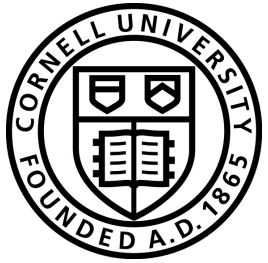
Passwords (3)

Tom Ristenpart
CS 6431



The game plan

- Refresh from last week
- Modeling password distributions
  - Melicher et al. paper (neural networks)
- Typo-tolerant password checking

Understanding password distributions

(1) Develop probabilistic model of passwords

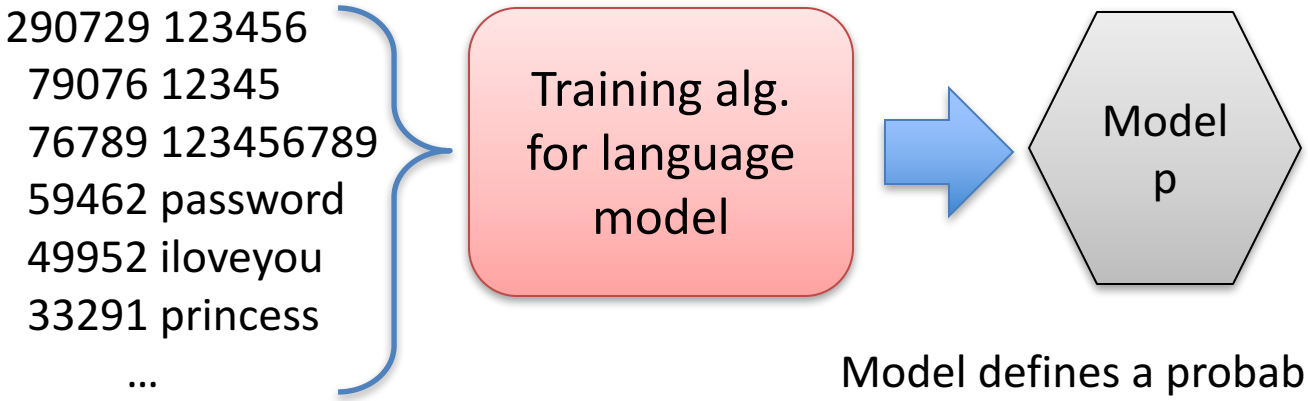
$pw_1, pw_2, \ldots, pw_N$

$p(pw_i) = p_i$ = probability user selects password pw_i

$$\sum_i p_i = 1$$

(2) Use p to educate brute-force crackers, strength meters, user interfaces

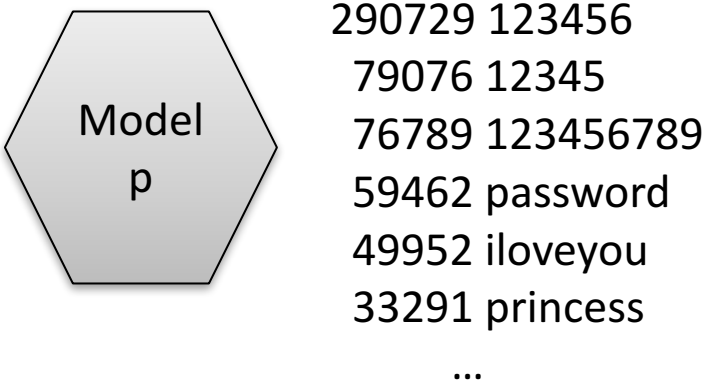
Train models from leaked passwords



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

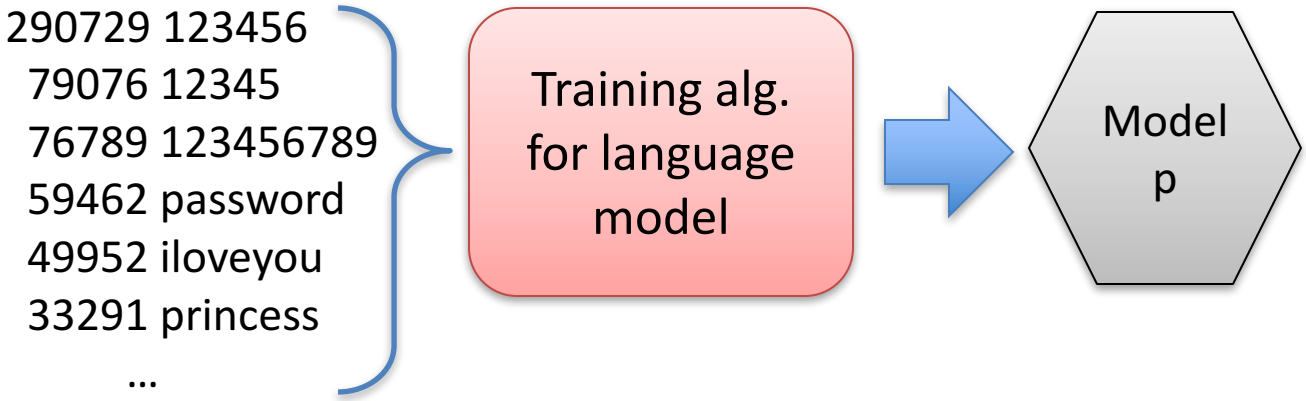
- sample passwords according to distribution
- enumerate passwords in order of likelihood

Trivial model is just the empirical CDF of the histogram itself



Supports all the above
Generalizability is quite poor
ML people would say this model is overfit

Train models from leaked passwords



Probabilistic context-free grammar (PCFG)

[Weir et al. “Password Cracking Using Probabilistic Context-free Grammars” 2009]

CFG with probability distribution associated to each rule

Fix a CFG, then learn probabilities by training on passwords

We can encode a string by its parse tree, the tree represented by probabilities in PCFG CDF

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

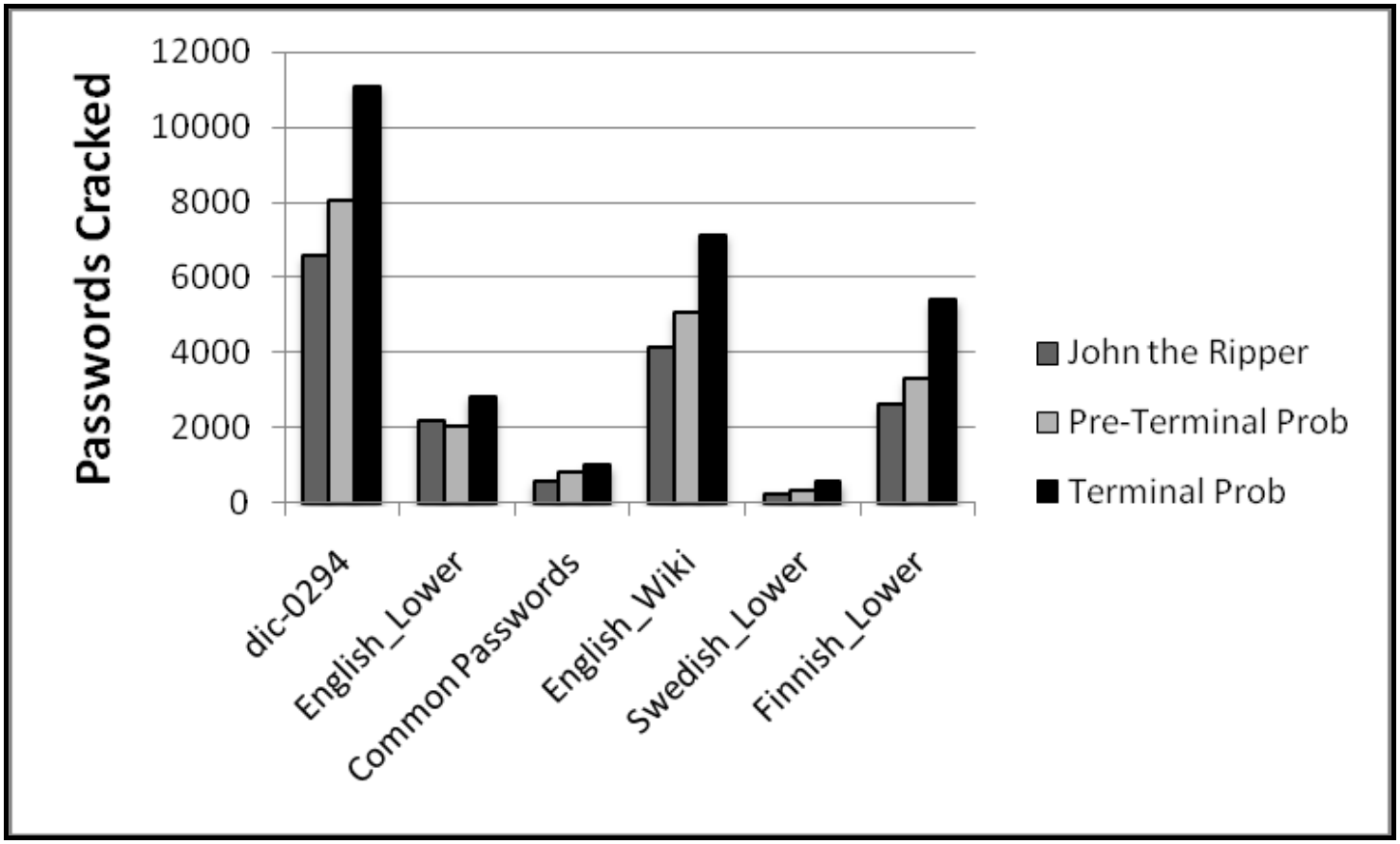
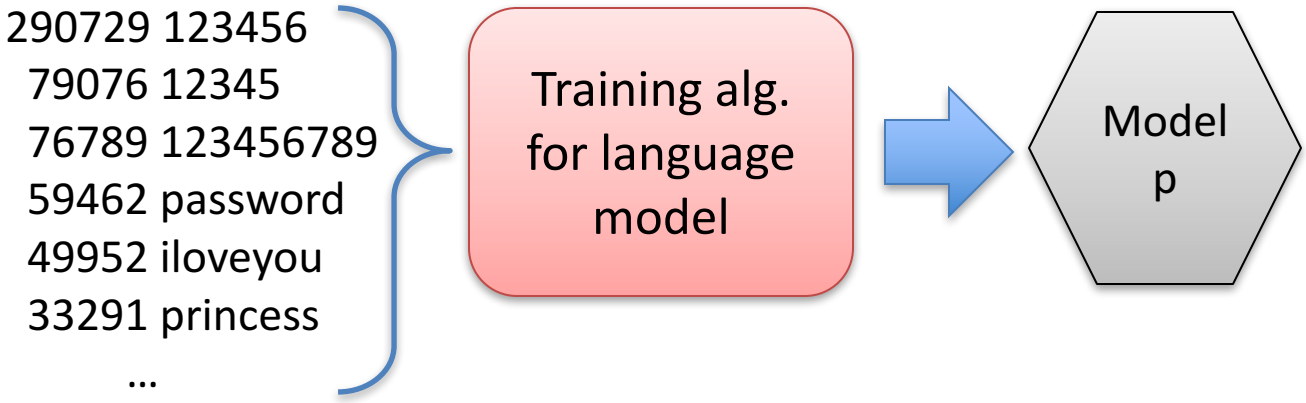


Fig. 4.4.1. Number of Passwords Cracked. Trained on the MySpace Training List. Tested on the MySpace Test List

Train models from leaked passwords



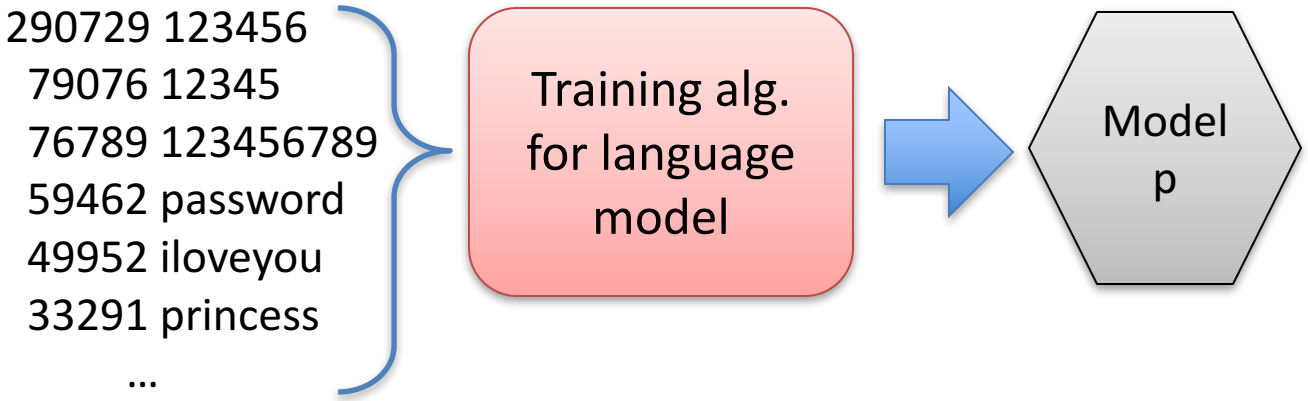
Probabilistic context-free grammar only one NLP modeling approach

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*)|

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

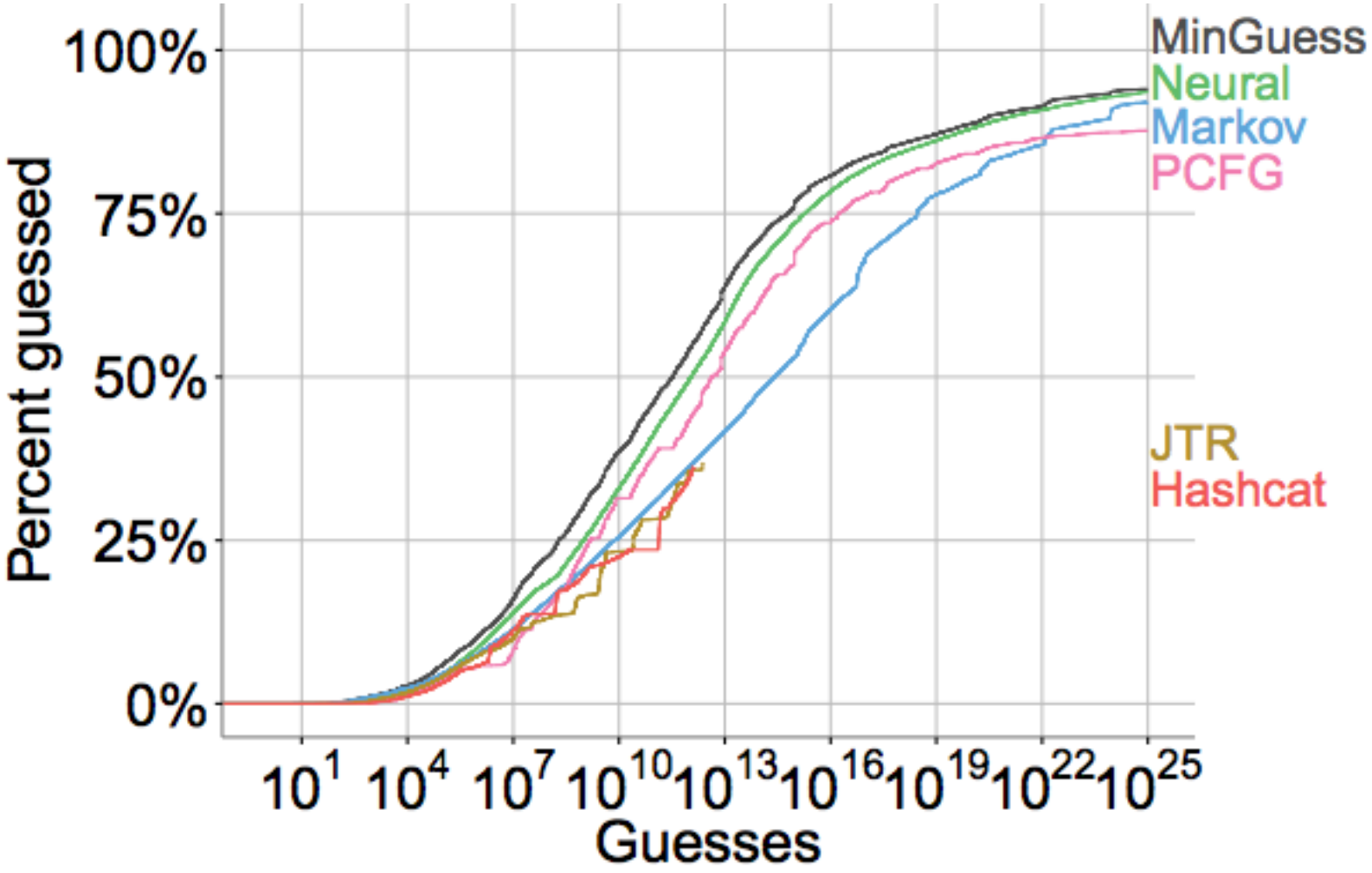
Can estimate S(pw*) using Monte-Carlo techniques [Dell'Amico, Filippone '15]

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

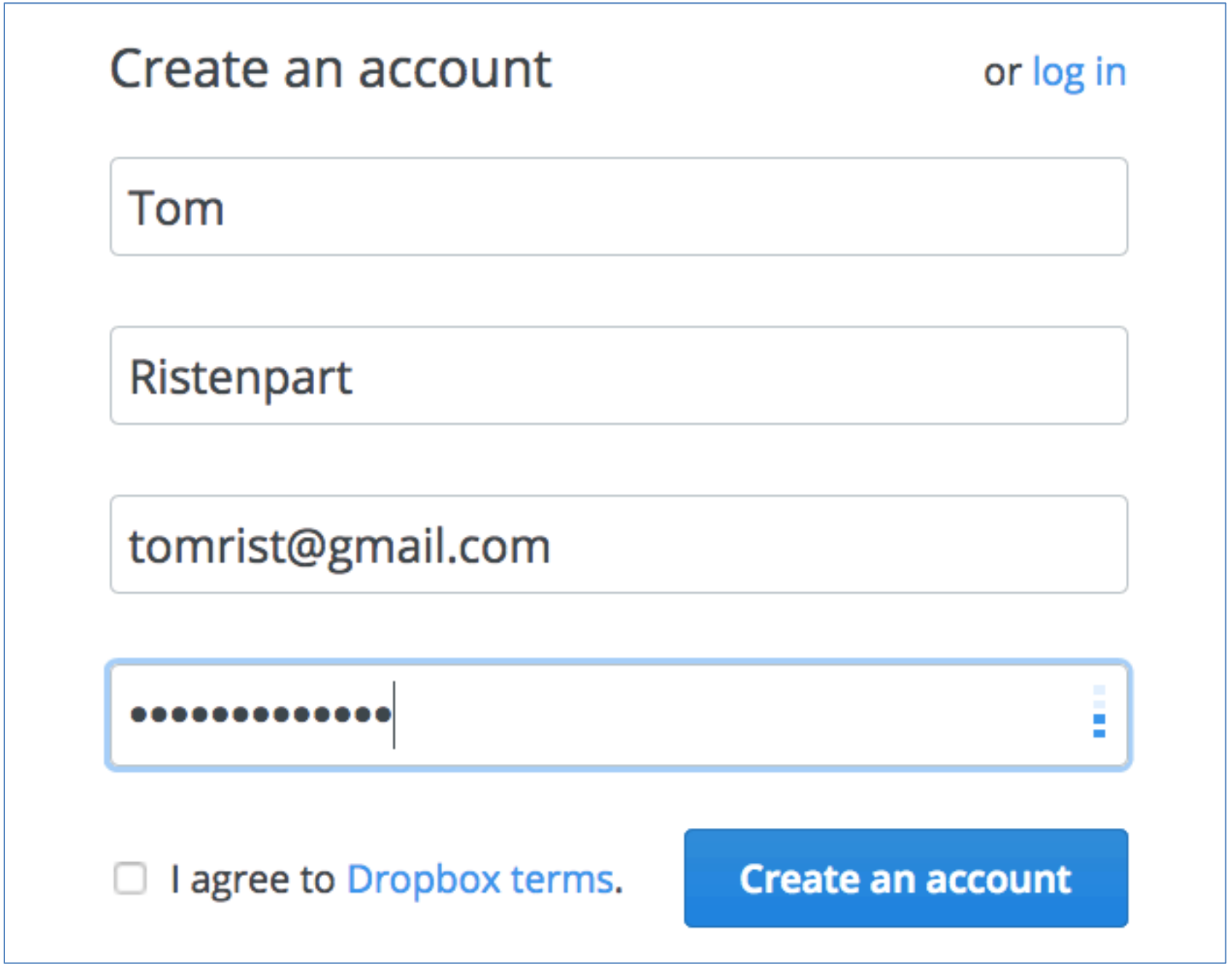
Set of n passwords sampled according to p

Prove 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


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

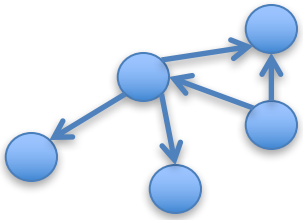
Chatterjee et al. 2017 paper:

Transcription experiment to identify frequent typos

↳ Many typos easily correctable

Analyze typos & their impact for Dropbox users (100s of millions)

↳ Typos huge burden on users / businesses



Formalize typo-tolerant checking
Prove "free corrections theorem"

↳ In theory, can correct typos w/o security loss

Analyze practical security via simulated attacks using password leaks as data

↳ In practice, too

MTurk transcription experiment

TYPE WORDS

The worker is requested to type the words **as they are** in the given input boxes.

- The words are case sensitive (e.g. "CoRrect" is not the same as "Correct").
- The typed characters will show up as dots.
- There is a timer in the page which will start the moment the turker types something on the input boxes and times out after **240 seconds**. The assignment will be auto-submitted once the timer expires. So, please do not start the task unless you are free for few minutes.
- The worker will get $0.07 for finishing the assignment, i.e., typing all the words.
- The worker **must type (and not copy-paste)**. This HIT records only the key presses in the input boxes, so copy-paste will not record the submission.

Please complete at most **one instance** of this HIT from this batch. *This HIT has a script that should prevent you from accidentally completing more than one HITs from the same batch.*

TIMER: **240 sec**

nerdesin1993**	sammie	sadie1
nerdesin1993**	sammie	sadie1
tenecesitomuxo	sawsaw	Evansgirl06
tenecesitomuxo	sawsaw	Evansgirl06
leandro3152	chapis.074	09106051337bikoy13
leandro3152	chapis.074	09106051337bikoy13

MTurk transcription experiment

Passwords sourced from RockYou password leak.

Transcription (as opposed to recall experiments) allowed faster data gathering. 100k passwords submitted for entry

CMU MTurk data provided qualitatively similar results

Typo	Example	% of errors
Caps lock	password1 -> PASSWORD1	10.9%
Case 1st letter	password1 -> Password1	4.5%
Extra char end	password1 -> password12	4.3%
Extra char front	password1 -> apassword1	1.3%
Last symbol to number	password! -> password1	0.2%
Proximity errors	password1 -> psssword1	21.8%
Transcription errors	passwordi -> password1	3.0%
Other	---	53.6%

Examples of *easily correctable typos*

Hard to correct

Relaxed typo-tolerant checkers

tom, Password1

tom	G_K(password1)
alice	G_K(123456)
bob	G_K(p@ssword!)

Easily-corrected typos admit simple correctors:

$f_{caps}(x)$ = x with case of all letters flipped

$f_{1st\text{-}case}(x)$ = x with first letter case flipped, if it is letter

…

Define set C of corrector functions. Let $h = G_K$(password1)

Relaxed checker:

If G_K(Password1) = h then Return 1
For each f in C:
    If G_K(f(Password1)) = h then Return 1
Return 0

Error-correction from error-detection via ball search

Dropbox experiments

tom, Password1

Dropbox

Instrumented password checks to evaluate typo prevalence:

If G_K(Password1) = h then Return 1
For each f in C:
    If G_K(f(Password1)) = h then Log f as successful
Return 0

Did not change effective authentication policy

100s of millions of users
Two 24-hour runs (Top 5 then Top 3 correctors)
Can't give all statistics due to confidentiality reasons

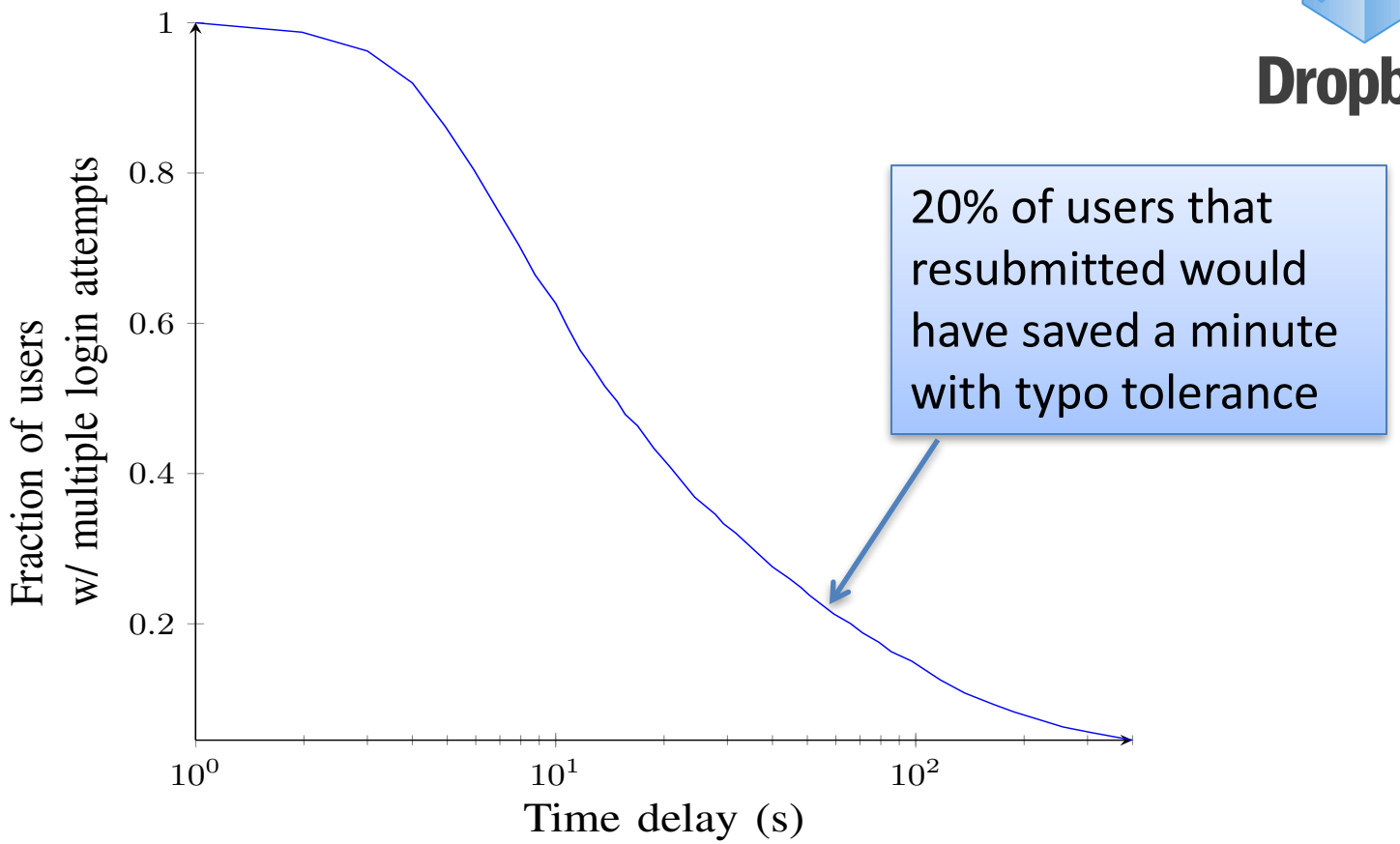
Dropbox experiments: Top 5

tom, Password1

Instrumented password checks to evaluate typo prevalence:

Typo	Example	% of failures
Caps lock	password1 -> PASSWORD1	1.13%
Case 1st letter	password1 -> Password1	5.56%
Extra char end	password1 -> password12	2.05%
Extra char front	password1 -> apassword1	0.35%
Last symbol to number	password! -> password1	0.21%
	Total:	9.30%

Dropbox experiments: Top 3



Typo tolerance would add several person-months of login time

3% of ***users*** couldn't log in during 24 hour period, but could have with one of Top 3 correctors

Online guessing speedup fallacy

tom, password

fail

tom	G_K(password1)
alice	G_K(123456)
bob	G_K(p@ssword!)

Can rule out 3 possible passwords if correct C_{top2}

3x speed-up in guessing attacks???

Is Vanguard Making It Too Easy for Cybercriminals to Access Your Account?

By **Susan Antilla** | 08/10/15 - 12:27 PM EDT

Online guessing attacks

Password list
pw_1 = 123456
pw_2 = 12345
pw_3 = password
pw_4 = iloveyou
pw_5 = Password
…

tom, 123456
fail
tom, 12345
fail
…

Stored password is G_K(password1)
Check G_K(123456)
Check G_K(12345)

Given budget q of queries
$Pr[success] \leq \lambda_q = \Sigma_{i=1..q} Pr[pw_i]$

Typo tolerant case (e.g., C_{top2})

Password list
pw_1 = 123456
pw_2 = 12345
pw_3 = password
pw_4 = iloveyou
pw_5 = Password
…

tom, 123456
fail
tom, password
fail
…

Stored password is G_K(password1)
Check G_K(123456)
Check G_K(password) G_K(Password) G_K(PASSWORD)

Let λ_q^* be maximal success probability
Finding optimal sequence of queries is NP-hard. Approximate w/ greedy

Security gap between exact and tolerant?

Best-case

No password typo of another

$\lambda_q^* = \lambda_q$

pw_1 = 123456
pw_2 = 12345
pw_3 = password
pw_4 = iloveyou
pw_5 = password1
pw_6 = password123

Worst-case

All passwords typos of another.
Passwords uniformly distributed

$\lambda_q^* = 3\,\lambda_q$

pw_1 = password
pw_2 = PASSWORD
pw_3 = Password
pw_4 = Loveyou
pw_5 = loveyou
pw_6 = LOVEYOU

Situations in practice will lie somewhere in between:

$$\lambda_q \leq \lambda_q^* \leq 3\,\lambda_q$$

Free corrections theorem

Thm. For any password distribution, set of correctors C, and adversarial query budget q, there exists a typo-tolerant checker for which $\lambda_q^* = \lambda_q$.

Construction:
For any password, check as many typos as one can while ensuring correctness and that $\Sigma_{\text{pw in ball}}\ p(\text{pw}) \leq p(\text{pw}_q)$

Ensures optimal adversarial strategy is to query $\text{pw}_1,\ldots,\text{pw}_q$ against typo-tolerant checker. Same as for strict checker

Remaining questions:

(1) Theorem rests on construction knowing exact password distribution

(2) Non-optimal attackers may get improvements

Empirical security analyses

Use password leaks as empirical password distributions

Typo-tolerant checker estimates distribution using RockYou

Simulate attacker following greedy strategy for typo-tolerant checker
Assume attacker knows challenge distribution exactly

q = 100

Challenge Distribution	Correctors	Exact success	Greedy strategy
phpbb	C_{top2}	5.50%	5.50%
phpbb	C_{top3}	5.50%	5.51%
Myspace	C_{top2}	2.86%	2.89%
Myspace	C_{top3}	2.86%	3.18%

Attackers that estimate challenge distribution incorrectly:

- Often perform worse when trying to take advantage of tolerance
- (See paper for details)

Carefully designed typo-tolerant checkers can improve user experience w/o degrading security

The research landscape since 1979…

- **Understanding user password selection**
  - Measuring password strength [see citations in Bonneau paper], [Li, Han `14], [CMU papers]
  - Measuring password reuse
- **Usability**
  - Strength meters, requirements, etc. [Komanduri et al. '11] [Dell'Amico, Filippone '15] [Wheeler '16] [Melicher et al. '16]
  - Password expiration [Zhang et al. '12]
  - Typo-tolerance [Chatterjee et al. `16]
- **Password transmission, login logic**
  - Single sign-on (SSO) technologies
  - Password-based authenticated key exchange [Bellovin, Merritt '92]
- **Password hashing**
  - New algorithms [PKCS standards], [Percival '09], [Biryukov, Khovratovich '15]
  - Proofs [Wagner, Goldberg '00] [Bellare, Ristenpart, Tessaro '12]
- **Improving offline brute-force attacks**
  - Time-space trade-offs (rainbow tables) [Hellman '80], [Oeschlin '03], [Narayanan, Shmatikov '05]
  - Better dictionaries [JohntheRipper], [Weir et al. '09], [Ma et al. '14]
- **Password managers**
  - Decoy-based [Bojinov et al. '10], [Chatterjee et al. '15]
  - Breaking password managers [Li et al. '14] [Silver et al. '15]
  - Stateless password managers [Ross et al. '05]