc.) and “Neutral passwords”. The tree then branches out to the different types of passwords that a user may create and their building constructs.

3 Problem

The main issue is that users keep using predictable, easy to crack, passwords. While system administrators try to prevent this behavior by creating password policies, users keep finding ways around to cope with the complexity the policies demand (Ur et al., 2016).

Further confirmed by Stavrou (2017), password policies seem to make users choose a semantic strategy when creating passwords which generates passwords with different levels of complexity. A study by Wier, Aggarwal, Collins and Stern (2014) shows for example that over 64% of their analyzed passwords only appended a number after a base password when forced to use 7+ character passwords. If users also had to include 1 non-digit character this percentage was raised to over 77%. This confirms the conclusions of both Ur et al. (2016) and Stavrou (2017) that users are unaware of password complexity and that semantic strategies are heavily used.

The lists of leaked passwords on the Internet also show that users still create weak, easy to crack passwords despite (or perhaps because) restrictive password policy enforcement (Stavrou, 2017). This seems to be somewhat contrary to the intention of the password policies. Despite the many password creation strategies suggested by standards, researches, essays, and communities there is little knowledge about how the different strategies are helping users to create secure memorable passwords (Yang, Li, Chowdhury, Xiong & Proctor, 2016). There is an explicit need to examine how different semantic strategies compare to each other as users seem to keep utilizing them for password creation.

The different strategies presented by Kävrestad et al. (2018) that are used when creating passwords will most likely generate passwords with dissimilar complexity. With focus on the technical aspects of password security the aim of this study is to define and create algorithms representing these strategies in order to:

Explore the cost related to cracking passwords generated with the different strategies

By defining classes and developing pseudo algorithms that represent the logic of cracking password generated by different strategies the main objective for this study is to calculate the cost for cracking passwords within each of the classes presented by Kävrestad et al. (2018) and compare them to each other. It is this study's intention to help system and network administrators, as well as regular users, to choose the strategies that generate more secure, harder to crack, passwords for systems and equipment by exploring which of these heavily used strategies are more costly to crack and hence more secure. Together with research on which classes might be more easy for users to remember this study hopes to contribute to a foundation for better password policies in organizations.

If administrators know the cost of cracking, and the memorability of each password creation technique, they can also create tools that analyze user passwords, while being created, and give recommendations on the usability regarding security and memorability.

The algorithms and results could also be used as a foundation to more easily create functional, language specific, scripts or dictionaries to crack passwords. From an IT forensics point of view these algorithms could reduce the time spent to crack passwords by focusing on specific methods instead of using brute force attacks. However, that would call for more research on which classes are more likely to be used as choosing the wrong method would be exceedingly time consuming.

3.1 Limitations

There are probably more strategies in existence than what is presented in this report. However, this study aims to focus on the strategies presented by Kävrestad et al. (2018) and calculate the cost for each class according to the collaborated definitions between researcher and interview subjects.

The term cost could mean many things, for example: It could be the monetary cost for buying the equipment used to crack passwords; it could be the time needed to crack the passwords: it could be the consequences of succeeding cracking the passwords or not. However, this study defines cost as the number of iterations that in worst case are needed to crack a password within each class (i.e. how many tries are needed to test all possible password combinations that can be created within each class by utilizing a given word list).

It is acknowledged by this study that the cost of cracking each class will be heavily dependent on the definition of the classes. Hence the results may vary if the definitions of the classes were to be different, which should be considered when interpreting the results. To address this issue, this study states that each class will be defined in its most general form and presented alongside each algorithm. As an example, the algorithm for the class "word-in-word" will be addressed as consisting of only 2 words, though theoretically it could be an arbitrary amount of words.

To further make the cost of the classes more comparable, the same foundation (word list, dictionary, alphanumeric characters etc.) will be used for all classes. The cost may differ as different languages consists of more or less words and letters. What is actually a "phrase" in one language might be seen as "random characters" in another. To address this problem this study clearly limits the language and alphabet to be used to be English. Other characters will be the 94 printable characters from the ASCII table.

As presented earlier, there are many aspects regarding the security of a password. To reach its goal this study will focus on the technical aspects of password security which include the computational complexity of cracking a password. Computational complexity in this study means the number of tries needed to test all possible passwords generated within the limitations of each class.

The pseudo algorithms presented with each class will only represent the logic used to crack passwords within that class and that logic can later be implemented in a case specific appropriate language. Hence, no specific code language will be used when writing the algorithms.

3.2 Expected Results

This study hopes to provide an initial comparison between the costs for cracking passwords generated by different strategies. The outcome will most likely not be a real world representation of the classes but rather a best to worst order of the strategies when defined in their most general form. With that information, hopefully a separation can be done between more and less secure strategies. This study hopes to contribute to a foundation for better password policies and help administrators and regular users to chose the semantic strategies that are the most time consuming for an attacker to crack.

4 Methodology

In order to achieve the research aim this study will have a qualitative focus where cracking algorithms will be created based on the researchers interpretation of the classes and how current literature define the classes. The algorithms and definitions will then be refined by conducting semi-constructed interviews with domain experts.

This study will utilize a hybrid approach where action research is mixed with a grounded theory approach for collecting and analyzing data.

4.1 Study Procedure

This section provides an overview of the study method.

Procedure for the study:

1. Study the topic
2. Define the classes
3. Create algorithms
4. Create interview questions
5. Do the interviews
6. Transcribe and analyze interviews
7. Refine the algorithms
8. Repeat step 5 if necessary
9. Rank algorithms according to cost
10. Draw conclusion

These steps can be divided into three logical phases which are presented below in Figure 2. Phase 1 is the preparation phase where initial data is collected and analyzed to provide input to phase 2, the execution phase. In phase 2, the actions to collect and analyze data with other methods are performed. This phase might be repeated depending on the outcome of the interviews and will be iterated until no more feedback is obtained from the interview subjects. In phase 3, the cost of cracking passwords within the classes will be calculated, results presented and conclusions will be drawn in accordance with the research aim.

The strategies provided by the study of Kävrestad et al. (2018) will be used as a foundation for this study. Initially the strategies will be defined based on current literature and the researcher's interpretations. Based on these definitions, pseudo algorithms will be created that represents how passwords could be cracked with a predefined word list. As shown below in figure 2, each step is affected by the one preceding it. Both the definitions and the algorithms will provide input for the interviews which will be conducted in order to apply potential refinements of either one. Finally, the costs will be calculated based on the resulting definitions.

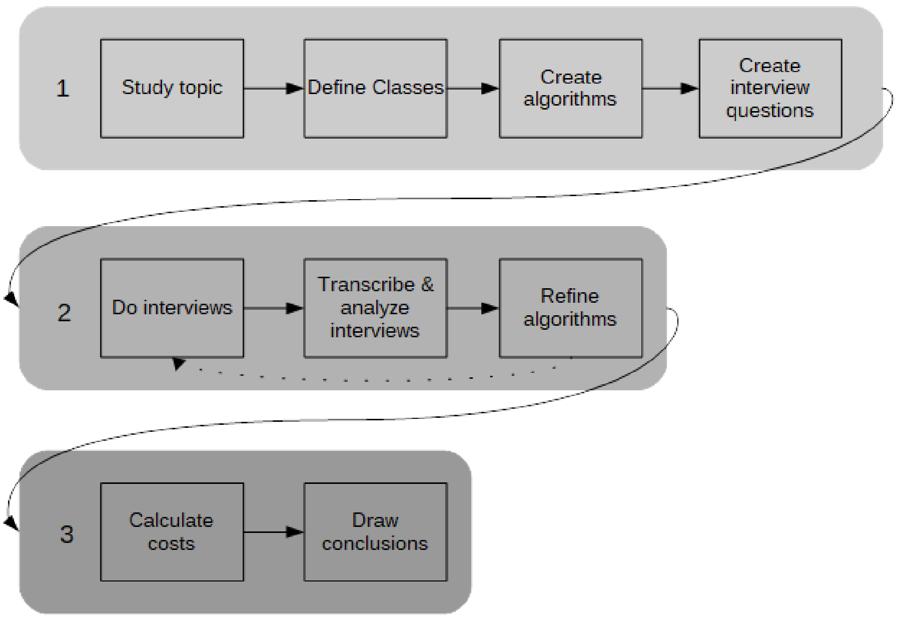


Figure 2. The three phases of the study process (author's own)

4.2 Action Research

According to Stringer (2014) an action research method is a collaborative method where systematic action is used to reach a specific desired goal. This method is often used to improve specific practices by systematically analyzing collected data. As the aim of the interviews is to improve the definitions and the algorithms in order to reach a conclusion about something specific this method is deemed highly relevant for the study. The Look, Think, Act procedure of an action research method, presented by Stringer (2014), is highly applicable to several steps in the study process including all steps of phase one presented above in Figure 2. As an example, reading (look) current literature and thinking (think) about how it could relate to the strategies will result in the action (act) of defining the classes. By analyzing (look and think) the defined classes the algorithms can be created (act), and so on.

In an action research the researcher has an active role in the study and a collaboration with the study participants in order to reach a conclusion. This seems to comply with the aim of this study as the active role of the researcher will be found both in the initial definitions of the strategies as well as in the interviews. The collaborative aspects will be found in the interviews where domain experts are engaged in the refinement of the definitions and algorithms which will have a direct impact on the results.

4.3 Grounded Theory

The aim of a grounded theory method is to build a theory based on data that is gathered and analyzed in a systematic way (Johnson & Christensen, 2014). Even though there is an underlying assumption that the strategies may differ in cost, the goal is not to prove nor invalidate this assumption but to explore if a difference among the strategies exist and thus create a theory grounded in collected and analyzed data.

A grounded theory is also an iterative study process where the analysis of one data set may provide input to the next phase which is done throughout both phase 1 and 2 of this study. This allows for different data collection methods to be used (Johnson & Christensen, 2014). The two methods for data collection used in this study are current research and semi-constructed interviews where the first provides input to the latter.

4.4 Selective literature review

A selective literature review is often used to start a qualitative study. As opposed to utilizing a literature review as the main method for a research, which would result in a structured and much more comprehensive review, a selective literature review can be used to get initial information and a broader perspective of the topic to be studied by finding seemingly relevant literature (Yin, 2011). To get the study process started, the work of Kävrestad et al. (2018) and its sources will be used as a foundation for initial definitions of the classes. If no definitions can be interpreted from these sources a search term relevant to the class (e.g. “mnemonic passwords”) will be used on databases to find relevant research. If still no literature can be found, the class will be defined solely on the researchers interpretations. The main goal of this step is merely to obtain information to create a draft of definitions and algorithms that will act as input to the interviews. As it is in the interviews where possible refinements will take place a more structured method at this point is deemed unnecessary. Based on these definitions, pseudo algorithms will be created that represents how passwords could be cracked with a predefined word list. Both the definitions and the algorithms will provide input for the interviews which are conducted in order to apply potential refinements of either one. An overview of the initial study phases is illustrated in Figure 3 below.

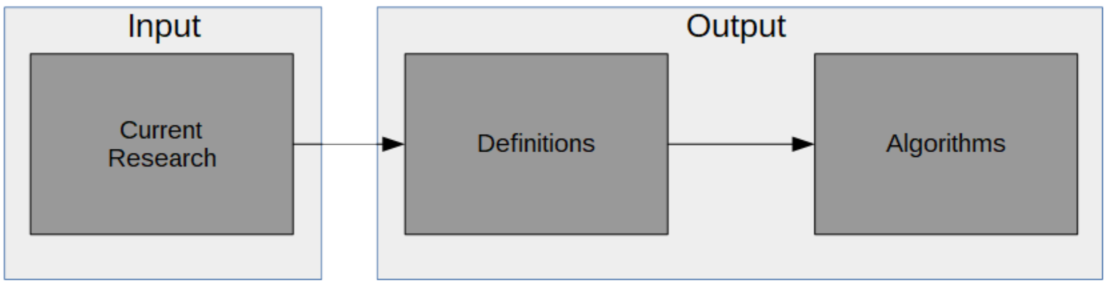


Figure 3. Overview of study phase 1 (author's own)

4.5 Semi-constructed Interviews

When conducting the interviews for data collection this study will utilize the method of a semi-constructed interview study. A semi-constructed interview study can be based on open questions that are planned in advance but do not necessarily follow a particular order. All questions may or may not be asked to all interview subjects as this method allows for improvisation and flexibility within the interview conversation (Wohlin et al., 2012). As the aim of the interviews is to get feedback and improve the algorithms and definitions based on the interviewee's thoughts and reflections it is also recommended by Berndtsson, Hansson, Olsson and Lundell (2008) to use open questions which will lead the discussion to what the interviewee emphasizes. This is common in qualitative research and a useful way to avoid bias from the researcher (Berndtsson et al., 2008).

This type of interview suits this research well as all participants may not have opinions on all or the same strategies and may have different ideas and suggestions regarding password cracking due to their background.

As suggested by Wohlin et al. (2012) the interviews will be divided into three phases. In phase one, the interviewer will shortly describe the topic and the purpose of the interview followed by questions regarding the interviewee's background. Phase two will consist of questions and discussions relating to the improvement of the algorithms or the definitions of the classes. Phase 3 will be a summary of the feedback obtained in order to avoid misunderstandings.

When conducting a qualitative study it is recommended to select interview subjects based on differences instead of similarities. As the aim for the interviews is to improve, it is important to account for different viewpoints and interpretations. If the same conclusion still can be drawn from multiple sources it will be of greater significance (Wohlin et al., 2012). To account for that, this study will consist of three interview subjects which all have expertise within IT but with different roles and backgrounds. An overview of this step is shown below in Figure 4.

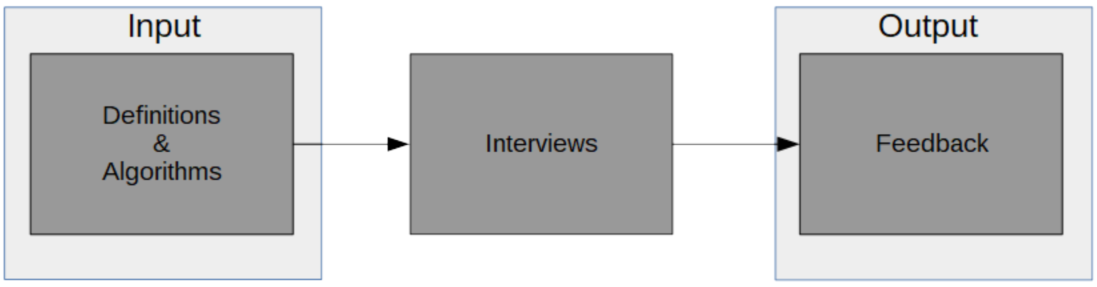


Figure 4. Input and output of the interview phase (author's own)

4.6 Analysis

The next step is to transcribe and analyze the recorded interviews. To help arrange the data from the interviews in a structured and meaningful way this study will borrow some concepts and frameworks related to grounded theory. A common data analysis approach in grounded theory is the notion of coding. Coding refers to the iterating process of analyzing the data systematically in order to categorize it. Different levels of categories are then created to find patterns and connections between the analyzed data (Johnson & Christensen, 2014). This study will make use of this concept of categorizing data in a systematic way. As mentioned in chapter 4.5 the interviews will consist of open questions. This may lead to other feedback than just refinements of the algorithms or definitions. Hence, the first step when analyzing the transcriptions will be to read them sentence by sentence and separate the data into two categories: refinements, and other thoughts. The refinements will consist of changes explicitly directed to the definitions or the algorithms. Each proposed refinement will then be categorized to its respective class and ultimately lead to a refinement of the same. Other thoughts will be further analyzed to see if the feedback is relevant to the aim of the study. If relevant to the study, the feedback will either lead to changes or be the topic of the discussion or future work. If other discussions arise that are not relevant to the research aim or topic they will be discarded (i.e. not included in this report). The analysis process is depicted below in Figure 5.

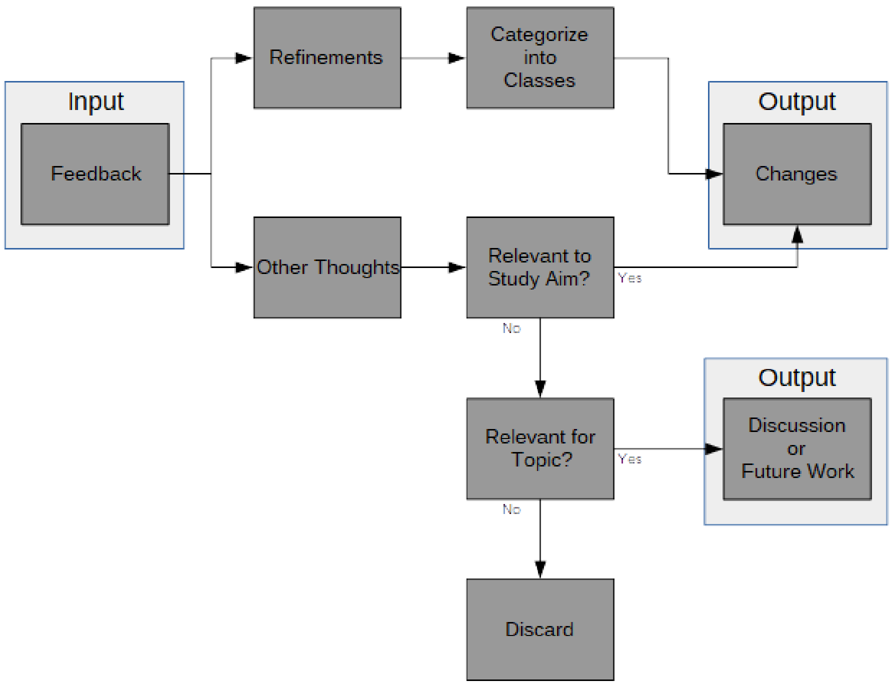


Figure 5. Flow chart of the analysis process (author's own)

As an example, feedback stating that passphrases should include special characters will be a direct refinement of the strategy passphrases and consequently lead to a change of the definition of the class passphrases. On the other hand, an idea regarding future research of passwords strategies will not be relevant to the aim of this study but is related to the topic and thus a valid contribution to the future work section.

If proposed refinements of the same class are different or contradict each other this study will utilize both refinements individually to present possible differences.

4.7 Refinements and Cost

The analyzed feedback will potentially result in refinements of the algorithms or definitions. The refinements will be implemented and communicated back to the interview participants. This is the part of the study where an iterating process might start. If the interviewees have new opinions after the refinements, additional interviews will be conducted and the analysis process will start over. If no additional feedback is to be collected the study will proceed as planned.

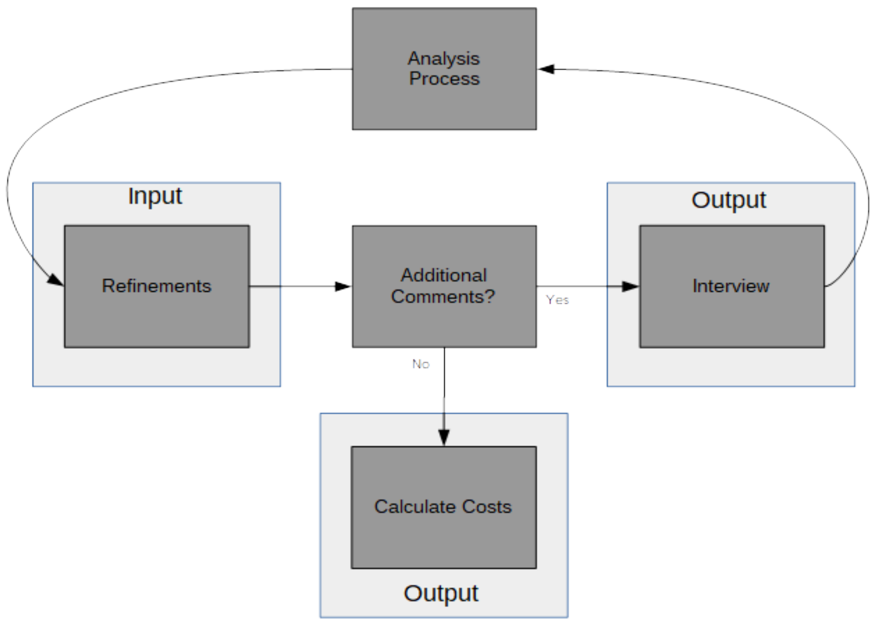


Figure 6. The iterating process of refinement (author's own)

The cost calculation process is assumed to be dependent on the resulting definitions and algorithms and can therefore not yet be exactly specified. The costs will, however, be based on how many possible passwords can be created by utilizing the strategies in conjunction with the predefined English dictionary. This will be further discussed the chapter 5.4.

4.8 Validity

According to Johnson and Christensen (2014) one big potential threat to validity in a qualitative study is researcher bias, which means that the results may be affected by what the researcher wants to find or lack of knowledge. The strategy recommended to avoid researcher bias is the notion of "reflexivity" where the researcher engagingly discusses and acknowledges the potential bias in a critical way throughout the study which, in this study, is done throughout this report.

By using a methodology called "triangulation" one can avoid the potential bias from using a single data source. Triangulation refers to the aim of reaching convergence between multiple data sources or methods to achieve higher conclusion validity. If the same conclusion can be drawn based on data from multiple sources the significance of the conclusions also become higher (Johnson & Christensen, 2014). This study will make use of three interview subjects (data sources) with different backgrounds within the field of IT to get more representative results. It is deemed sufficient by the researcher to only use three interview subjects as their different backgrounds are representative for the purpose of the study. Conducting more interviews would not be beneficial as they would include interview subjects with either similar or very different, irrelevant, backgrounds.

As mentioned, this study relies on the researchers interpretations of various aspects throughout the process. These aspects may be the definition of the classes, choice of language or other limitations. To address this, this research will follow what is recommended by Berndtsson et al. (2008) and use open questions to allow the interview subject to address any issue that it deems relevant and thus, potential bias from the researcher will at least have a chance to be questioned.

Interpretive validity refers to the degree to which the researcher understands or interprets the participants (interview subjects) in the research (Johnson & Christensen, 2014). As mentioned in the "Methodology" section, the interview process will consist of three phases where the third phase is dedicated to summarize and validate the feedback from the interview subjects which hopefully will reduce interpretation errors. This mitigation method is backed up by Wohlin et al. (2012) and by Johnson and Christensen (2014) who refers to it as "participant feedback".

4.9 Interview Material

An interview protocol is recommended by Yin (2011) to get the most out of the interviews. This is not meant to be a questionnaire but rather a guideline to keep the interview on track. As the purpose of the interview is to make something better it seems like a good idea to prepare the interview subjects beforehand. This will be done by putting together the algorithms and definitions in a document together with general information regarding the procedure of the interview so there will be less surprises. This material will be found in the appendices. As the interview subjects speak Swedish there will also be a Swedish version of the material.

5 Results

This section will provide results for the initial definitions and algorithms as well as the interview results and potential refinements. It concludes with the final costs for the strategies.

5.1 Algorithms

In this section the classes will be defined, and for each class, a corresponding pseudo algorithm for cracking that strategy will be presented. These definitions and algorithms will be the input to the interviews where potential refinements may emerge. Hence, the costs presented here should not be seen as results of the study as they are only representing the logical cost each class would have in its current state and without the word list. Potential changes to the definitions and the algorithms will be presented after the interview section.

If the algorithm for a specific class generates a higher cost than using brute force the class will be considered more secure and may not be in need of further ranking as the cost for cracking might be the same (i.e. brute force). However, if an algorithm can be created that generates a lower cost than its brute force equivalent it will be considered less secure and can be ranked from highest cost to lowest cost.

As described earlier, the cost is the number of tries needed to test all possible passwords that could be generated by utilizing the strategies in conjunction with a given word list. Therefore, the algorithms will not stop if the "right" password is found. Instead it implies to continue till all passwords are tested which will generate a worst-case scenario cost.

An overview of the strategies initially identified are presented below in Figure 7.

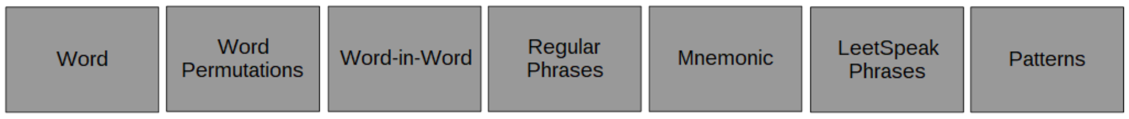


Figure 7. Initially identified strategies (author's own)

5.1.1 Regular Words

This class consists of words that can be found in dictionaries and the words are not altered in any way. Kävrestad et al. (2018) distinguish between regular words and biographic words which are words that relate to the users interests (e.g. family, hobbies, fictional characters etc.) many of which may also be found in regular dictionaries. Even if the word list used for cracking needs to be updated to include some specific biographical words the algorithm will stay the same. Therefore this study will make no such distinction. Neither does this class fractionate lower case letters from upper case letters as they would yield the same cost for cracking. Other word alternation will be presented in the "Permutation" chapter.

Algorithm:

```
for each word in list(
 test if word = password
)
```

As all words in the list are tested, this algorithm implies that the cost is the number of words in the list:

$Cost = \text{List}_{\text{Length}}$

5.1.2 Word in Word

As the name implies, this class is confined to consist of 2 words in total where each word in the word list, including the current word, can be placed in its whole between all letters in the current word. The words "car" and "**hug**" could result in "c**hug**ar". This class could also be defined as twining two words together by having every other letter in the password coming from word 2. Taking for example the words "car" and "**hug**" will result in "c**h**a**u**r**g**". This would however yield the same cost as there are as many spaces to start putting in word2 as there are writing it in its whole. The cost may represent either definitions but not both.

Algorithm:

```
for each word1 in list(
 for each word2 in list
  place word2 in word1 space X
)
```

As the number of spaces in a word always is one less than the length of the word the cost for each word will be:

$Cost = \text{Word}_{\text{length}} - 1 \times \text{List}_{\text{length}}$

5.1.3 Word Permutations

Permutations are different variations (combinations) in which one can change the characters in a word. Permutations can be applied to all classes where each character may be substituted with their corresponding leetspeak character alternatively lower case letters are substituted with upper case letters or vice versa. Leetspeak is, as described by Blashki and Nichol (2005) an alternative language where characters and symbols are substituting for letters with which they share a similar visual appearance (e.g. "A" is substituted by "@"). In both leetspeak substitution and capitalization of letters each character has only two possible states, the original or the altered. By using what Peckel (1999) calls the "binomial coefficient" one can calculate how many combinations that can be made if substituting one or more of the characters with another.

The formula is:

$\mathbf{C(n,r)=n!/(r!*(n-r)!)=n!/(r!*n!)}$

where n is the number of characters in the word and r is a subset of the characters to be substituted (i.e. how many unique combinations of size r can be made from n). In a two letter word the possible permutations are to change the first or the second character (r=1), or both characters (r=2) into either their respective leetspeak character or alternatively substituting lower case letters with upper case letters or vice versa. Thus, the total possible combinations are 3. Table 1 below shows the number of the possible permutations for different lengths.

Table 1. Possible substitution combinations for different lengths

Word/Phrase length	Possible permutations
5	31
6	63
7	127
8	255
9	511
...	...
18	262 143
27	134 217 727
36	68 719 476 735
45	35 184 372 088 831

5.1.4 Regular Phrases

A passphrase is similar to a regular password with the difference that it is usually longer as it consists of concatenated words and a password is defined as only one word. Passphrases can be random (i.e. have no logical meaning) or be a normal sentence to the user (Nielsen, Vedel & Jensen, 2014). This study will focus on both and define a passphrase as words concatenated with other words regardless of meaning. As the algorithm will test all possible combinations of the words in the given word list, passphrases may be both random and logical.

Algorithm:

```
for each word in list(
 for each word X
  append word with word X
)
```

This will present a logical cost of $\mathit{WordList}_{\mathrm{Length}}^{x}$ where X equals the number of words making up the passphrase.

5.1.5 LeetSpeak Phrases

Leetspeak phrases can be defined in different ways. The first is to have the whole phrase as leetspeak which can easily be done by converting the given word list to leetspeak and proceed with the algorithm as done with regular phrases. That will yield the same cost as for regular phrases. However, this study will focus on the possibility that any character combination of the phrase may be substituted with its leetspeak equivalent and therefore leetspeak phrases will be defined as phrases (from chapter 5.1.4) with permutations (from chapter 5.1.3). This way the base phrase algorithm will be the same as for regular phrases. However, after the phrase is created all alternation combinations have to be tested. This algorithm implies to test all alternations of a base phrase before proceeding to the next possible base phrase.

Algorithm:

```
for each word in list(
 for each word X in list
  append word with word X
  test all permutation combinations
)
```

As with the permutation of words, this algorithm covers substitutions regardless of whether it is leetspeak characters or combinations of upper and lower case letters, hence no distinct algorithm is needed to separate those classes. This would imply the same cost as for phrases but for each phrase created the cost will be multiplied with the number of possible combinations for that phrase's length from the "Word Permutation" chapter.

5.1.6 Mnemonic Passwords

Kuo, Romanosky and Cranor (2006) defines mnemonic passwords as using the first letter of each word in a memorable phrase. Yang et al. (2016) calls this the "Mnemonic sentence-based strategy" and claims it is the most recommended usage of this strategy. This study will define mnemonic passwords the same way with the assumption that if a seemingly random string is not based on a phrase it would be just random. As the user does not need to incorporate a logical structure of the phrases nor is there an easy way to check it before converting it to a mnemonic password the most general way is to create phrases based on the same algorithm as for regular phrases and make them mnemonic. The algorithm would need one nested iteration for each word that should be part of the phrase. The algorithm below is shortened for easier overview.

Algorithm:

```
for each word in list(
 for each word X in list
  append word with word X
  make it mnemonic
)
```

This algorithm would imply the same cost as for regular phrases which is $\mathit{WordList}_{\text{length}}^{X}$ but as the phrases are converted to mnemonic passwords the length of the phrases are no longer of importance and thus a brute force attack with all possible characters will result in a lower cost than this algorithm. To crack a 7 character long mnemonic password with this algorithm a 7 word phrase would have to be created which, even with a small word list of 1000 words, would result in a cost of 1000^7 which is much higher than the brute force equivalent of 26^7 using the 26 letter of the English alphabet. As an example:

Following the algorithm to create a 7 character long mnemonic password a 7 word long phrase has to be created. If the word list used consists of 1000 words, following the cost calculation for regular phrases, the cost for testing all possible phrases will be:

$1000^7 = 1\ 000\ 000\ 000\ 000\ 000\ 000\ 000$

If one instead used the 26 letters of the English alphabet to test all possible 7 character long passwords the cost would instead be:

$26^7 = 8\ 031\ 810\ 176$

5.1.7 Patterns

In the book "*Fundamentals of Digital Forensics*" by Kävrestad (2018) the class pattern is restricted and defined as adjacent characters within a given keyboard layout. When discussing patterns this notion of adjacent characters is also shared by Chou, Lee, Hsueh, and Lai (2012). This study will also share this definition and state that for the purpose of this study the class "Patterns" will be restricted to adjacent characters of the English keyboard layout where the 94 printable characters of the ASCII table will be included.

Algorithm:

```
for each char X in set(
 foreach adjacentchar Y(
  append char X with adjacentchar Y
  foreach adjacentchar Z(
   append char XY with adjacentchar Z
  )
 )
)
```

This algorithm implies to use the same start character for all possible combinations of length X before proceeding to the next starting character.

5.1.8 Alphanumeric and Special Characters

Alphanumeric characters are, in this study, defined as numbers, lower case letters and upper case letters. Special characters are those found in the ASCII table that do not fall into the alphanumeric definition. Passwords created with these characters that do not utilize some of the above mentioned strategies will be seen by this study as random passwords. This study states that using these characters without utilizing some of the above mentioned strategies would result in a random password and hence no specific algorithm will be created that generates a cost lower than brute force. The cost for random passwords will therefore be $character_{set}^{Length}$.

If using lower case letters from the English alphabet to generate a 7 character long "random" password the cost would be:

$$Cost = 26^7 = 8\,031\,810\,176$$

More example costs for random passwords of different lengths are presented in chapter 5.4 (Table 8).

5.2 Interviews

Semi-constructed interviews were held where three interview subjects with different backgrounds in IT were asked questions regarding refinements of the definitions or the algorithms. This study utilized the notion of open questions to let the interview subjects emphasize what they consider important and thus reducing researcher bias. As described earlier, the interviews are divided into topic sections where one section lets the interview subject present its thoughts and another where questions are asked about refinements of the algorithms. The algorithms were agreed upon and the main feedback concerned the definitions and other thoughts. Other thoughts included how to analyze how many percent of leaked passwords could be placed within each class and how the algorithms could be optimized for that. This type of feedback did not concern the aim of this study but will be the topic for the discussion and future work section. Summarized feedback from each interview will be presented individually in the following sections.

5.2.1 Interview 1

The first interview was conducted with an employee from the security company Assured. The interviewee has worked with security in fields from embedded systems to cryptography and application security as well as web penetration testing.

Refinements

When asked about refinements the main remarks were directed to the leetspeak substitutions. It was pointed out that the number of possible leetspeak alternatives for each letter are different which should be taken into consideration. Some letters may have more than one leetspeak equivalent and some may have none. It was thus recommended to define a specific leetspeak alphabet before calculating the cost.

Regarding the passphrases the use of separators should be considered. If a user chooses the words "car" and "pet" without a separator the resulting passphrase would be "carpet" which is also a word. This automatically puts the security of the phrase into that of the class word. To avoid this the user may choose to use a separator. The most common separators are, according to the interview, spaces or dashes (-). It was recommended to include these when cracking phrases.

When asked about mnemonic passwords an immediate response was that mnemonic passwords seems dumb. If a user aims to remember a long phrase to just use the first letter of each word it would be better to use the phrase as a whole. Perhaps it could be useful if the system has an upper limit for password length. Further discussion led to the conclusion that testing for mnemonic passwords using a word list would generate a skewed cost; two words beginning with the letter “A” would generate the same mnemonic password as two other words beginning with the letter “A”.

Other thoughts

When asked if there is something else to consider it was brought up that the cost should be expressed in some other way than writing out the whole number of tries. For example, instead of expressing the cost of hundred tries as “100” it should be written as 10^2 or something suitable for the class.

5.2.2 Interview 2

Interview subject 2 is a lecturer in information technology, mostly within networking, algorithms, and software testing and development. The interviewee is also a doctoral student.

Refinements

When asked about refinements of the algorithms the subject was content with the algorithms and had nothing to address regarding those. They were deemed sufficient for this study’s aim. Furthermore it was confirmed that for most classes it may be better to use a dictionary attack instead of brute force.

Other thoughts

The subject questioned the usage of brute force in today’s password cracking culture. It is extremely time consuming and most passwords are cracked by utilizing lists of leaked passwords or by appending said lists with other commonly used passwords. Curse words are brought up as an example. It was also of interest to the subject to see how the distribution of leaked passwords is within these classes to get some statistics about how they are utilized.

5.2.3 Interview 3

The third interview was conducted with a person that has been working within the Swedish police at the “National Forensics Center” for over 6 years. The main area of expertise is cracking passwords with the goal to access encrypted data. The person has requested to stay anonymous and not be recorded.

Refinements

The interview starts by confirming the first three categories (Word, Word-in-Word and permutations). They seem reasonable and correct. For the class “Patterns” the definition should be clarified and it is pointed out that the “Shift” and “AltGr” variations of the keys should be used to include all characters. When the interview subject was asked if the “Mnemonic” passwords algorithm could be improved the answer was yes as several phrases would result in the same password it was deemed unnecessary to check all possible phrases from the dictionary and instead use brute force on the character set.

When discussing the class “phrases” it is pointed out that the main grammar characters that are used in the English language (“!”, “?”, “.” and “,”) should be included. Not because it is statistically proven that these are common but as they are a logical part of a phrase and hence a strategic way for the user to include special characters in the password.

Other thoughts

It is also brought up that phrases usually consist of 4 words and that most users tend to construct logical sentences. This is a big challenge as it is neither practical to test all words nor is there a really good way to only test logical or grammatically correct sentences. It was also of great interest to see how these strategies are utilized in Sweden by analyzing leaked passwords.

To better define the class "Patterns" it was suggested that a study be constructed where users are told to create a pattern based password to get an idea what type of pattern is most utilized. In the password community passwords are considered to be in their default state when written with lower case characters. Upper case, leetspeak, and other substitution techniques are considered permutations of an original password.

5.3 Refinements

The interviews were analyzed and to maintain the notion of "reflexivity" all proposed refinements were presented in the previous chapter and are reflected upon in this chapter. Though the interview subjects did not have any suggestions regarding improvement of the algorithms, which will stay the same, they did have a significant amount of opinions regarding the definitions of the strategies. As these opinions were clearly presented during the interviews and the interviews included a summary phase it was deemed unnecessary to conduct follow up interviews as the question regarding additional comments after refinements had already been answered during the interviews. After the initial analysis, the found refinements and other thoughts were analyzed again to categorize them to their respective classes. As presented in the "Methodology" chapter, the feedback that did not concern this study's specific aim will be the topic for discussion or future work section. The proposed refinements that concerned the aim and specific classes will be presented in this section.

5.3.1 Refinement of Phrases

The biggest changes will concern the class "Phrases" and therefore indirectly "Mnemonic". Firstly, the usage of separators will be included. It will be assumed that they are either used between all words in the phrase or not at all. This assumption will have no effect on the algorithm or its cost as the number of phrases that can be created will stay the same. However, the number of characters used will increase and thus making the brute force cost higher.

The definition of a phrase will also be changed to include the special characters ("!", "?", "." and ",") presented in chapter 5.2. From a grammatically point of view these may be categorized into two sections: phrase ending characters ("!", "?" and ".") and between word characters (","). This is how this study will refer to those characters. A phrase with special characters will now include one from each category.

For all phrase based classes the cost for up to 5 word long phrases will be calculated. This to include the cost for 4 word long phrases which was common. The applied changes to the class “Phrases” are summarized below in Figure 8.

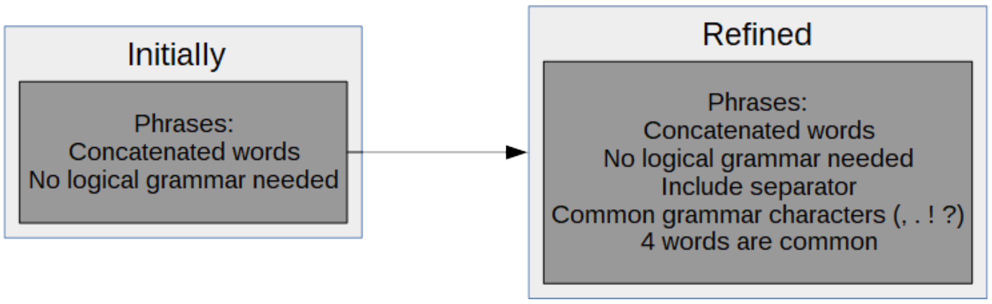


Figure 8. Summary of applied changes to the class phrases (author’s own)

5.3.2 Refinement of Mnemonic

The interviews suggested that the class “Mnemonic” should be cracked with brute force instead of the algorithm as it would generate a lower cost. However, to remain valid results this assumption can not be left unquestioned. The cost with the algorithm will still be calculated to validate the feedback and to present the differences between the algorithm and the brute force cost. As mnemonic passwords are based on phrases it would be interesting to include the special characters and separators of a phrase also for this class. The changes for mnemonic passwords are summarized and presented below in Figure 9.

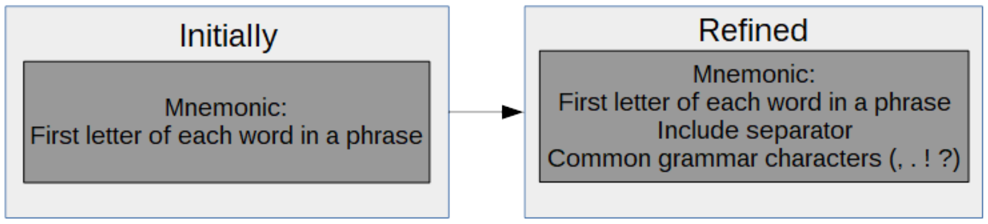

Initially
Mnemonic:
First letter of each word in a phrase
Refined
Mnemonic:
First letter of each word in a phrase
Include separator
Common grammar characters (, . ! ?)

Figure 9. Summary of applied changes to the class mnemonic (author’s own)

5.3.3 Refinement of LeetSpeak

According to Blashki and Nichol (2005) leetspeak is an ever changing language and in their example leetspeak alphabet almost all letters have only one leetspeak counterpart. This study will continue with the assumption that there is only one leetspeak equivalent for each letter.

This is to keep the study as general as possible and due to time constraints it is impractical to check all words and phrases to see how many characters have more than one leetspeak counterpart. Hence, no specific leetspeak alphabet will be defined.

5.3.4 Refinements of Patterns

Even though no changes are applied to this class from its initial definition, a clarification of the definition is necessary to avoid ambiguity. This study will define patterns as:

- Adjacent characters – not just the keys themselves but all the characters the keys represent (this implies the usage of the keys "Shift" and "AltGr")

Within the limitations of the:

- English keyboard layout
- 94 printable characters of the ASCII table.

5.3.5 General refinements

To make it more human comprehensible, all costs will be shortened to contain only the most significant numbers and presented with a base 10 logarithmic scale. As an example, the cost of 150 456 789 000 will be presented as $1,5 \times 10^{11}$.

As a final adjustment, passwords within all classes should, in their default state, be considered to be all in lower case. The resulting strategies for which the costs will be calculated are presented below in Figure 10. As the class "LeetSpeak Phrases" is also based on a regular phrase it was deemed interesting to the researcher to also include separator characters for this class as shown below in Figure 10.

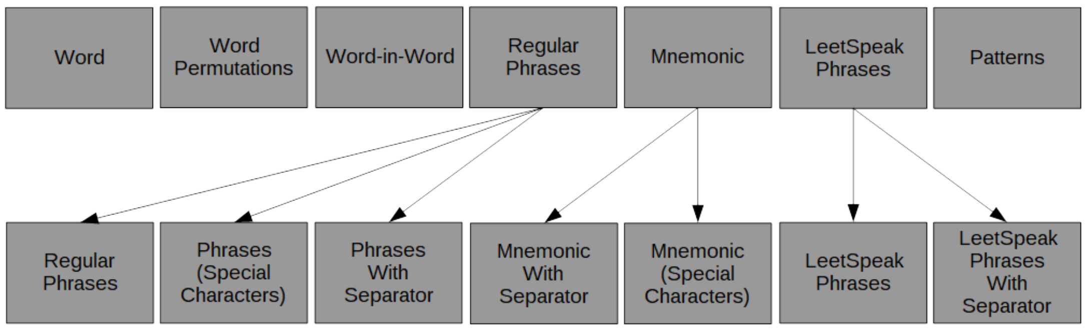


Figure 10. Classes after refinements (author's own)

5.4 Cost of the Classes

The costs of the classes are calculated based on the logic of the algorithms from the "Algorithms" chapter with potential changes from the "Refinements" chapter. Due to time constraints all possible unique passwords within each strategy could not be created and analyzed. Thus, a more general method for calculating the costs was needed. For this reason the average length of the words in the list was used together with the total amount of words (i.e. the list is used to get unbiased numbers for calculation).

Total number of words in the list is: 644 547

The average length of the words in the list is the total amount of characters divided by the number of words which results in:

$4\,396\,544 / 644\,547 = 9.423582190004437 \approx 9$

The costs will be based on the average word length of 9 characters. As an example, the strategy "Word-in-Word" will consist of 18 characters in total as the class was defined to include 2 words. The number of spaces in each word will therefore be 8. As all words can be put in 8 different places for all words in the list the calculation will be:

$644\,547 \times 8 \times 644\,547 = 3\,323\,526\,681\,672$

As shown in the above example, due to the large word list the costs (i.e. number of tries required to test all possible passwords) will be enormous. Converting the cost to the more readable form results in:

$3\,323\,526\,681\,672 = 3{,}3 \times 10^{12}$

When reading the results the main focus is the powers of 10. As in the calculation above, the 10^{12} will represent twelve zeros following the preceding two numbers (3,3). If two costs have the same amount of succeeding zeros the first two numbers can be used to differentiate them. As shown in Table 2 below, the cost for 9 character brute force has the same number of succeeding zeroes as the class Word-in-Word. However, the brute force cost's preceding numbers (5,4) are higher than that of the cost for Word-in-Word (3,3). Hence, the brute force cost is to be considered higher.

5.4.1 Overview

An overview of the different strategies are presented below in Figure 11. For each strategy a cost is presented together with an example password that could have been created with that strategy. For the word based strategies (word, word permutations, and word-in-word) the costs are the ones that are calculated in chapter 5.4.2. For the phrase based strategies, Figure 11 presents the costs for 4 word long phrases. The costs for patterns and random passwords are based on the length of 12 and 9 characters respectively. These lengths for the overview were chosen because they seem relevant in terms of usability and are thus assumed to be comparable. Costs for other passwords lengths are found in their respective subchapter. For each strategy the brute force equivalent cost is represented by the black line in the figure.

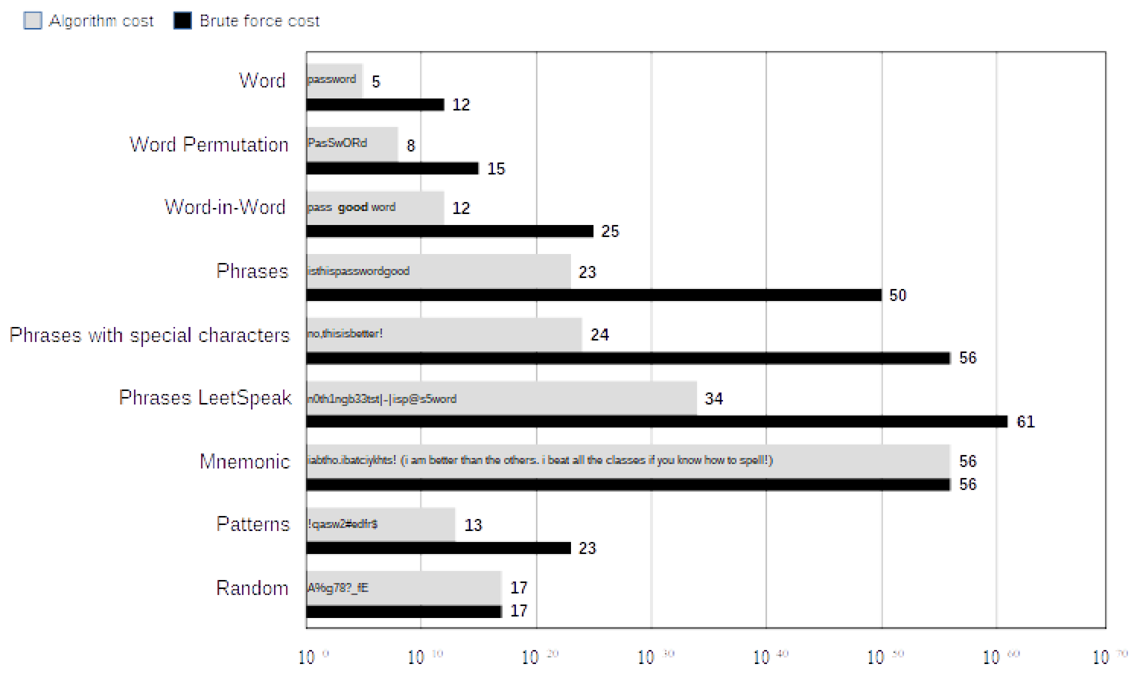


Figure 11. Overview of the costs (author's own)

5.4.2 Word Classes

The class "Word" has the cost of trying all the words in the list. Its brute force equivalent is based on the average word length of 9. For word permutations all words are multiplied with the possible combinations for 9 character long words. The brute force cost for "Word-in-Word" is based on the length of 18 as the class consists of 2 words (9 characters each on average). Table 2 below presents the cost of the word based classes.

Table 2. Costs for word based strategies

Class (*length*)	Cost	Brute Force ($\text{Character}_{\text{Set}}^{\text{length}}$)
Word (9 characters)	$6{,}4 \times 10^5$	$5{,}4 \times 10^{12}$ (26^9)
Word permutation (9 character)	$3{,}2 \times 10^8$	$2{,}7 \times 10^{15}$ (52^9)
Word-in-Word (18 characters)	$3{,}3 \times 10^{12}$	$2{,}9 \times 10^{25}$ (26^{18})

5.4.3 Regular Phrases

To show potential differences the separator character may provide the cost of regular phrases will be calculated both with and without separators. With the algorithm the cost will be the same with or without the separator. To compare it to the brute force cost which will cover both the inclusion and exclusion of the separator character the algorithm cost will have to be multiplied by 2.

As described earlier, the inclusion of a separator character will increase the number of characters used in the resulting password. This means that for the brute force cost not only will the character set increase by 1 but also the length that needs to be brute forced is also increased. To brute force a phrase with 2 words (18 characters) and one separator the total length will be 19 as there can be one separator between each word. Thus the calculation for brute force will be $Character_{set}^{NumberOfWords - 1 + Length}$. These calculations are the same for all classes where a separator is used.

The cost for regular phrases without the special grammar characters will be presented in Table 3 below.

Table 3. Costs for regular phrases of different lengths

Length of phrase	Cost	Brute Force (26^{length})	Brute Force with separator ($27^{NumberOfWords - 1 + length}$)
2 word phrases (18 characters)	$4,1 \times 10^{11}$	$2,9 \times 10^{25}$	$1,5 \times 10^{27}$
3 word phrases (27 characters)	$2,6 \times 10^{17}$	$1,6 \times 10^{38}$	$3,2 \times 10^{41}$
4 word phrases (36 characters)	$1,7 \times 10^{23}$	$8,6 \times 10^{50}$	$6,6 \times 10^{55}$
5 word phrases (45 characters)	$1,1 \times 10^{29}$	$4,7 \times 10^{63}$	$1,3 \times 10^{70}$

5.4.4 Phrases with Special Characters

As an interesting comparison, this study presents the costs for phrases and phrases with special characters individually. For phrases that include special characters the phrase length will be 9 × *numberofwords* + 2 as both the in-between character (“,”) and one of the phrase ending characters (“.”, “!”, “?”) will be included. For phrases that include special characters the cost will first be multiplied by the number of spaces in the phrase (number of words – 1) which represents the places where the comma sign (”,”) could be placed. This calculation will also include one of the phrase ending characters (”.”) as it was assumed earlier that a phrase can not be defined as a phrase with special characters without a phrase ending character.

The comma sign (“,”) will also be able to represent one of the other special characters if they were used in the middle of the phrase. For example “hey, how are you?” could be substituted by “hey! how are you?” and the cost will stay the same. In other words, the cost represents using one special character somewhere in the phrase and one at the end but not all possible combinations.

This cost is then multiplied by 3 to represent the other 2 possible phrase ending characters ("!", "?"). For the brute force cost the possible characters used will now be 30 (26 lower case letter and the 4 special characters). As an example, for the 2 word phrases the formula will be 30^{20}. Table 4 presents the cost for special character phrases.

Table 4. Costs for special character phrases of different lengths

Length of phrase	Cost	Brute Force (30^{length})	Brute Force with separator ($31^{NumberOfWords - 1 + length}$)
2 word phrases (20 characters)	$1,2 \times 10^{12}$	$3,4 \times 10^{29}$	$2,0 \times 10^{31}$
3 word phrases (29 characters)	$1,6 \times 10^{18}$	$6,8 \times 10^{42}$	$1,7 \times 10^{46}$
4 word phrases (38 characters)	$1,5 \times 10^{24}$	$1,3 \times 10^{56}$	$4,5 \times 10^{59}$
5 word phrases (47 characters)	$1,3 \times 10^{30}$	$2,6 \times 10^{69}$	$1,1 \times 10^{76}$

5.4.5 LeetSpeak Phrases

To calculate leetspeak phrases, the cost of regular phrases is multiplied with the possible permutation combinations for the specific phrase length. As mentioned, this also includes the substitution from lower case letters to upper case letters. Table 5 presents the cost for leetspeak phrases.

Table 5. Costs for phrases of different lengths with character substitution

Length of phrase	Cost	Brute Force (52^{length})	Brute Force with separator ($53^{NumberOfWords - 1 + length}$)
2 word phrases (18 characters)	$1,0 \times 10^{17}$	$7,7 \times 10^{30}$	$5,7 \times 10^{32}$
3 word phrases (27 characters)	$3,5 \times 10^{25}$	$2,1 \times 10^{46}$	$1,0\times 10^{50}$

Length of phrase	Cost	Brute Force (52^{length})	Brute Force with separator ($53^{NumberOfWords - 1 + length}$)
4 word phrases (36 characters)	$1,1 \times 10^{34}$	$5,9 \times 10^{61}$	$1,7 \times 10^{67}$
5 word phrases (45 characters)	$3,9 \times 10^{42}$	$1,6 \times 10^{77}$	$3,0 \times 10^{84}$

5.4.6 Mnemonic

Table 6 below presents the cost for mnemonic passwords. The costs are first calculated according to the algorithm which generates:

$$Cost = \text{Wordlist}_{\text{Length}}{}^{\text{NumberOfWords}}$$

As the mnemonic passwords now include special characters the mnemonic costs are then multiplied in the same way as presented for the special character phrases. The "Length of password" includes the number of words used and 2 special characters.

Table 6. Costs for mnemonic passwords of different lengths

Length of password	Cost	Brute Force (30^{length})	Brute Force with separator ($31^{NumberOfWords - 1 + length}$)
7 (5 words)	$1,3 \times 10^{30}$	$2,1 \times 10^{10}$	$2,5 \times 10^{16}$
8 (6 words)	$1,0 \times 10^{36}$	$6,5 \times 10^{11}$	$2,4 \times 10^{19}$
9 (7 words)	$8,3 \times 10^{41}$	$1,9 \times 10^{13}$	$2,3 \times 10^{22}$
...	…	...	...
20 (18 words)	$1,8 \times 10^{106}$	$3,4 \times 10^{29}$	$1,5 \times 10^{55}$
29 (27 words)	$5,5 \times 10^{158}$	$6,8 \times 10^{42}$	$1,0 \times 10^{82}$
38 (36 words)	$1,4 \times 10^{211}$	$1,3 \times 10^{56}$	$7,4 \times 10^{108}$
47 (45 words)	$3,4 \times 10^{263}$	$2,6 \times 10^{69}$	$5,1 \times 10^{135}$

5.4.7 Patterns

The process of calculating the cost, following the designed algorithm to create all possible patterns of adjacent characters, is not possible within the time limits of this project. Therefore the findings of Chou, Lee, Hsueh, and Lai (2012) will be used. They utilize the same principle for calculating the cost and include the 94 printable characters of the ASCII table. Table 7 below presents the cost of adjacent patterns in relation to their brute force equivalent.

Table 7. Costs for patterns of different lengths

Length	Cost	Brute Force (94^{length})
9	$4{,}1 \times 10^{10}$	$5{,}7 \times 10^{17}$
10	$4{,}7 \times 10^{11}$	$5{,}3 \times 10^{19}$
11	$5{,}8 \times 10^{12}$	$5{,}0 \times 10^{21}$
12	$7{,}0 \times 10^{13}$	$4{,}7 \times 10^{23}$
13	$8{,}5 \times 10^{14}$	$4{,}4 \times 10^{25}$
14	$1{,}0 \times 10^{16}$	$4{,}2 \times 10^{27}$
15	$1{,}2 \times 10^{17}$	$3{,}9 \times 10^{29}$
16	$1{,}5 \times 10^{18}$	$3{,}7 \times 10^{31}$

5.4.8 Alphanumeric and Special Characters

As described earlier, this study will categorize passwords not created by the above mentioned strategies as random passwords. The cost for random passwords are $character_{set}^{Length}$. Costs for different character sets are presented below in table 8.

Table 8. Brute force costs for different lengths and character sets

Length	Character set				
	0 – 9	a – z (or A – Z)	Aa – Zz	Aa0 – Zz9	ASCII (94)
7	$1{,}0 \times 10^{7}$	$8{,}0 \times 10^{9}$	$1{,}0 \times 10^{12}$	$3{,}5 \times 10^{12}$	$6{,}4 \times 10^{13}$
9	$1{,}0 \times 10^{9}$	$5{,}4 \times 10^{12}$	$2{,}7 \times 10^{15}$	$1{,}3 \times 10^{16}$	$5{,}7 \times 10^{17}$
16	$1{,}0 \times 10^{16}$	$4{,}3 \times 10^{22}$	$2{,}8 \times 10^{27}$	$4{,}7 \times 10^{28}$	$3{,}7 \times 10^{31}$
18	$1{,}0 \times 10^{18}$	$2{,}9 \times 10^{25}$	$7{,}7 \times 10^{30}$	$1{,}8 \times 10^{32}$	$3{,}2 \times 10^{35}$
29	$1{,}0 \times 10^{29}$	$1{,}0 \times 10^{41}$	$5{,}8 \times 10^{49}$	$9{,}5 \times 10^{51}$	$1{,}6 \times 10^{57}$

6 Conclusions

The results distinctly present a cost for the classes words, permutated words, and word-in-word that does not come near the other classes. This does not mean that they may not be usable. If for example an attacker was able to test 1000 billion passwords a day, a password generated by the strategy word-in-word would still take approximately 3,3 days to crack which might be more time than the attacker wants to spend on cracking one password. It would of course depend on the attacker's interest in succeeding. These classes do however fall into the lower cost category.

On the higher end the phrase based passwords are found. Due to the fact that only the first letter of each word that constitutes a phrase is used to make up a mnemonic password a very long phrase has to be made for cracking this type of password using the algorithm. The results present that it is extremely impractical to test all mnemonic passwords by generating all possible phrases from the words in the list. Hence, it is evident that for "Mnemonic" passwords the algorithm is too costly and that the best cracking solution is brute force. This also validates the feedback from the interviews.

Furthermore, the class "Mnemonic" is the only class whose algorithm generates a higher cost than its brute force equivalent which makes it the hardest class to crack. At least within the limitations of this study. Keep in mind that the brute force attack on "Mnemonic" passwords will always succeed but is independent of the list. Meanwhile, the other classes' costs rely on that the words are included in the list. If the words used in a leetspeak passphrase are not included in the list (or are misspelled) this class's cost converts to that of its brute force equivalent.

With this in mind, it would not be fair to neglect the fact that the brute force cost for leetspeak phrases is higher than that of the Mnemonic passwords and the brute force cost for phrases with special characters are the same as Mnemonic. This leads to the conclusions that if regular words from a dictionary, and standard grammar characters, are used to make up the phrase, the phrase should be made Mnemonic to make it harder to crack. Conversely, if made-up words, biographical words, cartoon characters or other words, that are believed not to be included in a word list, are used, the passphrase should not be made Mnemonic; Instead the phrase should include character substitution techniques (e.g. leetspeak) to make it harder to crack.

For example, if the phrase "best, password ever!" is used to create an 18 character long passphrase it would instead be better to create a 16 word long phrase (including 2 special characters) and turn it mnemonic.

If one were to utilize a non-existing word as a strategy, a phrase like "my dog krillex is cool" should be turned into "myDoGkriLLExiscooL" which would be harder to crack than an 18 character long mnemonic password which was based on a phrase containing the word "krillex".

7 Discussion

This study has tried to compare the costs of cracking passwords generated by different strategies. From the start it was apparent that the results would be heavily dependent on the definitions of the classes. Interviews with domain experts were made as an effort to avoid researcher bias and find common agreed upon definitions for the classes. Even though the study analyzed the feedback critically and willingly changed the definitions the factor of dependency remains. The results may still vary if the definitions were to be different

The goal was to compare the classes in their most general form and according to the interviews this may have been achieved. However, not everyone might agree with the definitions. As an example, this study defined the class “Word” as just a plain lower case word from a dictionary. Some might argue that many systems would not allow this as a password and that the general definition should include for example starting with an upper case letter and ending with a special character like for example “Bumblebee!”. Even with a definition like the one just mentioned the new cost would not change the order of the classes. Furthermore, as this was not a part of the classification model this study aimed to follow, nor was it brought up in the interviews this study remains confident that the most general form was defined.

To compare the classes the costs had to be calculated in a mutual way that made the classes comparable. As all classes (except for patterns) are based on words it made sense to use the average length of the words in the word list to calculate the costs. Even though one rarely uses nine character words the classes will still be comparable as long as they are based on the same foundation. Remember that it is not the cost per se that is important but the difference between them. The word list used is merely a way to get an unbiased estimation of the number of words that could be used and their average length. This will somewhat represent a real world cost if the user were to only make use of the English language as described in the “Limitations” chapter.

The issue of comparison is evident in the class “Mnemonic” as it would seem impractical from an attacker perspective to construct all possible phrases consisting of 45 words. Also, the agony of remembering a 45 word long phrase is clearly a usability issue. This might question the comparison between “Mnemonic” and “Phrases” as the class “Phrases” only need to consist of 5 words to make up 45 characters.

It was found that some of the interview subjects had a little trouble understanding what the study was about. Perhaps this was due to ambiguity in the material or that the material was not sent out in proper time. Nevertheless the interviews eventually provided the information needed. It just took a little longer than expected. A positive outcome from this was that the interviews were a great source of information on what the interviewees would like to see in future research on the subject of passwords.

It should also be noted that the results present costs for the strategies in their most basic and general form. If the strategy used happens to create a commonly used password it is likely that the password will be in an attackers list of commonly used passwords and hence bypass the complexity of the strategy. Similarly a “Mnemonic” password may result in a pattern of adjacent characters neglecting the costs for the chosen strategy. In conclusion, even though a strategy may seem more secure the cost will be dependent on both the strategy used and the resulting password.

7.1 Ethics

The most relevant ethical aspects to consider have been how the study was conducted and the implications the result may have. The interview subjects were chosen by asking seemingly relevant people in the business. As described in the “Methodology” section, it was important from the start to have diversity among the interview subjects and thus interview subjects were intentionally selected based on different professions and backgrounds.

This study emphasized the importance of reflexivity (transparency) to avoid researcher bias consequently the procedures of the study and interviews were presented in written form. All interview subjects were also given the content and procedure of the interview in written form to be able to prepare themselves beforehand. No distinction regarding the interview subject’s background were made when creating this material and all participants were given the same material at the same time.

Before each interview the interview subjects were asked whether they wanted to stay anonymous or not and also if recording of the interview was permitted. The decisions were accepted without questioning to give the interview subject a secure environment in which it could speak freely. Even though one agreed to have its name published all names were left out as it was later deemed to be irrelevant to the study. The interviewees’ backgrounds were more relevant for the aim of the study and were included.

Even though the aim of this study was to find out which semantic password creation strategy provided the highest cost to crack, the results also show, in order, which classes that has the lowest cost to crack. This may benefit an attacker as he could start by trying the strategies with the lowest cost and thus save a lot of time. This is an unavoidable fact and hopefully this study have already made users choose a password derived from the class(es) with the higher cost to crack.

As the results may help protect users and companies from malicious attackers there is also another backside; The results may be used by people performing other illegal activities. If someone were to use encryption to cover their illegal activities they could make use of the results from this study, create harder to crack passwords, and consequently make the investigation harder for the IT forensics analysts. It is hard to conduct this type of study without providing results that may benefit both legal and illegal activities. There are however already many publicly available sources on how to create strong passwords.

7.2 Societal and Scientific Contribution

The main contribution is that the results may educate individual users inspiring them to utilize a strategy that has a higher cost to crack when creating passwords. This hopefully leads to accounts that are less susceptible to malicious intrusion. Together with other research on how memorable these strategies may be, this study also hopes to contribute to the refinement of password policies for systems and organizations which may benefit the confidentiality for the company as well as the end user.

The scientific contribution is an initial comparison between some of the most commonly used strategies. If other studies were to compare different strategies by using other definitions or alternations these results may be used to see how the results differ. As described earlier, this study also contributes to a larger research project where the memorability of the these strategies are examined.

7.3 Future Work

This study tried to compare the classes in their most general form. Future research on this topic could be to calculate the costs for other definitions and limitations and thus show how different factors affect the costs or validating the results from this study. As suggested by the interviewees, a study on how users tend to define the notion of a pattern is of interest to get a more universal definition of what a pattern is. Future research could also include to analyze lists of leaked passwords to see how the strategies are utilized. The interviews provided an explicit need to do this within a local context.

References

Arends, R., Deussen, R., Green, B., Rush, J., Mache, J. & Weiss, R. (2018). Get a Clue: A Hands-On Exercise for Password Cracking. In *Proceedings of the 2018 International Conference on Security & Management*, 2018, 117 – 121. ISBN: 1-60132-488-X

Balagani, K.S., Conti, M., Gasti, P., Georgiev, M., Gurtler, T., Lain, D., … Wu, L. (2018). In J. Lopez et al. (Eds.), *Computer Security.* (pp. 263–280). Springer Verlag. DOI: 10.1007/978-3-319-99073-6

Berndtsson, M., Hansson, J., Olsson, B., & Lundell, B. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems, 2nd ed*. London: Springer Verlag. ISBN: 978-1-84800-008-7

Blashki, K. & Nichol, S. (2005). Game geek's goss: linguistic creativity in young males within an online university forum. In *Australian Journal of Emerging Technologies and Society*, Vol 3, 77–86. ISSN: 1449-0706

Choong, Y.Y. & Theofanos, M. (2015). What 4,500+ People Can Tell You – Employees' Attitudes Toward Organizational Password Policy Do Matter. In T. Tryfonas & I. Askoxylakis (Eds.), *Human Aspects of Information Security, Privacy, and Trust* (pp. 299–310). Switzerland: Springer. DOI: 10.1007/978-3-319-20376-8

Chou, H.C., Lee, H.C., Hsueh, C.W., & Lai, F.P. (2012). Password Cracking Based on Special Keyboard Patterns. In *International Journal of Innovative Computing, Information and Control*, Vol 8, 387 – 402.

Florencio, D. & Herley, C. (2007). A Large-Scale Study of Web Password Habbits. In *Proceedings of the 16th International Conference on World Wide Web,* 657–666. DOI: 10.1145/1242572.1242661

Johnson, R.B. & Christensen, L. (2014). *Educational research: Quantitative, qualitative, and mixed approaches* (5th ed.). California: SAGE. ISBN: 978-1-4522-4440-2

Komanduri, S., Shay, R., Gage, P., Mazurek. M. L., Bauer, L., Christin, N., Cranor, L. F., & Egelman, S. (2011). Of Passwords and People: Measuring the Effect of Password-Composition Policies. *In Proceedings of the Human Factors and Computing Systems*, *2595*–2604. ACM.

Kuo, C., Romanosky, S., & Cranor, L. (2006). Human Selection of Mnemonc Phrase-based Passwords. In *Proceedings of the Second Symposium on Usable Privacy and Security* Vol 149, 67-78

Kävrestad, J. (2018). *Fundamentals of Digital Forensics – Theory, Methods and Real-life application*. Switzerland: Springer Verlag. ISBN: 978-3-319-96318-1

Kävrestad, J., Eriksson, F. & Nohlberg, M. (2018). The Development of a Password Classifiation Model. *Journal of Information System Security*, Vol 14, 31-46

Nielsen, G., Vedel, M. & Jensen, C. D. (2014). Improving Usability of Passphrase Authentication. *2014 Twelfth Annual Conference on Privacy, Security and Trust (PST),* IEEE, Toronto, Canada. (2014). ISBN: 978-1-4799-3503-1/14

Peckel, E.W. (1999). Permutations and Combinations. In K. H. Rosen (Ed.), *Handbook of Discrete and Combinatorial Mathematics*. (pp. 136–140). CRC Press. ISBN: 978-0849301490

Pfleeger, C.P., Pfleeger, S.L. & Margulies, J. (2015). *Security in Computing* (5th ed.). New Jersey: Prentice Hall. ISBN: 978-0-13-408504-3

Stavrou, E. (2017, June). A situation-aware user interface to assess users' ability to construct strong passwords: A conceptual architecture. *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment*. DOI: 10.1109/CyberSA.2017.8073385

Stringer, E.T. (2014). *Action Research* (4th ed.). California: SAGE Publications. ISBN: 978-1-4522-0508-3.

Thakur, T., & Verma, R. (2014). Catching Classical and Hijack-Based Phishing Attacks. In A. Prakash & R. Shyamasundar (Eds.), *Information Systems Security.* (pp. 318–337). Springer Publishing. DOI: 10.1007/978-3-319-13841-1

Ur, B., Bees, J., Segreti, S.M., Bauer, L., Christin, N., & Cranor, L.F. (2016, May). Do Users' Perceptions of Password Security Match Reality?. *CHI*. DOI: 10.1145/2858036.2858546

Wier, M., Aggarwal, S., de Medeiros, B. & Glodek, B. (2009). Password Cracking Using Probabilistic Context-Free Grammars. In *30th IEEE Symposium on Security and Privacy*, 391 – 405. DOI: 10.1109/SP.2009.8

Wier, M., Aggarwal, S., Collins, M. & Stern, H. (2014). Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. *17th ACM conference on Computer and communications security*, Chicago, 2010, October 4–8. DOI: 10.1145/1866307.1866327

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin: Springer. ISBN: 978-3-642-29043-5

Yang, W., Li, N., Chowdhury, O., Xiong, A., & Proctor, R.W. (2016). An Empirical Study of Mnemonic Sentence-based Password Generation Strategies. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 1216–1229. DOI: 10.1145/2976749.2978346

Yin, R.K. (2011). *Qualitative Research from Start to Finish*. New York: Guilford Publications. ISBN: 978-1-60623-977-3

Yıldırım, M. & Mackie, I. (2019). Encouraging users to improve password security and memorability. *International Journal of Information Security, 18,* 1–19. DOI: 10.1007/s10207-019-00429-y

Appendix A – Interview Material (English translation)

Exploring the Cost Related to Cracking Passwords Generated with Different Strategies
Interview material

Thank you for participating in my interview study. This document includes information and the material to be discussed. I need your help to improve/validate the algorithms and/or the definitions of the classes to later be able to calculate and compare the costs using a given foundation (word list, character set etc.). During the interview I will ask questions but the main focus will be on your thought regarding my definitions and algorithms. Hence everything in this document is open for feedback.

Marcus Birath

Definitions

Class – Strategy for creating passwords

Algorithm – Pseudoalgoritm/semantic approach

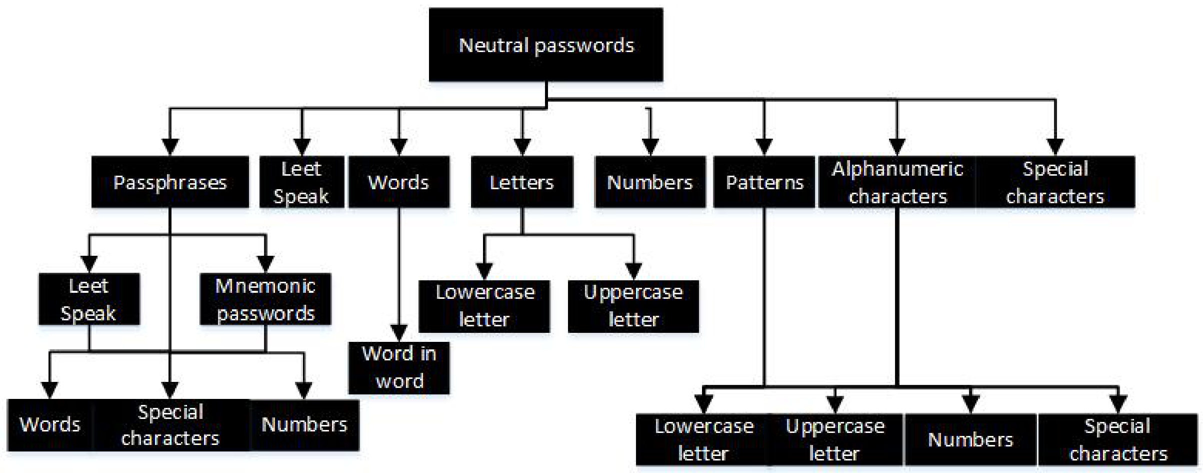
Cost – Number of tries

Definitions – How I'm interpreting the classes (within the study's limitations)

Purpose

Users may utilize different strategies to create passwords. In my study I will explore the cost for cracking passwords generated by these strategies. The goal is to find algorithms that, with a given foundation (word list), can generate a lower cost for cracking than a brute force attack. If no such algorithm can be created the class will be considered ”secure” as the cost will be brute force. If an algorithm can be created that generates a lower cost these classes will be considered less secure and ranked accordingly. Together with other research, regarding the memorability of the classes, this will hopefully serve as a foundation for what strategy is more secure and easy to remember.

The strategies I will use is shown in the figure below (from Joakim Kävrestad's research).



Limitations

To keep the study from blowing out of proportions I have to make some limitations. When I calculate costs, defines classes and construct algorithms I will stick to the following:

- English dictionary
- English alphabet (Aa – Zz och 0 – 9)
- The ASCII Table's 94 printable character (num. 33 – 126)

Interview

The interview is expected to be 30 – 60 minutes long and is divided into three parts.

1. You may freely adress what you think is important, what I have missed, disagreements etc.

2. I fill in with questions regarding the definitions and algorithms if needed.

3. I make a short summary of the feedback provided to avoid misunderstandings.

The interview encourages that you bring up what you think is important.

Algorithms

This section defines the classes and a pseudo algorithm is presented for how one could crack a password generated by the respective class. To make the classes more comparable they have to be defined as generally as possible. To define the class ”word-in-word” as 5 words twined together will not be considered to be as general as it could be as it would give the class a much different cost than if it consisted of 2 words which would be the most general.

Word

This class consists of words that can be found in dictionaries and the words are not altered in any way. Kävrestad et al. (2018) distinguish between regular words and biographic words which are words that relate to the users interests (e.g. family, hobbies, fictional characters etc.) many of which may also be found in regular dictionaries. Even if the word list used for cracking needs to be updated to include some specific biographical words the algorithm will stay the same.

Algorithm

```
for each word in list(
 test if word = password
)
```

Cost = Wordlist length

Word-in-Word

As the name implies, this class is confined to consist of 2 words in total where each word in the word list, including the current word, can be placed in its whole between all letters in the current word. The words “car” and “**hug**” could result in “c**hug**ar”. This class could also be defined as braiding two words together by having every other letter in the password coming from word 2. Taking for example the words “car” and “**hug**” will result in “c**h**a**u**r**g**”. This would however yield the same cost as there are as many spaces to start putting in word2 as there are writing it in its whole. The cost may represent either definitions but not both.

Algorithm:

```
for each word1 in list(
 for each word2 in list
  place word2 in word1 space X
)
```

As the number of spaces in a word always is one less than the length of the word the cost for each word will be $\text{Word}_{\text{length}}$ -1× $\text{List}_{\text{length}}$

Word Permutations

To use leetspeak without any logic would be just a random password and can not be threated as separate class. It has to be used in conjunction with other classes. Permutations can be applied to all classes where each character may be substituted with their corresponding leetspeak character alternatively lower case letters are substituted with upper case letters or vice versa. In a two letter word the possible permutations are to change the first or the second character, or both characters. This makes the total possible combinations are 3. The table below shows the number of the possible permutations for different word lengths.

Word length (L)	Possible permutations	Number of words of length (L) in list	Cost	Brute force keyspace equivalent
1	1	X	X × 1	52
2	3	X	X × 3	2704
3	7	X	X × 7	140 608
4	15	X	X × 15	7 311 616
5	31	X	X × 31	380 204 032
6	63	X	X × 63	19 770 609 664
7	127	X	X × 127	1 028 071 702 528
8	255	X	X × 255	53 459 728 532 456
9	511	X	X × 511	2 779 905 883 635 715
10	1023	X	X × 1023	144 555 105 949 057 024

To combine substitution with the class “Word” the following algorithm could be used.

Algorithm:

```
for each word in list(
 test all substitution combinations
)
```

Cost for length (X) = Number of words of length X × possible combinations for length X

Passphrases

A passphrase is defined as words concatenated with other words regardless of meaning. The algorithm will test all possible combinations of the words in the given word list.

Algorithm:

```
for each word in list(
 for each word X in list
  append word with word X
)
```

Cost = WordlistLength^X where X is the number of words making up the passphrase.

LeetSpeak Phrases

Leet speak phrases can be defined in different ways. The first is to have the whole phrase as leet speak which can easily be done by converting the given word list to leet speak and proceed with the algorithm as done with regular phrases. This will yield the same cost as regular phrases. However, this study will focus on the possibility that any character combination of the phrase may be substituted with its leetspeak equivalent and therefore leetspeak phrases will be defined as phrases (from “phrases”) with permutations (from “Word permutation”). This way the base phrase algorithm will be the same as for regular phrases. However, after the phrase is created all alternation combinations have to be tested. This algorithm implies to test all alternations of a base phrase before proceeding to the next possible base phrase.

Algorithm:

```
for each word in list(
 for each word X in list
  append word with word X
  test all permutation combinations
)
```

Cost for phrase length (X) = Number of phrases of length X × possible combinations for length X

As with substitution of words, this class includes substitution to both leetspeak and lower case to upper case.

Mnemonic

This study will define mnemonic passwords as using the first letter from each word in a phrase. As the user does not need to incorporate a logical structure of the phrases nor is there an easy way to check it before converting it to a mnemonic password the most general way is to create phrases based on the same algorithm as for regular phrases and make them mnemonic. The algorithm would need one nested iteration for each word that should be part of the phrase. The algorithm below is shortened for easier overview.

Algorithm:

```
for each word in list(
 for each word X in list
  append word with word X
  make it mnemonic
)
```

This algorithm would imply the same cost as for regular phrases which is $\mathit{WordList}_{\text{length}}^{x}$ but as the phrases are converted to mnemonic passwords the length of the phrases are no longer of importance and thus a brute force attack with all possible characters will result in a lower cost than this algorithm. To crack a 7 character long mnemonic password with this algorithm a 7 word phrase would have to be created which, even with a small word list of 1000 words, would result in a cost of 1000^7 which is much higher than the brute force equivalent of 94^7 using the 94 common characters of ASCII table.

Patterns

In a study by Kävrestad the class pattern is restricted and defined as adjacent characters within a given keyboard layout. This study will also share this definition and state that for the purpose of this study the class "Pattern" will be restricted to only adjacent characters where the 94 printable characters of the ASCII table will be included.

Algorithm:

```
for each char X in set(
 foreach adjacentchar Y(
  append char X with adjacentchar Y
  foreach adjacentchar Z(
   append char XY with adjacentchar Z
  )
 )
)
```

This algorithm implies to use the same start character for all possible combinations of length X before proceeding to the next starting character.

I think this is just a small part of what a pattern could be. What a patterns could be depends on the user. Thus this algorithm will only be applicable for patterns with the definition ”adjacent characters”.

Alphanumeric and Special Characters

Alphanumeric characters are, in this study, defined as numbers, lower case letters and upper case letters. Special characters are those found in the ASCII table that do not fall into the alphanumeric definition. Passwords created with these characters that do not utilize some of the above mentioned strategies will be seen by this study as random passwords. This study states that using these characters without utilizing some of the above mentioned strategies would result in a random password and hence no specific algorithm will be created that generates a cost lower than brute force. The cost for random passwords will therefore be $character_{set}^{Length}$.

Interview questions

This study is based on semi-constructed interviews where focus lies on what the interviewee wants to bring up. Hence there will probably be specific follow-up questions for each interview but following are some general questions:

- What do you work with? (What is your background?)
- What have you found that you want to comment?
- Do you miss any classes?
- Do you want to change any definitions? Why?
- Do you want to change any algorithms? Why?
- Do you want to change anything else?
- The definition of the class ”patterns” only constitutes a part of what a pattern could be. How would you define the class ”Patterns”?
- Do you think that a class whose cost is lower than brute force could be considered secure?

Appendix B – Interview Material (Swedish original)

Exploring the Cost Related to Cracking Passwords Generated with Different Strategies Intervjumaterial

Tack för att du vill medverka i min intervjustudie. Nedan följer information samt det material som kommer diskuteras. Jag behöver din hjälp för att förbättra/validera algoritmerna eller definitionerna av klasserna för att jag sen ska kunna beräkna och jämföra kostnader utifrån en given grund (ordlista, teckentabell etc.). Under intervjun kommer jag ställa frågor men stort fokus kommer ligga på dina tankar kring mina definitioner av klasserna samt algoritmerna och således är allt i detta dokument öppet för feedback.

Marcus Birath

Definitions

Klass – Strategi för att skapa ett lösenord

Algoritm – Pseudoalgoritm/semantiskt tillvägagångssätt

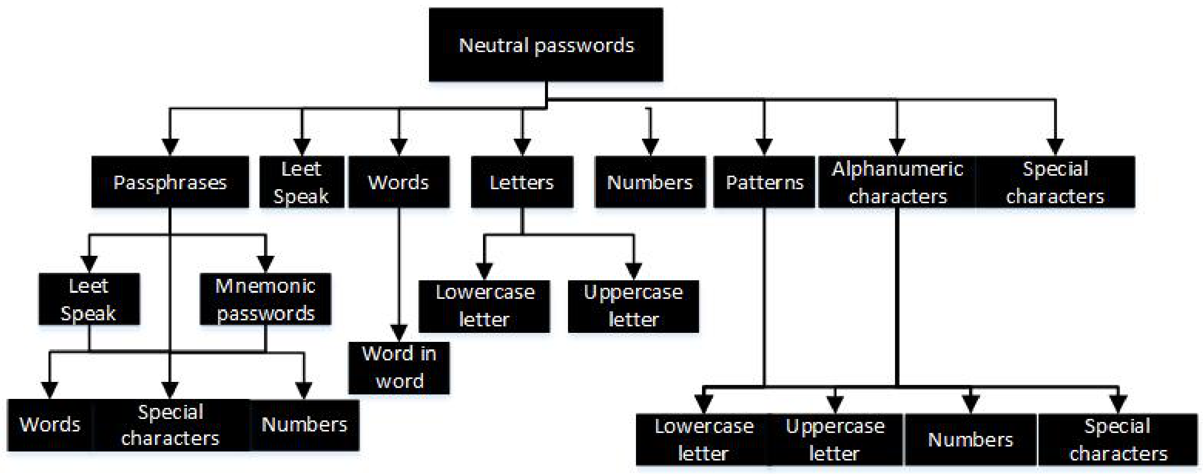
Kostnad – Antal försök

Definitioner – Hur jag tolkar klasserna (inom mina begränsningar)

Syfte

Användare kan använda sig av olika strategier för att skapa lösenord. Jag ska i mitt arbete utforska kostnaden för att knäcka lösenord som är skapade med dessa strategier. Målet är att försöka hitta algoritmer för varje klass som, med hjälp av en given bas (ordlista etc.) kan generera en lägre kostnad än en ”brute force” attack. Kan ingen sådan algoritm skapas så anses klassen vara ”säker” då kostnaden blir ”brute force”. Om en algoritm kan skapas som genererar en lägre kostnad kommer dessa klasser istället anses vara mindre säkra och rangordnas därefter. Tillsammans med annan forskning, gällande hur lätta de olika klasserna är att komma ihåg för användare, är detta tänkt att utgöra underlag för vilken strategi som säkrast och lättast att komma ihåg.

Strategierna jag utgår från visas i nedanstående figur (från Joakim Kävrestads forskning)



Begränsningar

För att arbetet inte ska skena iväg har jag behövt sätta vissa begränsningar. När jag räknar på kostnader och definierar klasser och algoritmer kommer jag förhålla mig till:

- Engelsk ordbok
- Engelska alfabetet (Aa – Zz och 0 – 9)
- ASCII teckentabell 94 tecken (nr 33 – 126)

Intervju

Intervjun är tänkt att vara ca 30 – 60 minuter lång och indelad i tre delar.

1. Du får fritt ge feedback på vad du tycker är viktigt, vad jag missat, vad vi inte är överens om etc.

2. Jag fyller i med frågor gällande t.ex. definitionerna och algoritmerna om det behövs.

3. Jag sammanfattar kort den eventuella feedbacken jag fått för att undvika missförstånd.

Intervjun är öppen för att du trycker på det du anser vara mest viktigt.

Algorithms

Nedan definieras klasserna och en pseudoalgoritm presenteras för hur man skulle gå tillväga för att knäcka motsvarande lösenord. För att göra klasserna jämförbara måste alla klasser definieras med utgångspunkt att de ska vara så generella som möjligt. Att t.ex. definiera klassen ”word-in-word” som 5 ord nästlade i varandra anses inte generell då det skulle ge klassen en helt annan kostnad än om den bestod av 2 ord vilket anses vara det enklast möjliga.

Word

Klassen ”Word” består av ord som återfinns i en ordbok. Tanken är att användaren valt ett ord från sin ordlista och använder detta i sin helhet. Kävrestad skiljer på vanliga och biografiska ord, vilket kan vara ord som relaterar till användarens intressen (hobby, seriefigurer etc.) men som kanske inte finns i en vanlig ordbok. Denna studie anser att även om dessa ord inkluderas så förblir algoritmen densamma.

Algoritm

```
for each word in list(
 test if word = password
)
```

Kostnad = Ordlistans längd

Word-in-Word

Som nämnt ovan anses denna klass bestå av 2 ord där man placerar det ena ordet (i sin helhet) mellan några av bokstäverna i det andra ordet. Om man t.ex. använder sig av ordet ”bil” och ”apa” kan det resultera i ”b*apa*il”. Man kan alltså placera ett ord i något av alla mellanrum och antalet mellanrum för varje ord blir alltid ett mindre än ordets längd.

Algoritm:

```
for each word1 in list(
 for each word2 in list
  place word2 in word1 space X
)
```

Kostnad för varje ord = $\text{Word}_{\text{length}} - 1 \times \text{List}_{\text{length}}$

Denna klass skulle också kunna definieras som att man ”flätar samman” 2 ord genom att varannan bokstav i lösenordet är en bokstav från ord 2. T.ex. ordet ”bil” och ”apa” blir b*a*i*p*l*a*. Detta skulle dock generera samma kostnad då det finns lika många ställen att börja placera in ord2 på som det finns att skriva ord2 i sin helhet. Så oavsett strategin anser denna studie att kostnaden blir densamma så länge man utgår från endast en av definitionerna.

Word Permutations

Att använda LeetSpeak utan någon logik anses endast vara ett slumpmässigt lösenord och kan således inte behandlas som en egen klass utan behöver istället kombineras med andra klasser och ses som substitution av bokstäver. Substitution kan användas för alla klasser där varje tecken kan bytas ut till motsvarande LeetSpeak-tecken eller små bokstäver byts ut till stora eller tvärtom. I ett ord med 2 tecken kan man antingen byta ut det första, det andra eller båda. Detta ger 3 unika kombinationer. Jag har tagit fram hur många unika kombinationer man kan göra för en given ordlängd som visas och jämförs med motsvarande kostnad för ”brute force” i tabellen nedan.

Word length (L)	Possible permutations	Number of words of length (L) in list	Cost	Brute force keyspace equivalent
1	1	X	X × 1	52
2	3	X	X × 3	2704
3	7	X	X × 7	140 608
4	15	X	X × 15	7 311 616
5	31	X	X × 31	380 204 032
6	63	X	X × 63	19 770 609 664
7	127	X	X × 127	1 028 071 702 528
8	255	X	X × 255	53 459 728 532 456
9	511	X	X × 511	2 779 905 883 635 715
10	1023	X	X × 1023	144 555 105 949 057 024

Att kombinera substitution med klassen ”Word” skulle följande algoritm kunna användas:

Algoritm:

```
for each word in list(
 test all substitution combinations
)
```

Kostnad för ordlängd (X) = Antalet ord av längden X × Möjliga kombinationer för längd X

Passphrases

En lösenordsfras definieras som ett ord som hänger ihop med ett annat utan någon nödvändig grammatisk korrekthet eller logik. Därför kommer alla ord från ordlistan att sättas ihop med varandra.

Algoritm:

```
for each word in list(
 for each word X in list
  append word with word X
)
```

Kostnad = $OrdlistansLängd^X$ där X är antalet ord som frasen består av.

LeetSpeak Phrases

Man kan definiera denna klass som fraser som översatts i sin helhet till ”LeetSpeak”. Detta skulle endast kräva att man översatte ordlistan till ”LeatSpeak” och fortsätter med samma algoritm som för vanliga fraser. Denna studien kommer istället anta att vilken kombination av bokstäver som helst kan bytas ut mot sina motsvarande ”LeetSpeak-tecken”. Denna klassen kommer således definieras som lösenordsfraser (från ”Passphrases”) med substitution (från ”Word permutation”). Grundalgoritmen blir samma som för fraser med skillnaden att man för varje fras måste testa alla kombinationer av substitution. Denna algoritm medför att man, på gott eller ont, för varje grundfras testar alla kombinationer innan man går vidare till nästa grundfras.

Algoritm:

```
for each word in list(
 for each word X in list
  append word with word X
  test all permutation combinations
)
```

Kostnad för fraslängd (X) = Antalet fraser av längden $X \times$ Möjliga kombinationer för längd X

Precis som substitution av ord så inkluderas i denna algoritm substitution av tecken oavsett om det är till ”LeetSpeak” eller om små bokstäver byts mot stora eller tvärtom.

Mnemonic

Denna studien definierar ”Mnemonic passwords” som att använda första bokstaven i varje ord från en X ord lång fras. Eftersom användaren inte behöver ha någon logik i sin fras och inte heller finns det ett lätt sätt att kolla detta innan frasen görs om till ”Mnemonic” så anses det mest generella sättet vara att skapa grundfraser (enligt ”Passphrases”) och sen konvertera dom till ”Mnemonic”. Algoritmen behöver en ny ”loop” för varje ord som frasen ska bestå av men är nerkortad för enklare översikt.

Algoritm:

```
for each word in list(
 for each word X in list
  append word with word X
  make it mnemonic
)
```

Denna algoritmen ger en kostnad på *OrdlistansLängd*^{X} men eftersom fraserna görs mnemoniska spelar inte längre grundfrasens längd någon roll när de jämförs med en ”brute force” attack och således kommer alltid en ”brute force” attack generera lägre kostnad än denna algoritm förutsatt att ordlistan är längre än antalet tecken som används vid ”brute force”. För att knäcka ett 7 tecken långt lösenord med denna algoritm behöver en 7 ord lång fras skapas som även med en kort ordbok på 1000 ord genererar en kostnad på 1000^{7} vilket är betydligt mer än motsvarande ”brute force” på 94^{7} (94 tecken från ASCII-tabellen).

Patterns

Kävrestad själv ger i sin studie ett exempel på att mönster kan vara närliggande tangenter i en given tangentbordslayout. Denna studie väljer att förhålla sig till samma definition där tecken som inkluderas är ASCII-tabellens 94 skrivbara tecken vilket genererar följande algoritm. Precis som med fraser så behövs en ny iteration för varje tecken som lösenordet kan bestå av men är nerkortad för enklare översikt.

Algoritm:

```
for each char X in set(
 foreach adjacentchar Y(
  append char X with adjacentchar Y
  foreach adjacentchar Z(
   append char XY with adjacentchar Z
  )
 )
)
```

Denna algoritmen antyder att man använder sig av samma starttecken tills man testat alla möjliga kombinationer (för längd X) innan man byter starttecken.

Denna studie anser att detta bara utgör en ytterst liten del av vad ett mönster kan vara och att vad som anses vara ett mönster är helt upp till användaren. Således blir denna algoritm endast representativ för mönster med definitionen ”närliggande tecken”.

Alphanumeric and Special Characters

Alfanumeriska tecken anses i denna studie bestå av små och stora bokstäver samt siffror 0 – 9. Specialtecken är de tecken från ASCII-tabellen som inte är alfanumeriska. Denna studie definierar resten av klasserna som slumpmässiga lösenord eftersom att använda sig av dessa tecken på ett logiskt sätt utan att använda någon av ovanstående strategier anses omöjligt. Därav kan ingen algoritm skapas som genererar en lägre kostnad än ”brute force”. Kostnaden för övriga lösenord kommer då att bli *Teckenantal*$^{\textit{Lösenordslängd.}}$.

Intervjufrågor

Denna studien bygger ”semi-konstruerade” intervjuer där stor vikt läggs på vad intervjuobjektet vill lyfta fram. Således kommer troligtvis följdfrågor att tillkomma under intervjun men nedan listas några övergripande huvudfrågor:

- Vad jobbar du med?
- Vad har du hittat som du vill kommentera?
- Saknar du några klasser?
- Vill du ändra på någon/några definitioner? Varför?
- Vill du ändra på någon/några algoritmer? Varför?
- Vill du ändra på något annat?
- Definitionen av klassen utgör bara en del av strategin, hur ser du på klassen ”Patterns”?
- Tycker du att en klass vars kostnad är lägre än brute force kan anses säker?