Investigating the Distribution of Password Choices

David Malone and Kevin Maher,
Hamilton Institute, NUI Maynooth.

19 April 2012

How to Guess a Password?

Passwords are everywhere.
If you dont know the password, can you guess it?

1. Make a list of passwords.
2. Assess the probability that each was used.
3. Guess from most likely to least likely.

A dictionary attack, but with optimal ordering.

(Applies to computers and keys too.)

How long will that take?

If we knew probability P_i of i^{th} password.
Rank the passwords from 1 (most likely) to N (least likely).
Average number of guesses is:

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

Note, not the same as Entropy (Massey '94, Arikan '96).
Does this P_i really make sense?
Is there a distribution with which passwords are chosen?

Outline

- Is there password distribution?
  Is knowing it better than a crude guess?
- Are there any general features?
  Do different user groups behave in a similar way?
- Some distributions better than others.
  Can we help users make better decisions?

Getting data

Want a collection of passwords to study distribution.
Asked Yahoo, Google.

- ...

Crackers eventually obliged.

- 2006: flirtlife, 98930 users, 43936 passwords, 0.44.
- 2009: hotmail, 7300 users, 6670 passwords, 0.91.
- 2009: computerbits, 1795 users, 1656 passwords, 0.92.
- 2009: rockyou, 32603043 users, 14344386 passwords, 0.44.

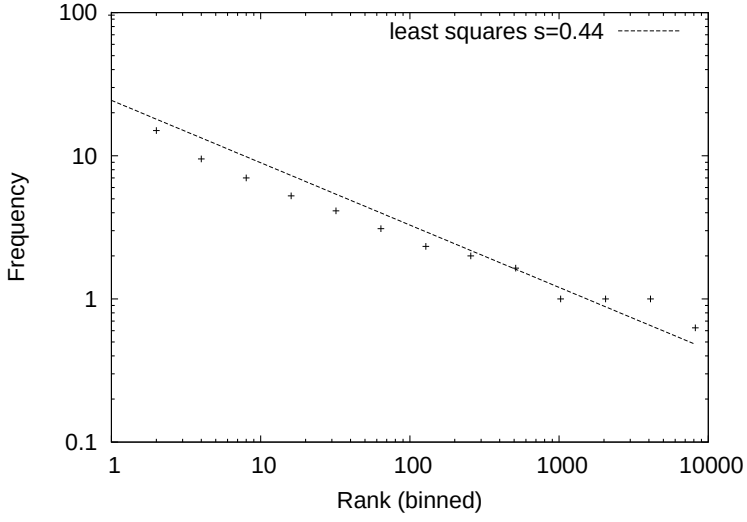
Good: cleartext!
Bad: Had to clean up data.

Top Ten

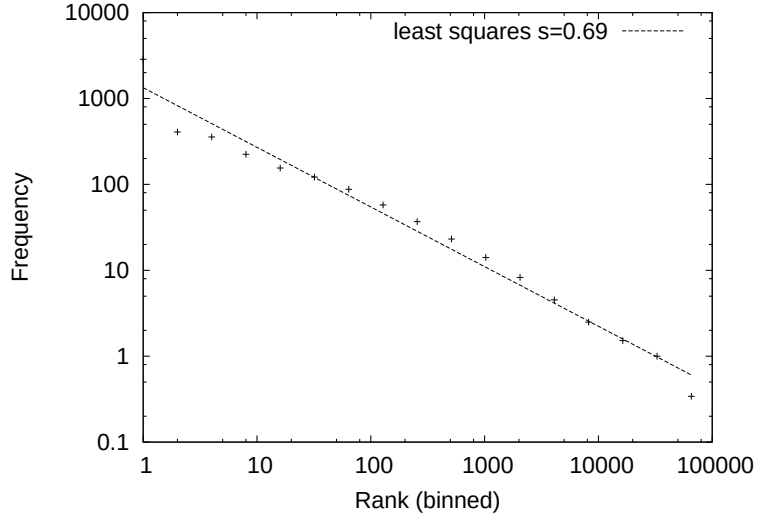
Rank	hotmail	#users	flirtlife	#users	computerbits	#users	rockyou	#users
1	123456	48	123456	1432	password	20	123456	290729
2	123456789	15	ficken	407	computerbits	10	12345	79076
3	111111	10	12345	365	123456	7	123456789	76789
4	12345678	9	hallo	348	dublin	6	password	59462
5	tequiero	8	123456789	258	letmein	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

(c.f. Imperva analysis of Rockyou data, 2010)

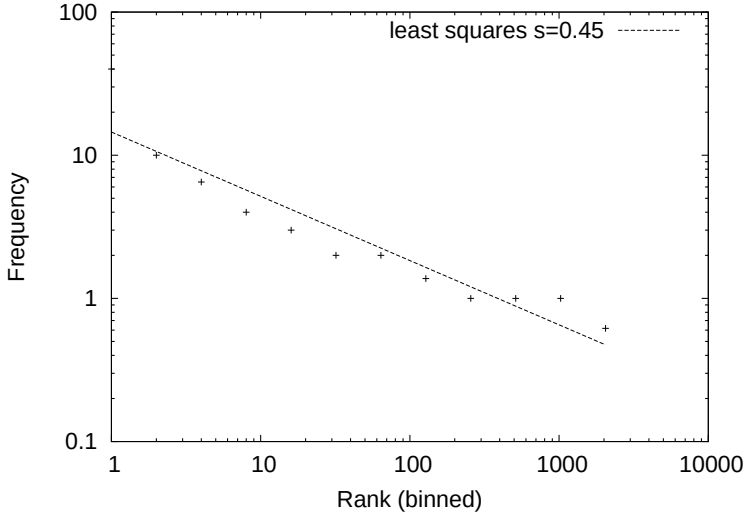
Distribution?



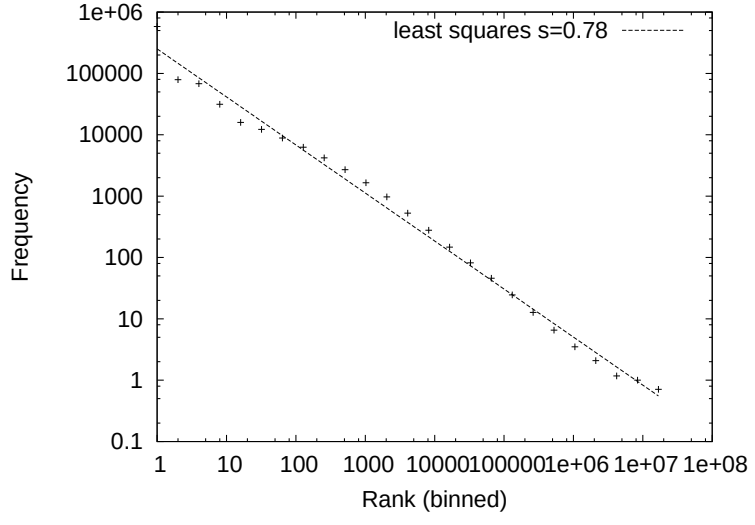
hotmail



flirtlife



computerbits



rockyou

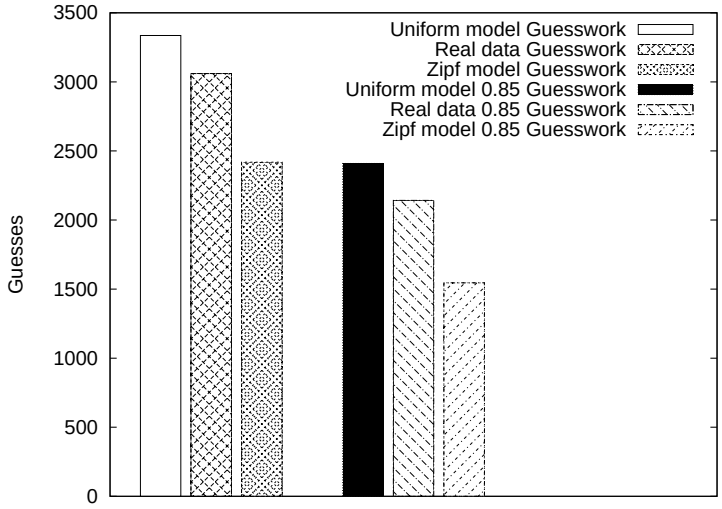
Zipf?

- A straight line on a log-log plot points towards heavy tail.
- Zipf?

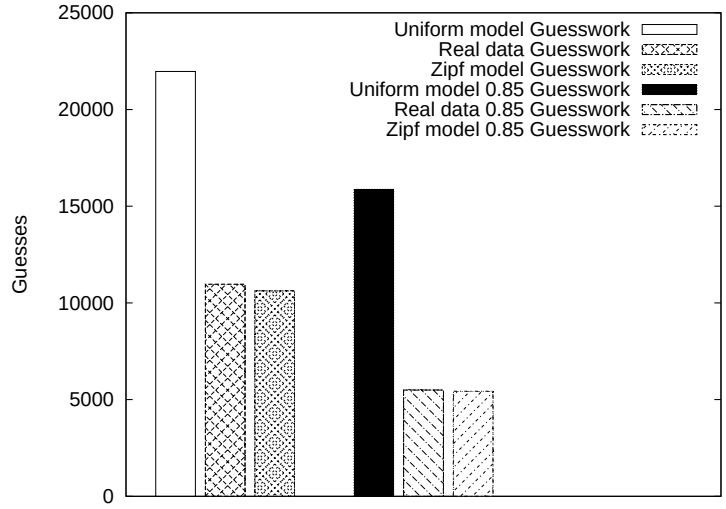
$$P_r \propto \frac{1}{r^s}$$

- Slope gives s.
- Can check p-values (Clauset '09).
- s is small, less than 1.

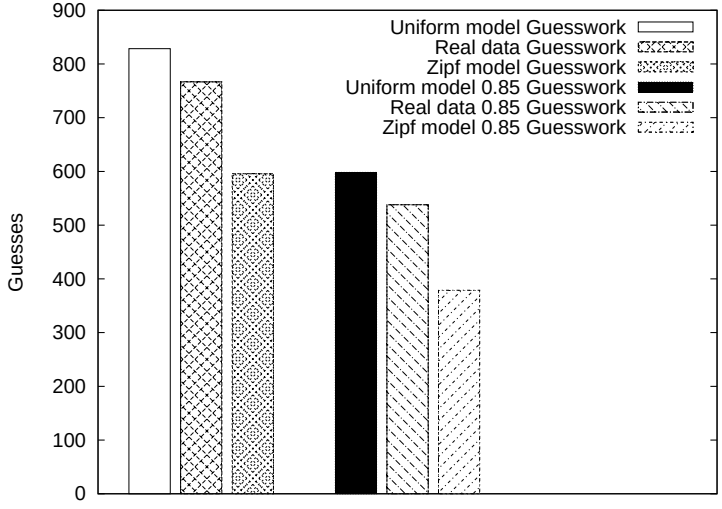
Guesswork Predictions



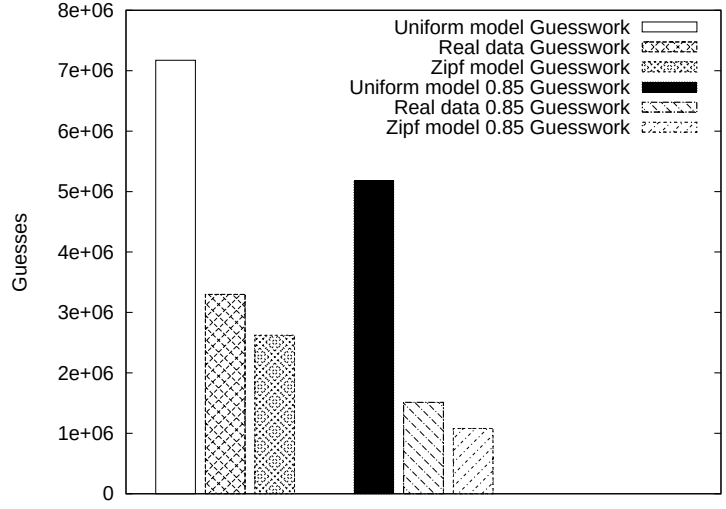
hotmail



flirtlife



computerbits



rockyou

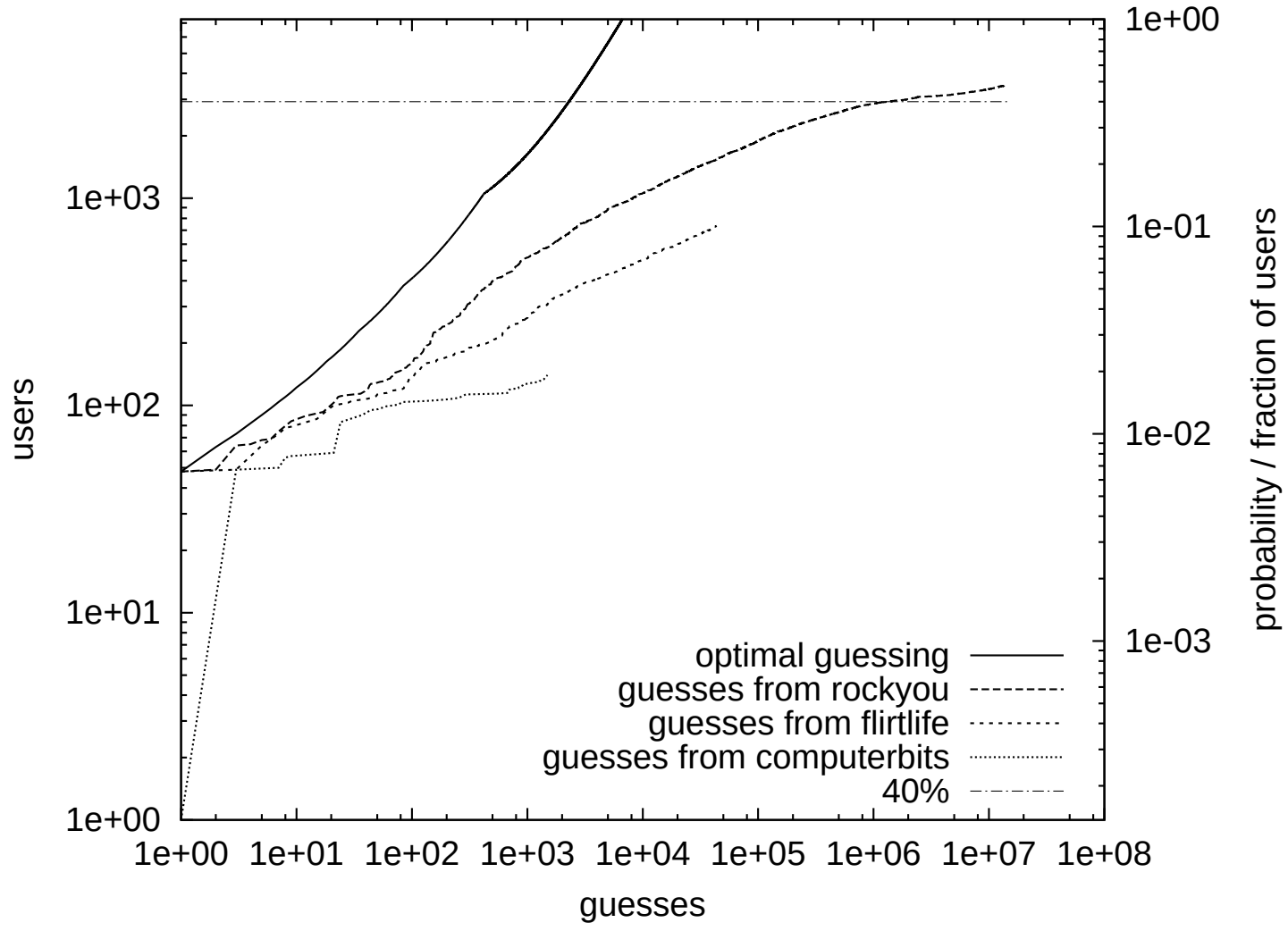
Who cares?

- Algorithm Design — exploit heavy tail?
- Can we get close to optimal dictionary attack?
- Can we make dictionary attack less effective?

2 and 3 answer questions about common behavior and helping users.

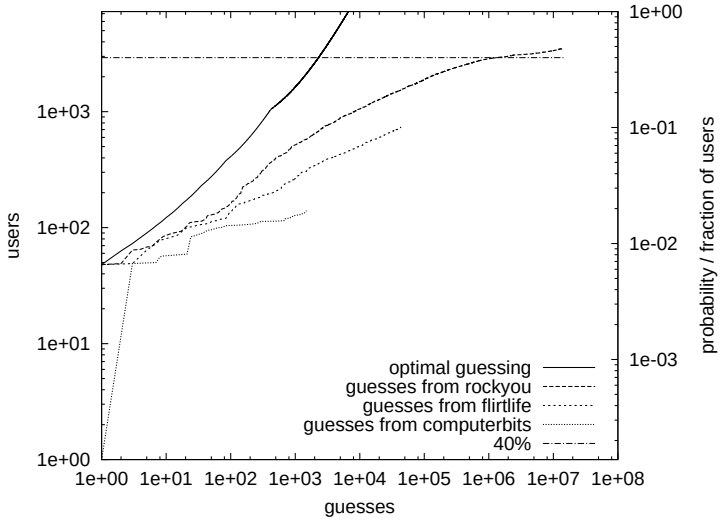
Dictionary Attack

Suppose we use one dataset as a dictionary to attack another.

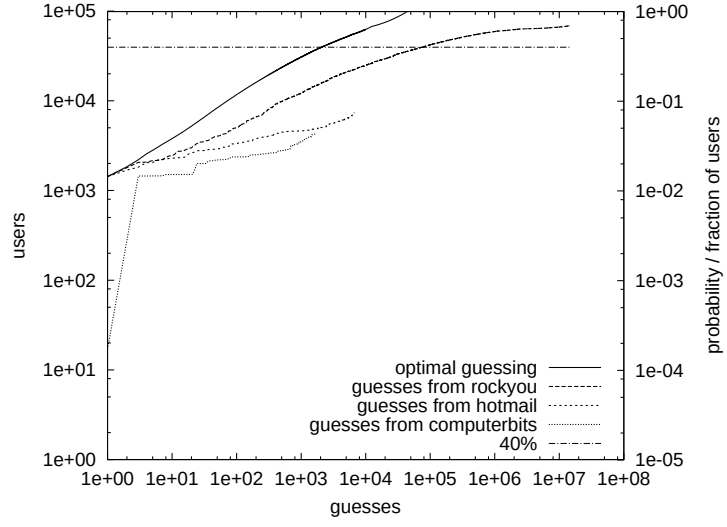


hotmail

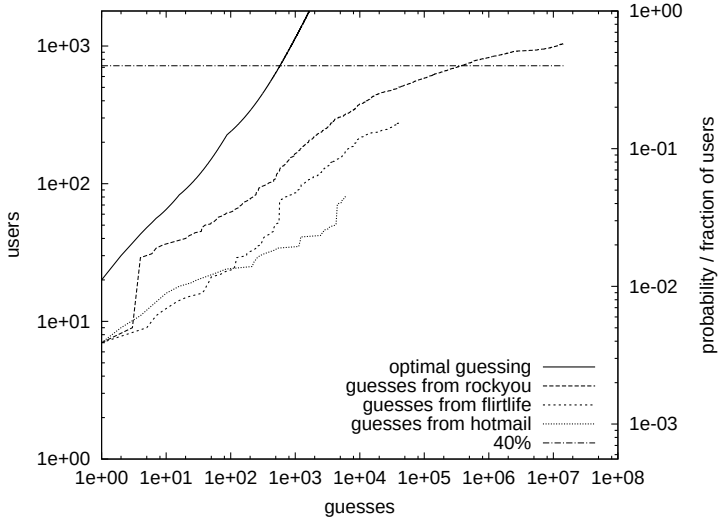
Dictionary Attack — Same Story



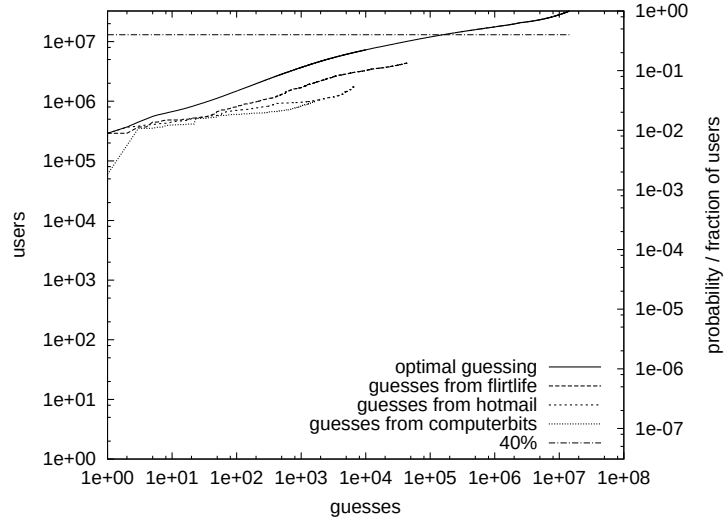
hotmail



flirtlife



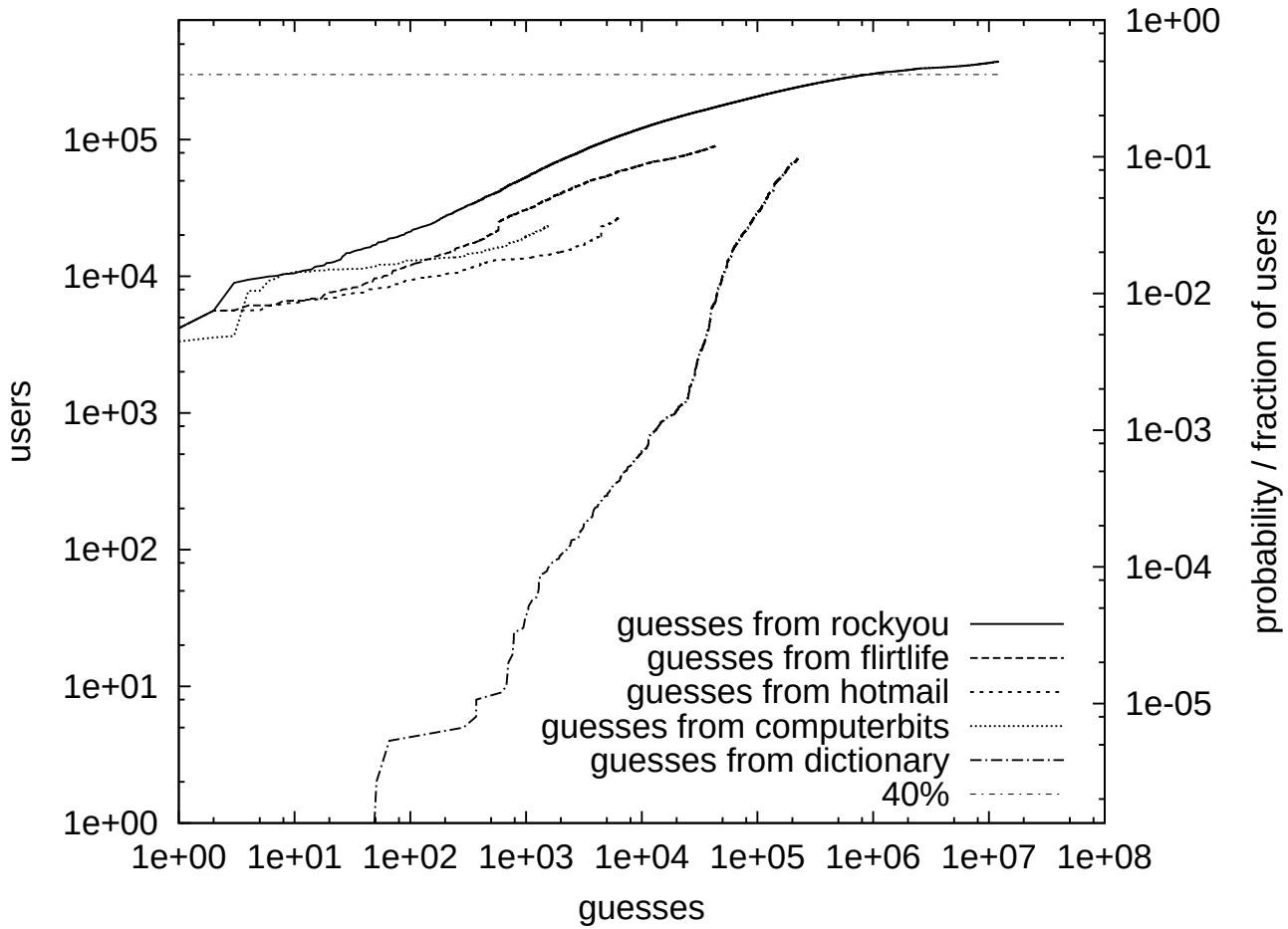
computerbits



rockyou

Dictionary Attack Gawker

December 2010, Gawker, 748090 DES Hashes, well salted.



Results in paper for % passwords.

Dell'Amico'10 review smart generators. This looks $\times 10$!

Helping Users

If users select passwords 'randomly', can we make them a better generator?

- Banned list (e.g. twitter),
- Password rules (e.g. numbers and letters).
- Act like a cracker (e.g. cracklib),
- Cap peak of password distribution (e.g. Schechter'10),
- Aim for uniform?

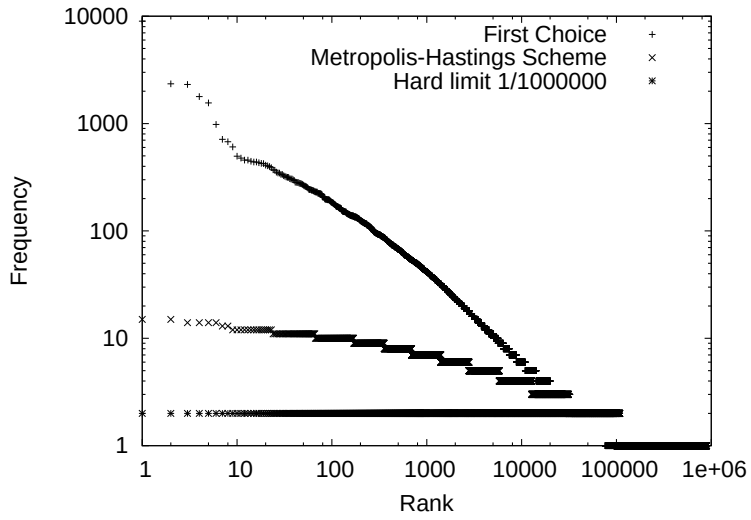
Metropolis-Hastings algorithm takes bad random number generator and makes it good.

Metropolis-Hastings for Uniform Passwords

Keep a frequency table $F(x)$ for requests to use password x.

1. Uniformly choose x from all previously seen passwords.
2. Ask user for a new password x'.
3. Generate a uniform real number u in the range $[0, F(x')]$ and then increment $F(x')$. If $u \leq F(x)$ go to step 4 (accept), otherwise return to step 2 (reject).
4. Accept use of x' as password.

How does it do?



Rockyou-based test, 1000000 users, mean tries 1.28, variance 0.61.

Could be implemented using min-count sketch.
Doesn't store actual *use* frequencies.
No parameters, aims to flatten whole distribution.

Conclusions

- Idea of distribution of password choices seems useful.
- Zipf is OK, but not perfect match.
- Different user groups have a lot in common (not peak).
- Dictionaries not great for dictionary attacks.
- Treat users as random password generators?
- Future: Generalise beyond web passwords?
- Future: Field test of Metropolis-Hastings?
- Future: What does optimal banned list look like?

From Reviews

- So much cool literature from (at least) 1979–2012.
- *In security, passwords are the gift that keeps on giving.*