Project: Password Security

Documentation

BAUHAUS UNIVERSITY WEIMAR

Asif Iqbal and Jula McGibbon

Supervisor:

Dr. Andreas Jakoby

Weimar, April 13, 2016

Abstract

Passwords are used to protect one's information from attacks, when the password itself is secure enough. In our project, password security means to create strength and effective passwords that are less likely to guess and resistent to general attacks. The strength of a password is a function of length, complexity, and unpredictability. On the other hand, the effectiveness of a password of a given strength is strongly determined by the design and implementation of the factors: Knowledge, ownership, inheritance. After analyzing various factors and human behaviour, we ended up with the project implementing various methods, mainly with genetic engineering, in order to generate strong passwords from a given user input, e.g. a favorite quote or a line respectively. For testing purpose, 20 million passwords were generated by the system from a book called ”Time Machine”(cf. [1]). However, one of the most important challenges is to remember the passwords after choosing a set of passwords which is the future task of password security.

Contents

1 Introduction 5

1.1 Background of project 5

2 Human psychology and password analysis 5

3 Password Analysis 6

3.1 Most common passwords 6

3.2 Analyzing three data bases 6

3.3 Analyzing the web portal RockYou 9

3.3.1 Most common passwords findings 10

3.4 Analyzing Hasso Platner data 10

4 Types of attacks 12

5 Password generator 12

5.1 Mutation methods 12

5.1.1 Substitution 13

5.1.2 Cutting, mixing and sticking back together 13

5.1.3 Reversing 14

5.2 Approach 14

5.3 The problem of some operations 16

6 Guessing Metrics 16

6.1 Shannon entropy 17

6.2 Guess work 17

6.3 Cross entropy 18

6.4 Strength metrics 19

6.5 Password probabilities according to our algorithm 20

7 Code **20**

7.1 Eclipse 20

7.2 Program 21

8 Survey **21**

8.1 Analysis of the survey results 22

8.1.1 How familiar are you with computers, the www, . . .? 22

8.1.2 How many active passwords\accounts do you have approximately? . 23

8.1.3 Long-time passwords and resets 24

8.1.4 Password's security 25

8.1.5 How to generate a password and where to safe it? 26

8.1.6 Passwords with personality 28

8.1.7 Password lengths and password's consistency 28

8.1.8 How about our generator method? 30

8.2 Survey questions 32

9 Conclusion **40**

10 Attachments **42**

1 Introduction

1.1 Background of project

Passwords have become a mandatory element and play a large role in applications or devices. They are nearly universal, that means, they play an important role for gaining access to accounts of all kinds such as email, bank, portal, dating or various social networking accounts. The more applications people are using, the more they become affected by security threat, if the passwords are not strong enough. Usually, people are notoriously poor at achieving sufficient entropy to produce appropriate passwords. According to a study (cf.[2]), 40.54 bits is the average password entropy which was estimated from half a million users. Ordinarily, people are requested to pick a password guided by suggestions or by a limited set of rules, while creating a password for a system. However, by doing so, they get passwords of weak strength, since humans have a tendency to follow some specific patterns in creating passwords, which can usually be attacked easily. To resolve this security issue, the project password security came up for generating passwords with sufficient entropy by using different modifications within the passwords. In this project, we describe the issues in terms of creating passwords. The project ends up with a short survey after implementing the password generator to get a rough feedback from users.

2 Human psychology and password analysis

Human psychology of creating or retrieving passwords is the investigation of what makes passwords simple to remember and guessable. In order for making a password work effectively and providing sufficient security to its user, it must be kept as a mystery and non-guessable. This likewise requires the user to remember their secret word. The psychology behind choosing a password is a unique balance between memorability, security and convenience. Password security involves many psychological and social issues, including whether or not to share a password, the feeling of security and the eventual choice of whether or not to change a password. Passwords may also reflect personality. Those who are more uptight or security-oriented may choose longer or more complicated passwords. Those who are lax or who feel more secure in their everyday lives may never change their passwords.

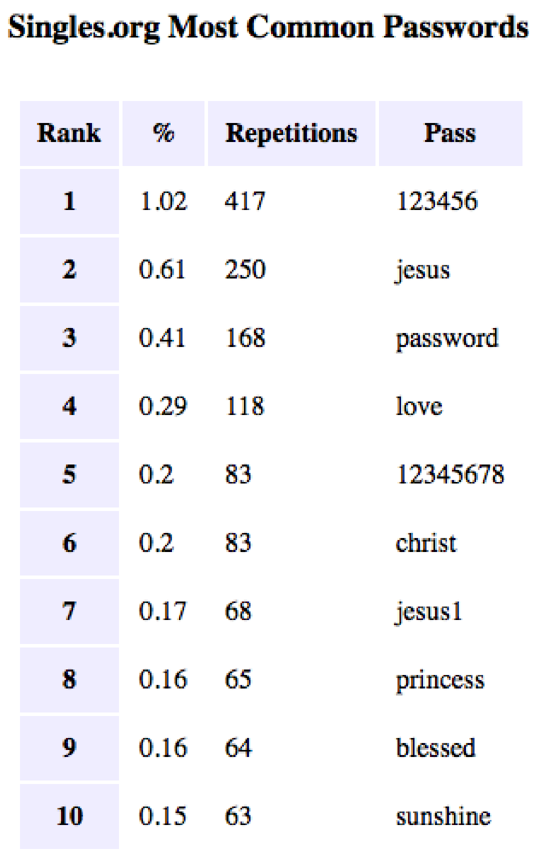
3 Password Analysis

3.1 Most common passwords

We will now have a look at the most common used passwords considering different web portals. The distribution of these passwords can change a lot over time. As we will see, password selection is extremely dependent on peoples interests. But interests change when time passes by and so do passwords. Another dependency is the people's consciousness about security. It can become greater in future times with an increase of attacks on accounts or if people get more informed about security issues beforehand, for example at school.

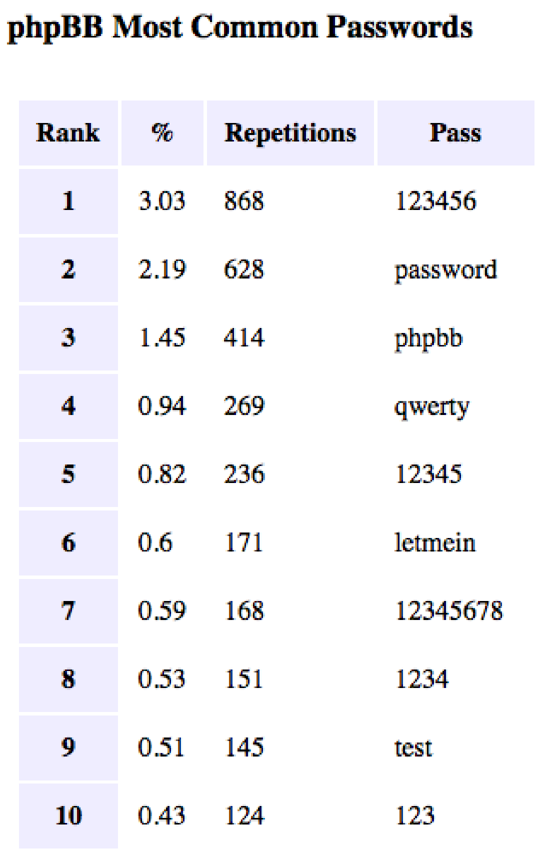
3.2 Analyzing three data bases

At first we evaluate the top 10 most common passwords from the three databases Singles.org, phpBB and MySpace published in February 2009 by [5]. In total, these three data bases include 116782 passwords classified into three categories: password rank, password length, password characters.



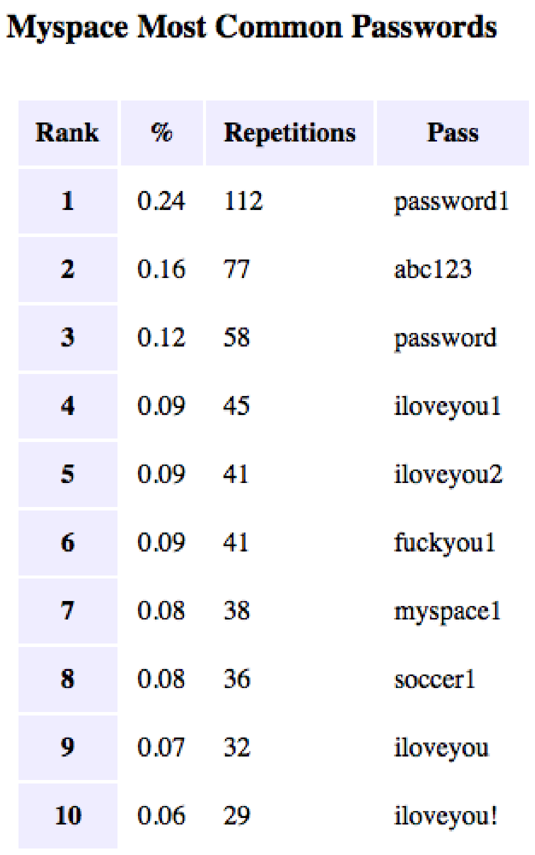
Singles.org Most Common Passwords

Rank	%	Repetitions	Pass
1	1.02	417	123456
2	0.61	250	jesus
3	0.41	168	password
4	0.29	118	love
5	0.2	83	12345678
6	0.2	83	christ
7	0.17	68	jesus1
8	0.16	65	princess
9	0.16	64	blessed
10	0.15	63	sunshine



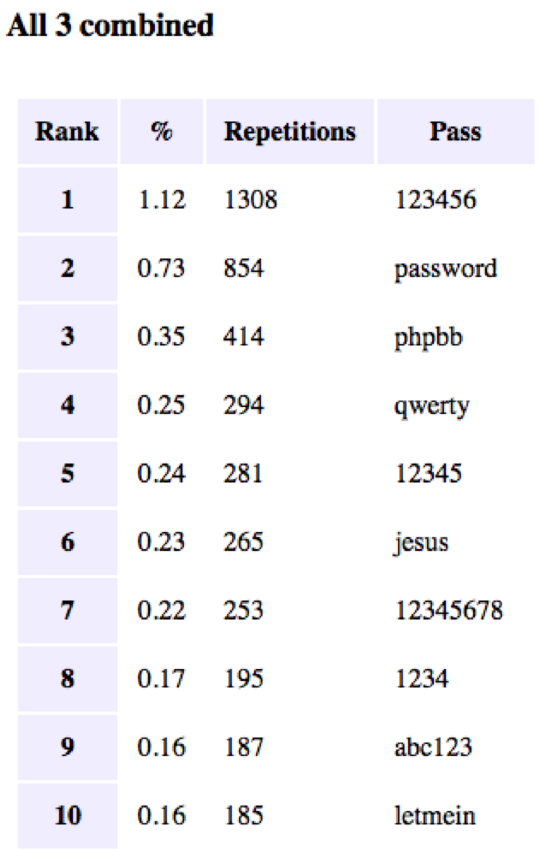
phpBB Most Common Passwords

Rank	%	Repetitions	Pass
1	3.03	868	123456
2	2.19	628	password
3	1.45	414	phpbb
4	0.94	269	qwerty
5	0.82	236	12345
6	0.6	171	letmein
7	0.59	168	12345678
8	0.53	151	1234
9	0.51	145	test
10	0.43	124	123



Myspace Most Common Passwords

Rank	%	Repetitions	Pass
1	0.24	112	password1
2	0.16	77	abc123
3	0.12	58	password
4	0.09	45	iloveyou1
5	0.09	41	iloveyou2
6	0.09	41	fuckyou1
7	0.08	38	myspace1
8	0.08	36	soccer1
9	0.07	32	iloveyou
10	0.06	29	iloveyou!



All 3 combined

Rank	%	Repetitions	Pass
1	1.12	1308	123456
2	0.73	854	password
3	0.35	414	phpbb
4	0.25	294	qwerty
5	0.24	281	12345
6	0.23	265	jesus
7	0.22	253	12345678
8	0.17	195	1234
9	0.16	187	abc123
10	0.16	185	letmein

What becomes immediately apparent is the simplicity of the passwords. The passwords have the following salient characters: The password is . . .

a) . . . a simple number or character sequence like **123456** or **abc123**,

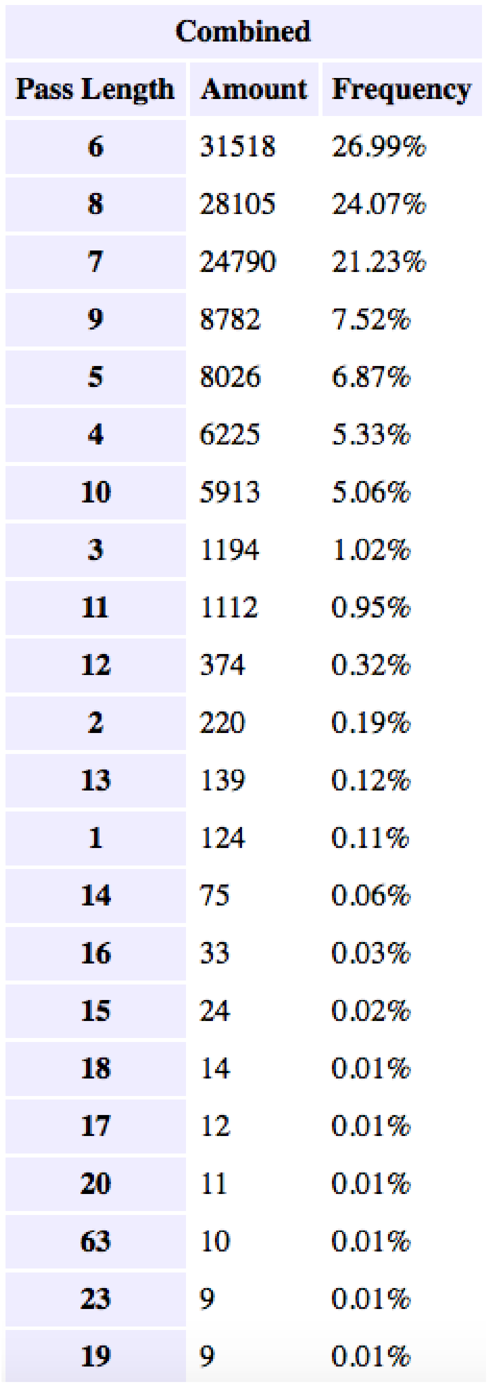
b) . . . a word, which is dependent on the context, like **love** on Singles.org, **phpbb** on phpBB, **myspace1** on MySpace, **password** or **qwerty**,

c) . . . a common used emotional sentence like **iloveyou**, **fuckyou** or an expanded version of it with a number or a symbol at the end, e.g. **iloveyou1**, **iloveyou!**,

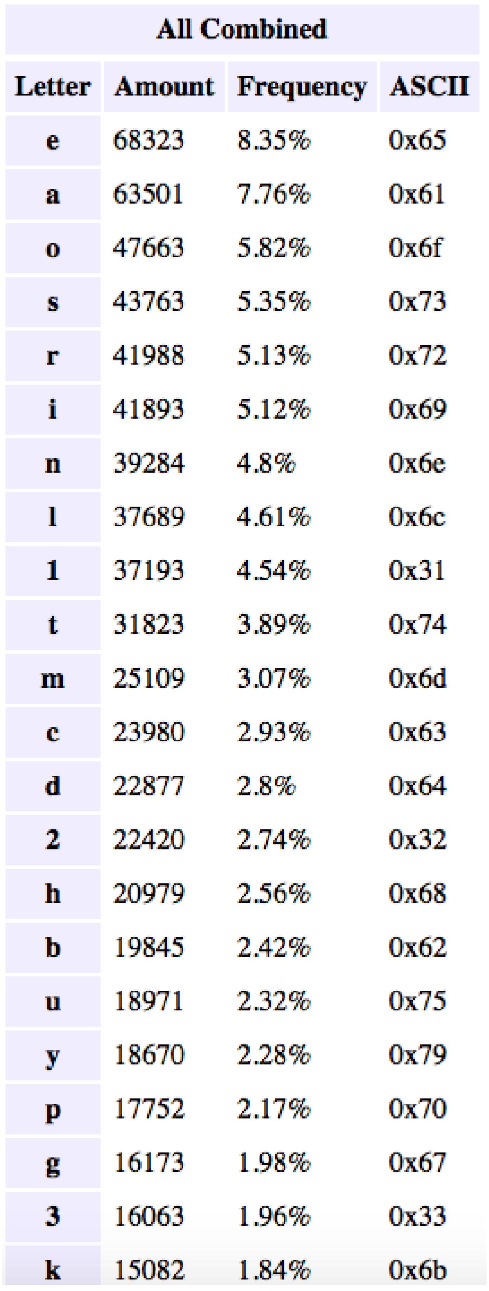
d) . . . a word which has something to do with people's interests, like **soccer1**, **sunshine** or **jesus**,

e) . . . a word which has something to do with people's personality, like a name, an address or a phone number. These passwords are trivially not in the top10, but very common names, like **michael**, are even in the top 40.

Consequently, the password length is mostly not dependent on security measures but on the word(s) of which people think they can remember them.



Combined		
Pass Length	**Amount**	**Frequency**
6	31518	26.99%
8	28105	24.07%
7	24790	21.23%
9	8782	7.52%
5	8026	6.87%
4	6225	5.33%
10	5913	5.06%
3	1194	1.02%
11	1112	0.95%
12	374	0.32%
2	220	0.19%
13	139	0.12%
1	124	0.11%
14	75	0.06%
16	33	0.03%
15	24	0.02%
18	14	0.01%
17	12	0.01%
20	11	0.01%
63	10	0.01%
23	9	0.01%
19	9	0.01%

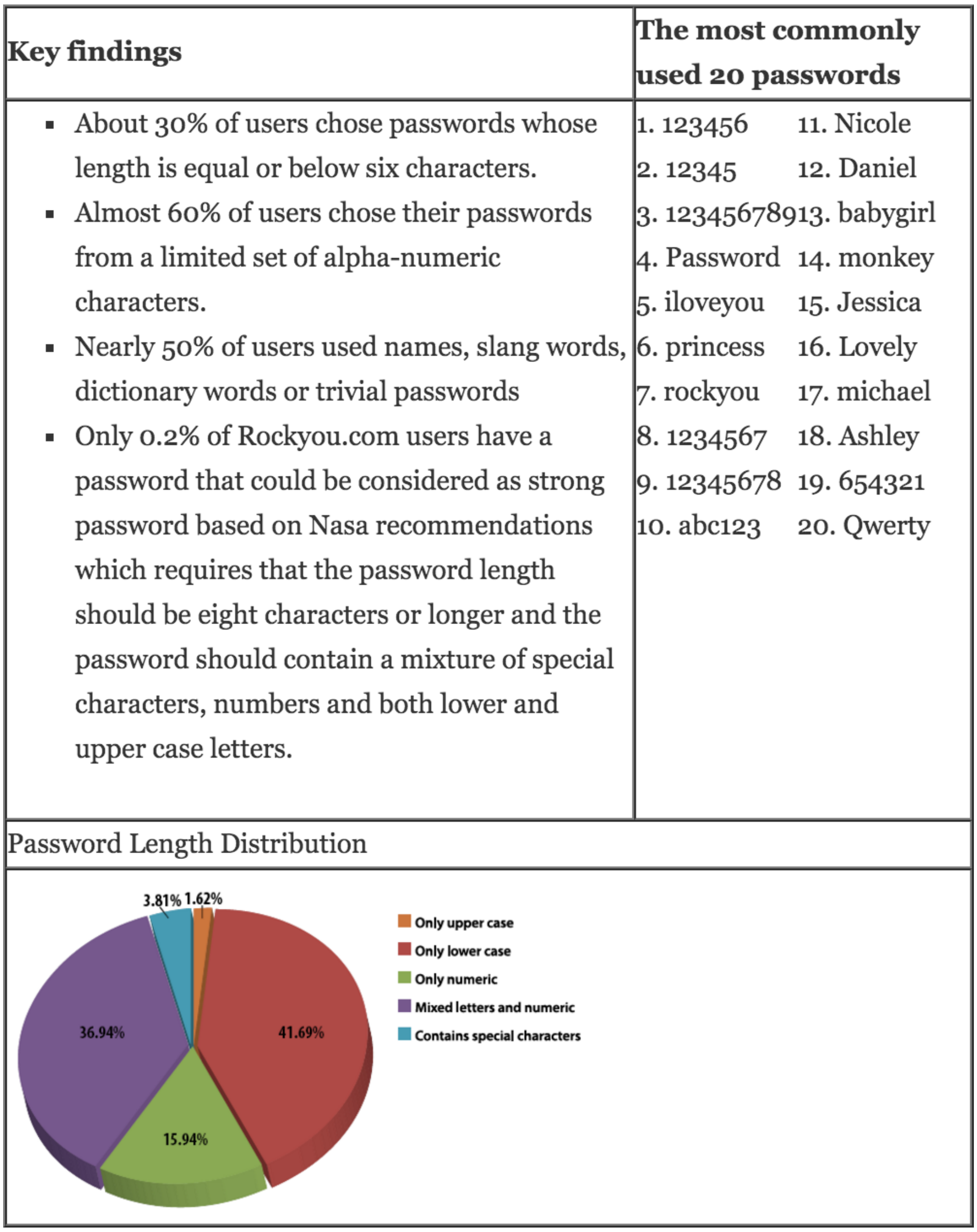


All Combined			
Letter	**Amount**	**Frequency**	**ASCII**
e	68323	8.35%	0x65
a	63501	7.76%	0x61
o	47663	5.82%	0x6f
s	43763	5.35%	0x73
r	41988	5.13%	0x72
i	41893	5.12%	0x69
n	39284	4.8%	0x6e
l	37689	4.61%	0x6c
1	37193	4.54%	0x31
t	31823	3.89%	0x74
m	25109	3.07%	0x6d
c	23980	2.93%	0x63
d	22877	2.8%	0x64
2	22420	2.74%	0x32
h	20979	2.56%	0x68
b	19845	2.42%	0x62
u	18971	2.32%	0x75
y	18670	2.28%	0x79
p	17752	2.17%	0x70
g	16173	1.98%	0x67
3	16063	1.96%	0x33
k	15082	1.84%	0x6b

Looking at the table with the most used password lengths, we can see that most passwords have a length of 7 ± 2, which is exactly the number of chunks a person is said to remember in the short term memory for about 10 seconds (cf.[6]). This could be one reason for this order. Another possible reason is that people actually think this length of a password provides enough security. Particularly alarming are almost all of the lengths under 8 or even 9, especially if they are not randomly chosen passwords which should also include numbers, upper and lower case letters and symbols. Combined with the frequency of the letters any attacker could easily run a Brute Force attack with an underlying dictionary build on these charts and would end up with a remarkable amount of broken accounts.

3.3 Analyzing the web portal RockYou

In December 2009, the Internet experienced a critical data attack. Around 32 million user passwords of the web portal RockYou (cf.[7]) got stolen by a hacker using SQL injection as attack. He got all passwords and made them anonymously (i.e. without usernames) available as download in the Internet. Security experts started analyzing the passwords and Imperva (a provider of cyber and data security products) released a study regarding the security level of the passwords. They came up with the following results:



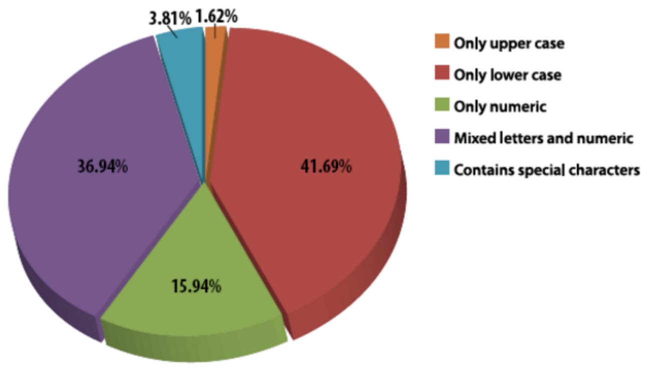
Key findings

- About 30% of users chose passwords whose length is equal or below six characters.
- Almost 60% of users chose their passwords from a limited set of alpha-numeric characters.
- Nearly 50% of users used names, slang words, dictionary words or trivial passwords
- Only 0.2% of Rockyou.com users have a password that could be considered as strong password based on Nasa recommendations which requires that the password length should be eight characters or longer and the password should contain a mixture of special characters, numbers and both lower and upper case letters.

The most commonly used 20 passwords

1. 123456	11. Nicole
2. 12345	12. Daniel
3. 123456789	13. babygirl
4. Password	14. monkey
5. iloveyou	15. Jessica
6. princess	16. Lovely
7. rockyou	17. michael
8. 1234567	18. Ashley
9. 12345678	19. 654321
10. abc123	20. Qwerty

Password Length Distribution



3.3.1 Most common passwords findings

We classify the most common password findings with the categories from 3.2, whereas f) is an additional category.

a) **Simple sequence**: 123456, 123, 123123, 01234, 2468, 987654, 123abc, abc123, 246abc, aaa, eee, llll, 999999, qwerty, asdf, and other keyboard rolls

d) **People's interests**: Favorite Band or Song, favorite cartoon or movie character, favorite sport or sports star, favorite animal, favorite color or food, etc.

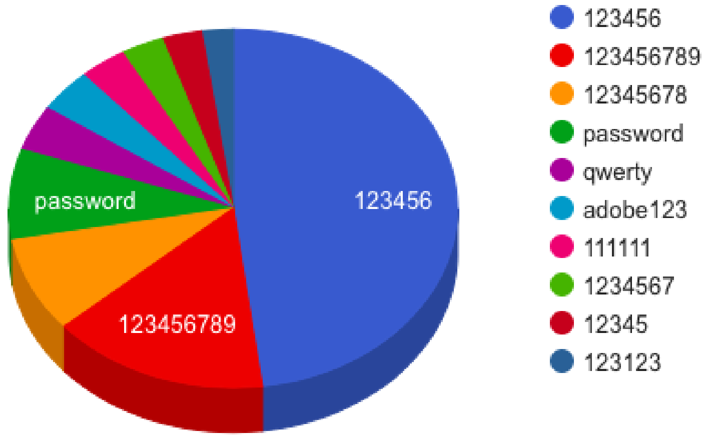
e) **People's personality**: First Name, first letter of given name then surname, country of origin

f) **Variations of the passwords from above**: Combining two words of the different fields, spelling the password backwards, adding a symbol or a number to a given password

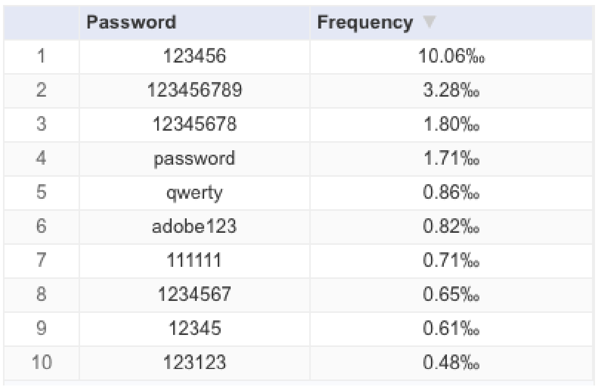
3.4 Analyzing Hasso Platner data

The Hasso-Platner Institut for software system technology in Potsdam collects leaked identity data giving people a opportunity to find out whether their e-mail address, along with other personal data (e.g. telephone number, date of birth or address), has been made public on the Internet. One only needs to type in their e-mail address in a security checker, which compares the e-mail address to the Hasso Platner data base [9] containing 235,733,863 leaked identities (state of the 3rd of April 2016). With this, they want to prevent further misuse of these identities by others and animate people to change their passwords using more secure passwords the next time. Surely the institut cannot assure that the checked e-mail address was not hacked, since their data base does not contain all stolen data of the Internet.

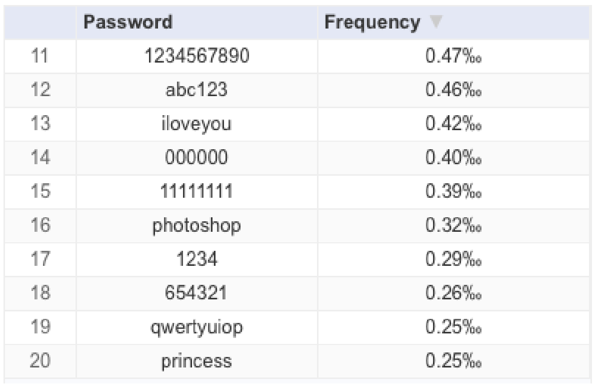
Looking at the Hasso Platner statistics [8], we can still find the simplest passwords to be the most present ones. The distribution of the 10 most common plaintext passwords can be illustrated in a pie chart:



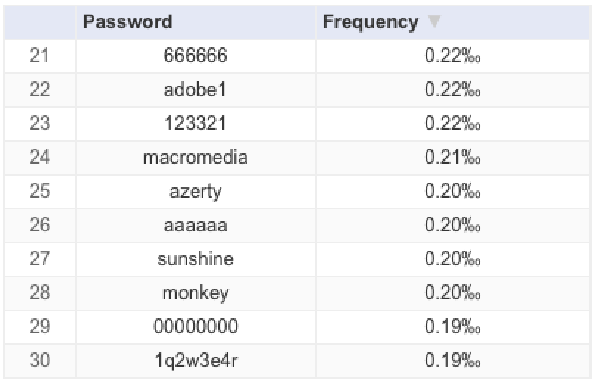
The top 40 passwords are:



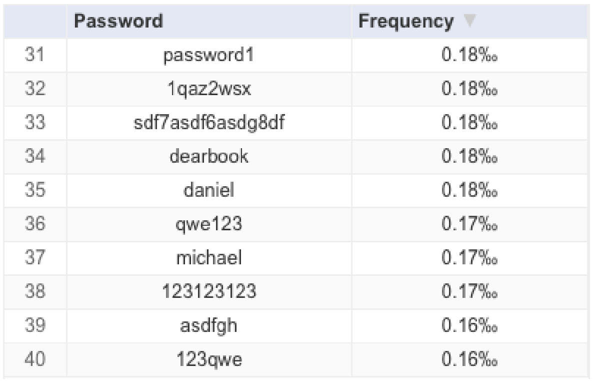
	Password	Frequency
1	123456	10.06‰
2	123456789	3.28‰
3	12345678	1.80‰
4	password	1.71‰
5	qwerty	0.86‰
6	adobe123	0.82‰
7	111111	0.71‰
8	1234567	0.65‰
9	12345	0.61‰
10	123123	0.48‰



	Password	Frequency
11	1234567890	0.47‰
12	abc123	0.46‰
13	iloveyou	0.42‰
14	000000	0.40‰
15	11111111	0.39‰
16	photoshop	0.32‰
17	1234	0.29‰
18	654321	0.26‰
19	qwertyuiop	0.25‰
20	princess	0.25‰



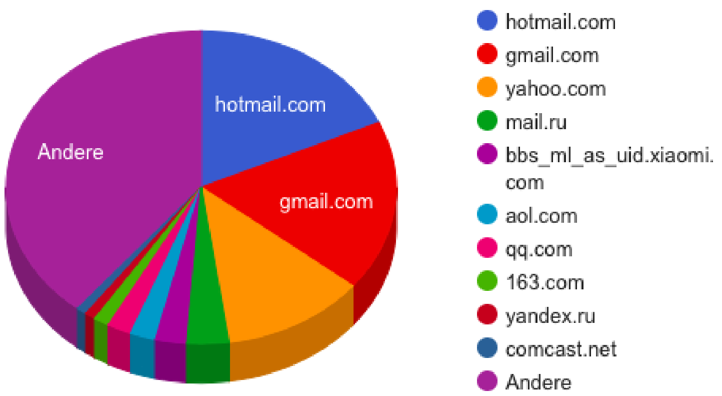
	Password	Frequency
21	666666	0.22‰
22	adobe1	0.22‰
23	123321	0.22‰
24	macromedia	0.21‰
25	azerty	0.20‰
26	aaaaaa	0.20‰
27	sunshine	0.20‰
28	monkey	0.20‰
29	00000000	0.19‰
30	1q2w3e4r	0.19‰



	Password	Frequency
31	password1	0.18‰
32	1qaz2wsx	0.18‰
33	sdf7asdf6asdg8df	0.18‰
34	dearbook	0.18‰
35	daniel	0.18‰
36	qwe123	0.17‰
37	michael	0.17‰
38	123123123	0.17‰
39	asdfgh	0.16‰
40	123qwe	0.16‰

One can also find the top 100 on the Hasso Platner page. The most common passwords consist of simple number sequences or keyboard sequences containing keys close to each other, e.g. the zigzag sequence **1q2w3e4r** at rank 30.

The following pie charts shows the distribution of all identities in domains. The most common ones are hotmail.com and gmail.com, all domains beneath the top ten domains are listed as "Andere":



4 Types of attacks

To understand how to protect password systems, it is mandatory to become familiar with the most commonly used types of attacks. We listed some of them below:

- **Brute force attack**: Systematically checking all possible keys or passwords until the correct one is found.
- **Dictionary attack**: Trying hundreds or sometimes millions of likely possibilities, such as words in a dictionary.
- **Password resetting**:
  - Using the option ”I forgot my password” of an account and correctly answering the password reset questions.
  - Hacking into an email account and fishing the new login data for the reset of a password.
- **Rainbow table**: Precomputed table for reversing cryptographic hash functions.
- **Password sniffing**: Listening to all incoming and outgoing network traffic to record any instance of a data package that contains a password.

5 Password generator

In this section, we try to construct a password generator in theory. Our goal is to build a password generator which creates passwords with high security and high memorability. These two factors are not easy to combine. Our approach is to get passwords out of books as mnemonics. This means, we are only taking the first letters of the words and all special characters, e.g. a point, a comma or a question mark. Each sentence should give us one mnemonic. Accordingly, each mnemonic will end with a dot and start with an upper case letter. It is obvious that this lowers the security of the password, since there exists kind of a pattern, if not necessarily of a big consequence. To blur this pattern and to make the password more secure, we apply some mutation operations on the password, like substitution, reversion or cutting, mixing and sticking back together. Conversely, this procedure will lower the memorability of the password. Therefore, we need to keep the operations simple and easy to remember, but still with enough complexity to ensure a certain security.

5.1 Mutation methods

In the following, we explain operations for modifying a password which ones will be used in our program (cf. 7).

5.1.1 Substitution

Substitution will replace certain letters with different but similar letters, numbers or symbols. The positions for replacement are randomly chosen. We will use:

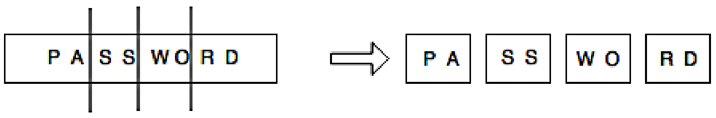
- a → @
- t or T → 7
- e → 3
- g or B → 8
- i or I → 1
- x or X → *
- O or o → 0
- b → 6
- y → 4
- . . . see 7 for more

The memorability of which letters get exchanged with which symbols is probably quite good to remember if the letters are very similar to the symbols. We assume this is mostly the case with our chosen substitutions. The challenge for the password owner is to remember which letters get substituted and which don't, since this is randomly chosen by the algorithm. The procedure of substitution is quite public among hackers and it is also not very hard to guess which letter might got exchanged with which symbol due to the similarity of the letters and symbols. Thus, the randomly chosen substitution is necessary to assure security.

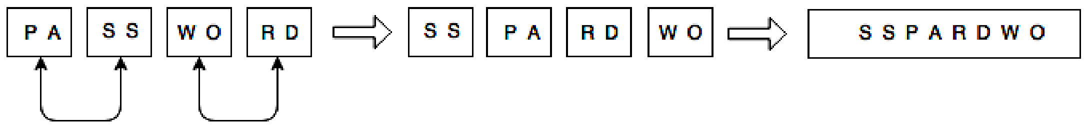
5.1.2 Cutting, mixing and sticking back together

Cutting will cut the password at different positions, mixes them up and puts them back together as one password.

There are two important steps to remember. First, the positions of the cuttings, which can be chosen uniformly to enhance memorability, e.g.



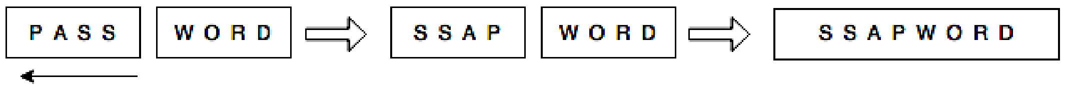
Second, the swaps of the individual cut parts, which do not exceed two swaps for our generator:



Since there are only two simple systematical and picturesque steps to remember, this will probably be the easier part of the mutation process to keep in memory. Rather more challenging than remembering is to write down a password, when doing swap operations at the same time. The swapping part will mess up the sentence structure and is likely to result in a senseless order, which can be followed by confusion and demotivation of the performing person. This is why we should keep these mutations simple but not too simple to provide a certain security.

5.1.3 Reversing

Reversing will either reverse the whole string or parts of the string. Therefore, it can be a mixture of first of all cutting, reversing the cut parts and sticking them back together:

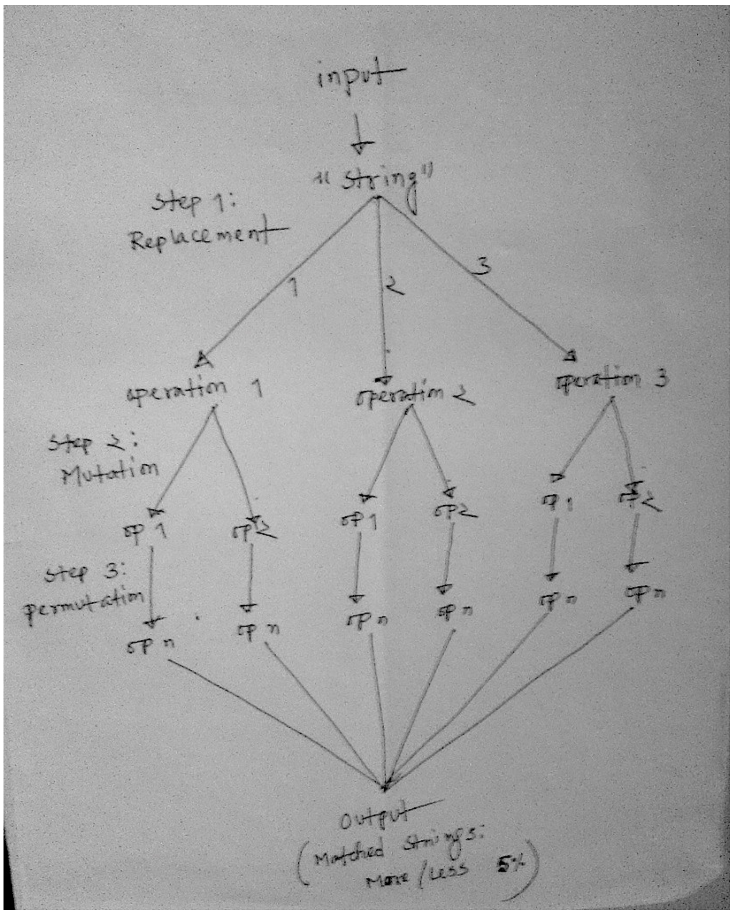


As in the previous part, the operation itself won't be as hard to remember. But to write down the first characters of a sentence while reversing some characters is definitely the most difficult step. Therefore, reversing should only happen in one and mostly two parts within the cut parts.

5.2 Approach

Our approach was taking an arbitrary book, getting out all possible mnemonics and applying the operations discussed in 5.1. The book we used is called "Time Machine" (cf.[1]). As result, our program generated in the first place 2000 passwords respectively mnemonics out of this book. On each resulting password we applied the mutation techniques discussed above and stopped our program after it generated about 2 million passwords. Having these 2 million passwords, it is possible to extend the amount of passwords even more with some simple tricks. For instance, we can take one password out of the data set and combine it with all the other ones. Then it has double length and is therefore more secure. Of course, the effort to remember such a password doubles itself as well. Apart from this, we can do this with every password in the set which gives us 2 million times

2 million passwords, i.e. 4 trillion ($4 \cdot 10^{12}$) passwords generated out of one book. The following figure shows all operations used for generating the passwords step by step:



After applying the different algorithm for all possible combinations, we got the same result and from one string it generates all unique combinations without any other operations. It is a long process to generate strings from many and its run time is huge. The following calculations is to get all possible strings of different lengths....

a(l=1) = 1
ab(l=2) = 2
abc(l=3) = 6
abcd(l=4) = 24
abcde(l=5) = 120
abcdef(l=6) = 720
abcdefg(l=7) = 5040
abcdefgh(l=8) = 40320
abcdefghi(l=9) = 362880
abcdefghij(l=10) = 3628800
abcdefghijk(l=11) = 39916800
abcdefghijkl(l=12) = 479001600
abcdefghijklm(l=13) = 6227020800

abcdefghijklmn(l=14) = 87178291200
. . . and so on.

5.3 The problem of some operations

The question which arises when generating different passwords out of one by various operations is, if these passwords are really different in the end or if operations generated in some cases the same passwords out of the former ones.

For example, we could have generated the mnemonic **ToboT** out of our book. If we revers this one, it will still be the same afterwards, that means the operation is invariant to this password. For substitution, the words **Ueal** and **U3@l** will result in the same password, if we only apply the substitution a $\rightarrow$ @ and e $\rightarrow$ 3. Also cutting can have this effect, e.g. cutting **abccba** in half, cutting **aacbbc** in three parts and swapping the first to blocks in both cases leads to the password **cbaabc**.

The question is, how severely are our passwords effected by this?! First of all, we always have a symbol at the end of a sentence, like a point, question mark or exclamation mark. Therefore, reversing the whole password won't result in the same password, since there is no symbol at the beginning of the sentence. The probability that substitution will lead to same passwords is also quite low, since sentences with numbers or symbols (others than the normal grammar symbols) don't appear in general.

Nevertheless, we took some actions to be sure, there won't appear an unfavourable case like the one discussed above. If same passwords already appear in the first password selection without modifications, we will delete them or apply different operations on these passwords. Furthermore, we will avoid a password with similarities within itself, like **Teee**. This password will be reduced to **Te**.

6 Guessing Metrics

Since we try to generate passwords which are easy to remember, these passwords still need to provide enough security. Therefore, we will examine some metrics helping us to measure the guessing difficulty of our generated passwords. The main source, we are using for this, is the dissertation "Guessing human-chosen secrets" by Joseph Bonneau (cf.[4]). In the following, let $\mathcal{X}$ be the underlying distribution and p_i the probability of a given symbol.

6.1 Shannon entropy

The most common metric to measure the expected value of the information in a message is the Shannon entropy:

$$H_1(\mathcal{X}) = \sum_{i=1}^{N} -p_i log_2(p_i).$$

However, this entropy measure is not compatible with our purpose, because it has no correlation to guessing difficulty, since the attacker guesses all possible values individually and strives to guess the most likely values at first.

6.2 Guess work

One option to measure the guessing difficulty, when the attacker proceeds in optimal order, is the **Guess work** metric, given by:

$$G_1(\mathcal{X}) = E[\#_{\text{guesses}}(X \leftarrow \mathcal{X})] = \sum_{i=1}^{N} p_i \cdot i.$$

This metric comes closer to the procedure of a real-world attacker. But it will still produce absurd results without having a stopping criterion. In the real world the attacker would stop guessing against the most difficult values, like 32-character-heximaldecimal strings.

Now, let us assume, there are k accounts which are attacked at once. The attacker will first of all guess the most likely value, then he will go on with the second most likely value and so on until $l \geq k$ accounts are broken. Thus, we can measure the expected success for an attacker limited to β guesses with the β **-success-rate**:

$$\lambda_\beta(\mathcal{X}) = \sum_{i=1}^{\beta} p_i.$$

If we want to find out the number of guesses needed to guess a certain proportion α, we can sum up the probabilities of the single passwords, beginning with the most likely one, and count the guesses until reaching the desired proportion. This job can be done by the α**-work-factor**:

$$\mu_\alpha(\mathcal{X}) = \min\{\mu | \sum_{i=1}^{\mu} p_i \geq \alpha\}, \quad \alpha = \frac{\ell}{k}.$$

Combining these two formulas into one, we obtain the α**-Guesswork metric:**

$$G_\alpha(\mathcal{X}) = (1 - \lceil\alpha\rceil) \cdot \mu_\alpha(\mathcal{X}) + \sum_{i=1}^{\mu_\alpha(\mathcal{X})} p_i \cdot i,$$

where

$$\lceil\alpha\rceil = \lambda_{\mu_\alpha(\mathcal{X})}(\mathcal{X}) = \sum_{i=1}^{\mu_\alpha(\mathcal{X})} p_i.$$

The result is the expected number of guesses per account to achieve a success rate α. Breaking all accounts is represented by $\alpha = 1$. Since $\lceil \alpha \rceil$ rounds up the expected success for an attacker limited to $\mu_\alpha(\mathcal{X})$ guesses, the expression $(1 - \lceil \alpha \rceil)$ rounds down the proportion of unsuccessfully guessed values, because the attacker may actually succeed with probability greater than α.

For comparing the α-Guesswork to other entropy measures, we need to convert it into units of bits or an ”effective key-length” respectively. This will represent the size of a randomly chosen cryptographic key which gives equivalent security. A metric's effective key-length is the (logarithmic) size of a uniform distribution $\mathcal{U}_N$. The β-success-rate of a uniform distribution is $\frac{\beta}{N} = \lambda_\beta(\mathcal{X})$. Thus, any distribution $\mathcal{X}$ with respect to $\lambda_\beta(\mathcal{X})$ is equivalent to a uniform distribution of size $N = \frac{\beta}{\lambda_\beta(\mathcal{X})}$. Accordingly we derive the effective key metrics:

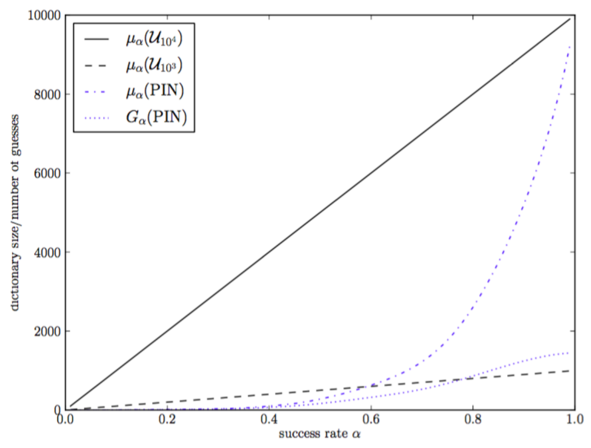
$$\tilde{\lambda}_\beta(\mathcal{X}) = log_2\left(\frac{\beta}{\lambda_\beta(\mathcal{X})}\right)$$

and

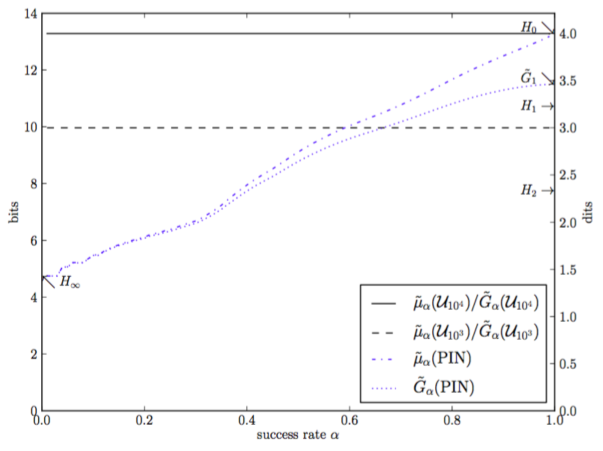
$$\tilde{\mu}_\alpha(\mathcal{X}) = log_2\left(\frac{\mu_\alpha(\mathcal{X})}{\lceil \alpha \rceil}\right).$$

For the α-guesswork, we would get an equation which is a bit more complicated. To simplify comparing in this case, we use the effective key-length:

$$\tilde{G}_\alpha(\mathcal{X}) = log_2(N).$$



(k) Plots the dictionary size μ_α needed to have a chance of success α, as well as the expected number of guesses per account G_α.



(l) Converts both metrics into an effective key-length. Note: $\tilde{\mu}_\alpha$ and $\tilde{G}_\alpha$ are horizontal lines for a uniform distribution, i.e. an attacker gains no efficiency advantage at a lower desired success rate α.

6.3 Cross entropy

In section 6.2 we considered that the attacker was guessing a certain proportion of passwords in an optimal order and stopped when guessing got too expensive. Now, we look at an attacker who guesses an distribution $\mathcal{Y}$ against an underlying distribution $\mathcal{X}$. This makes sense, since many factors can lead to a variety of possible guessing dictionaries, e.g. by different language skills or different interests of the users. In the following, let $p_j^{\mathcal{X}}$ be the probability over $\mathcal{X}$ of the j^{th} most common event in $\mathcal{Y}$ and $p_j^{\mathcal{Y}}$ be the probability over $\mathcal{Y}$ of the j^{th} most common event in $\mathcal{Y}$. We measure the expected number of bits needed to encode an event drawn from $\mathcal{X}$ using an optimal coding scheme based on $\mathcal{Y}$ with the **Cross entropy of $\mathcal{X}||\mathcal{Y}$** computed by:

$$H_1(\mathcal{X}||\mathcal{Y}) = -\sum_{j=1}^{|\mathcal{Y}|} p_j^{\mathcal{X}} log_2(p_j^{\mathcal{Y}}).$$

To measure the number of extra bits needed to encode events from $\mathcal{X}$ using a code from $\mathcal{Y}$, we can use the **Kullback-Leibler-divergence** given by:

$$D_{\text{KL}}(\mathcal{X}||\mathcal{Y}) = \sum_{j=1}^{|\mathcal{Y}|} p_j^{\mathcal{X}} log_2\left(\frac{p_j^{\mathcal{Y}}}{p_j^{\mathcal{X}}}\right).$$

If there are events in $\mathcal{X}$ which don't exist in $\mathcal{Y}$, then $p_j^{\mathcal{Y}}$ is 0 for $j \leq \mathcal{Y}$, therefore $H_1(\mathcal{X}||\mathcal{Y})$ and $D_{\text{KL}}(\mathcal{X}||\mathcal{Y})$ would be ∞, because they include $log_2(0)$. Hence, they are not defined in this case.

6.4 Strength metrics

Regarding our project it is also useful to estimate the resistance of guessing provided by a specific password x, instead of analyzing the guessing resistance of an entire distribution $\mathcal{X}$. Giving the user knowledge about it, could be an enlightenment about the security of their chosen passwords. While language difference does not make a major difference in guessing attacks on the entirety, it can make a huge difference for individual passwords, since the attacker could have some knowledge about the nationality of the user and also about some interests. This can shrink the distribution of possible passwords enormously.

Our strength metric denoted by $S_{\mathcal{X}}(x)$ should have the following **Key properties**:

1. Consistency for uniform distributions: $\forall_{x \in \mathcal{U}_N} : S_{\mathcal{U}_N}(x) = log_2(N)$,
2. Monotonicity: $\forall_{x,x' \in \mathcal{X}} : p_x \geq p_{x'} \Leftrightarrow S_{\mathcal{X}}(x) \leq S_{\mathcal{X}}(x')$.

Let i_x be the index of a password x which is the number of items in $\mathcal{X}$ of greater probability than p_x. We can measure the strength of an individual password by estimating when it would be guessed by a password-cracking software using the **Index metric**:

$$S_{\mathcal{X}}^{I}(x) = log_2(2 \cdot i_x - 1).$$

6.5 Password probabilities according to our algorithm

Evaluating our data set of 2 million passwords, we found out that we generated almost no duplicate passwords. Mnemonics of the same length including the mutation operations are quite uniformly distributed, thus the probability that an attacker would guess any of the passwords earlier than another one is very low. Measuring the strength of such a data set, with the Shannon entropy or the Guess work, we will be close to the maximum outcome, because the maximum is reached if the distribution is uniform (cf. 6.2, figure g). In other words, uncertainty is highest when all possible events are equiprobable. Therefore, a source alphabet with a non-uniform distribution will have less entropy. If we include passwords of different lengths, which is the case in our data set, we will get less entropy.

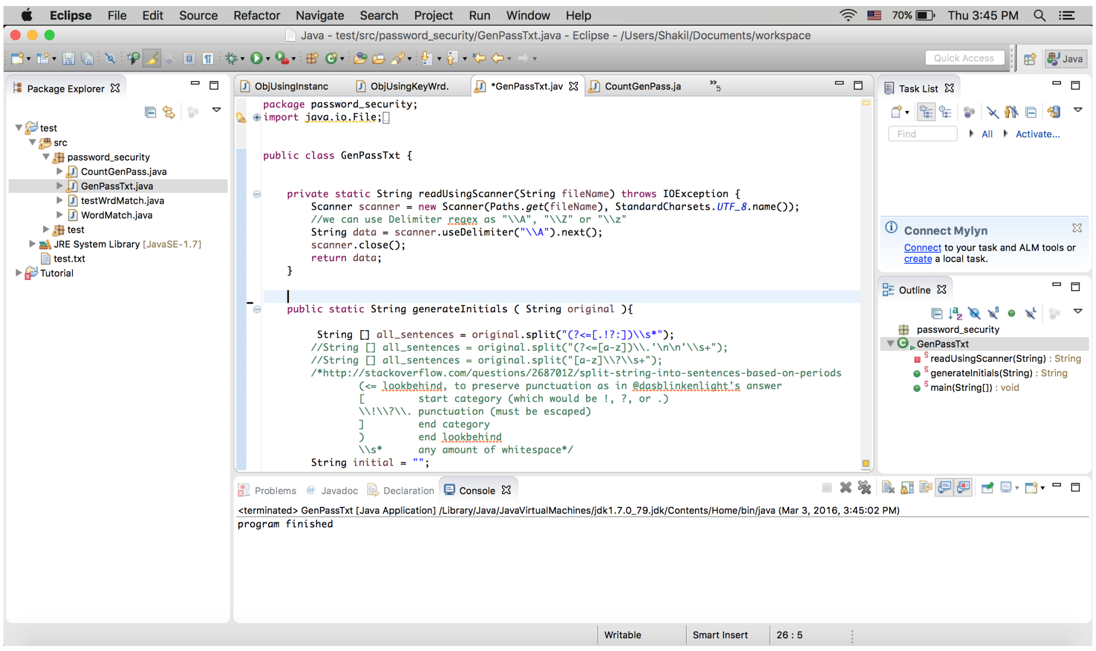
Regarding the cross entropy, our generated password data set has the distribution $\mathcal{X}$ an attacker would need to guess against. It is also called the true distribution. The attacker needs to approximate this distribution with a distribution $\mathcal{Y}$. If the attacker knew something about the system, he would also take a book or some more books and generates his own passwords according to his knowledge. The approximation will be close to the true distribution, if the attacker knows specific details about the system, i.e. about specific mutation operations or about the used book. Then, the entropy measure will be close to zero - It is zero for equal distributions. The less the attacker knows about the system, the bigger will be the distance between the two distributions and thus the higher the entropy measure.

Concerning the Index metric, we come to rather unsatisfying results for a uniform distribution. The Index metric measures the strength of an individual password by estimating when it would be guessed by an attacker and uses in this case an index of a password, which is the number of items in $\mathcal{X}$ of greater probability then p_x. This is a problem, if we look at a uniform distribution, because then we cannot order our probabilities. Therefore, it is not recommended to use this metric in this case. Since our data set includes passwords of the same kind but of different lengths, we could order them according to their lengths, because shorter passwords will be guessed before longer passwords.

7 Code

7.1 Eclipse

We had used Eclipse throughout the project for the implementation of each and every step. Eclipse is an Integrated Development Environment (IDE) developed by IBM. It is mostly written in java, used to develop applications. It involves many different task e.g. coding, testing, compiling, running and debugging etc. While opening Eclipse, it always asks for workspace selection to achieve better organizational structure of work files.


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Java - test/src/password_security/GenPassTxt.java - Eclipse - /Users/Shakil/Documents/workspace
Package Explorer
test
src
password_security
CountGenPass.java
GenPassTxt.java
testWrdMatch.java
WordMatch.java
test
JRE System Library [JavaSE-1.7]
test.txt
Tutorial
ObjUsingInstanc
ObjUsingKeyWrd.
*GenPassTxt.jav
CountGenPass.ja
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
password_security
GenPassTxt
readUsingScanner(String) : String
generateInitials(String) : String
main(String[]) : void
Problems Javadoc Declaration Console
Writable Smart Insert 26 : 5

```
package password_security;
import java.io.File;


public class GenPassTxt {


    private static String readUsingScanner(String fileName) throws IOException {
        Scanner scanner = new Scanner(Paths.get(fileName), StandardCharsets.UTF_8.name());
        //we can use Delimiter regex as "\\A", "\\Z" or "\\z"
        String data = scanner.useDelimiter("\\A").next();
        scanner.close();
        return data;
    }


    public static String generateInitials ( String original ){

         String [] all_sentences = original.split("(?<=[.!?:])\\s*");
        //String [] all_sentences = original.split("(?<=[a-z])\\.'\n\n'\\s+");
        //String [] all_sentences = original.split("[a-z]\\?\\s+");
        /*http://stackoverflow.com/questions/2687012/split-string-into-sentences-based-on-periods
                (<= lookbehind, to preserve punctuation as in @dasblinkenlight's answer
                [         start category (which would be !, ?, or .)
                \\!\\?\\. punctuation (must be escaped)
                ]         end category
                )         end lookbehind
                \\s*      any amount of whitespace*/
        String initial = "";
```

```
<terminated> GenPassTxt [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java (Mar 3, 2016, 3:45:02 PM)
program finished
```

More information about this software can be obtained here:

`http://en.wikipedia.org/wiki/Eclipse_(software)`.

Downloaded from: `https://eclipse.org/downloads/`

7.2 Program

The biggest challenge to build up our program for the generator were the mutation operations. To prevent some patterns, we wanted them to be randomly chosen. In the main method we read in the file 'TimeMachine.txt' by calling the 'readUsingScanner' method. For the replacement we call the 'rand_replacement' method. All generated strings are stored in text files. See 10 for the main program.

8 Survey

At the end of our project, we created a questionnaire asking people about the choice and properties of their passwords. All together we asked about 40 people, including an eight grade school class from the Friedrich-Schiller-Gymnasium in Weimar. We only took a sample of 10 people out of this school class, otherwise they would have influenced our study about 50 percent. We also covert the age classes 16-25 and 25-35 quite well. Together they have a weight on the result of 50 percent. All other age classes are less representative. In the end of our questionnaire, we asked some questions about our password generator. For creation of the questionnaire, we used Google Formulare [10], which also created all pie diagrams. All bar diagrams were build by Meta charts [11].

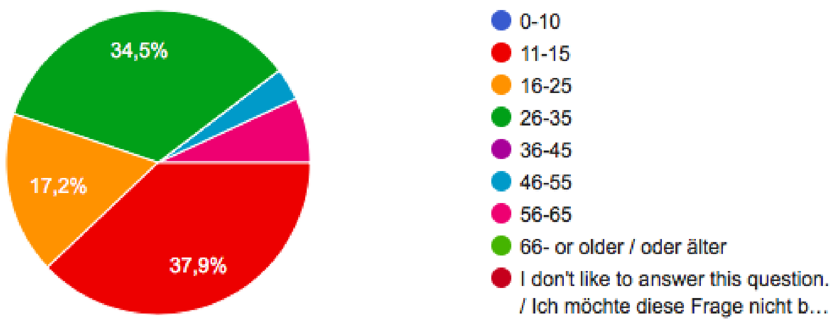


Figure 1: Age classes

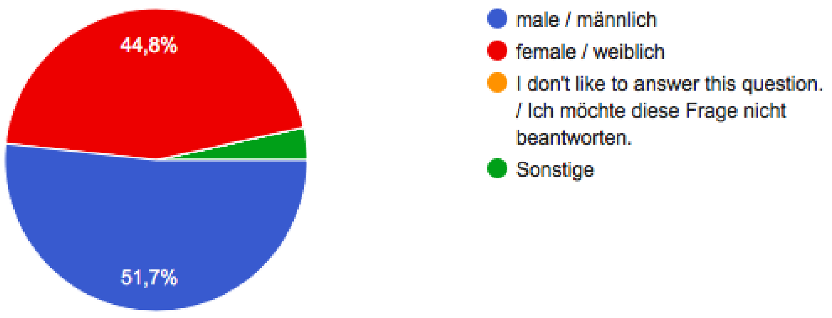


Figure 2: Gender

8.1 Analysis of the survey results

We will now analyze our results from the survey. At first, we will only list facts to the given statistics. Then we will give reasons why the results could be like this respectively what influence led to results like this. If needed, we have a look at the utility and flaws of the question. The question in 5.5. and 6.5. was not considered for further analysis, since it turned out to be misunderstood with respect to the results. The sense of the question was to ask people if they reuse passwords and if yes, we wanted to know the number of reuses of the password with the maximum reuses. Instead, we used question 1 and 2 to analyze this question. We won't present detailed results of every person, so we can provide a certain anonymity for them.

8.1.1 How familiar are you with computers, the www, . . . ?

Keyfindings from Question 12:

- Most of the questionees answered, they are 'very or quite familiar' with computers or the World Wide Web.

- Particularly noticeable is that seven of eleven 8 grade students answered, that they are 'very familiar' with it.
- Only people from the age classes 46-55 and 56-65 answered the question with a 'not familiar'.

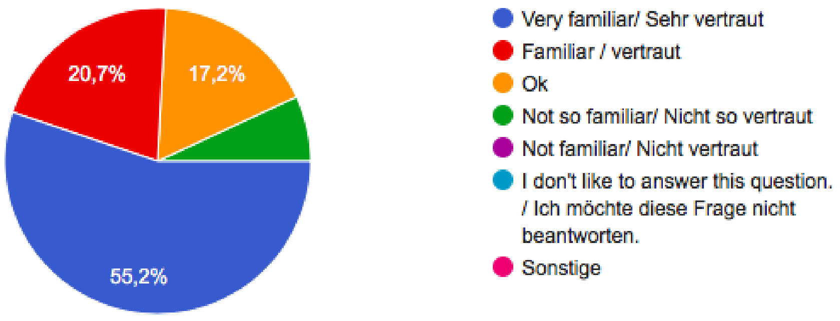


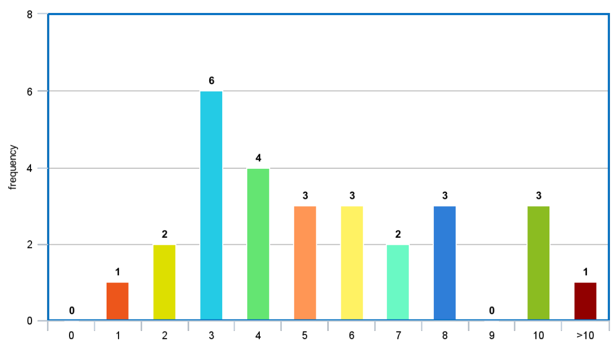
Figure 3: How familiar are you with computers, the www ...?

The question shows how present computers and the internet are nowadays. But it also shows that the question was asked very generally. If it was something like "Are you an computer expert or not?", the results might be different. To get a more precise view about the humans' choice of passwords, it would be necessary to ask more people without good knowledge about computers and the World Wide Web.

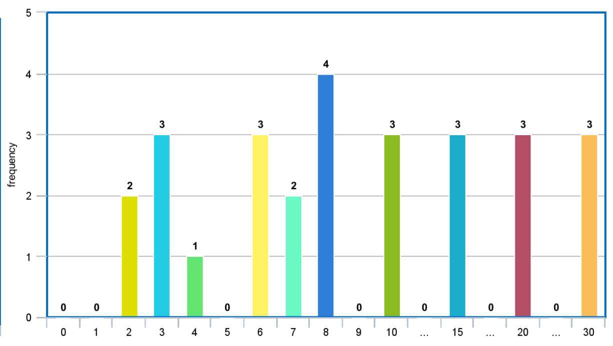
8.1.2 How many active passwords\accounts do you have approximately?

Keyfindings from Question 1 and 2:

- 4 persons with more passwords than accounts.
- 8 persons have more or the same amount of passwords as accounts.
- 5 people have 4-6 times more accounts than passwords.
- 5 people have about twice as much accounts as passwords
- Two people didn't answer completely



(a) Number of passwords



(b) Number of accounts

The fact, that some people stated they have more passwords than accounts might refer to the fact, that people usually underestimate their number of active accounts. When answering the questionnaire, they probably only think about the most necessary accounts, like an email account or a social network account. But usually they have a lot of other accounts which they only need in special situations, like an account for a run registration, an account on a shopping page, an account for a phone contract and so on. Since it is more necessary to remember passwords, the number of passwords they remember in this moment, is more present in their minds and therefore higher than the number of accounts. In total, we can observe that people generally have more accounts than passwords, thus they mostly reuse passwords for different accounts. Possible reasons are: They don't see any sense in having many different passwords; They don't want to or think they can't remember more than a certain amount of passwords without any additional help like a key manager. Why are they not using additional help like a key manager? This question can have different answers, e.g.: They started with a few accounts for which they could remember all passwords; They are uninformed about other options for storing passwords, ... and so on. Furthermore, we found no correlation between the number of passwords with respect to the number of accounts and the gender or age of a person.

8.1.3 Long-time passwords and resets

Keyfindings from Question 3 and 4:

- 3/4 of the people use passwords from inactive accounts for active accounts
- About 70 percent are resetting their password 'not very often' (55,2 %) or 'never' (13,8%)
- 17% reset their password 'very often' or 'often'

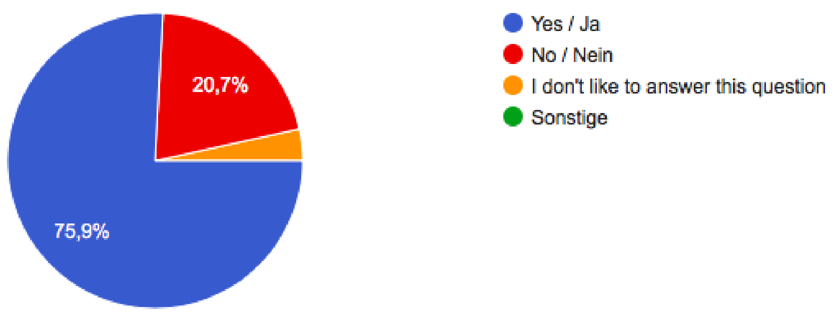


Figure 4: Are you using passwords on active accounts, which you already used on currently inactive accounts?

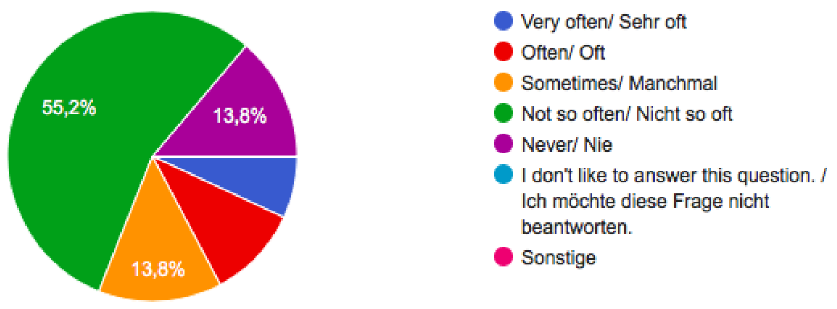


Figure 5: How often are you resetting your password, because you forgot about it?

The question ”Are you using passwords on active accounts, which you already used on currently inactive accounts?” reveals how stuck most of the people are with passwords, which they are already using for a years. Certainly, long-time passwords are burnt into people’s minds and provide them with a certain guaranty of remembering them. Therefore, they accept negative side effects of this behaviour. Usually, most of the older passwords are not chosen quite well, since people started with them, when coming in touch with computer or the www. Thus, if one account with such a password is hacked, the attackers also have access to all other accounts with the same password.

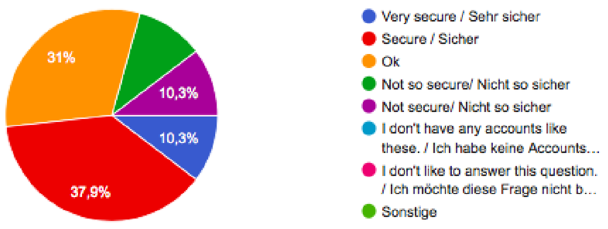
Since there are also attacks exploiting password resets (cf. 4), we asked the question ”How often are you resetting your password, because you forgot about it?”. Quite a few people (17%) answered, that they do resets very often or often. This is presumable due to the fact, that they only use their memory for remembering their passwords and have thus no other option than doing a reset.

8.1.4 Password’s security

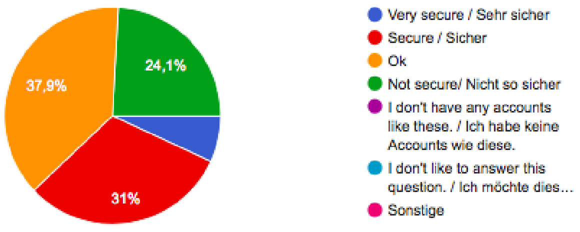
Keyfindings from Question 5 and 6:

- Missing option ’Not so secure’ in NOT top secret accounts

- About 38% find their passwords 'very secure' or 'secure' for their NOT top secret accounts
- About 48% find their passwords 'very secure' or 'secure' for their top secret accounts
- Almost the same amount in both cases (20,6% and 24,1%) think their passwords are 'not secure'
- 3 people find their passwords for top secret accounts less secure than their passwords for NOT top secret accounts
- 6 people find their passwords for top secret accounts more secure than their passwords for NOT top secret accounts
- 20 people find their passwords for top secret accounts as secure as their passwords for NOT top secret accounts
- One Girl said, she doesn't know what 'secure' means



(a) Top secret accounts: Security



(b) NOT top secret accounts: Security

Unfortunately, the option 'not so secure' got somehow lost, while preparing the questionnaire. Nevertheless, the pie charts give us an insight into the security of the humans' passwords. Only a few more people rated their passwords for top secret accounts more secure than their passwords for NOT top secret accounts. Most of all, the distribution of the two pie charts seems about the same. But we encountered some problems about this question. On the one hand, it is hard to say which accounts are 'top secret' and which ones are 'NOT top secret'. For instance, a social network was listed as 'NOT top secret', although it also contains personal information. Moreover, one girl arose the question "What does security actually mean?". Due to the fact that 3 people found their passwords for top secret accounts less secure than their passwords for NOT top secret accounts, she was obviously not the only one struggling with this question.

8.1.5 How to generate a password and where to safe it?

Keyfindings from Question 5.1. and 6.1.:

- The same for top secret accounts and NOT top secret accounts

- Only one person chooses his passwords with a password generator
- All others use their own inventions for passwords

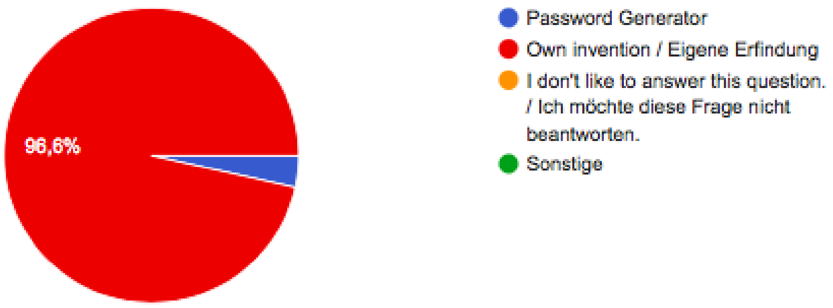


Figure 6: Method used for password choice.

Keyfindings from Question 5.6. and 6.6.:

- People use the same method in both cases to safe their passwords
- Most people like to keep their passwords only in memory
- Only 4 people use a technical device to save their passwords

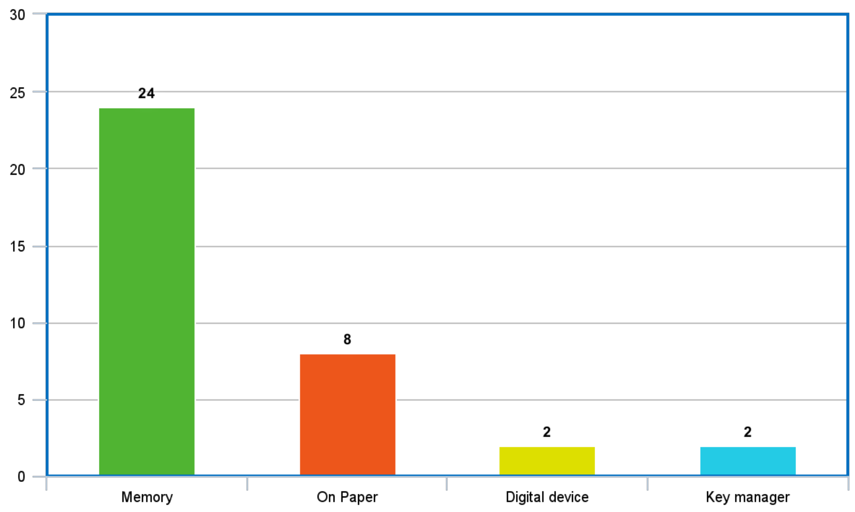


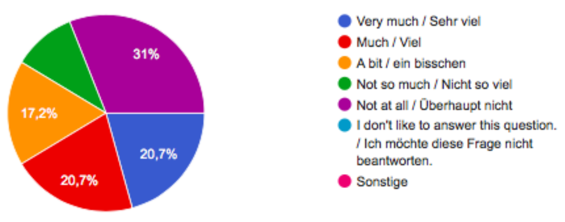
Figure 7: Password safe

We can see that the two questions ”How are you generating your passwords?” and ”Where are you saving your passwords?” stand in a close relation. If people are not using additional help to safe their passwords besides keeping them in mind, they surely can’t use very complicated passwords produced by a password generator. And yet, the majority of the people does not use additonal help, possibly because they are uninformed and are too lazy to get informed. Generally people also like to stuck with old habits, especially if it is a topic which you don't discuss with the people in your social environment and which they think is not presented to the outside in another way. Passwords are quite a taboo issue to talk about in public. But if people don't exchange opinions about an issue, they don't feel a need to keep up with others in this field. Furthermore, it can also be about a trust issue using digital device for saving their passwords.

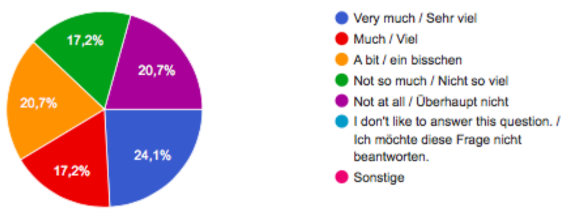
8.1.6 Passwords with personality

Keyfindings from Question 5.2. and 6.2.:

- In both cases people choose passwords quite much related to their personality (40-60%)
- More people stated that their passwords have nothing in common with their personality for top secret accounts (31%) than for NOT top secret accounts (20,7%)
- If we combine the option 'not so much' with 'not at all' in both cases, the pie diagrams are about the same



(a) Top secret accounts: Relation between passwords and personality.



(b) NOT top secret accounts: Relation between passwords and personality.

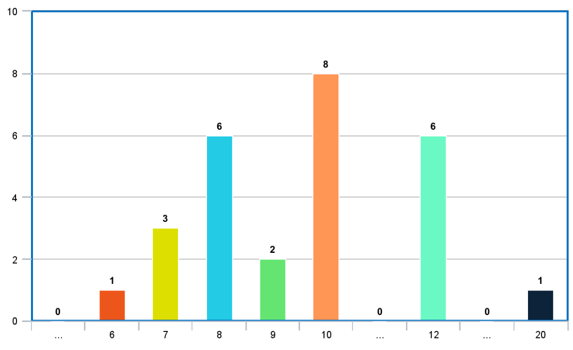
Although the pie diagrams don't show a big difference between personality in passwords for top secret accounts and NOT top secret accounts, the pie diagram reveals us a remarkable number of passwords related to the people's personality. Up to 60% stated that their passwords are have 'very much', 'much' or 'a bit' in common with their personality. Even 5 people judged their passwords as 'very secure' or 'secure' and stated at the same time that their passwords have 'much' or 'very much' in common with their personality. Apparently, they don't feel threatened by attackers.

8.1.7 Password lengths and password's consistency

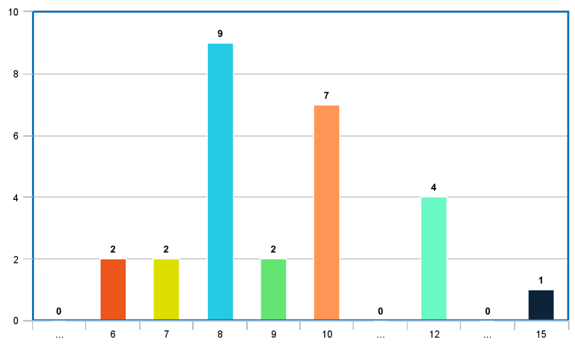
Keyfindings from Question 5.3. and 6.3.:

- Most people use a length between 8 and 12 for top secret accounts

- Most people use a length between 8 and 10 for NOT top secret accounts
- Only one person uses a length bigger than 12 in both cases



(c) Top secret accounts: Password length

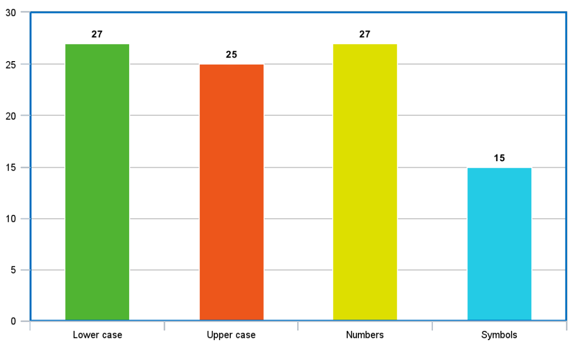


(d) NOT top secret accounts: Password length

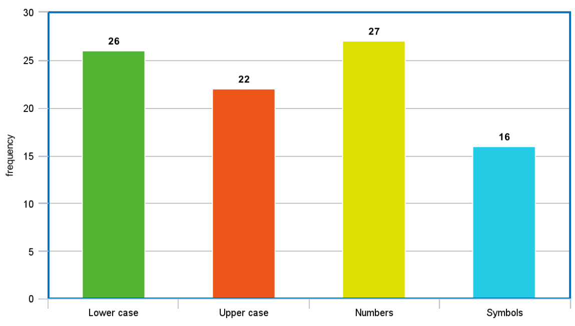
Certainly, also the password length is connected to people choice of how they safe their passwords. The majority of the people, with password lengths of 12 or bigger, use a key manager or paper for saving their passwords. All others rather use shorter passwords. We can observe a little tendency to longer passwords for top secret accounts.

Keyfindings from Question 5.4. and 6.4.:

- Distribution about the same in both cases
- Symbols are almost only used of every second person, whereas lower case, upper case and numbers are almost always used



(e) Top secret accounts: Consistency of the password



(f) NOT top secret accounts: Consistency of the password

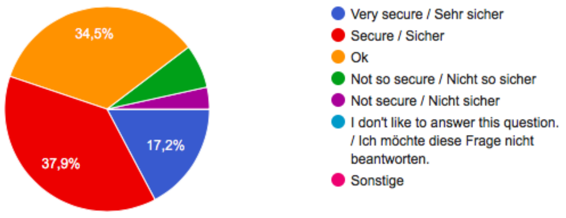
Since symbols are not involved in our everyday language, they are also not always used, when choosing a password. Nevertheless, about 50% of the people reported their passwords include one or more symbols. Somehow, it seems to be rooted in the general knowledge of the majority that it is important to use symbols. Unfortunately, this fact seems to be one of the exceptions people are aware about.

8.1.8 How about our generator method?

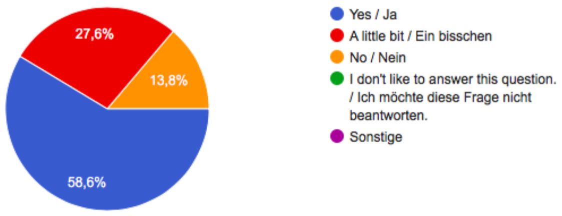
We will now analyze some answers about our procedure of creating new passwords as mnemonics and applying some operations on those.

Keyfindings from Question 7.1. and 8.1.:

- Over 50% judge a mnemonic of length 8 as 'very secure' or 'secure'
- Only about 10% judge it as 'not secure' or 'not so secure '
- People mostly think security increases with additional operations, only 13,8% think it doesn't



(g) How secure is a mnemonic of length 8?



(h) Does security increase with additional operations?

Again, we encounter the question "What does security mean?". Almost 90% of our questionees judged a mnemonic of length 8 as 'very secure', 'secure' or 'ok'. In the first place, a mnemonic looks quite random, although certain patterns can be observed (like a dot, exclamation mark or question mark at the end). And it might be quite secure, if attackers are not considering any of such methods. Surely, it is definitely more secure than passwords, which include personal information of the corresponding persons.

Keyfindings from Question 7.2.:

- Split opinions: 50% would not use the method to generate passwords
- About 40% would use is
- One person would only use it for NOT top secret accounts (male, 11-15)
- Another one would only use it for top secret accounts (male, 26-35)
- Two people didn't answer this question

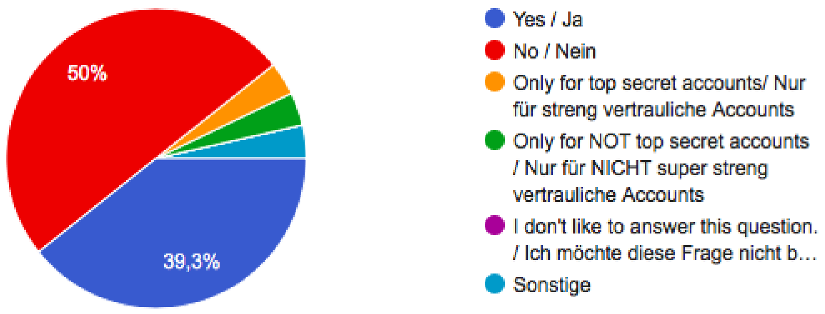
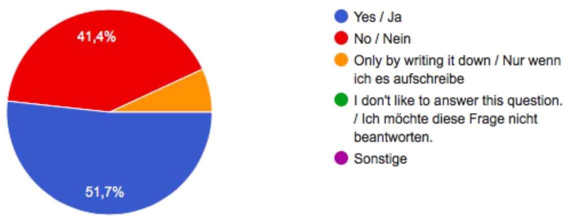


Figure 8: Would you use such a method?

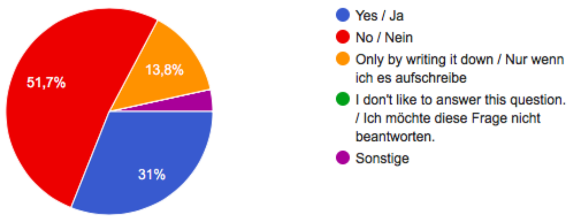
Surprisingly many people stated, they would like to use a mnemonics to generate their passwords. It is surprising, because previous result presume that most of the questionees are not using such a method, since their passwords show a close relation to their personality. One explanation could be, that they were not aware of such a method.

Keyfindings from Question 7.3. and 8.2..:

- About 50% of the people think, they can remember a mnemonic over time
- About 30% of the people think, they can still remember it with additional operations
- One person answer to question 8.3. with "Yes, but not too many"



(a) Mnemonic: Can you remember such a password over time?



(b) With additional operations: Can you remember such a password over time?

First of all, the option 'Only by writing it down' can be combined with the option 'No'. Definitely more people (51,7%) think they can remember a mnemonic without operations over time than a mnemonic with operations (31,7%). The reason is comprehensible, since they need to remember even more. It is questionable how usable such a method with operations is in everyday use and if it can be used without any additional help, like a key manager, to memorize these passwords.

8.2 Survey questions

In the following, we present the whole survey, how it was given to the people:

Password Security

In this survey we want to find out more about the humans choice of passwords, in case to build a password generator which is fitted to humans needs and which is still offering enough security. All collected data will be treated anonymously, not used for any criminal manipulations and not given to any other second parties. We thank you in advance for joining our survey. You will maximally need to answer 27 questions, which will approximately take about 10-15 minutes. Since we don't want you to lie about answers, we are always offering you the choice "I don't like to answer this question". If there is no choice question, write down "No answer" in this case.

Basic Questions

1. How many active passwords do you have approximately?

------------------

2. How many active accounts do you have approximately?

------------------

3. Are you using passwords on active accounts, which you already used on currently inactive accounts?

O Yes

O No

O I don't like to answer this question

O Other:

4. How often are you resetting your password, because you forgot about it?

O Very often/ Sehr oft

O Often/ Oft

O Sometimes/ Manchmal

O Not so often/ Nicht so oft

O Never/ Nie

O I don't like to answer this question.

O Other:

Specific passwords

5. How secure do you judge your passwords for pages storing top secret information (e.g. email accounts)?

O Very secure

O Secure

O Ok

O Not so secure

O Not secure

O I don't have any accounts like these. (Go to 6)

O I don't like to answer this question.

O Other:

Specific passwords continued

5.1. Which methods are you using to choose such passwords?

O Password Generator

O Own invention

O I don't like to answer this question.

O Other:

5.2. How much are such passwords related to your personality (e.g. your favorite song or first name)?

O Very much

O Much

O A bit

O Not so much

O Not at all

O I don't like to answer this question.

O Other:

5.3. Which length do your passwords have in general for pages like these?

5.4. Do your passwords of such kind include . . .

O . . .lower case letters?

O . . .upper case letters?

O . . .numbers?

O . . .symbols (like *, $, :, ...)?

O I don't like to answer this question.

5.5. What is the highest number of times you are reusing such a password for different accounts?

5.6. Where are you saving such passwords (memory, paper, digital device, key manager)?

O Memory

O On Paper

O Digital device

O Key manager

O I don't like to answer this question.

Specific passwords

6. How secure do you judge your passwords for pages storing NOT super top secret information (e.g. newspaper accounts, social network)?

O Very secure

O Secure

O Ok

O Not secure

O I don't have any accounts like these. (Go to 7)

O I don't like to answer this question.

O Other:

Specific passwords continued

6.1. Which methods are you using to choose such passwords?

O Password Generator

O Own invention

O I don't like to answer this question.

O Other:

6.2. How much are such passwords related to your personality (e.g. your favorite song or first name)?

O Very much

O Much

O A bit

O Not so much

O Not at all

O I don't like to answer this question.

O Other:

6.3. Which length do your passwords have in general for pages like these?

6.4. Do your passwords of such kind include . . .

O . . .lower case letters?

O . . .upper case letters?

O . . .numbers?

O . . .symbols (like *, $, :, ...)?

O I don't like to answer this question.

6.5. What is the highest number of times you are reusing such a password for different accounts?

6.6. Where are you saving such passwords (memory, paper, digital device, key manager)?

O Memory

O On Paper

O Digital device

O Key manager

O I don't like to answer this question.

Password Generator

7. Image the following: You want to generate a password for a new account. To do so you grab your favorite book and choose an arbitrary sentence out of it. Now you write down all the first letters and all symbols as they are ordered in the sentence (e.g. "He went to the beach!" becomes "Hwttb!")

7.1. How secure would you judge a password like this with a length of 8 characters?

O Very secure

O Secure

O Ok

O Not so secure

O Not secure

O I don't like to answer this question.

O Other:

7.2. Would you use a method like this to generate your passwords?

O Yes

O No

O Only for top secret accounts

O Only for NOT top secret accounts

O I don't like to answer this question.

O Other:

7.3. Do you think you can remember a password like this in memory over time?

O Yes

O No

O Only by writing it down

O I don't like to answer this question.

O Other:

Password Generator extended

8. Now, you are doing the following: You take your generated password from before and apply some operations to it. For example you cut the password in half and swap the two halfs. Then you are substituting some letters with similar symbols, e.g. the letter a becomes @ and the letter e becomes 3.

8.1. Do you think this method makes the password even more secure?

O Yes

O A little bit

O No

O I don't like to answer this question.

O Other:

8.2. Do you think you can remember a password like this in memory over time?

O Yes

O No

O Only by writing it down

O I don't like to answer this question.

O Other:

Personal questions

9. How old are you?

O 0-10

O 11-15

O 16-25

O 26-35

O 36-45

O 46-55

O 56-65

O 66- or older

O I don't like to answer this question.

10. What is your gender?

O male

O female

O I don't like to answer this question.

O Other:

11. Which of the following applies to you?

O Yes

O No

O I did/do a school degree

O I did/ do a university degree

O I did/do a job training

O I don't like to answer this question.

12. How familiar are you with computers, the www, ...?

O Very familiar

O Familiar

O Ok

O Not so familiar

O Not familiar

O I don't like to answer this question.

O Sonstiges:

13. Remarks: ____

9 Conclusion

Looking at several statistics, it became obvious to us that there exists a large deficit of choosing effective passwords. So we started with an idea of a generator, which outputs passwords fulfilling two essential properties: memorability and security. To provide a certain security, the passwords needed to look random or hard to guess respectively. Additionally, they should underlie a system to provide memorability. Eventually, we came up with generating mnemonics, which ones we judged as quite easy to remember. Since these passwords are not that hard to crack, if the attacker knows about the used technique, we needed to bring in some additional security. The discussed operations could fulfill this task.

If people are not using additional help to safe their passwords (e.g. key managers), generating passwords will always be a pondering between memorarization and security. Since the majority, as we could see it in the survey, is not using additional help, we can provide them with an useful idea to generate their passwords, which doesn't necessarily need to be saved additionally on paper, in a key manager or somewhere else. Our survey showed that about 40% of the questionees would use a mnemonic as a password and about 50 % think they can remember such a method. About 30% answered they can also remember a mnemonic with additional operations. Nevertheless, the procedure of typing in a mnemonic with additional operations requires a lot of concentration, since one needs to do all of those operations in mind while typing. Doing so, it will take quite a while and also mistakes can happen quickly. Both things can lead to demotivation.

Our method might not be the best one, but it is a method which can keep up with other common methods, e.g making up a sentence without sense, like "The donkey is under the sunny battery having some chicken cocktails" using this as a password (maybe without whitespaces). This method would provide security, because on the one hand it is a very long sentence and on the other hand, attackers would rather guess sentences which make sense. It also provides memorability, since people's mind is made to remember strange, striking or suprising things better than everyday things.

Looking for methods, which can fulfill all our needs, we should maybe turn the perspective. In other words, if people got more informed about technical help to generate and save theirs passwords, we would not need to concentrate that much on memorability any more. But nowadays, these options seem not user-friendly or not present enough. And even with technical help, we will still need to keep some passwords only in memory or on paper. First of all, also key managers need a password to enable access. Secondly, people might not have a digital device with them in some situations to look up their password, e.g. if they have a password-locked frontdoor to their house and forgot their mobile phone.

Until now, it remains an open question which method is the right one to use. And this might not be too bad. Because if everyone is using the same method... Can such a method provide security over time?

References

[1] http://www.gutenberg.org/cache/epub/35/pg35-images.html

[2] http://research.microsoft.com/pubs/74164/www2007.pdf

[3] http://wayback.archive.org/web/20040712152833/http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63v6_3_3.pdf

[4] http://www.jbonneau.com/doc/2012-jbonneau-phd_thesis.pdf

[5] http://blog.jimmyr.com/Password_analysis_of_databases_that_were_hacked_28_2009.php

[6] http://psychclassics.yorku.ca/Miller/

[7] www.rockyou.com

[8] https://sec.hpi.uni-potsdam.de/leak-checker/statistics

[9] https://sec.hpi.de/leak-checker/search

[10] https://www.google.com/intl/de_de/forms/about/

[11] https://www.meta-chart.com/pie#/display

10 Attachments

```
package project_pass_sec;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Generator {


public static long a=0;
public static int z=0;

//method: for replacing characters with specific character

public static String rand_replacement(String s){

    int m=s.length();
    int n=0;

    while(n<m){

    Random random = new Random();
    int index = random.nextInt(s.length());

    //System.out.println(index);
   // return s.charAt(index);

    if (s.charAt(index)=='a')
    {
     char[] myNameChars = s.toCharArray();
     myNameChars[index] = '@';
     s = String.valueOf(myNameChars);
    }

    else if(s.charAt(index)=='t'||s.charAt(index)=='T')
    {
     char[] myNameChars = s.toCharArray();
     myNameChars[index] = '7';
```

```
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='e')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '3';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='g'||s.charAt(index)=='B')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '8';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='i'||s.charAt(index)=='I')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '1';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='x'||s.charAt(index)=='X')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '*';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='O'||s.charAt(index)=='o')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '0';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='b')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '6';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='y')
```

```
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '4';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='i')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '!';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='i')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '1';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='S'||s.charAt(index)=='s')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '$';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='s')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '5';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='V'||s.charAt(index)=='v')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '^';
 s = String.valueOf(myNameChars);
}

else if(s.charAt(index)=='R'||s.charAt(index)=='W'||s.charAt(index)=='P')
{
 char[] myNameChars = s.toCharArray();
 myNameChars[index] = '&';
 s = String.valueOf(myNameChars);
```

```
    }
   ++n;
   }

    return s;
}


 private static String readUsingScanner(String fileName) throws IOException {

Scanner scanner = new Scanner(Paths.get(fileName), StandardCharsets.UTF_8.name());
        //we can use Delimiter regex as "\\A", "\\Z" or "\\z"
        String data = scanner.useDelimiter("\\A").next();
        scanner.close();
        return data;
    }


public static void main(String[] args) throws IOException{

String str = "abcdef"; // for testing purpose
String[] uniStr;
System.out.println("Started....wait & see all generated files......");
String file_name = "C:/Users/SharfUddin/workspace/pass_sec/try1.txt";

String new_file = readUsingScanner(file_name); // method calling to read file

        /*PrintWriter printWriter = new PrintWriter ("C:/Users/SharfUddin/workspace
        printWriter.println (new_file);
        printWriter.close ();*/

 //String input = "verylo"; // for testing purpose
 String newSt1,newSt2,newSt3;
 //String s1,s2;
         newSt1=rand_replacement(new_file);
         PrintWriter printWriter1 = new PrintWriter ("C:/Users/SharfUddin/workspace
         printWriter1.println (newSt1);
         printWriter1.close ();
         newSt2=rand_replacement(new_file);
         PrintWriter printWriter2 = new PrintWriter ("C:/Users/SharfUddin/workspace
         printWriter2.println (newSt2);
         printWriter2.close ();
         newSt3=rand_replacement(new_file);
         PrintWriter printWriter3 = new PrintWriter ("C:/Users/SharfUddin/workspace
         printWriter3.println (newSt3);
         printWriter3.close ();
         //System.out.println(newSt1+" "+newSt2+" "+newSt3);
         int j,m;

         for(int i=1;i<=3;i++)
         {
          switch(i)
          {
          case 1:

         String st1 = newSt1;
          //String[] words1 = st1.split(" ");
         String[] words1 = st1.split("\\s+");
         for (j = 0; j < words1.length; j++) {

              words1[j] = words1[j].replaceAll(" ", "");
              //System.out.print(words1[j]+" ");
          }
         //System.out.println();

         m=j;

          for(j=0;j<m;j++){
          int l=words1[j].length();
          if(l>=6)

          {
          //System.out.println(words1[j]);

              uniStr= uniform(words1[j]);
              permute(uniStr, uniStr.length);
              uniStr=uniform(words1[j]);
              permute(uniStr, uniStr.length);
                      //System.out.println("------------------");
}
          }

          break;



          case 2:

         String st2 = newSt2;
          String[] words2 = st2.split("\\s+");
         for (j = 0; j < words2.length; j++) {
              words2[j] = words2[j].replaceAll(" ", "");
             // System.out.print(words2[j]+" ");
          }
         //System.out.println();

         m=j;

          for(j=0;j<m;j++){
          int l=words2[j].length();
          if(l>=6)
          {
//System.out.println(words2[j]);
uniStr= uniform(words2[j]);
permute(uniStr, uniStr.length);
uniStr=uniform(words2[j]);
permute(uniStr, uniStr.length);
// System.out.println("------------------");

}
          }

              break;


          case 3:

         String st3 = newSt3;
          String[] words3 = st3.split("\\s+");
         for (j = 0; j < words3.length; j++) {

              words3[j] = words3[j].replaceAll(" ", "");
              //System.out.print(words3[j]+" ");
          }
         //System.out.println();

         m=j;

          for(j=0;j<m;j++){
          int l=words3[j].length();
          if(l>=6)

          {
          //System.out.println(words3[j]);

               uniStr= uniform(words3[j]);
               permute(uniStr, uniStr.length);
               uniStr=uniform(words3[j]);
               permute(uniStr, uniStr.length);
              //System.out.println("------------------");
}
          }

              break;
          }
          //System.out.println("finished !!!!!!!!!");


         }
         System.out.println("Total generated strings/pass="+" "+a);
         System.out.println("successfully finished !!!!!!!!!");

}


public static String[] uniform(String words3){

//String[]
String s1,s2;
String[] erg;
z++;
if(z<2){

s1=words3.substring(0, 3);
//System.out.println(s1);
         s2 = words3.substring(3);
         //System.out.println(s2);
 String[] array31={s1};
      String[] array33=s2.split("");
      //String[] erg1=result.split("");
      erg = new String[array31.length + array33.length];
      System.arraycopy(array31, 0, erg, 0, array31.length);
      System.arraycopy(array33, 0, erg, array31.length, array33.length);
}

else{
s1=words3.substring(0, words3.length() - 3 );
s2 = words3.substring(words3.length() - 3);
//System.out.println(s1);
        //System.out.println(s2);
 String[] array31=s1.split("");
     String[] array33={s2};
     //String[] erg1=result.split("");
     erg = new String[array31.length + array33.length];
     System.arraycopy(array31, 0, erg, 0, array31.length);
     System.arraycopy(array33, 0, erg, array31.length, array33.length);
     z=0;
}
return erg;
}




private static void swap(String[] ourarray, int right, int left) {

        String temp = ourarray[right];
        ourarray[right] = ourarray[left];
        ourarray[left] = temp;
    }

    public static void permute(String[] ourArray, int currentPosition) {

        if (currentPosition == 1) {

         a++;
            //System.out.println(a+" "+" "+Arrays.toString(ourArray));
            String str1 = Arrays.toString(ourArray);
            str1=str1.substring(1, str1.length()-1).replaceAll(", ","");
            //System.out.println(str1);
            //System.out.println(a+" "+" "+str1);

            File file = new File("C:/Users/SharfUddin/workspace/pass_sec/final_file
         try{
             //System.out.println(file.createNewFile());
         file.createNewFile();
             PrintWriter pw = new PrintWriter(new FileOutputStream(file,true));
             //pw.print(a+"."+str1+" ");
             pw.print(str1+" ");
             pw.flush();
             pw.close();
         }catch(IOException ex){
             ex.printStackTrace();
         }

        } else {
            for (int i = 0; i < currentPosition; i++) {
                // subtract one from the last position (here is where you are
                // selecting the the next last item
                permute(ourArray, currentPosition - 1);

                // if it's odd position
                if (currentPosition % 2 == 1) {
                    swap(ourArray, 0, currentPosition - 1);
                } else {
                    swap(ourArray, i, currentPosition - 1);
                }
            }
        }
    }
}
```