Investigating the Relationship Between Password Distribution and Zipf's Law

David Malone, Kevin Maher
Hamilton Institute,
NUI Maynooth.

January 27, 2011

Abstract

In this paper we want to look at the distribution with which passwords are chosen. Zipf's Law is commonly observed in lists of chosen words. Using password lists from four different on-line sources, we will investigate if Zipf's law is a good candidate for describing the frequency with which passwords are chosen. We look at a number of standard statistics, used to measure the security of password distributions, and see if modeling the data using Zipf's Law produces good estimates of these statistics. Finally, we assess how similar the common passwords in different instances of these password lists are by using guessing as a metric.

1 Introduction

In this paper we investigate if password frequency distribution can be modelled by Zipf's Law. Zipf's Law is a probability distribution where the frequency of an event is inversely proportional to its rank on a frequency table. Here the rank of the most common event is 1, the rank of the second most common is 2, and so on. Zipf's Law has been observed when looking at the frequencies with which words are used in natural language. In our case, the events will be the use of a particular password by a user. To study this, we use lists of users and passwords from hotmail.com, flirtlife.de, computerbits.ie and rockyou.com. In each case, the list of usernames and passwords were made public after a security incident. The lists have between 1800 users to 32 million users.

There are a number of substantial differences between the choice of passwords and the use of words in natural language. In particular, passwords are usually chosen to be hard to guess and there is a large literature of advice on how to choose passwords (e.g. [7]). However, there are many reasons for not choosing a password according to the recommendations: the advice is arduous, it takes a long time to produce a recommended password, the resulting password is likely to be a hash of letters and numbers which will be hard to memorise. It has even been argued that in general, the cost of choosing one of these passwords is far greater than the cost of loosing the information it is trying to protect [4].

Thus, rather than uniformly choosing from a list of non-dictionary words the average user is likely to choose a password which they can easily remember, such as their name, city where they live, favourite team and so on which leads to certain passwords being used more frequently than others. Since Zipf's Law has been observed in many empirical data sets, we would like to investigate if it provides a reasonable model for the passwords we see.

Seeing Zipf's Law, or any other distribution that is skewed in favour of a number of passwords, has implications for security. If the right distribution of passwords can be identified, the cost of guessing a password can be reduced. One naturally expects that, say, the demographic of users of a site could be used to target an attack; a website with a .ie domain is more likely to have Irish themed passwords than a site with a .fr domain.

Zipf's law tells us that the number of occurrences of something is inversely proportional to its rank on a frequency table, $y \sim r^{-s}$, where s is a parameter which is close to 1. By plotting our datasets and fitting a value for s, we will see if a Zipf distribution provides a good model. Then,

Site	#users	#pass	$\frac{\#pass}{\#users}$
hotmail	7300	6670	0.91
flirtlife	98930	43936	0.44
computerbits	1795	1656	0.92
rockyou	32603043	14344386	0.44

Table 1: Number of users and number of distinct passwords seen for each site.

using the metrics as the guesswork [8] and Shannon entropy, we will see if these models provide good predictions of these statistics.

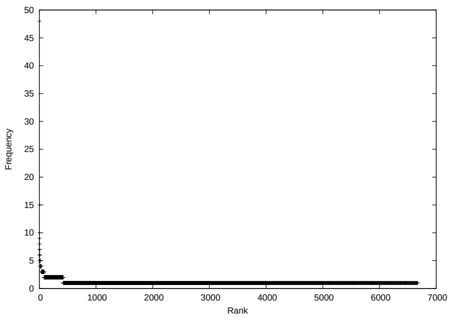
We will also compare the similarity of our different data sets using guessing as a metric. We will show that by using common passwords from one list, a speed up can be obtained when guessing the passwords from another list.

2 Overview of Datasets

We collected sets of passwords belonging to sites which were previously hacked and the lists of passwords leaked. Since the sets were gathered by different methods, such as key-logging, network sniffing or database dumps, the lists may only contain a random samples of users. Our lists are from the 2009 hotmail.com list, the 2006 flirtlife.de list, the 2009 computerbits.ie list and the 2009 rockyou.com list.

Some of the lists also give multiple passwords for a small number of users. In this case, we cleaned up the sets by taking the user's password as the last entry seen for that user, which would hopefully correspond to a user initially typing the wrong password and then typing the correct one, or in the case that the password was changed, the most recent password. We also omitted any user with a whitespace password. After the data was cleaned up, we produced a table ranking passwords in order of decreasing frequency of use by users. Table 1 shows the number of users and the number of distinct passwords for each set of data. As is obvious from the table, for smaller lists there are relatively more unique passwords.

Table 2 summarises the top 10 passwords in each list. We see that passwords such as **123456** and **password** are very common. The most common password, "123456" accounts for 0.7% of the total passwords in the hotmail data, 3.3% in the flirtlife list and 2.0% in the rockyou list; "password" accounts for 1.2% of the total passwords of the computerbits list. This indicates that the password distribution is skewed in favour of some common passwords.



(a) hotmail

Figure 3.1: Plot of rank vs frequency on a linear scale

The demographic of the users from each list is quite clear form the first 10 passwords of the lists with each ccTLD. Note that the hotmail.com data is believed to have been collected with phishing targeted at the Latino community. The flirtlife list shows clear signs of users speaking German and Turkish, and the computerbits list contains place names of Irish interest. If we look at the data from computerbits and rockyou we see that the name of the website appears in the top ten of each list. It seems likely that this method for choosing a password will also be used on other sites.

By taking obvious passwords and knowing the demographic of the users at which the site is aimed one could build a comprehensive dictionary that could be expected to cover the most common passwords in use at a site. This implies that users, if they are concerned about their accounts being hacked, should use less common passwords. Even something as simple as changing some of the characters to upper-case or writing the word in 'leet speak', could be expected to move the password out of the most-common list.

3 Distribution of Passwords

In this section we will look at how passwords are distributed in our lists, and see how well these distributions match a Zipf model. We will be interested in the frequency f_i with which we see the i^{th} most popular password. Where passwords are seen equal numbers of times, we break

Rank	hotmail	#users	flirtlife	#users	computerbits	#users	rockyou	#users
1	123456	48	123456	1432	password	20	123456	290729
2	123456789	15	ficken	407	computerbits	10	12345	79076
3	111111	10	12345	365	123456	7	123456789	76789
4	12345678	9	hallo	348	dublin	6	password	59462
5	tequiero	8	123456789	258	letmein	5	iloveyou	49952
6	000000	7	schatz	230	qwerty	4	princess	33291
7	alejandro	7	12345678	223	ireland	4	1234567	21725
8	sebastian	6	daniel	185	1234567	3	rockyou	20901
9	estrella	6	1234	175	liverpool	3	12345678	20553
10	1234567	6	askim	171	munster	3	abc123	16648

Table 2: Top 10 Passwords for each list

the tie randomly.

Fig. 3.1 shows the rank vs. frequency of our data plotted on a linear scale for the hotmail list. There are a small number of passwords with a high frequency, and many passwords with a frequency of 1 or 2, which makes the graphs hard to read. Instead, we plot the frequency versus rank on a log-log scale in Figure 3.2. These graphs certainly show evidence of heavy-tailed behaviour, with the frequency dropping much more slowly than exponential. A Zipf distribution would appear as a straight line on a log-log plot, where the parameter s is the negative of the slope.

If we fit a least-squares line to this data, as shown in Figure 3.2, we get a slope which is too shallow because a large fraction of the points have frequency 1 or 2, which biases the slope towards 0. To account for this, we follow the method in [1] and bin the data logarithmically, as shown in Figure 3.3. Here, we sum the frequency of all ranks between 2^n and $2^{n+1}-1$. We see that this gives us a slope which better fits our data, and that the line appears a relatively good fit. We use this binned slope as a basis for modeling our data with a Zipf distribution.

An alternate way to view the data is to look at the number of passwords n_k that are each used by exactly k users. We plot this in Figure 3.4 on a log-log scale. As explained in [1], if the data is Zipf distributed, we expect this graph to also be a straight line with slope $-(1 + 1/s)$. As we can see, we also need to bin this data before fitting a line, and the result it shown in Figure 3.5. Again, we see a line is a relatively good fit, with the largest discrepancy appearing for computerbits, the smallest list. The resulting slopes are summarised in Table 3.

	hotmail	flirtlife	c-bits	rockyou
s raw	0.16	0.64	0.15	0.51
s binned	0.44	0.69	0.45	0.78
$-m$ raw	2.46	1.72	2.56	1.37
$-m$ binned	3.07	2.32	3.10	2.23
$1 + 1/s$ binned	3.27	2.45	3.22	2.28

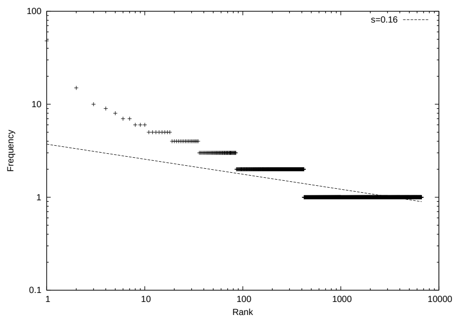
Table 3: Estimated parameters for Zipf distribution and n_k vs. k graphs.

4 Password Statistics

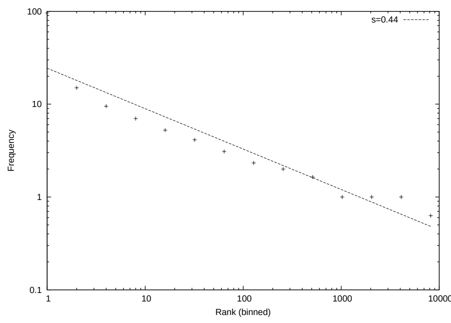
In this section we will look at a number of statistics relevant to passwords that can be derived from the distribution of how passwords are chosen. We will look at these statistics when calculated directly from our lists, and when calculated using two simple models of the lists. For the real lists, we calculate our statistics assuming the probability of the password of rank i appearing is f_i/N, where f_i is the frequency with which we observed that password and N is the total number of passwords observed.

The first model assumes password choices are uniform over all passwords seen, i.e., if the number of passwords is N then a password is chosen with probability $1/N$. The second model assumes that password choices are distributed with a Zipf distribution, i.e., the probability of password with rank i being used is $P_i = Ki^{-s}$, where s is the parameter found in Section 3 and K is a normalising constant.

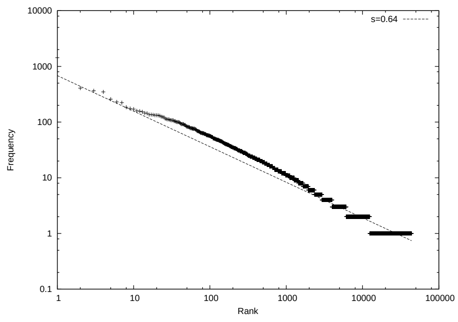
Now let us consider the statistics of interest.



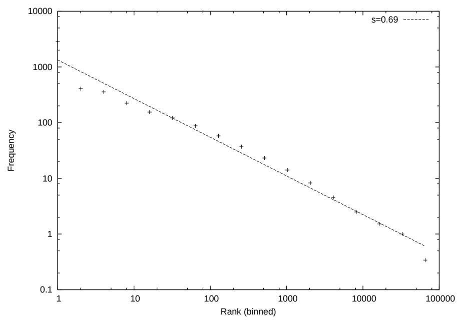
(a) hotmail



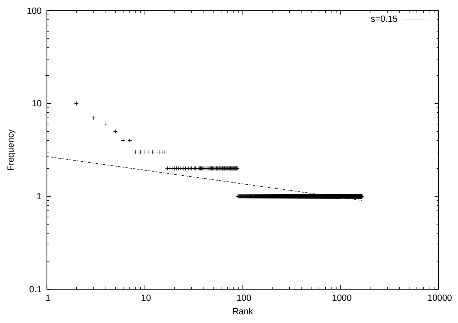
(a) hotmail



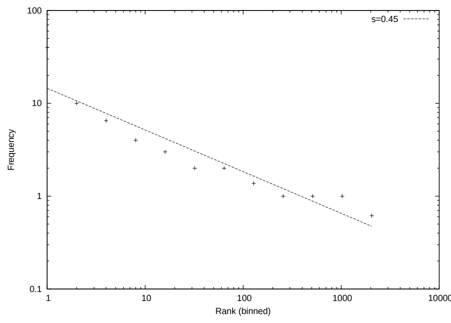
(b) flirtlife



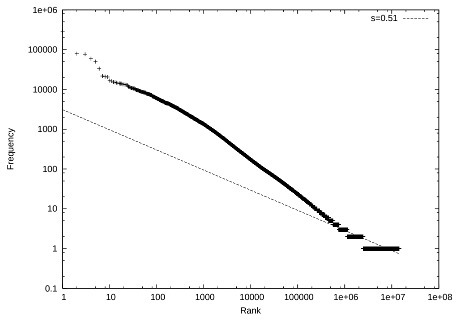
(b) flirtlife



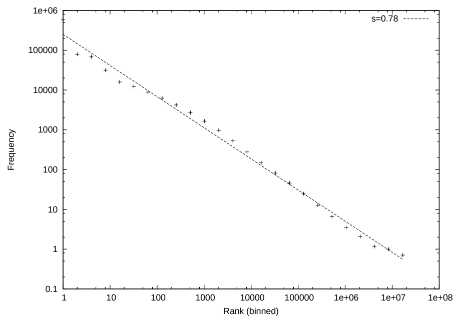
(c) computerbits



(c) computerbits



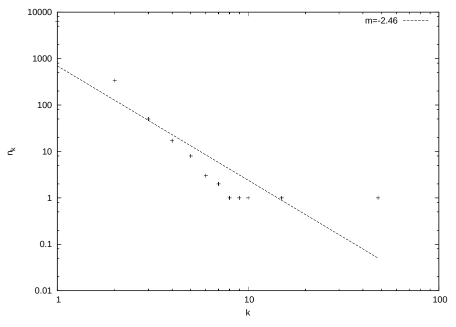
(d) rockyou



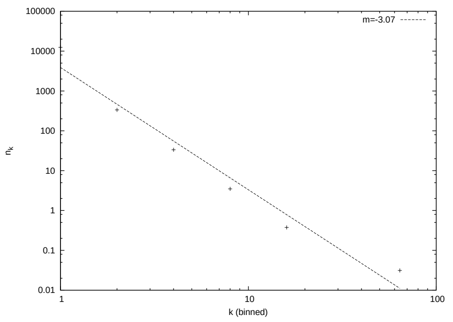
(d) rockyou

Figure 3.2: Plot of plot rank vs. frequency on log-log scale

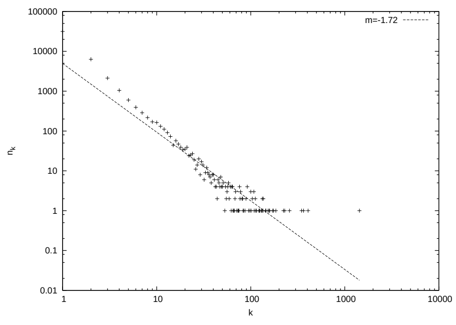
Figure 3.3: Exponentially binned plot of Figure 3.2



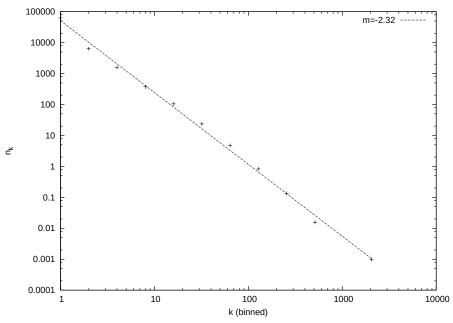
(a) hotmail



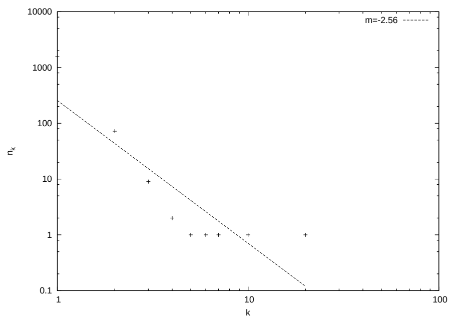
(a) hotmail



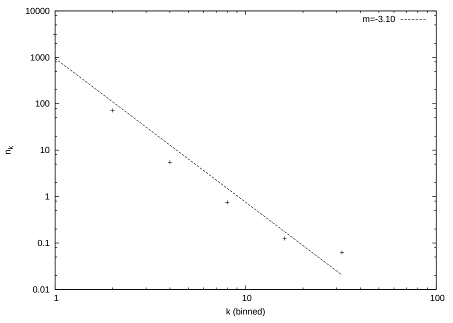
(b) flirtlife



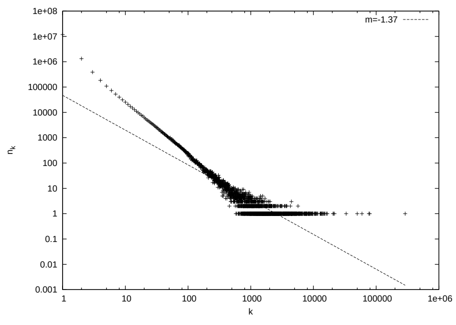
(b) flirtlife



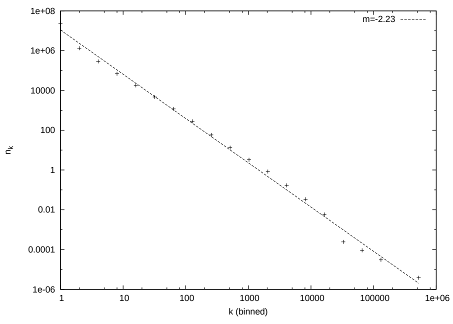
(c) computerbits



(c) computerbits



(d) rockyou



(d) rockyou

Figure 3.4: Plot of k vs. n_k on log-log scale.

Figure 3.5: Plot of k vs. n_k, exponentially binned on log-log scale.

Some of these statistics place considerable emphasis on the tail of the distribution, and can be sensitive to relatively small gaps between the model and the data. The first statistic is the *guesswork*, which is the mean number of guesses needed to correctly guess the password [8], when the password distribution is known, but the exact password is not. Guesswork is given by

$$G = \sum_{i=1}^{N} iP_i,$$

where P_i is the probability of the password of rank i.

Another strategy for guessing passwords, given the distribution, is to try the common passwords, but to give up before when some fraction α of the distribution has been covered. The mean number of guesses associated with this is known as the α-guesswork, G_α [8]. Its value is given by

$$G_\alpha = \sum_{i=1}^{r_\alpha} iP_i,$$

where r_α is the rank of the password when the cumulative probability of being successful is at least α. We will work with $\alpha = 0.85$, so that we cover most of the distribution, but avoid the tail.

The Shannon Entropy,

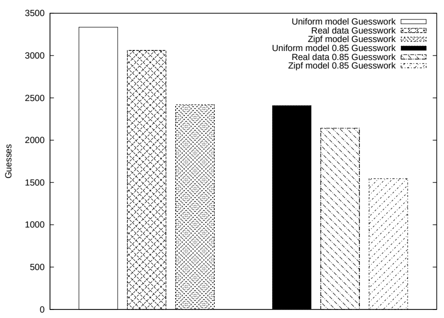
$$H = -\sum P_i \log_2 P_i,$$

is a common measure of the number of bits of uncertainty associated with a random variable. While Shannon Entropy has been used as a measure of security of password and key distributions, it does not relate directly to how easy it is to guess a password [6, 5]. Renyi entropy, which is a generalization of Shannon entropy, is given by

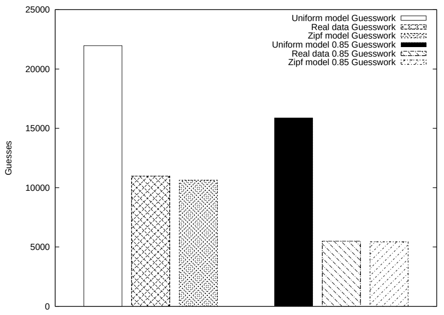
$$R = \log_2(\sum \sqrt{P_i})^2.$$

It is asymptotically related to the guessability of a password. The min-Entropy is also used as a conservative measure of password/key security [3].

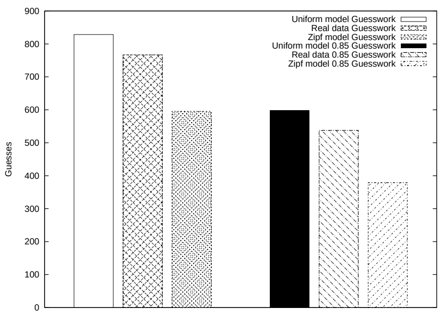
Figure 4.1 compares the guesswork statistics for the uniform model, the real data and the Zipf model. The bars on the left show the guesswork and the bars on the right show the 0.85-guesswork. As expected, the guesswork estimates for the uniform model overestimate the



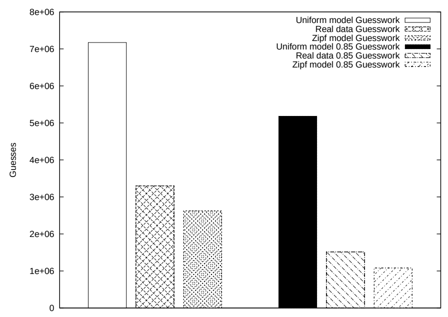
(a) hotmail



(b) flirtlife



(c) computerbits



(d) rockyou

Figure 4.1: Guesswork statistics for Uniform model, Real data and Zipf model.

required number of guesses. As only a small percentage of the total number of passwords seen in both the hotmail.com and computerbits.ie list, the passwords, in general, are closer to having uniform probability, so the guesswork is better modelled. We see that when passwords that shared among many users make up a larger percentage of the total passwords, as in the flirtlife.de and rockyou.com list, the guesswork is far lower than the uniform guesswork. The Zipf model seems to underestimate the required guesswork.

Figure 4.2 shows the Entropy values for the actual data and models. Shannon Entropy is shown on the left, Renyi Entropy in the middle and min-entropy on the right. The Uniform model, again as expected, tends to overestimate the Entropy. However, for the Renyi Entropy both models and the data seem to give results that are close together. The Zipf model seems to provide relatively good approximations in all cases.

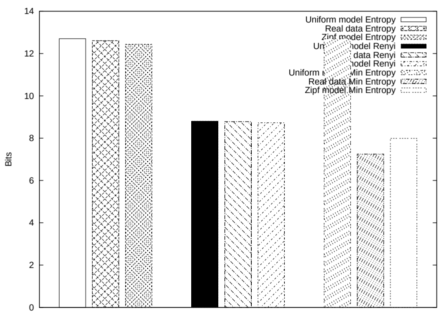
5 The Relation between Distributions

We have seen that while the popular passwords in our lists have things in common (for example, the password 123456), they also show features specific to the website or service. One thing all the lists have in common is a number of relatively frequently used passwords followed by a long tail of uncommon passwords. In this section, we will try to quantify how much we learn about one list, based on the others.

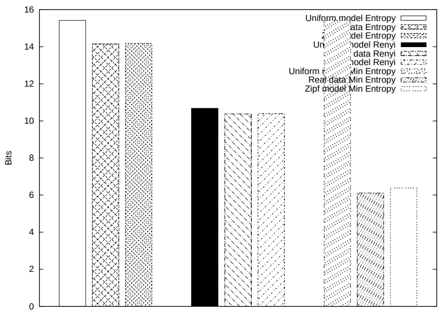
Consider the problem of guessing the password of a randomly selected user from one of our lists. If we guess the passwords from most popular to least popular, then after t guesses we will have guessed the passwords used by

$$C(t) = \sum_{i=1}^{t} f_i,$$

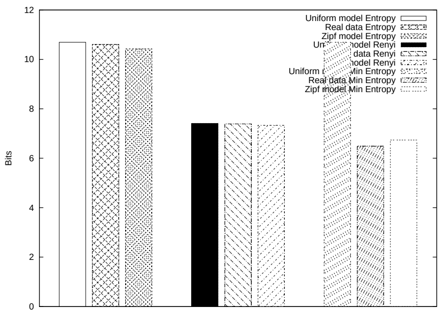
users. If we guess one password at each trial, guessing in this order recovers passwords as quickly as possible, and is in this sense optimal. Figure 5.1 shows $C(t)$ for each of our datasets, as the optimal guessing strategy. The right hand axis shows $C(t)/N$, which we can interpret as the probability of successfully guessing in t guesses or the fraction of users whose passwords have been guessed.



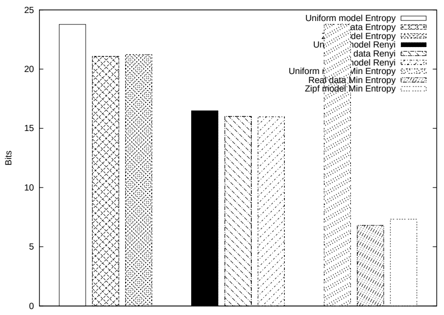
(a) hotmail



(b) flirtlife



(c) computerbits



(d) rockyou

Figure 4.2: Entropy values for Uniform model, Real data and Zipf model.

If we do not know the optimal order in which to guess the passwords, we may instead guess them in the optimal order σ for another reference data set. Now, after t guesses, we will have guessed the passwords of

$$C(t||\sigma) = \sum_{i=1}^{t} f_{\sigma(i)},$$

users, where we assume $f_{\sigma(i)}$ is zero if password i is not in the list we are guessing. If the ordering of the passwords by popularity is the same for both lists, then this function will be $C(t||\sigma) = C(t)$, otherwise $C(t||\sigma) \leq C(t)$.

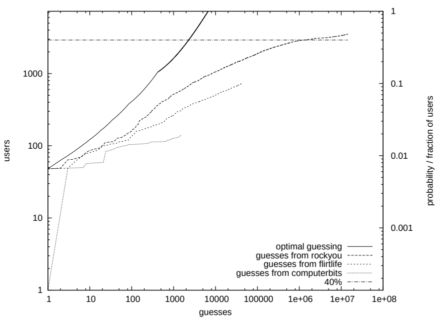
Figure 5.1 shows $C(t||\sigma)$ for each of our lists, when using each of the other lists as a reference. Consider the situation after 1000 trial guesses. The number of users whose passwords match one of these 1000 guesses, $C(1000||\sigma)$, can be seen to vary by almost an order of magnitude, depending on the list used as a reference. Thus, to guess passwords quickly, we would like a good reference list.

Observe that for any list, once we have made more than 10–100 guesses, the larger reference lists lead to more successes than smaller reference lists. This suggests that the impact of demographics is limited to the most popular passwords, and a larger list gives a better ordering of passwords, that cannot be determined from a smaller data set. The rockyou list is quite effective and the top one million rockyou passwords cover close to 40% of users in the other lists.

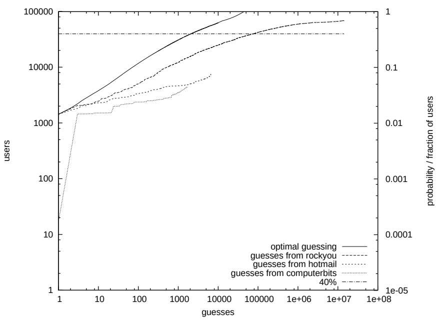
We can apply this directly to a password cracking problem. In December 2010, the password database from Gawker.com was leaked. This database did not contain plaintext passwords, but instead contained hashes of passwords using the well-known DES and Blowfish password hashing schemes. We can use the words in our list as guesses in an off-line cracking attack against the Gawker hashes.

The Gawker dataset contained 748,090 users potentially valid (i.e. 13 character) DES hashes. The hashes use 3844 different salts. A simple perl script can attempt passwords at a rate of approximately 80,000 trials per hour per core on a modern CPU. As the DES hash truncates passwords to 8 characters, we truncate long passwords and reaggregate our previous lists. The number of users whose passwords were cracked after t trials is shown in Figure 5.2.

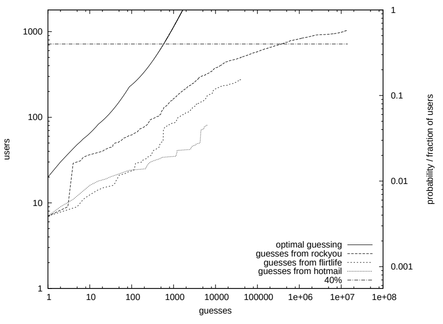
Again, the large lists provide the fastest recovery of passwords, and recovers 40% of users



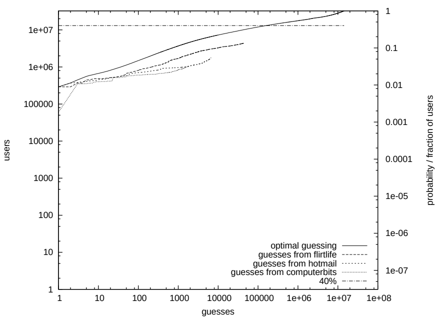
(a) hotmail



(b) flirtlife



(c) computerbits



(d) rockyou

Figure 5.1: Using password ordering from one distribution to guess another.

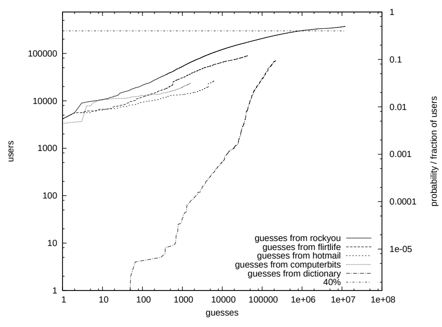


Figure 5.2: Using orderings from each list to crack Gawker password hashes.

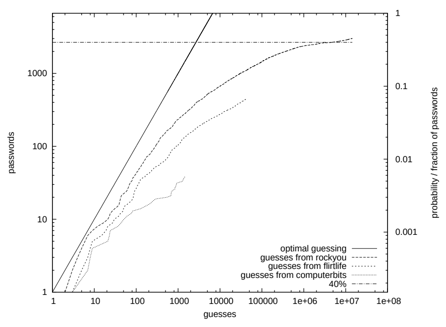
in less than one million trials. Even our smaller lists do well, recovering the passwords of more than 10,000 users in around 1,000 trials (less than one minute of CPU time).

For comparison, we show the results of using a dictionary in lexical order as list of guesses. The dictionary is based on the contents of `/usr/share/dict/` on Mac OS X, truncated to 8 characters and sorted using the Unix sort command. This results in a return on effort that is substantially lower than with ranked password lists.

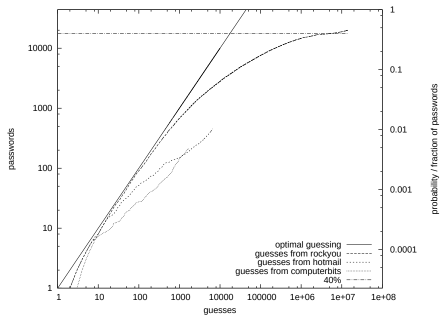
In [2], the authors consider various techniques for generating candidate passwords for guessing, including dictionary attacks, mangled dictionaries and Markov generators, which can be trained on sample passwords. They assess these techniques in terms of the fraction of passwords recovered in three data sets. For comparison, we will show the fraction of passwords recovered using the ranked password lists.

Figure 5.3 shows the results for our main lists. The optimal rate at which we can recover passwords is 1 per guess, so we plot the optimal line $y = x$. We see that with about 500,000–5,000,0000 guesses we can obtain about 40% of the passwords, except when guessing rockyou passwords, when the other lists simply do not have enough guesses to reach 40%. Note, that in the rockyou graph, the curves for the other lists stay close to the optimal line, showing that there is a relatively good return for each guess.

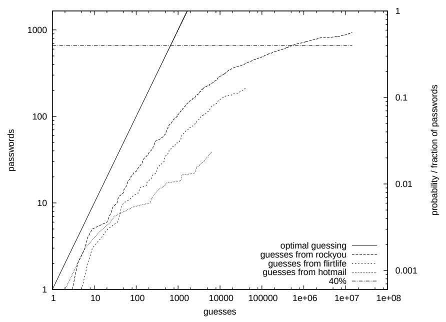
Figure 5.4 shows the results for guessing the Gawker passwords, in terms of fraction of passwords recovered. As not all hashes have been cracked and the hashes are salted, we do not know the total number of distinct passwords. However, we can upper bound the number by as-



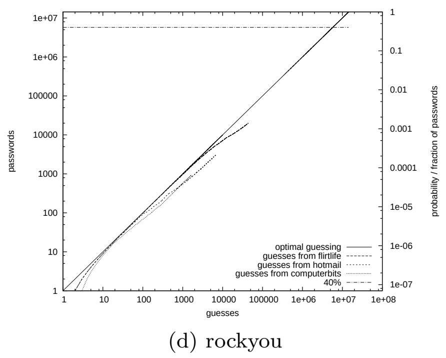
(a) hotmail



(b) flirtlife



(c) computerbits



(d) rockyou

Figure 5.3: Using password ordering from one distribution to guess another.

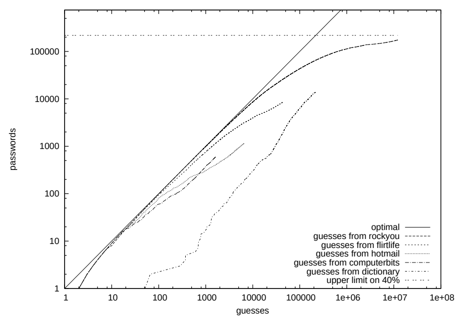


Figure 5.4: Using orderings from each list to crack Gawker password hashes.

suming that all uncracked passwords are unique.

First we note that using other password lists to guess still provides significantly better return than using a dictionary. Indeed, the curves stay relatively close to the optimal line for guesses based on the rockyou data set for between 10 and 10,000 guesses, indicating a success rate of almost 100%. After using all 14 million passwords in this list, we have cracked close to 40% of the passwords.

This compares favourably with the techniques in [2], where in order to achieve success rates in the region of 40% (without direct use of the training data) the number of required guesses is over 100 million. Of course, once exhausted, a list provides no more candidate guesses, whereas the mangling and Markov techniques can theoretically yield unbounded data sets.

6 Discussion

From our basic results, we have seen that Zipf's law seems to provide a reasonable description of the frequencies with which passwords are chosen, though with a parameter $s < 1$. We also see that fitting a distribution provides good approximations of the Shannon Entropy, guesswork and other statistics that are of interest when assessing a password distribution. As expected, we see that a model where all passwords are equally likely provides a poor approximation.

We have seen that demography of the userbase choosing the passwords can be evident in the most popular passwords, and even the name of the website is a likely password. Some sites, for example Twitter, implement banned password lists [9], which includes many of the more common passwords, including the name of the site. Banning more commonly chosen passwords may result in a more even spread of password choices.

The Zipf distribution decays relatively slowly, so we expect there to be a large number of relatively commonly chosen passwords. If these passwords do not change much from one list to another, we expect that one list can provide a pretty good guide about the popularity of passwords in another. We see that this is the case, and that while not optimal, larger lists provide good guidance about the ranking of passwords in other lists. We've demonstrated that this can provide a significant speedup in guessing or cracking passwords with moderate numbers of guesses, particularly over a simple dictionary attack.

An attacker who has collected leaked passwords from a collection of websites has useful starting point for cracking a password. If a hashed password is exposed, the time for an attacker to try, say, 20 million passwords is relatively small, even on a single CPU. We note that this adds some extra weight to the advice that reusing passwords between websites is a risk, even if there is no way for an attacker to identify common users.

7 Conclusion

We have seen that a Zipf distribution is a relatively good match for the frequencies with which users choose passwords. We have also seen that the passwords found in the lists that we have studied have relatively similar orderings. Consequently, passwords from one list provide good candidates when guessing or cracking passwords from another list.

Acknowledgment: The authors would like to thank Ken Duffy for thought-provoking comments and Dermot Frost for the offer of spare CPU cycles.

References

[1] L.A. Adamic. Zipf, power-laws, and pareto-a ranking tutorial. Xerox Palo Alto Research Center, `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`, 2000.

[2] M. Dell'Amico, P. Michiardi, and Y. Roudier. Password Strength: An Empirical Analysis. *INFOCOM*, pages 1–47, 2010.

[3] D. Eastlake, J. Schiller, and S. Crocker. RFC 4086: Randomness Requirements for Security. *Theory of Cryptography Library: Record*, pages 1–47, 2005.

[4] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on New security paradigms*, pages 133–144. ACM, 2009.

[5] D. Malone and W. Sullivan. Guesswork is not a substitute for Entropy. In *Proceedings of the Information Technology and Telecommunications Conference*. Citeseer, 2005.

[6] James L. Massey. Guessing and entropy. In *In Proceedings of the 1994 IEEE International Symposium on Information Theory*, page 204, 1994.

[7] Mindi McDowell, Jason Rafail, and Shawn Hernan. Cyber security tip st04-002. `http://www.us-cert.gov/cas/tips/ST04-002.html`, 2009.

[8] J. Pliam. The disparity between work and entropy in cryptology. *Theory of Cryptography Library: Record*, pages 98–24, 1998.

[9] twitter.com. Source code from twitter registration page. `view-source:https://twitter.com/signup` (search for twttr.BANNED_PASSWORDS), 2010.