# Passwords

Tom Ristenpart
CS 6431

# The game plan

- Basics, history
  - Morris and Thompson 1979
- The research landscape
- Current practices in industry
- Measuring password distributions:
  - Florencio & Herley (client-side measurement)
  - Bonneau (server-side measurement)
  - Understanding password strength metrics

# Password use cases

- OS login
- Website / service login
- PINs
- Encryption

Authentication

Confidentiality

# Morris-Thompson paper

Register:
tom, pw

Authentication service

Store tom, pw in some form.

login:
tom, pw'

Check that pw' = pw

What are security threats?

# Brute-force attacks

- **Offline brute-force attacks**
  - Compromise database
  - E.g.: “cracking” via dictionary attacks
  - *Countermeasures*: hash passwords with purposefully slow-to-compute cryptographic hash function

    (was: MD5, SHA-1  now: argon2, scrypt)
- **Online brute-force attacks**
  - E.g: Submit guesses to web site
  - *Countermeasures*: Rate limit, account lockout

- **Shoulder surfing, compelled password disclosure, malware, side-channels, ...**

# Brute-force attacks

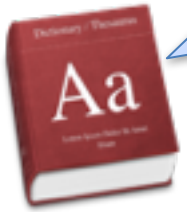
- **Offline brute-force attacks**
  - Compromise database
  - E.g.: “cracking” via dictionary attacks
  - *Countermeasures*: hash passwords with purposefully slow-to-compute cryptographic hash function

    (was: MD5, SHA-1  now: argon2, scrypt)
- **Online brute-force attacks**
  - E.g: Submit guesses to web site
  - *Countermeasures*: Rate limit, account lockout

- **Shoulder surfing, compelled password disclosure, malware, side-channels, ...**

# Offline brute-force attacks

$h_1 = H(pw_1)$
$h_2 = H(pw_2)$
…
$h_m = H(pw_m)$




Nowadays:
Use password leaks to inform dictionary

Dictionary:
List of probable passwords
or way of generating them

$H(guess_1)$ , $H(guess_2)$ , …

Check if any guesses equal any of $h_1, \ldots , h_m$

“Human beings being what they are, there is a strong tendency for people to choose relatively short and simple passwords that they can remember. “

[Morris, Thompson 1979]

# “The results were disappointing, except to the bad guy.”

“In a collection of 3,289 passwords gathered from many users over a long period of time,

15 were a single ASCII character;
72 were strings of two ASCII characters;
464 were strings of three ASCII characters;
477 were strings of four alphamerics;
706 were five letters, all upper-case or all lower-case;
605 were six letters, all lower-case.”

[Morris, Thompson 1979]

86% crackable

Morris, Thompson suggest:

- Slow hashing (they called it encryption)
- Less predictable passwords (pw requirements)
- Salt before hashing
- Use custom version of DES to avoid hardware
- Avoid timing attacks to distinguish bad login

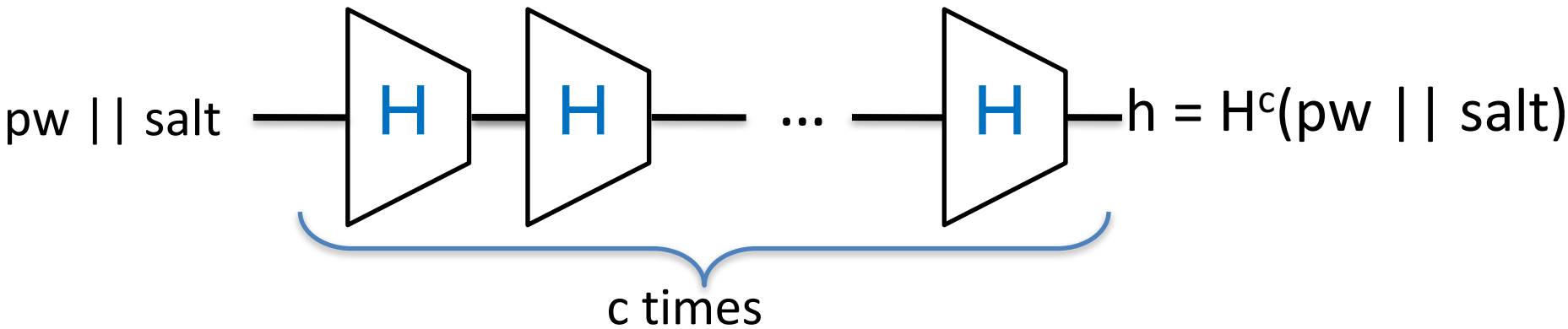
"We did not attempt to hide the security aspects of the operating system, thereby playing the customary make-believe game in which weaknesses of the system are not discussed no matter how apparent."

# 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]

# Password hashing

- Recall Morris, Thompson goal: slow down brute-force attacks
- PKCS#5 approach:




$H : \{0,1\}^* \rightarrow \{0,1\}^n$ is cryptographic hash function (e.g., SHA-256)

salt should be random bit string large enough to be unpredictable

# Password hashing

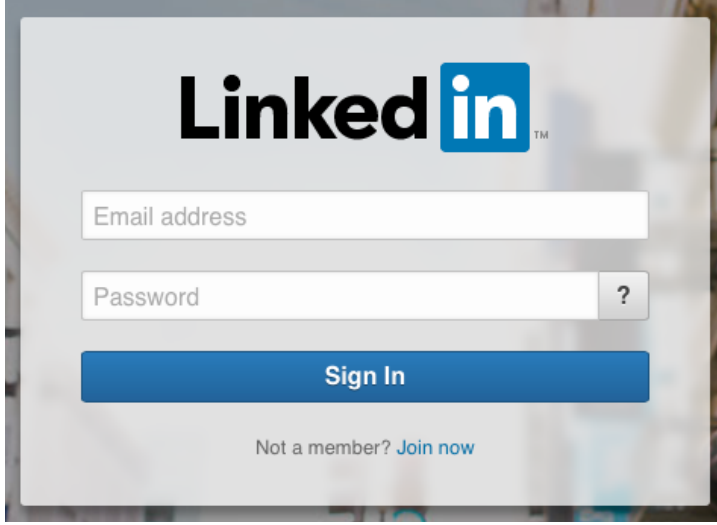
- Recall Morris, Thompson goal: slow down brute-force attacks
- The role of salts:
  - Prevents use of time-memory trade-offs (rainbow tables)
  - Cracking m accounts requires m times the work

$$h_1 = H^c(pw_1, salt_1)$$
$$h_2 = H^c(pw_2, salt_2)$$
$$\ldots$$
$$h_m = H^c(pw_m, salt_m)$$

Proofs:
See [Bellare, Ristenpart, Tessaro '12]

Linked in circa 2012 stored passwords as which of:

- pw
- MD5(pw)
- H(salt,pw)
- $H^c$(salt,pw)



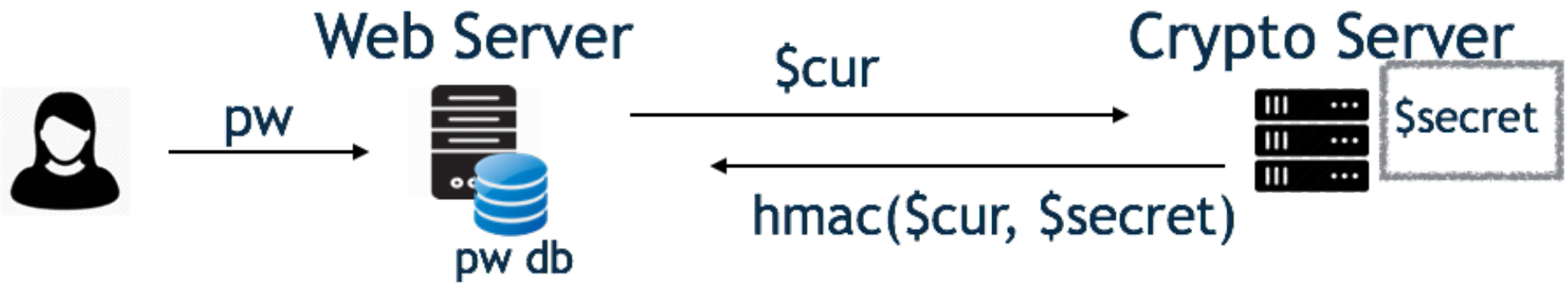

2012: 6.5 million hashes leaked onto Internet
90% cracked in 2 weeks

2016: 177.5 million more hashes leaked
98% cracked in 1 week

http://arstechnica.com/security/2016/06/how-linkedins-password-sloppiness-hurts-us-all/

# Facebook password “onion”

```
$cur  = ‘password’
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```




Split-trust model:
must compromise both servers to mount offline brute-force

# Understanding password strength

(1) Empirical studies of user passwords

Leaks

Instrumentation of large web systems (Bonneau paper)

Instrumentation of clients (Florencio & Herley)

(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

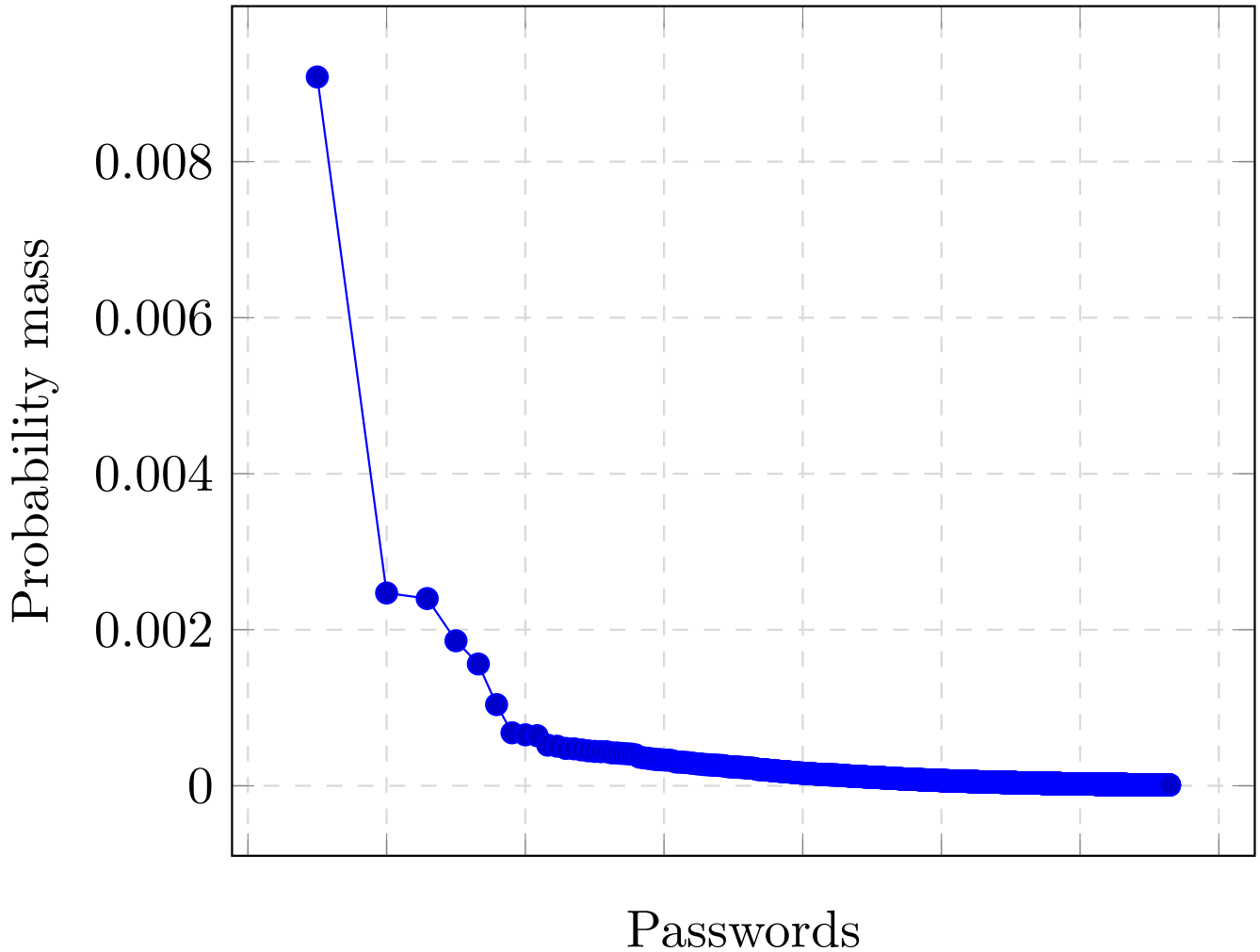
290729 123456
79076 12345
76789 123456789
59462 password
49952 iloveyou
33291 princess
21725 1234567
20901 rockyou
20553 12345678
16648 abc123
16227 nicole
15308 daniel
15163 babygirl
14726 monkey
14331 lovely

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)

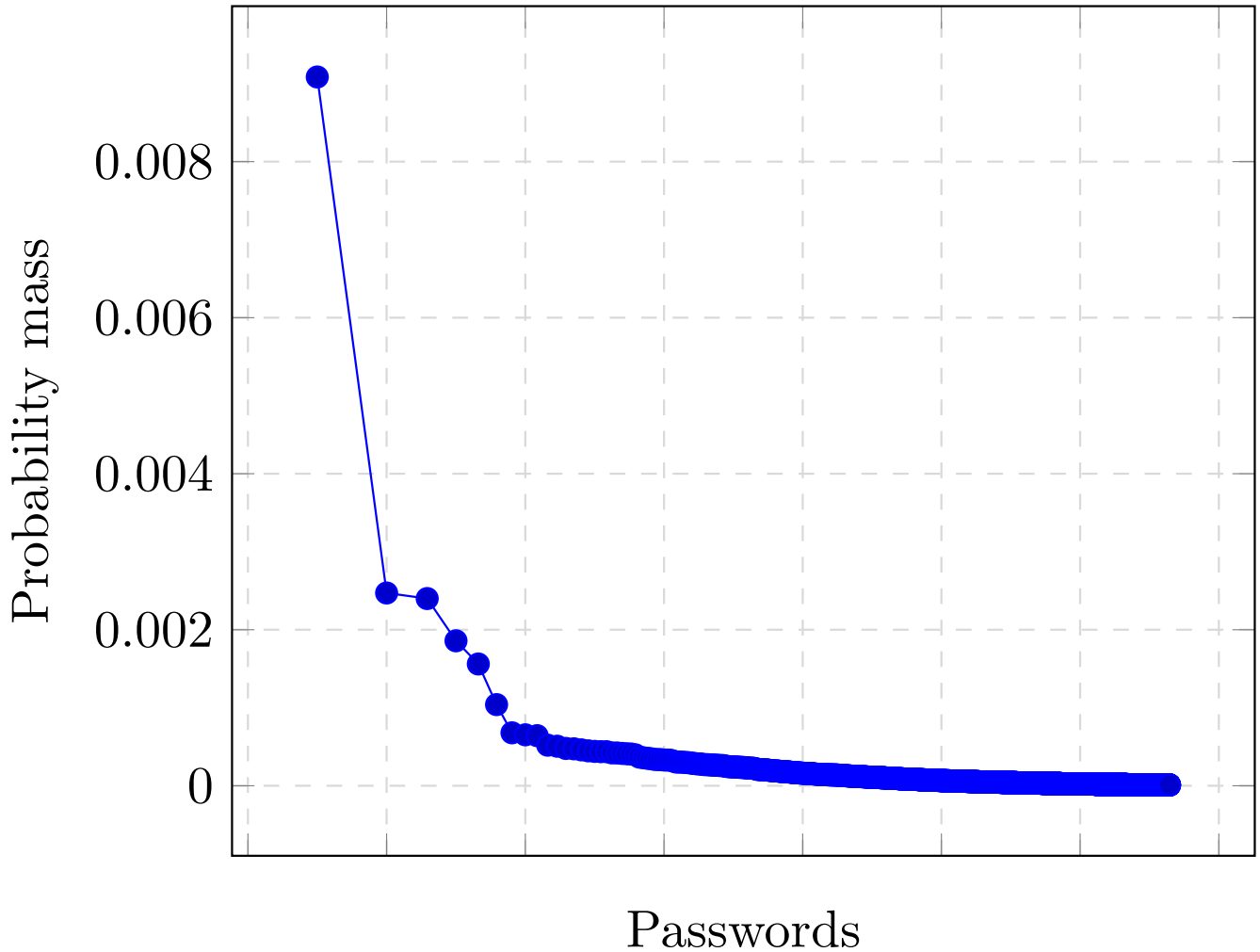
# Florencio & Herley study

- Instrument Windows Live toolbar
  - 544,960 clients opted-in to study
- Captured passwords typed into browser
  - Hashed and stored locally
  - Sent report to server about (quantized) password strength, associated URL, etc.

# Florencio & Herley 2007

- Avg user:
  - Has 6.5 passwords, each used at 3.9 different sites
  - Has 25 accounts requiring passwords
  - Types 8 passwords per day
  - Selects 40.54 “bitstring” password
- ~1.5% of Yahoo users forget their passwords each month (!)

# Rockyou empirical probability mass function




(Only first 5,000 points shown)

# Bonneau Yahoo password study

- Instrument login infrastructure
  - 69 million accounts monitored
- Hash passwords with key H(K,pw) and store result in histogram
- Throw away K
  - Can’t do brute-force attacks later on
  - Only learn empirical distribution of passwords
- Also stored some demographic information
- How do we measure strength of password distribution?

# 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) = $\text{Alphasize(pw)}^{\text{len(pw)}}$
- Simpler than classical NIST entropy estimate

# Password strength metrics

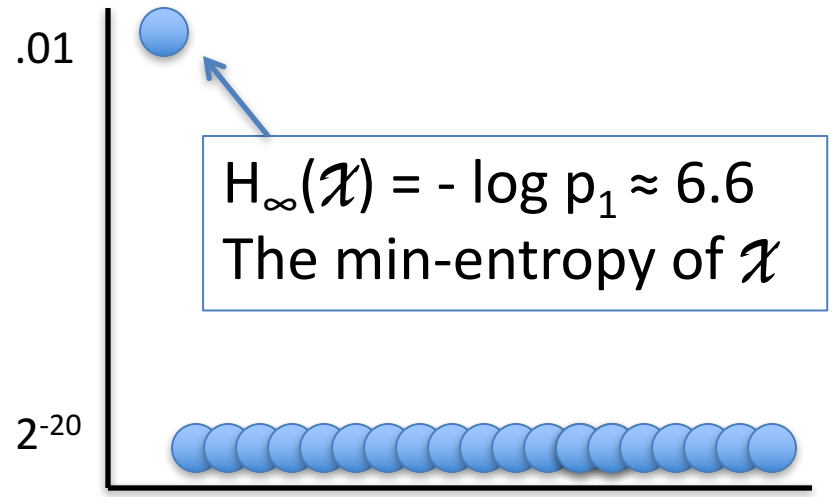
Let $\mathcal{X}$ be password distribution.
Passwords are drawn iid from $\mathcal{X}$
N is size of support of $\mathcal{X}$
$p_1$ , $p_2$ , …, $p_N$ are probabilities of passwords in decreasing order

Shannon entropy:

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

# 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**

# Password strength metrics

Beta-success rate:

$$\lambda_\beta(\mathcal{X}) = \sum_{i=1}^{\beta} p_i \qquad \tilde{\lambda}(\mathcal{X}) = \log(\beta/\lambda_\beta(\mathcal{X}))$$

Alpha-work-factor:

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

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

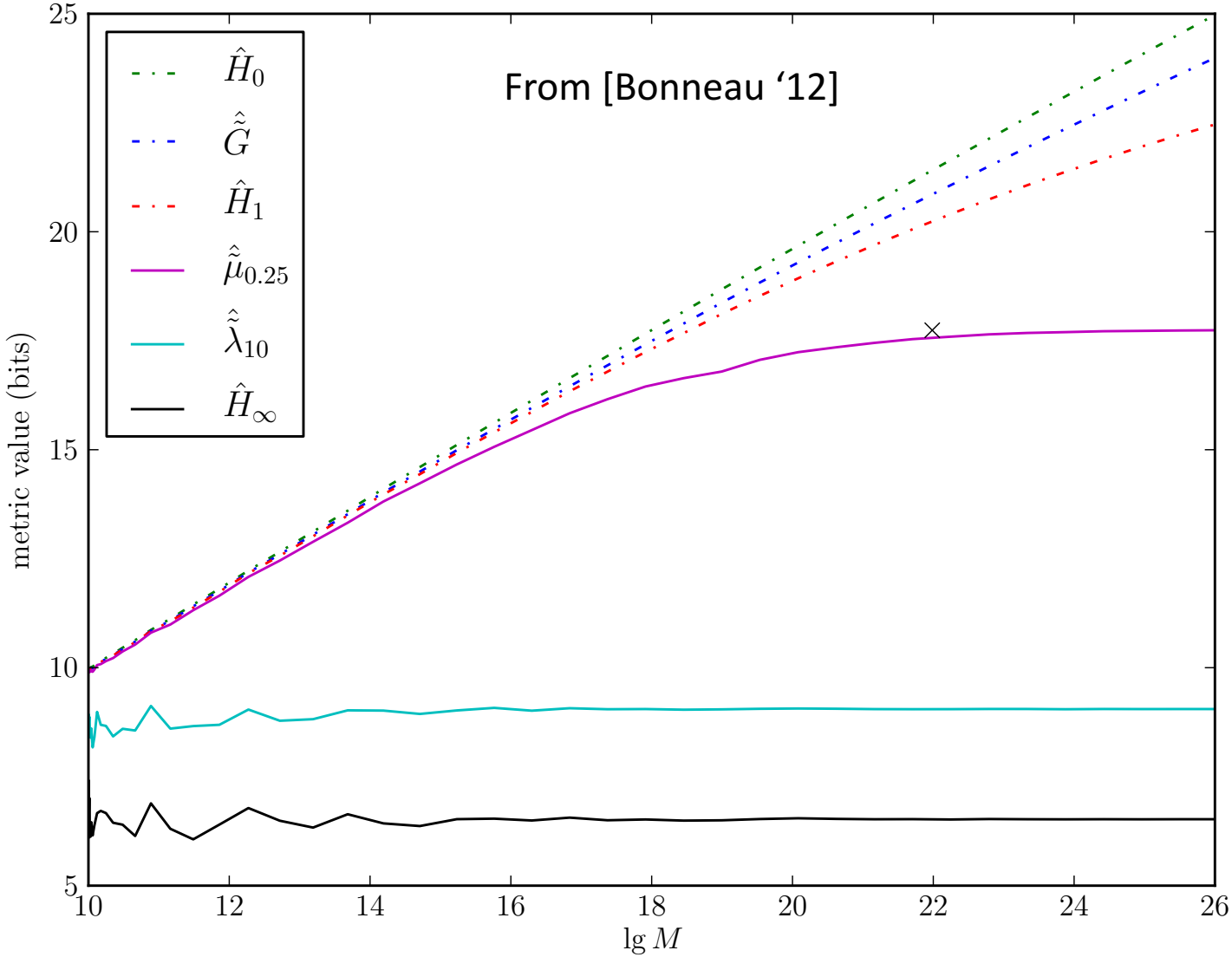
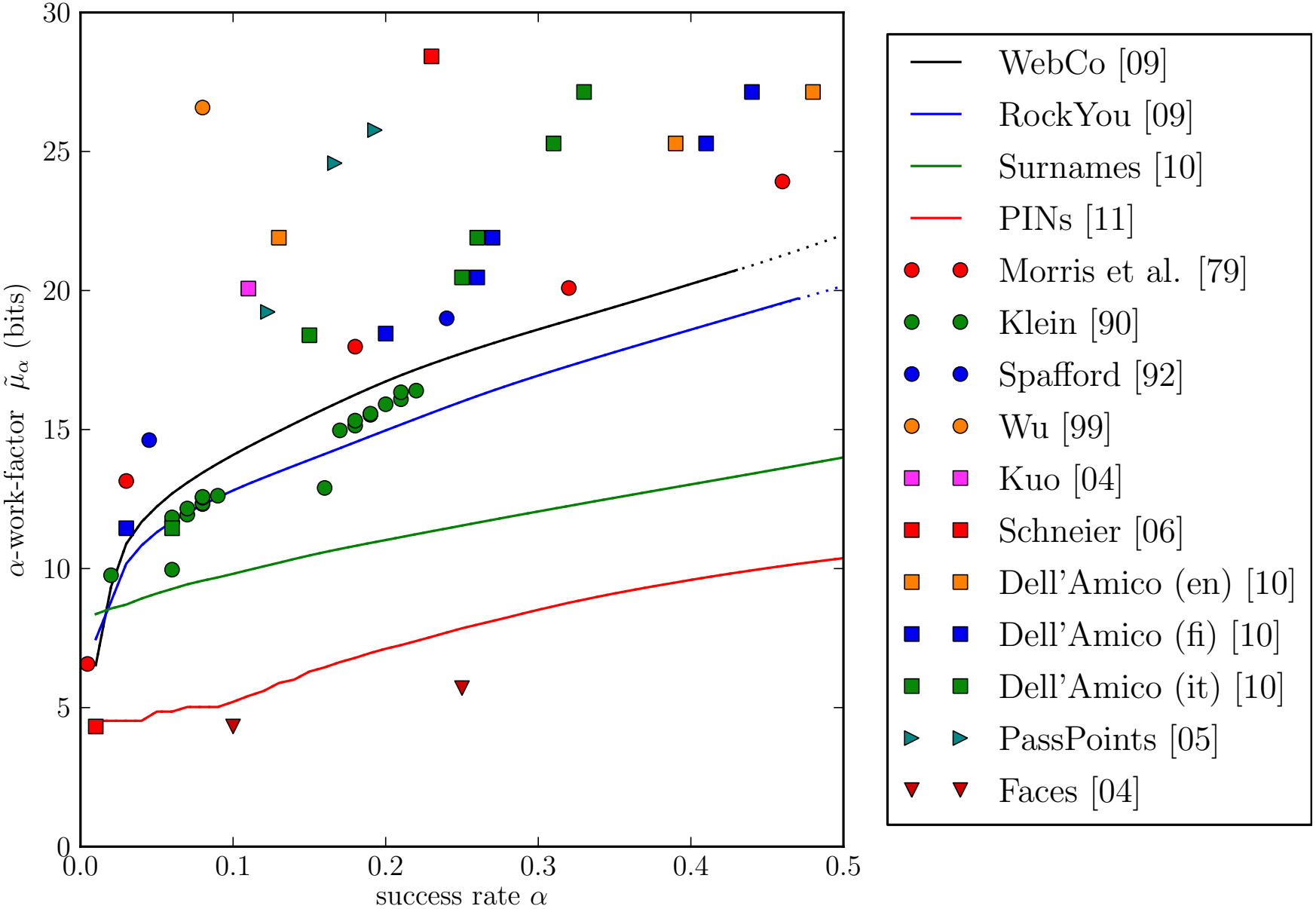



Figure 3. Changing estimates of guessing metrics with increasing sample size $M$. Estimates for $H_\infty$ and $\tilde{\lambda}_{10}$ converge very quickly; estimates for $\tilde{\mu}_{0.25}$ converge around $M = 2^{22}$ (marked $\times$) as predicted in Section V-A. Estimates for $H_0$, $H_1$, and $\tilde{G}$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




From [Bonneau '12]

# Bonneau takeaways

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