



THE UNIVERSITY OF
CHICAGO



Security, Usability, & Privacy
Education & Research

RUHR
UNIVERSITÄT
BOCHUM

RUB

Password

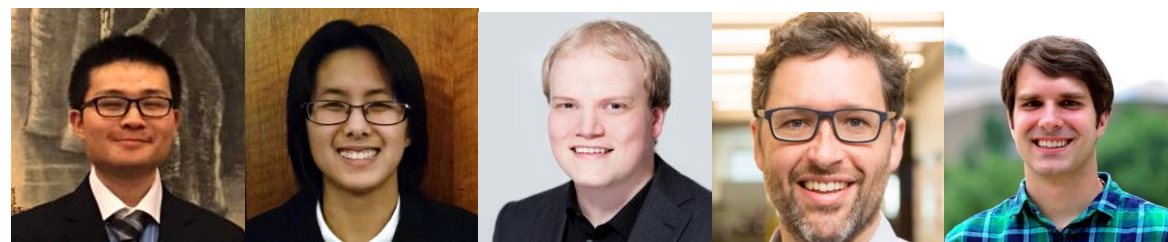


Weak



Reasoning Analytically About Password-Cracking Software

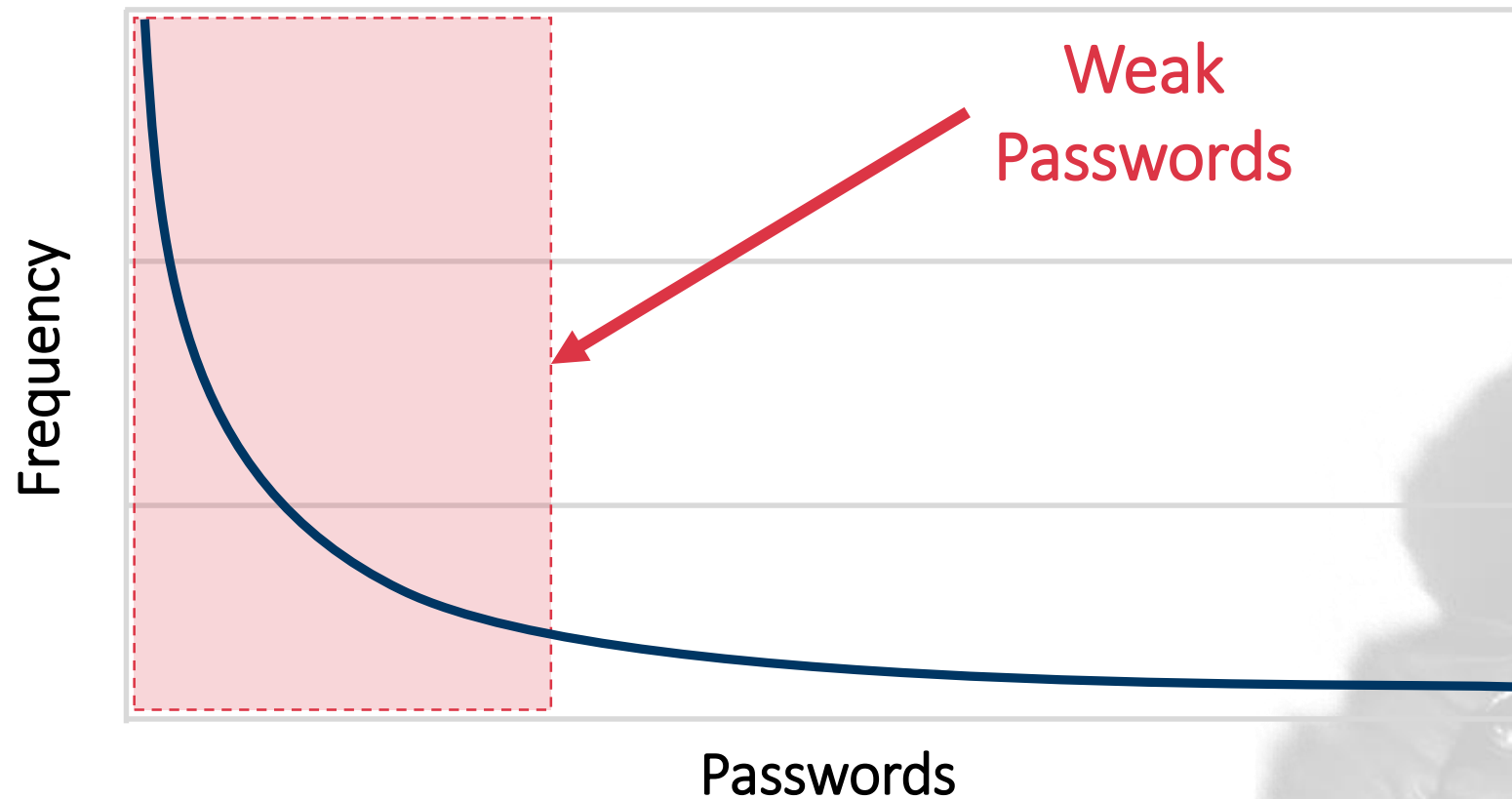
Enze “Alex” Liu, Amanda Nakanishi, Maximilian Golla, David Cash, and Blase Ur



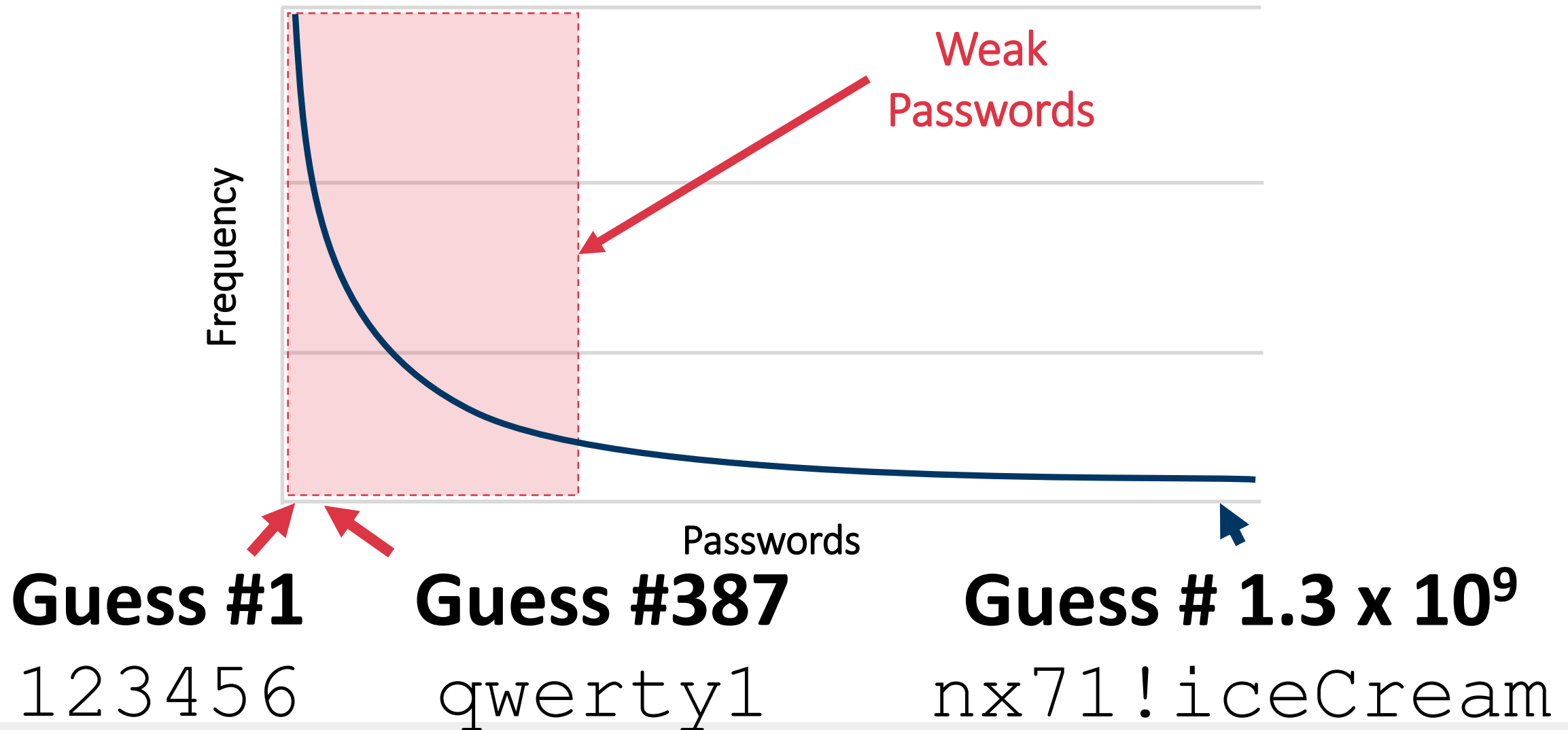
People Choose Weak Passwords



What Makes a Password “Weak”?



What Makes a Password “Weak”?



Guess Number = Approximate Strength

Example:

Johnny14!

Guess #:
390,000



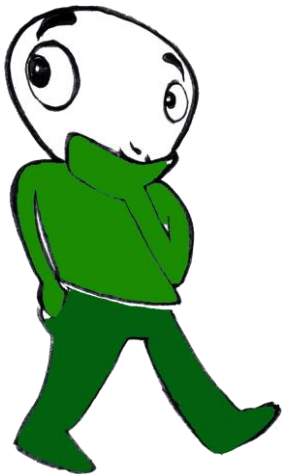
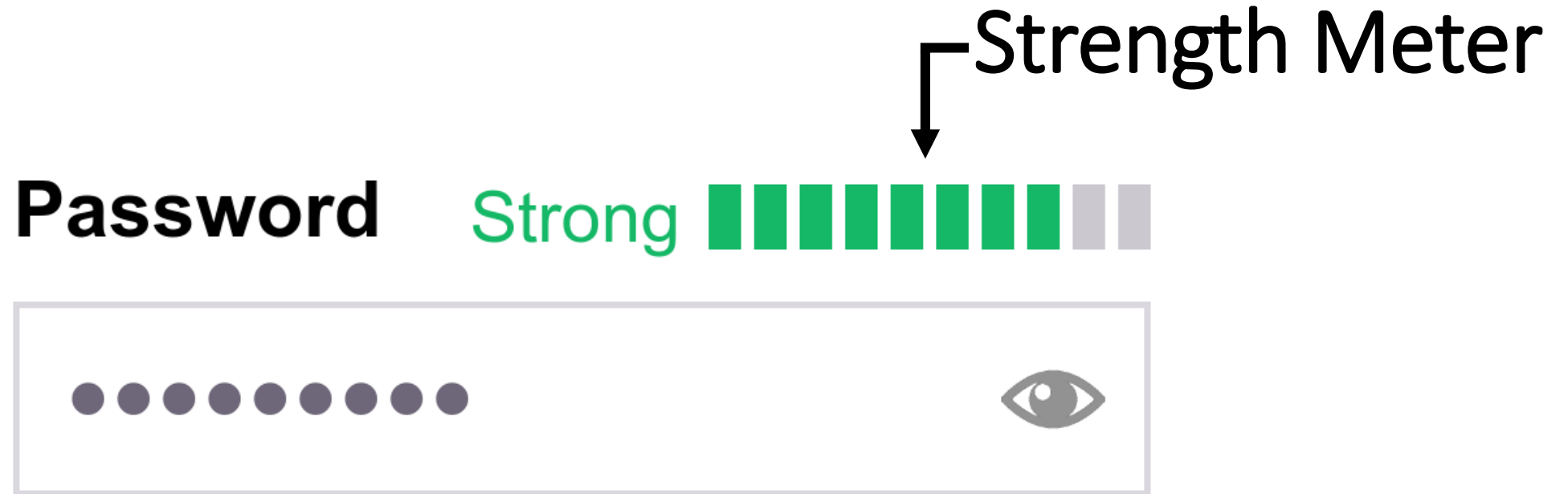
Guess Number

/ɡɛs 'nʌmbə/

noun

1. The number of guesses required to guess a password.

Application 1: Strength Meters



Application 2: Proactive Checking

Password123!

Application 2: Proactive Checking



Application 3: Academic Research

Do Users' Perceptions of Password Security Match Reality?

Blase Ur, Jonathan Bees*, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor
Carnegie Mellon University, *The Pennsylvania State University
(bur, ssegreti, lbauer, nicolasc, lorrie)@cmu.edu, *jfb5406@psu.edu

ABSTRACT

Although many users create predictable passwords to which users realize these passwords are not well understood. We investigate the relationship between users' perceptions of the strength of their actual strength. In this 165-participant study, we ask participants to rate the comparative security of fully juxtaposed pairs of passwords, as well as the memorability of both existing passwords and password-creation strategies. Participants had varying perceptions about the impact of basing passwords on phrases and including digits and keyboard path words. However, in most other cases, participants' perceptions of what characteristics make a password secure did not match the performance of current password creation. We find large variance in participants' understanding of password security, potentially explaining why passwords may be attacked, potentially explaining why passwords are not predictable passwords. We design directions for helping users make better password choices.

Author Keywords

User behavior; perceptions of security; password authentication; users' folk models; usable security

ACM Classification Keywords

H.5.m Information Interfaces and Presentation Miscellaneous; K.6.5 Security and Protection: Authentication

INTRODUCTION

For better or worse, passwords remain today's de facto method of user authentication [11]. While the predicted chosen passwords have been widely documented [74, 77, 80], very little research has investigated the relationship between users' perceptions of password security. That is, do users rate selecting terrible passwords and choose to do so or are they unwittingly creating weak passwords because they believe they are making secure ones?

In this paper, we report on a 165-participant study of users' perceptions of password security. Participants' perceptions about the security and memorability of passwords are compared to the actual security of their passwords.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee for profit or commercial advantage and that copies that bear this notice and the full copyright notice must appear on the first page. Copyright for third-party components of this work must be acknowledged. Copyright is held by the author(s). 0736-1075/15/07-1234-1234. ACM 978-1-4558-2805-6/15/07-1234-1234. http://dx.doi.org/10.1145/2805636.2805636

"I Added '!' at the End to Make It Secure": Observing Password Creation in the Lab

Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor
Carnegie Mellon University
(bur, fnoma, jbees, ssegreti, rshay, lbauer, nicolasc, lorrie)@cmu.edu

ABSTRACT

Users often make passwords that are easy for attackers to guess. Prior studies have documented features that lead to easily guessed passwords, but have not probed why users craft weak passwords. To understand the genesis of common password patterns and uncover average users' misconceptions about password strength, we conducted a qualitative interview study. In our lab, 49 participants each created passwords for fictitious banking, email, and news website accounts while thinking aloud. We then interviewed them about their general strategies and inspirations. Most participants had a well-defined process for creating passwords. In other cases, participants consciously made weak passwords. In other cases, however, weak passwords resulted from misconceptions, such as the belief that adding "!" to the end of a password instantly makes it secure or that words that are difficult to spell are more secure than easy-to-spell words. Participants commonly anticipated only very targeted attacks, believing that using a birthday or name is secure if those data are not on Facebook. In contrast, some participants made secure passwords using unpredictable phrases or non-standard capitalization. Based on our data, we identify aspects of password creation ripe for improved guidance or automated intervention.

1. INTRODUCTION

Despite decades of research investigating passwords, many users still make passwords that are easy for attackers to guess [9, 22, 35, 62]. Predictable passwords continue to cause problems, as evidenced by the recent release of celebrities' private photos obtained in part through a password-guessing attack on Apple's iCloud [11, 37]. While most everyone would prefer a world without the burden of remembering a portfolio of passwords [18, 53], passwords are familiar, easy to implement, and do not require that users carry anything. As a result, passwords are unlikely to disappear entirely in the near future [7]. Although expecting users to remember complex and distinct passwords for dozens of accounts is absurd, single-sign-on systems, software password managers, and biometrics [4] promise to reduce this burden. Passwords also remain useful for frequently accessed accounts, as master passwords for password managers, and as an integral part of two-factor authentication.

Copyright is held by the author(s). Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee. Symposium on Usable Privacy and Security (SUSOPS) 2015, July 22-24, 2015, Ottawa, Canada.

USENIX Association

2015 Symposium on Usable Privacy and Security 123

Measuring Password Guessability for an Entire University

Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley*, Richard Shay, and Blase Ur

Carnegie Mellon University
Pittsburgh, PA
sarangak, tvidas, lbauer, p, rshay, burj@cmu.edu

*University of New Mexico
Albuquerque, NM
pgk@cs.unm.edu

1. INTRODUCTION

Empirical studies of password guessability have been specifically collected. Properties of password guessability are poorly understood. In-on passwords used at a research university, we describe the many precautions we take to analyze how guessable passwords are by attacking algorithms. We find that the number of demographic factors that influence password guessability is more complex than we anticipated. For example, we find that school make passwords more guessable. We find more common passwords and password composition policies and subsets of passwords that appear to comply with password composition policies.

Information Systems)

This work for personal or for profit or commercial advantage and that copies that bear this notice and the full copyright notice must appear on the first page. Copyright is held by the author(s).

Researchers have documented the numerous problems of text passwords for decades — passwords are easy to guess, hard to remember, easily stolen, and vulnerable to observation and replay attacks (e.g., [28, 38]). The research community has invested significant effort in alternatives including biometrics, graphical passwords, hardware tokens, and federated identity; however, text passwords remain the dominant mechanism for authenticating people to computers, and seem likely to remain that way for the foreseeable future [5, 23]. Better understanding of text passwords therefore remains important.

Considerable effort has been spent studying the usage and characteristics of passwords (e.g., [13, 17, 34, 35, 45]), but password research is consistently hampered by the difficulty in collecting realistic data to analyze. Prior password studies all have one or more of the following drawbacks: very small data sets [36], data from experimental studies rather than from deployed authentication systems [31], no access to plaintext passwords [3], self-reported password information [47], leaked data of questionable validity, or accounts of minimal value [26, 53]. As a result, the importance of whether the results apply to real, high-value passwords has remained open.

In this paper, we study more than 25,000 passwords making up the entire user base of Carnegie Mellon University (CMU). Notably, these passwords are the high-value gatekeeper to most end-user (i.e., non-administrative) online functions within the university, including email, grading systems, transcripts, financial data, health data, payroll, and course content. Furthermore, these passwords were created under a password-composition policy among the stricter of those in common use [18], requiring a minimum of eight characters and four different character classes. Using indirect access to the plaintext of these passwords, we measure their strength. In addition, we obtain contextual information from personnel databases, authentication logs, and a survey about password creation and management, and correlate these factors with password strength. To acquire this data, we established a partnership with the CMU information technology division; the research was also vetted by our Institutional Review Board (IRB). Our approach to analyzing this sensitive data securely provides a blueprint for future research involving security-sensitive data in the wild.

Using this data, we make two important and novel contributions to the field of password research. First, we identify interesting trends in password strength, measured as resistance to offline guessing attacks, in which an attacker attempts to recover plaintext passwords from their hashes [2, 6, 44]. Using statistical methods adopted from survival analysis, we find that users associated with science and technology colleges within the university make



Guess # Depends on Model

We don't think in "cracks," we think in guess numbers!



Guess #:
1,928,730,033

!=



Guess #:
8,346,290,721

!=




Guess #:
inf.

!=



Guess #:
390,000

Password Cracking:
Johnny14! - cracked 

Guess Number:
Depends on
"trained" model

Goals For Guess Numbers

1. Compute guess numbers efficiently
2. Configure guessing method systematically
3. Approximate real-world attack

Password



Efficient

Guess
Number

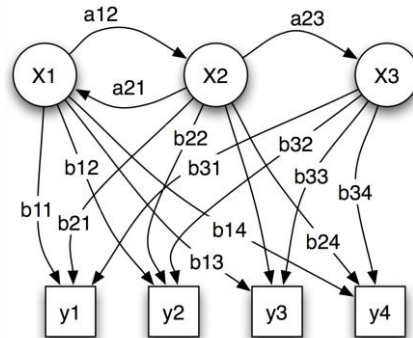


Outline

1. State of the art
2. How software password-cracking tools work
3. Our efficient techniques for guess numbers
4. Our techniques for systematic configuration

Password-Cracking Methods

Probabilistic Models



Software Tools



Probabilistic Models



Markov Models [Narayanan and Shmatikov, CCS 2005]

Probabilistic Context-Free Grammars [Weir et al., S&P 2009]

Neural Networks [Melicher et al., USENIX Security 2016]

Guess #



Configuration

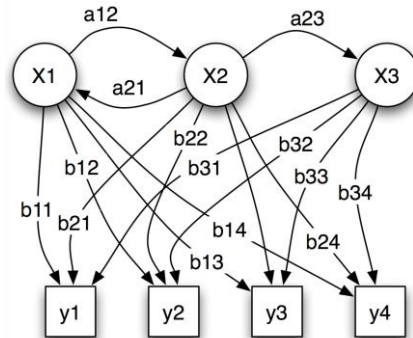


Real



Password-Cracking Methods

Probabilistic Models



Software Tools



Software Tools



John the Ripper



Hashcat



Guess Number by Enumeration

1. 123456
2. password

Does Not Scale !!!

5. p@ssw0rd
6. Johnny14!



Software Tools



John the Ripper



Hashcat



Guess #
Configuration
Real



Reasoning Analytically About
Password-Cracking Software

[S&P 2019]

Enze Liu, Amanda Nakanishi, Maximilian Golla[†], David Cash, Blase Ur
University of Chicago, [†] Ruhr University Bochum

Outline

1. State of the art
2. How software password-cracking tools work
3. Our efficient techniques for guess numbers
4. Our techniques for systematic configuration

Mangled Wordlist Attack



Wordlist

Super
Password
Chicago

Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go

Mangled Wordlist Attack



Wordlist

Super
Password
Chicago

Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go
super
password
chicago

Example Wordlists and Rulelists



Wordlist

Rulelist

Linkedin ($\approx 60,000,000$)

HIBP ($\approx 500,000,000$)

Korelogic ($\approx 5,000$)

Megatron ($\approx 15,000$)

Generated2 ($\approx 65,000$)



$10^9 - 10^{15+}$
guesses

+ Professionals' private word/rule lists

Outline

1. State of the art
2. How software password-cracking tools work
3. **Our efficient techniques for guess numbers**
4. Our techniques for systematic configuration

Is This Password in the Guesses?

Chic4go

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go
super
password
chicago

Is This Password in the Guesses?

Wordlist

Super
Password
Chicago

Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go
super
password
chicago

Insight

We can work backwards!

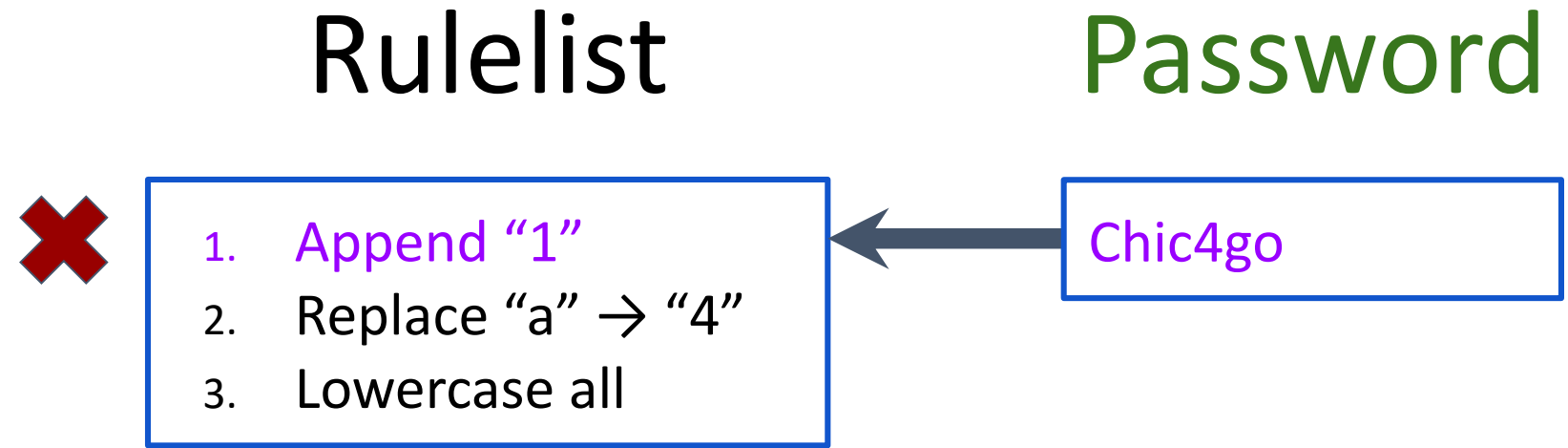
“Rule Reversal”

Marechal (PasswordsCon 2012)

Kacherginsky (PasswordsCon 2013)

and many others

Inversion Process

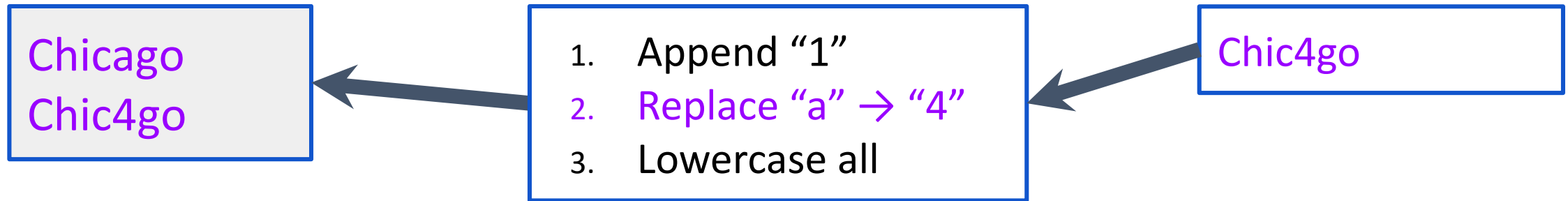


Inversion Process

Preimages

Rulelist

Password



Count Guesses

Wordlist

Super
Password
Chicago

Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super1
Password1
Chicago1

Super
P4ssword
Chic4go
super
password
chicago

Count Guesses

Wordlist

Sup
Pas
Chic




Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super
Pass
Chicago



Super
P4ssword
Chic4go
super
password
chicago

Count Guesses

Wordlist

Super
Password
Chicago

Rulelist

1. Append "1"
2. Replace "a" → "4"
3. Lowercase all

Guesses

Super1
Password1
Chicago1

Super
P4ssword
Chic4go

super
password
chicago

Approach

- **Invert** each password for each rule
 - Identify the first rule, if any, that guesses it
 - Sum guesses made by previous rules
- **Count guesses** per rule (JtR) / word (Hashcat)
 - Do this once per wordlist / rulelist combo

Why is this non-trivial?

Inverting Passwords

Name	Function	Description	Example Rule	Input Word	Output Word	Note
Nothing	:	do nothing	:	p@ssW0rd	p@ssW0rd	
Lower-case	l	Lowercase all letters	l	p@ssW0rd	p@ssw0rd	
Upper-case	u	Uppercase all letters	u	p@ssW0rd	P@SSWORD	
Capitalize	c	Capitalize the first letter and lower the rest	c	p@ssW0rd	P@ssw0rd	
Invert Capitalize	C	Lowercase first found character, uppercase the rest	C	p@ssW0rd	p@SSWORD	
Toggle Case	t	Toggle the case of all characters in word.	t	p@ssW0rd	P@SSWORD	
Toggle @	TN	Toggle the case of characters at position N	T3	p@ssW0rd	p@sSW0rd	*
Reverse	r	Reverse the entire word	r	p@ssW0rd	dr0Wss@p	
Duplicate	d	Duplicate entire word	d	p@ssW0rd	p@ssW0rdp@ssW0rd	
Duplicate N	pN	Append duplicated word N times	p2	p@ssW0rd	p@ssW0rdp@ssW0rdp@ssW0rd	
Reflect	f	Duplicate word reversed	f	p@ssW0rd	p@ssW0rd-dr0Wss@p	
Rotate Left	{	Rotates the word left.	{	p@ssW0rd	@ssW0rdp	
Rotate Right	}	Rotates the word right	}	p@ssW0rd	dp@ssW0r	
Append Character	\$X	Append character X to end	\$1	p@ssW0rd	p@ssW0rd1	
Prepend Character	^X	Prepend character X to front	^1	p@ssW0rd	1p@ssW0rd	
Truncate left	[Deletes first character	[p@ssW0rd	@ssW0rd	
Truncate right]	Deletes last character]	p@ssW0rd	p@ssW0r	
Delete @ N	DN	Deletes character at position N	D3	p@ssW0rd	p@sW0rd	*
Extract range	xNM	Extracts M characters, starting at position N	x04	p@ssW0rd	p@ss	* #
Omit range	ONM	Deletes M characters, starting at position N	O12	p@ssW0rd	psW0rd	*
Insert @ N	iNX	Inserts character X at position N	i4!	p@ssW0rd	p@ss!W0rd	*
Overwrite @ N	oNX	Overwrites character at position N with X	o3\$	p@ssW0rd	p@s\$W0rd	*
Truncate @ N	'N	Truncate word at position N	'6	p@ssW0rd	p@ssW0	*
Replace	sXY	Replace all instances of X with Y	ss\$	p@ssW0rd	p@\$SW0rd	
Purge	@X	Purge all instances of X	@s	p@ssW0rd	p@W0rd	+

Name	Function	Description	Example Rule	Note
Reject less	<N	Reject plains if their length is greater than N	<G	*
Reject greater	>N	Reject plains if their length is less or equal to N	>8	*
Reject equal	_N	Reject plains of length not equal to N	_7	*
Reject contain	!X	Reject plains which contain char X	!z	
Reject not contain	/X	Reject plains which do not contain char X	/e	
Reject equal first	(X	Reject plains which do not start with X	(h	
Reject equal last)X	Reject plains which do not end with X)t	
Reject equal at	=NX	Reject plains which do not have char X at position N	=1a	*
Reject contains	%NX	Reject plains which contain char X less than N times	%2a	*
Reject contains	Q	Reject plains where the memory saved matches current word	rMrQ	e.g. for palindrome

Name	Function	Description	Example Rule	Input Word	Output Word	Note
Swap front	k	Swaps first two characters	k	p@ssW0rd	@pssW0rd	
Swap back	K	Swaps last two characters	K	p@ssW0rd	p@ssW0dr	
Swap @ N	*NM	Swaps character at position N with character at position M	*34	p@ssW0rd	p@sWs0rd	*
Bitwise shift left	LN	Bitwise shift left character @ N	L2	p@ssW0rd	p@æssW0rd	*
Bitwise shift right	RN	Bitwise shift right character @ N	R2	p@ssW0rd	p@9ssW0rd	*
Ascii increment	+N	Increment character @ N by 1 ascii value	+2	p@ssW0rd	p@tsW0rd	*
Ascii decrement	-N	Decrement character @ N by 1 ascii value	-1	p@ssW0rd	p?ssW0rd	*
Replace N + 1	.N	Replaces character @ N with value at @ N plus 1	.1	p@ssW0rd	psssW0rd	*
Replace N - 1	,N	Replaces character @ N with value at @ N minus 1	,1	p@ssW0rd	ppssW0rd	*
Duplicate block front	yN	Duplicates first N characters	y2	p@ssW0rd	p@p@ssW0rd	*
Duplicate block back	YN	Duplicates last N characters	Y2	p@ssW0rd	p@ssW0r-drd	*
Title	E	Lower case the whole line, then upper case the first letter and every letter after a space	E	p@ssW0rdw0rd	P@ssw0rdW0rd	+
Title w/separator	eX	Lower case the whole line, then upper case the first letter and every letter after a custom separator character	e-	p@ssW0rd-w0rd	P@ssw0rd-W0rd	+

Approach to Inverting Passwords

Chic4go

- Represent preimages as \approx regex
- Few: [{C} {h} {i} {c} {a,4} {g} {o}]
- Many: 4 4 4 4 \rightarrow [{a,4} {a,4} {a,4} {a,4}]
- (“Purge 1” reversed): [{1}* {C} {1}* {h} {1}* {i} {1}* {c} {1}* {a,4} {1}* {g} {1}* {o} {1}*]
- Represent wordlist as trie

Counting Guesses For Each Rule

Wordlist

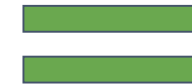
Rule

Guesses

Super
Password
Chicago



Reject if no “a”;
Replace a → 4

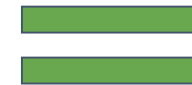


2

Super
Password
Chicago



Replace e → a
Reject if no “a”;
Replace a → 4



3

Advantages and Disadvantages

- Method is preferable:
 - Few target passwords
 - Need guess number quickly
- Not preferable:
 - Many target passwords

Fast Guess Number Estimation

LinkedIn + SpiderLabs  3.01×10^{14} Guesses

	Enumeration	Our Approach
Size	~ 3 PB	~ 10 GB
Preprocessing	> 2 years	< 1 day
Mean Lookup	???	< 1 second

Outline

1. State of the art
2. How software password-cracking tools work
3. Our efficient techniques for guess numbers
4. **Our techniques for systematic configuration**

Software Tools Depend On

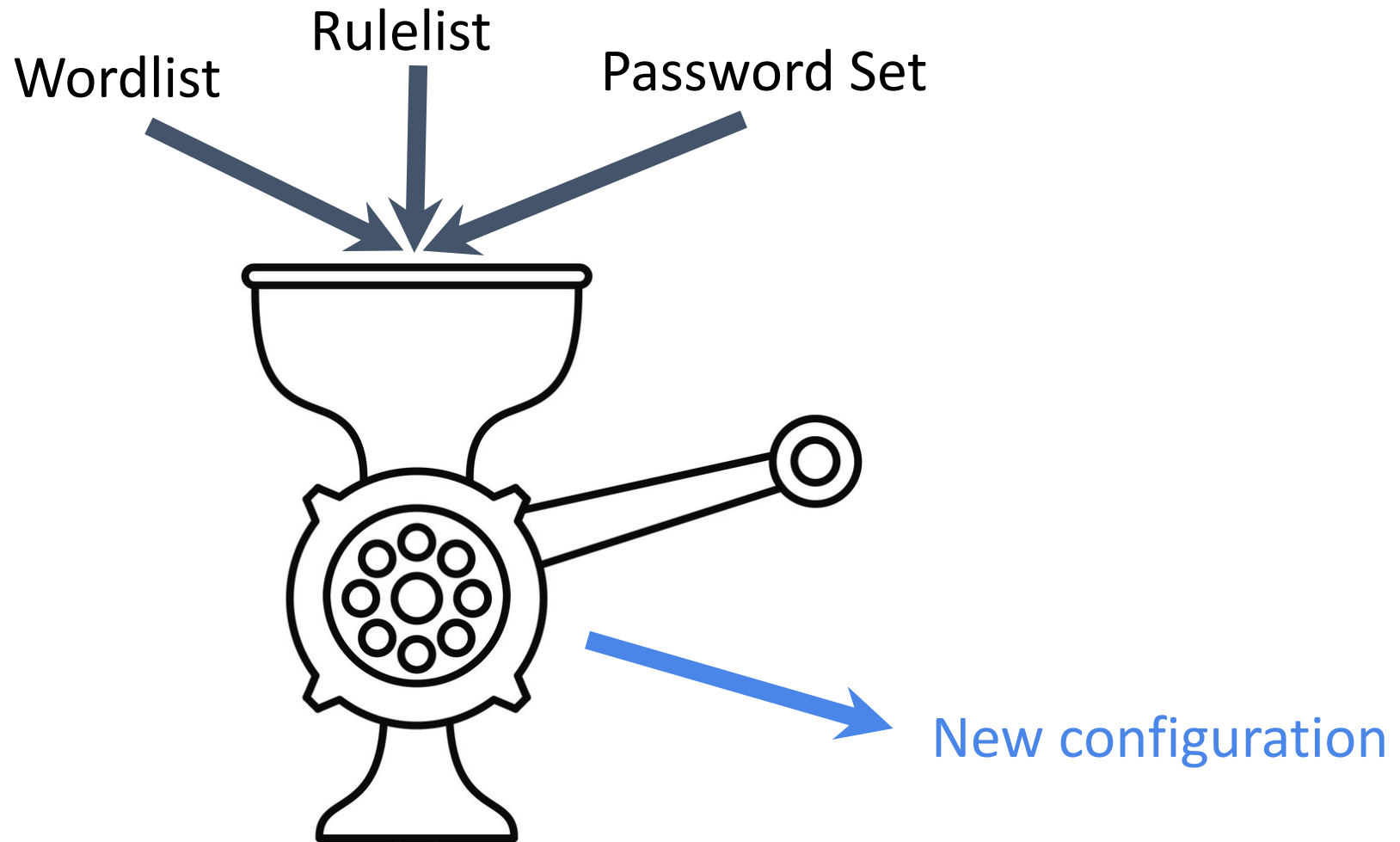
Contents of the wordlist

Order of words

Contents of the rulelist

Order of rules

Insight: Data-Driven Configuration



Data-Driven Configuration

Contents of the wordlist

Order of words

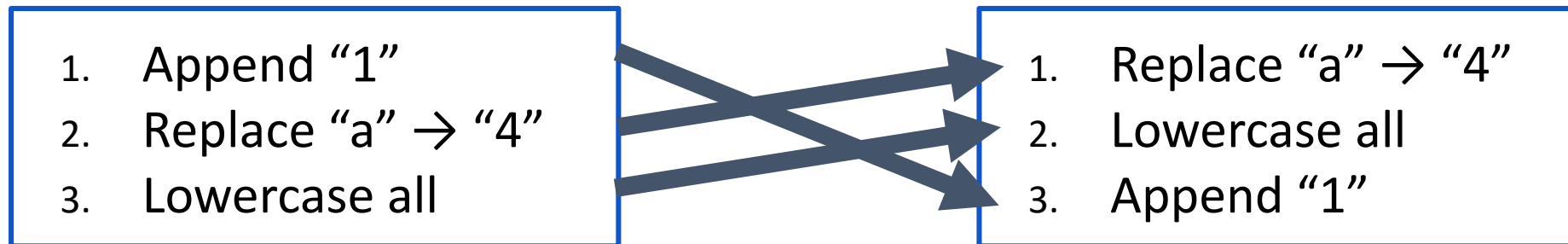
Contents of the rulelist

Order of rules

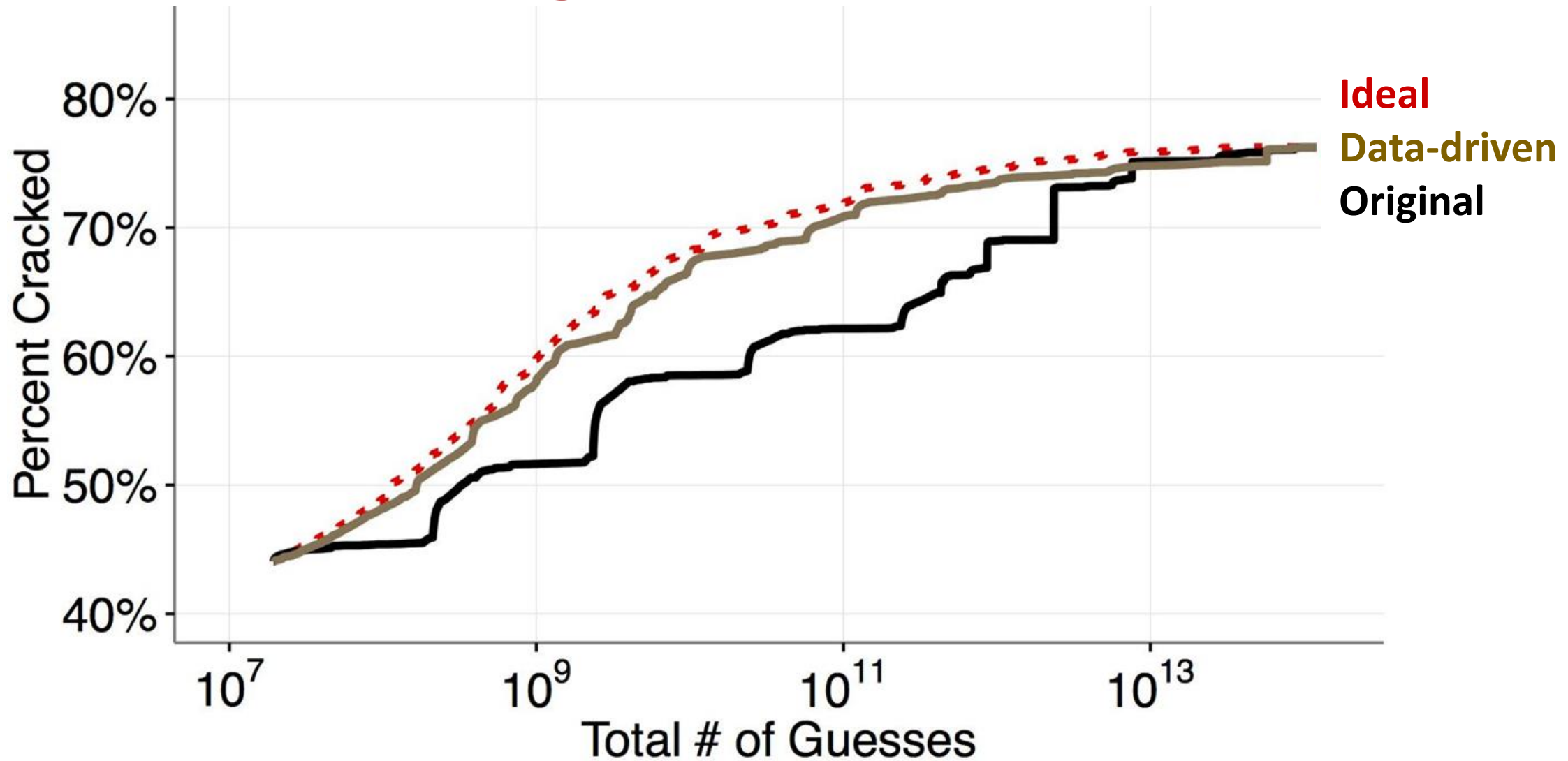
Rule Ordering

Should the rules be in a different order?

Key idea: Order by # cracks per guess



Rule Ordering Results



Word Completeness

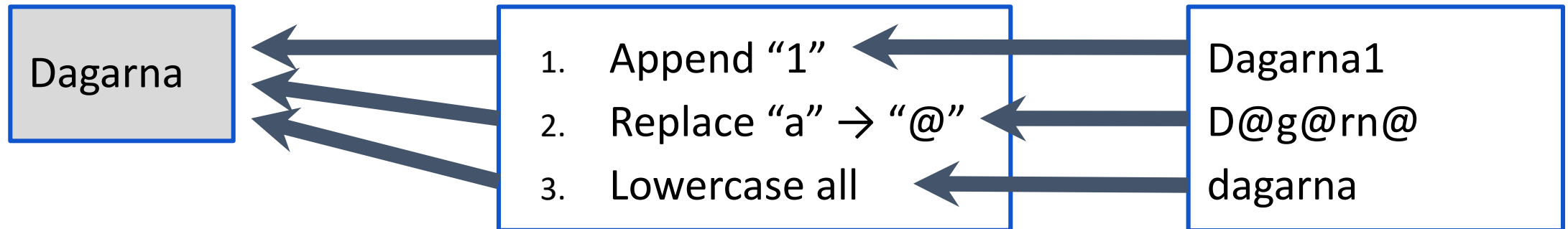
Should other words be in the wordlist?

Key idea: Add frequent preimage “misses”

Preimages

Rulelist

Passwords



Word Completeness (Sample Results)

Category	Examples
Set-specific	bfheros; ilovmyneopets""""



Word Completeness (Sample Results)

Category	Examples
Set-specific	bfheros; ilovmyneopets""""
Meaningful	MaSterBrain; la la la
Short strings	a2; a23; 7a; b2; q2

Takeaway



<https://github.com/UChicagoSUPERgroup>

Guess Number

Configuration

Analytical Tools

Reasoning Analytically About Password-Cracking Software

Enze “Alex” Liu, Amanda Nakanishi,
Maximilian Golla, David Cash, Blase Ur

