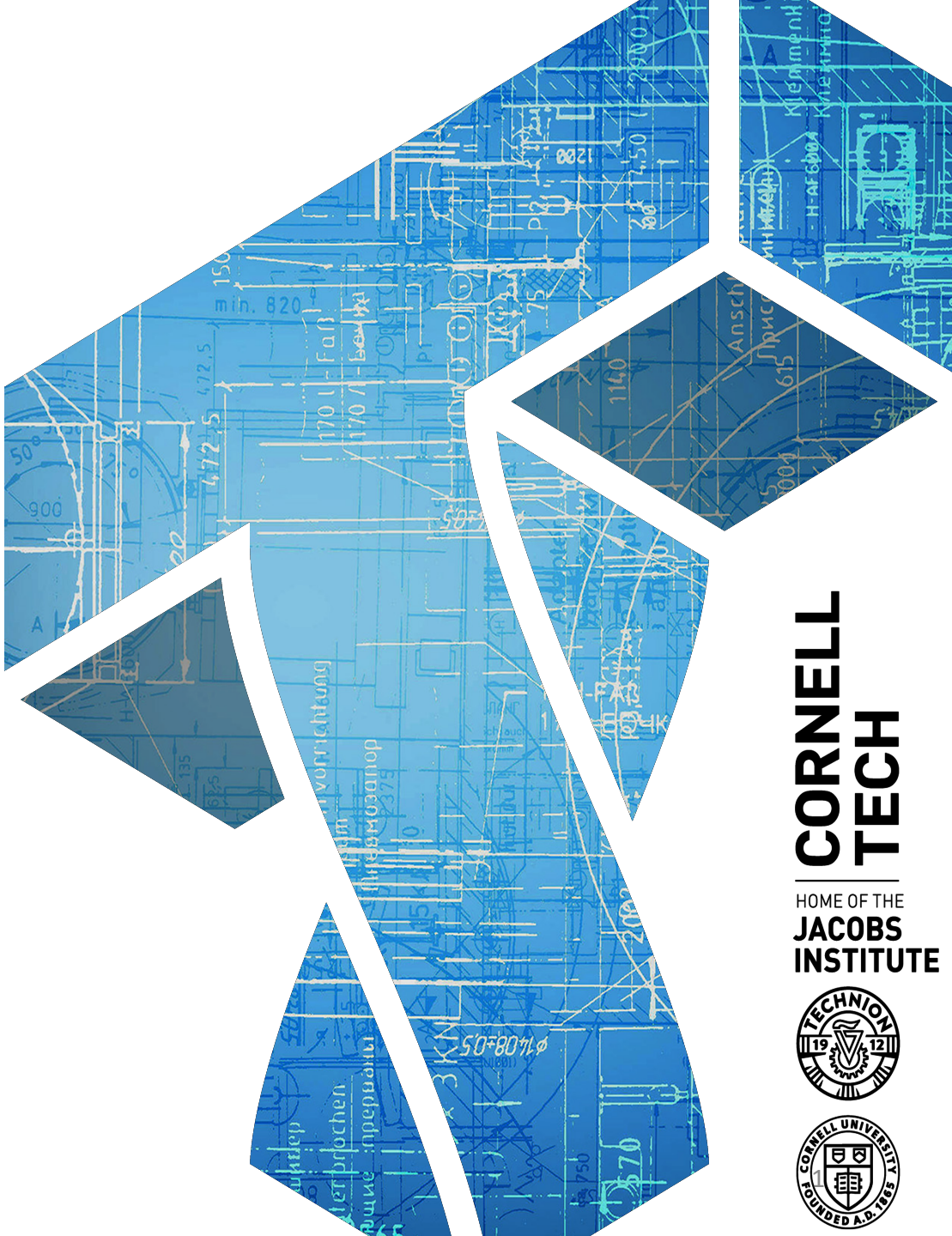
CS 6431: Passwords



Instructor: Tom Ristenpart

https://github.com/tomrist/cs6431-fall2021

CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Summary from last time

- Morris & Thompson UNIX case study
- Password hashing
- Florencio & Herley measurement study

- Today:
  - Understanding password distributions
  - Bonneau study
  - Melicher et al.

Understanding password strength

(1) Empirical studies of user passwords

Password database leaks
Instrumentation of large web systems (Bonneau paper)
Instrumentation of clients (Florencio & Herley)
In-lab studies, online studies, surveys

(2) Develop probabilistic model of passwords

$pw_1, pw_2, \ldots, pw_N$
$p(pw_i) = p_i$ = probability user selects password pw_i

$$\sum_i p_i = 1$$

(3) Use p to educate brute-force crackers, strength meters

Internet users ditch “password” as password, upgrade to “123456”

Contest for most commonly used terrible password has a new champion.

by **Jon Brodkin** - Jan 20 2014, 4:00pm GMT

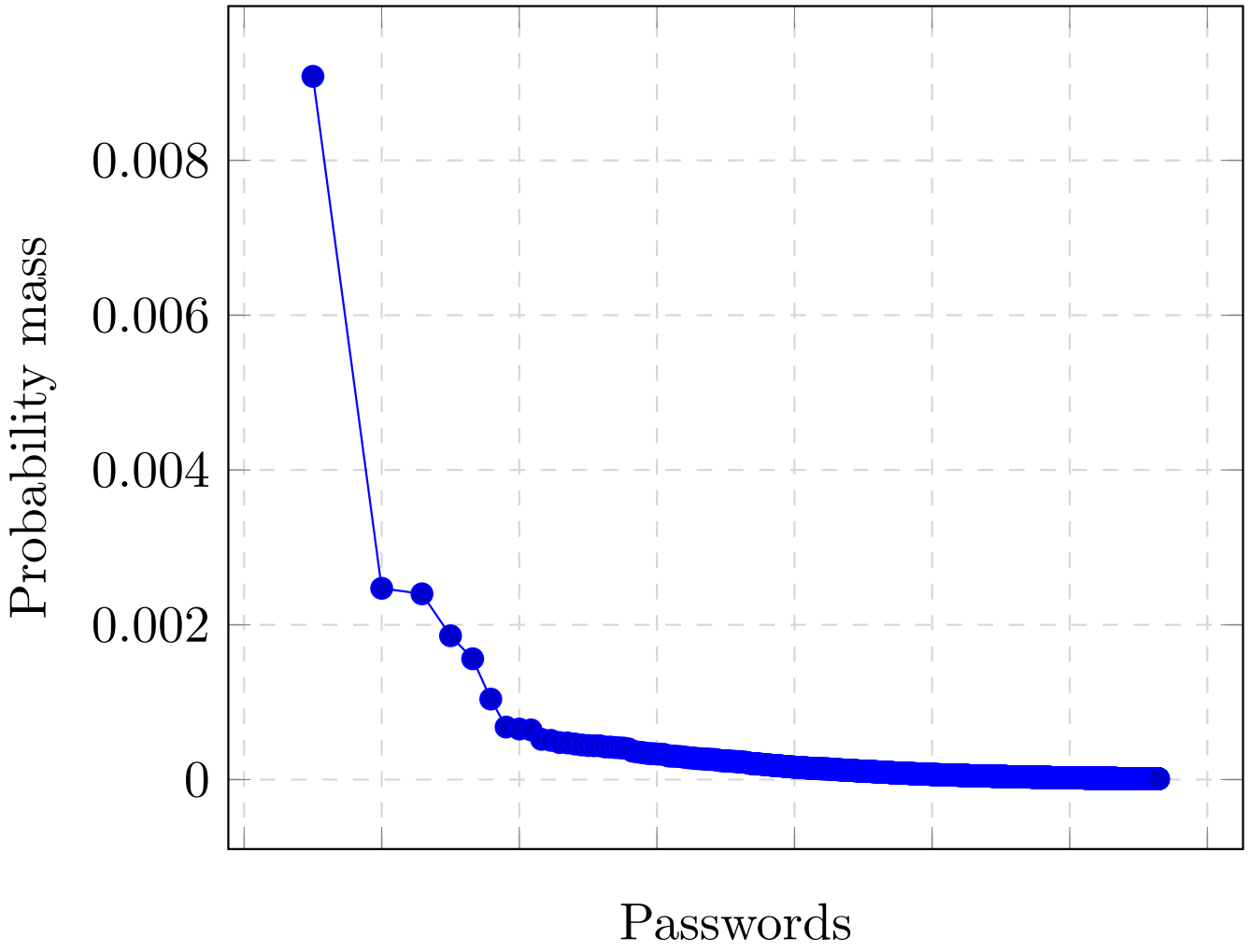
290729 123456
79076 12345
76789 123456789
59462 password
49952 iloveyou
33291 princess
21725 1234567
20901 rockyou
20553 12345678
16648 abc123
16227 nicole
15308 daniel
15163 babygirl
14726 monkey
14331 lovely

Rockyou data breach:
32 million social gaming accounts

Most common password used by almost 1%

[Bonneau 2012]
69 million Yahoo! Passwords
1.1% of users pick same password

Rockyou empirical probability mass function



(Only first 5,000 points shown)

Password strength metrics

- Florencio and Herley approach?
  - Alphasize(pw) = sum of the sizes of character classes observed in password
    - Hello12! Has alphabet size = 26 + 26 + 10 + 22 = 84
  - Bitstrength(pw) = $\log_2(\text{Alphasize(pw)}^{\text{len(pw)}})$
    - Bitstrength(Hello12!) = 51.1 bits of entropy
- Similar to, simpler than classical NIST entropy estimate

Password strength metrics

Let $\mathcal{X}$ be password distribution

Passwords are drawn iid from $\mathcal{X}$, denoted $x \leftarrow\!\!\$\ \mathcal{X}$

N is size of support of $\mathcal{X}$

p_1 , p_2 , …, p_N are probabilities of passwords in decreasing order

Shannon entropy: $$H_1(\mathcal{X}) = \sum_{i=1}^{N} -p_i \log p_i$$

As an expectation: $$H_1(\mathcal{X}) = \mathbf{E}(-\log p(\mathcal{X}))$$

Class breakout discussion: entropy measures

- Bitstrength(pw) = $\log_2$(Alphasize(pw)$^{\text{len(pw)}}$)

- Shannon entropy: $H_1(\mathcal{X}) = \sum_{i=1}^{N} -p_i \log p_i$

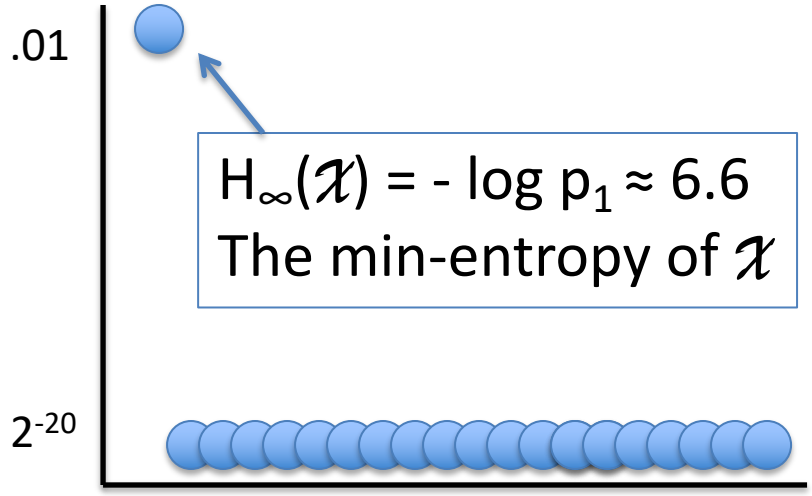
- Breakouts into small groups (10 minutes):
  - Intros: name, program, research interests, one personal thing
  - Discuss entropy measures so everyone understands them
  - Give example distribution w/ high Bitstrength & Shannon, but easy to guess passwords.
  - Why is this important?
  - How would you measure password strength?

Clarifications

- Bitstrength gives estimate for a ***particular*** password
- Shannon entropy gives estimate for ***distribution***
  - Formally these are clearly different
- Any examples of distributions?

Shannon entropy is poor measure (for password unpredictability)

N = 1,000,000
$p_1 = 1 / 100$
$p_2 = (1 - 1/100)/999{,}999 \approx 1 / 2^{20}$
…
$p_N = (1 - 1/100)/999{,}999 \approx 1 / 2^{20}$



$H_1(\mathcal{X}) \approx 19$

19 bits of "unpredictability". Probability of success about $1/2^{19}$

What is probability of success if attacker makes one guess?

Shannon entropy is almost never useful measure for security

Min-entropy as a measure?

- Min-entropy $H_\infty(\mathcal{X}) = -\log p_1$
- p_1 is best-case probability of succeeding on ***first*** guess.
  - What is that guess? Most probable password
- What about in q guesses?
- Min-entropy gives (conservative) upper bound:

$$\sum_{i=1}^{q} p_i \leq q \cdot p_1$$

- Can generalize to q-min-entropy (c.f., [Woodage et al. 2017])

Password strength metrics

Beta-success rate:

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

$$\tilde{\lambda}_\beta(\mathcal{X}) = \log(\beta/\lambda_\beta(\mathcal{X})$$

Alpha-work-factor:

$$\mu_\alpha(\mathcal{X}) = \min\left\{ j \;\middle|\; \sum_{i=1}^{j} p_i \geq \alpha \right\}$$

$$\tilde{\mu}_\alpha(\mathcal{X}) = \log(\mu_\alpha(\mathcal{X})/\lambda_{mu_\alpha}(\mathcal{X}))$$

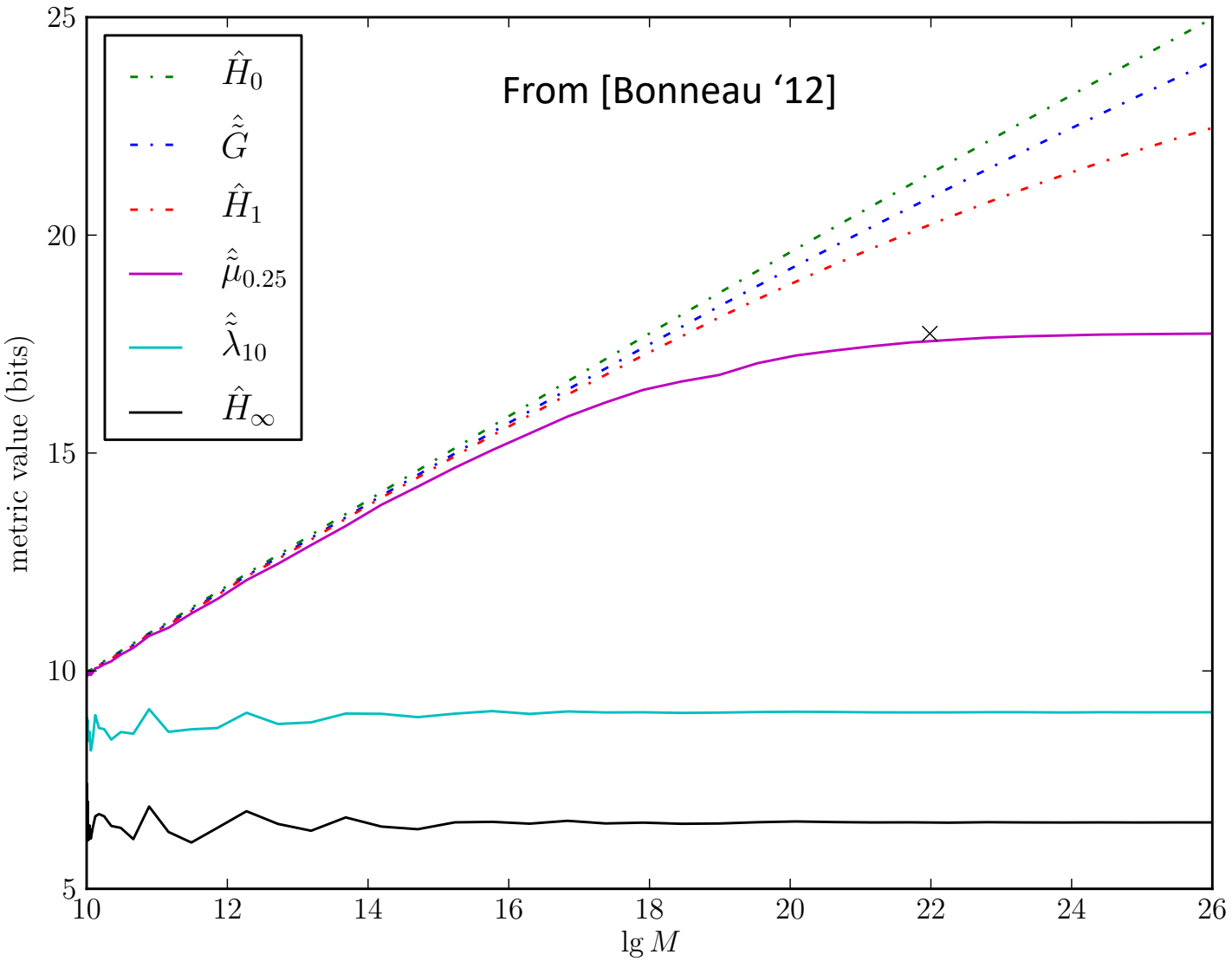


Figure 3. Changing estimates of guessing metrics with increasing sample size M. Estimates for H_∞ and $\tilde{\lambda}_{10}$ converge very quickly; estimates for $\tilde{\mu}_{0.25}$ converge around $M = 2^{22}$ (marked $\times$) as predicted in Section V-A. Estimates for H_0, H_1, and $\tilde{G}$ are not close to converging.

Bonneau's measurement study

Login:
tom, pw'

Yahoo login service

Check that H(pw') = H(pw)

- Florencio, Herley measured on client side
- Bonneau looked at server side. What were instrumentation challenges?
  - Keyed hash H(r || pw) and throw away key r
  - Don't store in a way that allows linking user to hash

Bonneau's measurement study

	M	$\hat{H}_\infty$	$\hat{\tilde{\lambda}}_{10}$	$\hat{\tilde{G}}_{0.25}$	$\hat{\tilde{G}}_{0.5}$
Yahoo! (2011)	69301337	6.5	9.1	17.6	21.6
RockYou (2009)	32603388	6.8	8.9	15.9	19.8
Battlefield Heroes (2011)	548774	7.7	9.8	16.5	20.0

Table III
COMPARISON OF YAHOO! DATA WITH LEAKED DATA SETS

- Lots more data in paper: demographics, comparison with cracking experiments from prior work, …

Bonneau takeaways

- Use appropriate strength measures for password distributions
- Yahoo study: people pick lousy passwords
- What does Bonneau paper not give us?

Understanding password strength

(1) Empirical studies of user passwords

Password database leaks
Instrumentation of large web systems (Bonneau paper)
Instrumentation of clients (Florencio & Herley)
In-lab studies, online studies, surveys

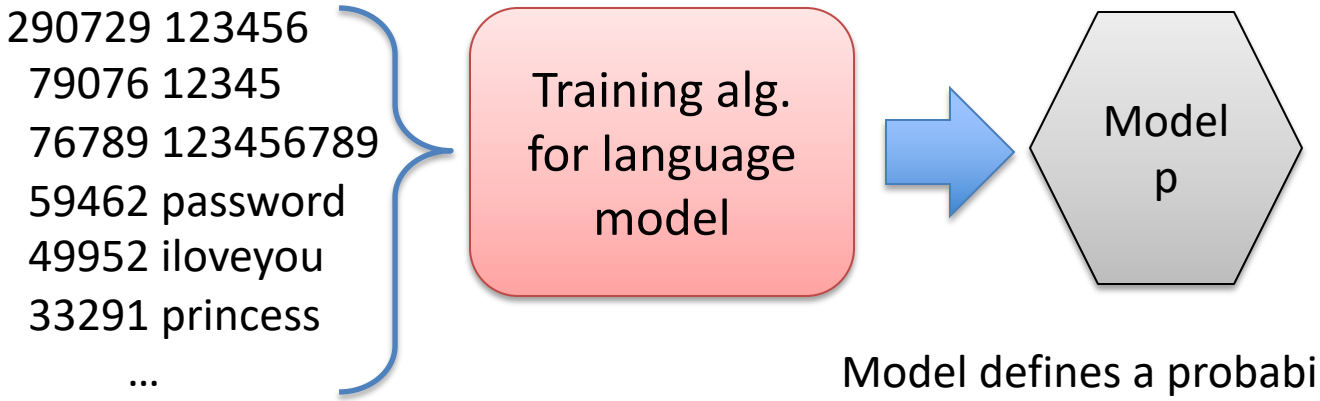
(2) Develop probabilistic model of passwords

$pw_1, pw_2, \ldots, pw_N$
$p(pw_i) = p_i$ = probability user selects password pw_i

$$\sum_i p_i = 1$$

(3) Use p to educate brute-force crackers, strength meters

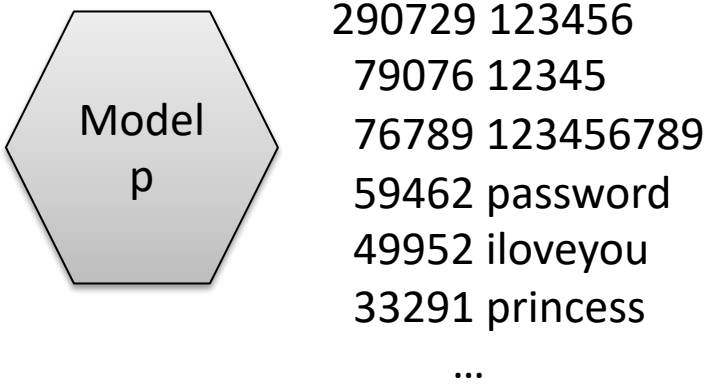
Train models from leaked passwords



Model defines a probability distribution over passwords. Can use to:

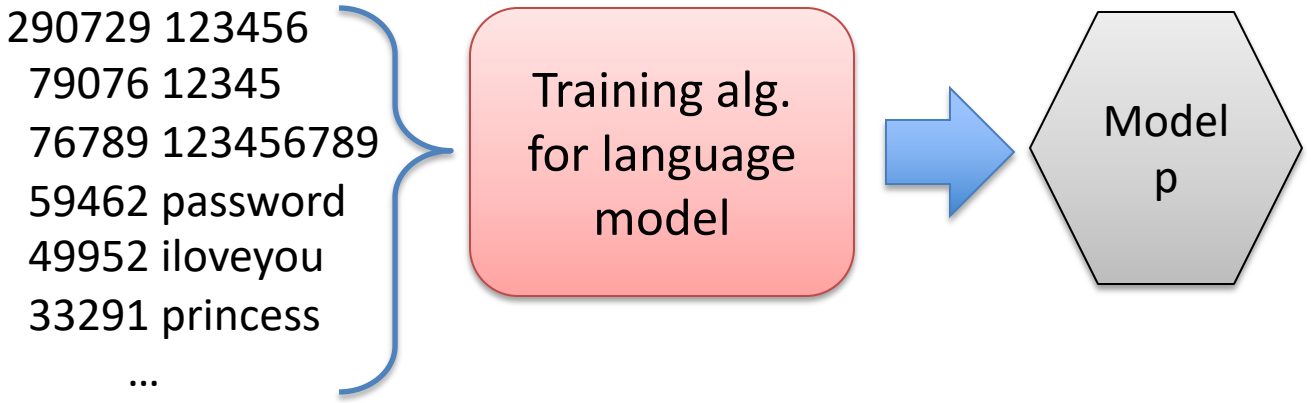
- sample passwords according to distribution
- enumerate passwords in order of likelihood

Trivial model is just the empirical CDF of the histogram itself



Supports all the above
Generalizability is quite poor
ML terminology: model is overfit

Train models from leaked passwords



Probabilistic context-free grammar (PCFG) [Weir et al. 2009]

CFG with probability distribution associated to each rule

Fix a CFG, then learn probabilities by training on passwords

We can encode a string (password) by its parse tree, and assign probability by accumulating edge probabilities of PCFG

TABLE 3.2.1
Example probabilistic context-free grammar

LHS	**RHS**	**Probability**
$S \rightarrow$	$D_1L_3\ S_2D_1$	0.75
$S \rightarrow$	$L_3D_1S_1$	0.25
$D_1 \rightarrow$	4	0.60
$D_1 \rightarrow$	5	0.20
$D_1 \rightarrow$	6	0.20
$S_1 \rightarrow$	!	0.65
$S_1 \rightarrow$	%	0.30
$S_1 \rightarrow$	#	0.05
$S_2 \rightarrow$	$$	0.70
$S_2 \rightarrow$	**	0.30

$S \rightarrow L_3D_1S_1 \rightarrow L_34S_1 \rightarrow L_34!$

$Pr[\ L_34!\ ] = 0.25 * 0.60 * 0.65 = 0.0975$

With good training data: Works better than JtR

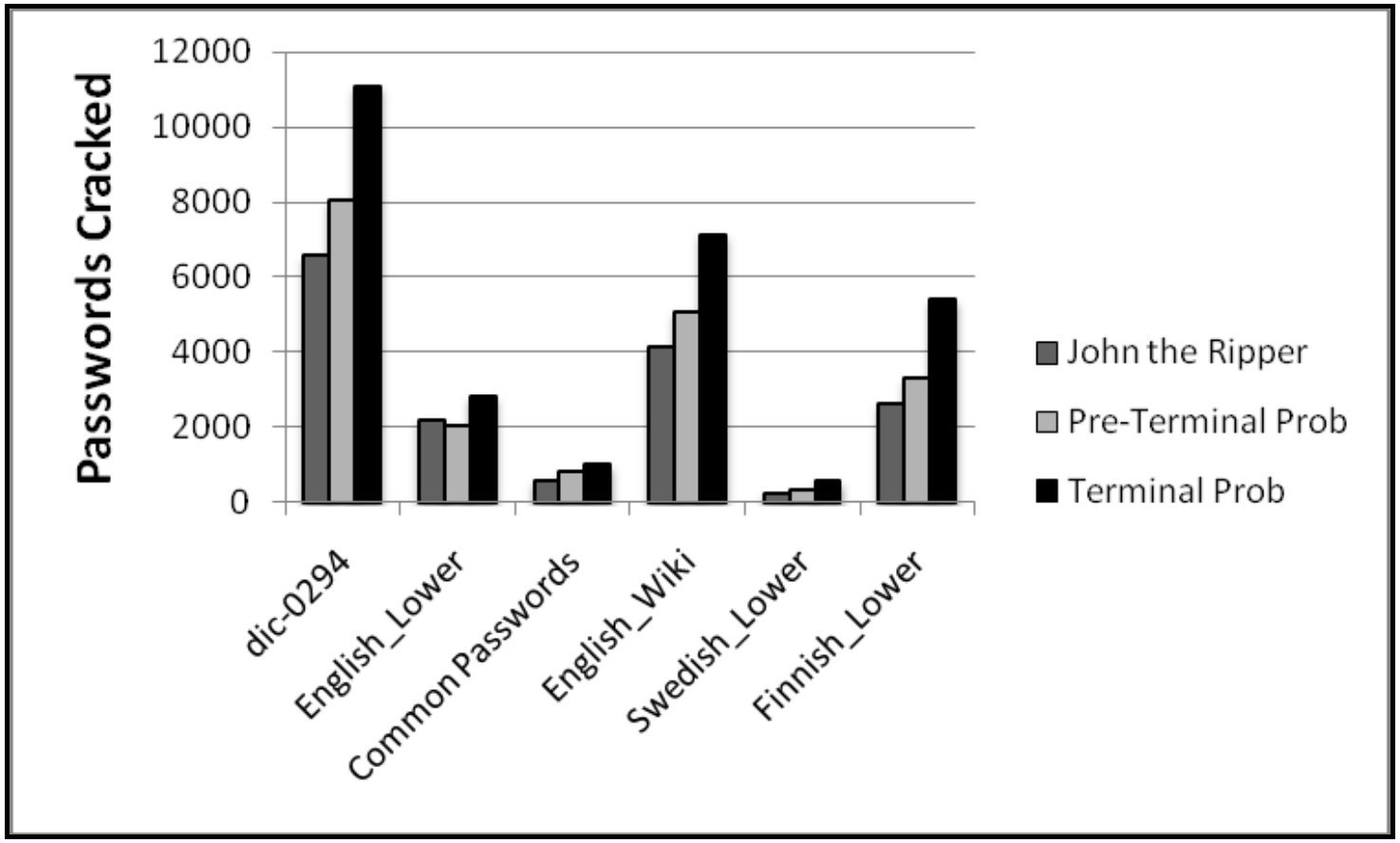
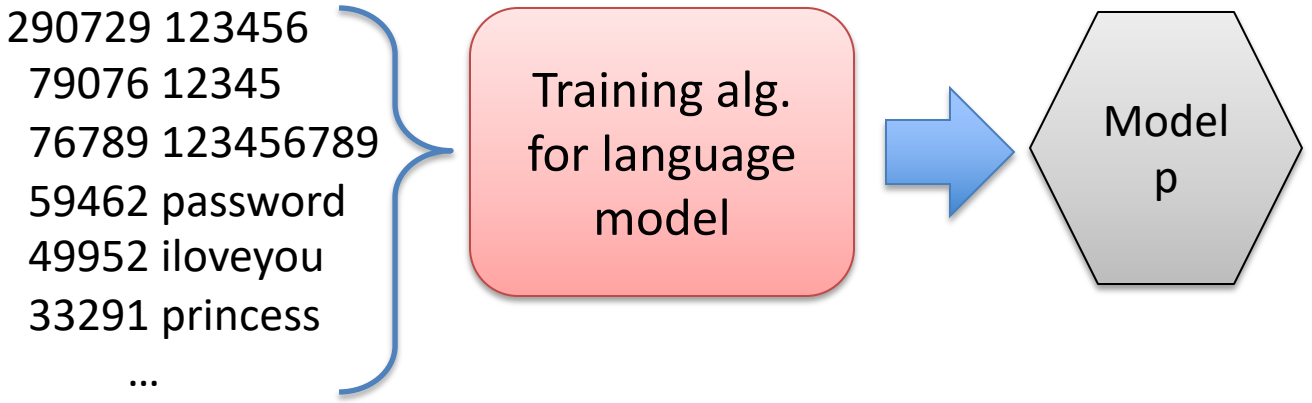


Fig. 4.4.1. Number of Passwords Cracked. Trained on the MySpace Training List. Tested on the MySpace Test List

Train models from leaked passwords



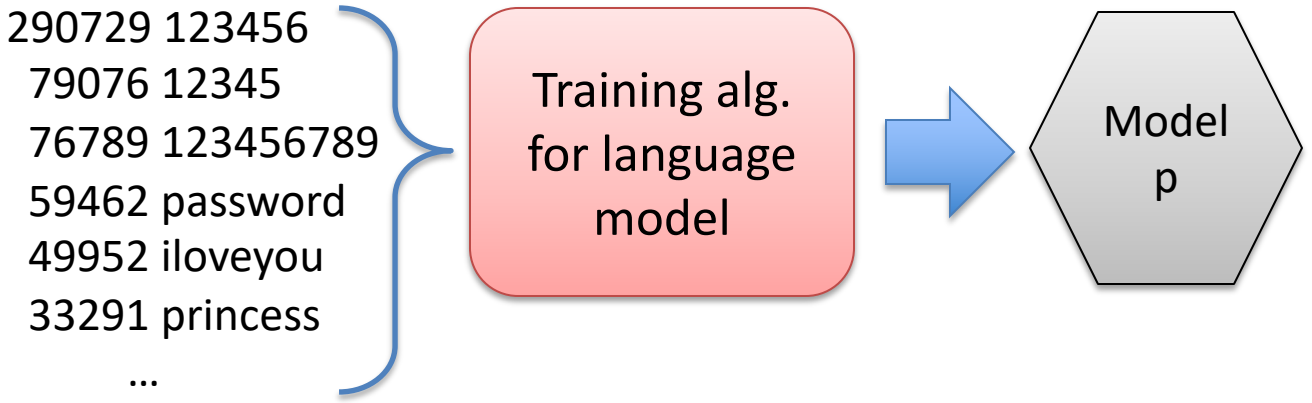
Probabilistic context-free grammar only one NLP modeling approach

n-gram Markov models another popular choice.:

$$\Pr[\, w_1 w_2 \cdots w_k \,] \approx \prod_{i=1}^{k} \Pr[\, w_i \mid w_{i-(n-1)} \cdots w_{i-1} \,]$$

[Ma et al. ’14] show carefully chosen Markov model beats Weir et al. PCFG

Train models from leaked passwords



Neural network approach of [Melicher et al. 2016]

Use Long short-term (LSTM) recurrent neural network trained from large number of leaks (RockYou, Yahoo!, many others)

They primarily target using it as a strength meter:

For any pw, use p(pw*) to estimate the guess rank |S(pw*)|

$S(pw^*) = \{ pw \mid p(pw) > p(pw^*) \}$

Can estimate using Monte-Carlo techniques [Dell'Amico, Filippone '15]

Can estimate S(pw*) using Monte-Carlo techniques

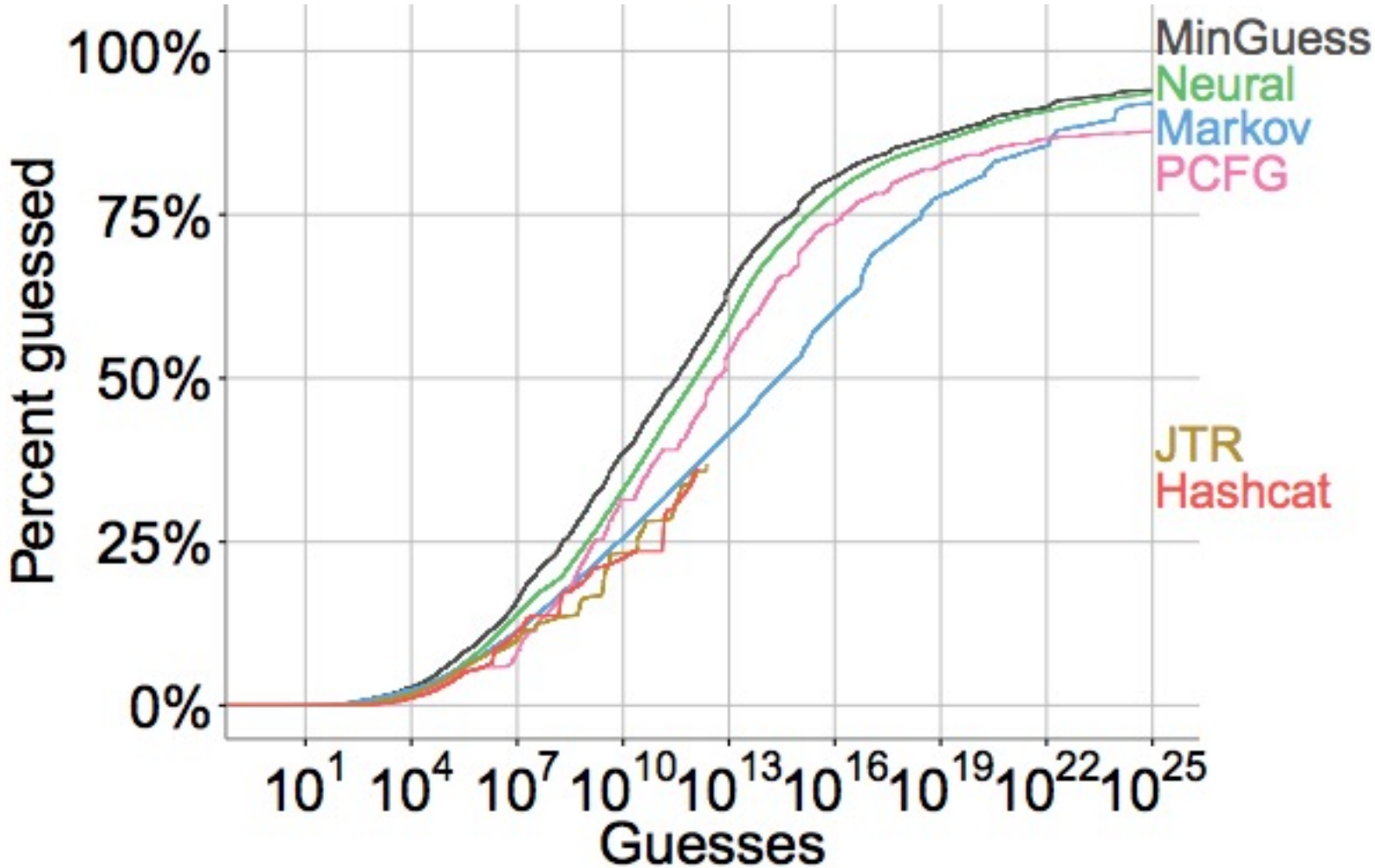
[Dell'Amico, Filippone '15]

$$\hat{S}(pw^*) = \sum_{pw \in \Theta} \begin{cases} \frac{1}{p(pw) \cdot n} & \text{if } p(pw) > p(pw^*) \\ 0 & \text{otherwise} \end{cases}$$

Set of n passwords sampled according to p

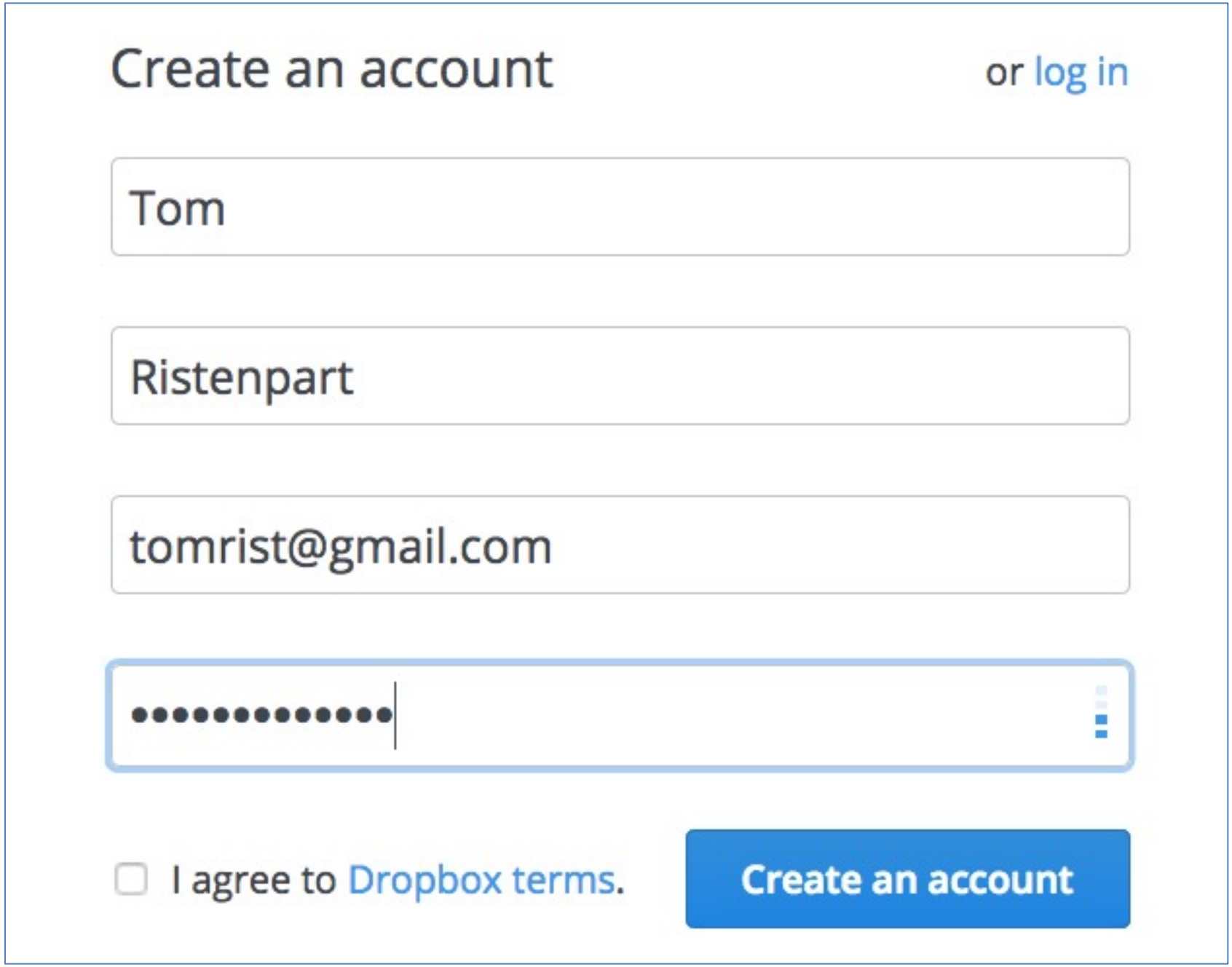
Proved that as n grows, a good estimate of true guess rank S(pw*)

Can precompute tables and do binary search to compute guess rank in space O(n) and time O(log n)



(b) Webhost passwords

[Melicher et al. 2016]


Create an account
or log in
Tom
Ristenpart
tomrist@gmail.com
I agree to Dropbox terms.
Create an account

Password selection policies

- [Komanduri et al. ‘11], [Vu et al. ‘07], [Proctor et al. ‘02] and others studied with Amazon Mturk
  - General consensus that it increases resistance (but by exactly how much is not clear to me), but decreases usability
  - Just have a length requirement, and nudge with strength meter
- Password expiration policies [Zhang et al. ’10]
  - They don’t work
  - Attacker with old password can crack new one very often

Summary: password strength

- Older entropy measures, estimates inaccurate
- Needed new empirical approaches:
  - measurement studies, breach data
- Password models power:
  - Strength meters use guess rank estimate for best known guessing attacks
  - Simulation-based studies utilize these models as well
  - Improvements to cracking tools (CMU password guessing tool)

When does password strength *not matter?*