THE UNIVERSITY OF CHICAGO
Security, Usability, & Privacy Education & Research
RUHR UNIVERSITÄT BOCHUM RUB

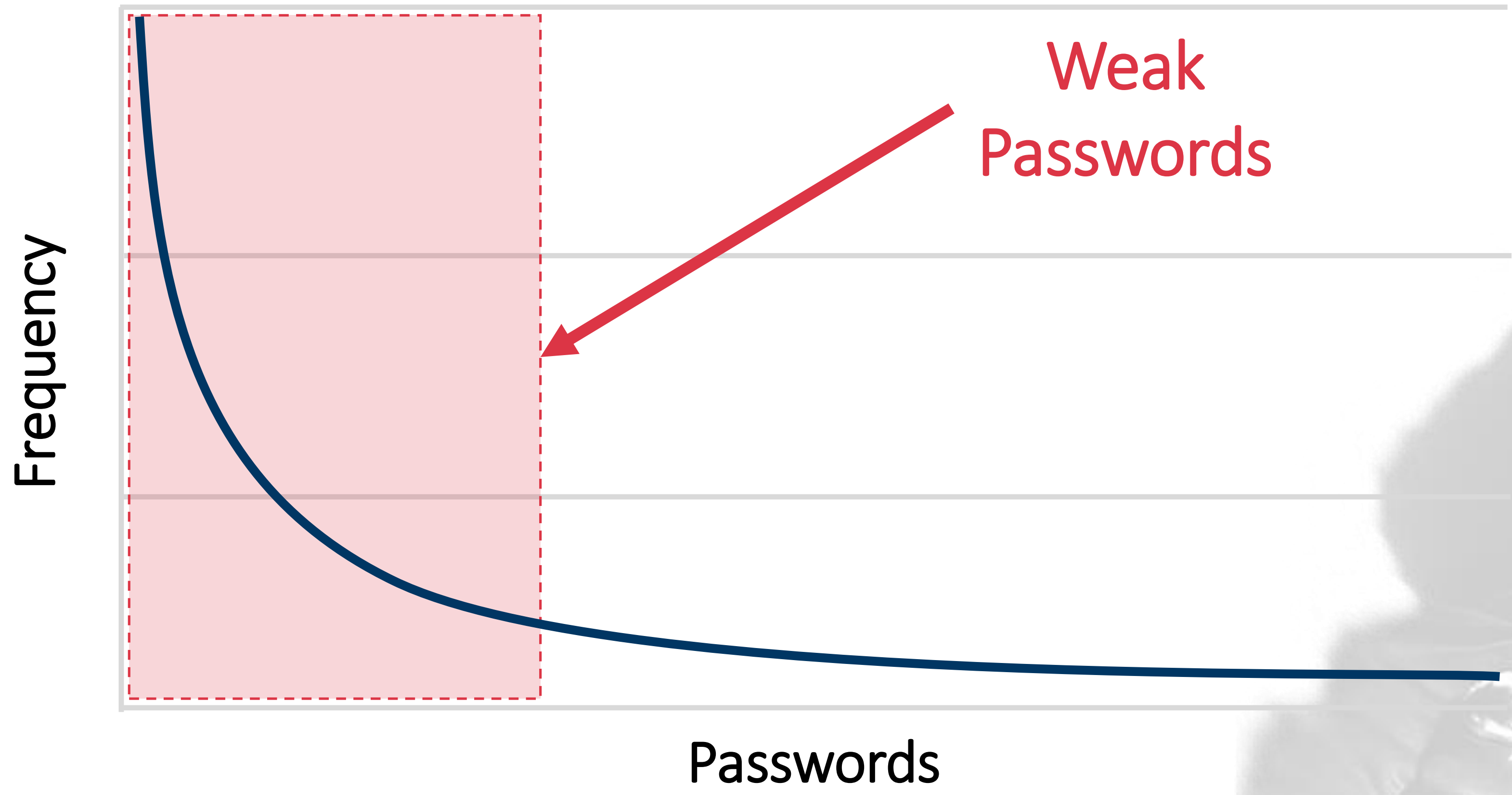
Reasoning Analytically About Password-Cracking Software

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

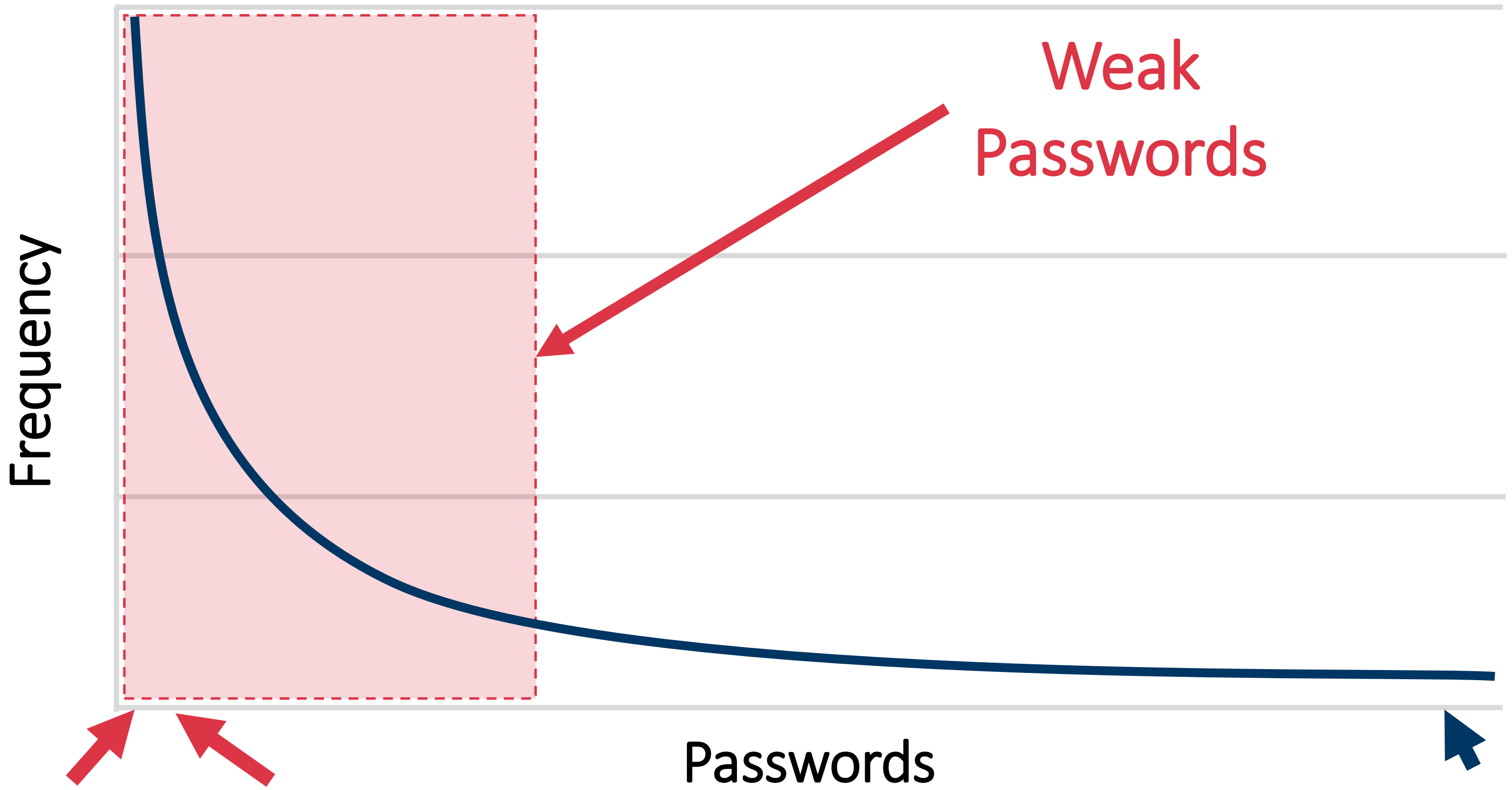
November 26, 2019 | PasswordsCon | Stockholm, Sweden

People Choose Weak Passwords

What Makes a Password “Weak”?



What Makes a Password “Weak”?



Guess #1
`123456`

Guess #387
`qwerty1`

Guess # 1.3 x 10^9
`nx71!iceCream`

Guess Number = Approximate Strength

Example:

`Johnny14!`

Guess #:

390,000

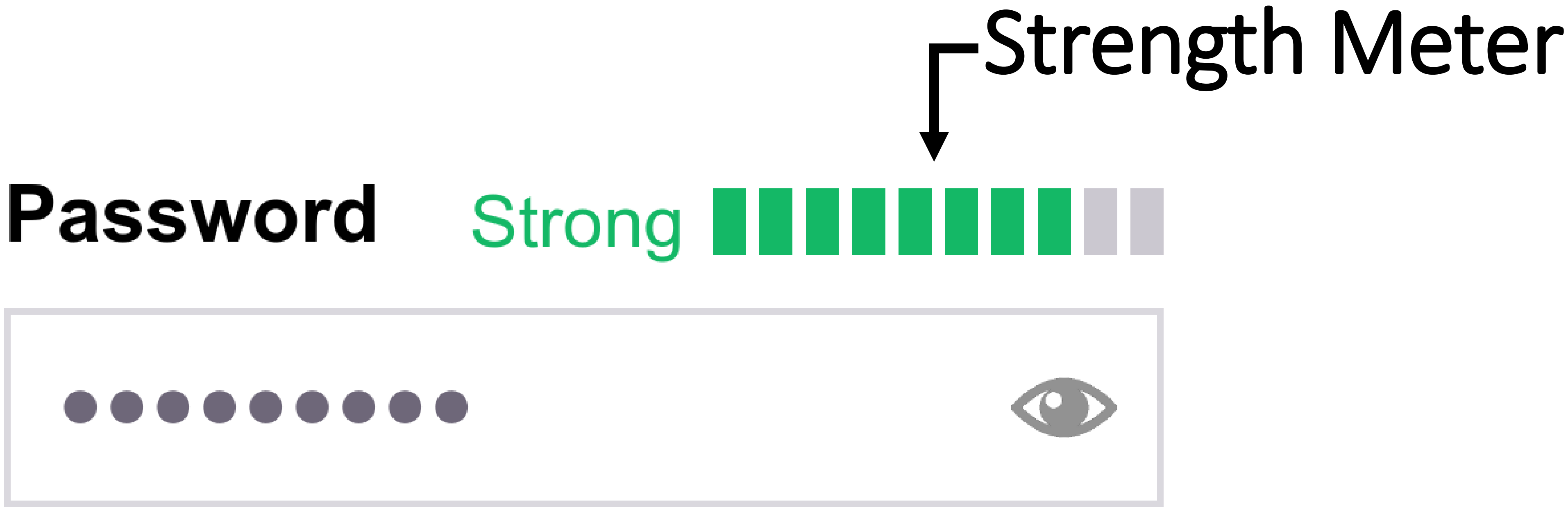
Guess Number

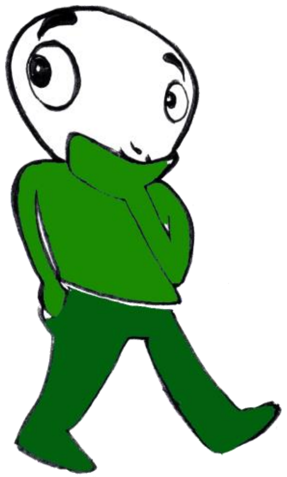
/gɛs ˈnʌmbə/

noun

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

Application 1: Strength Meters





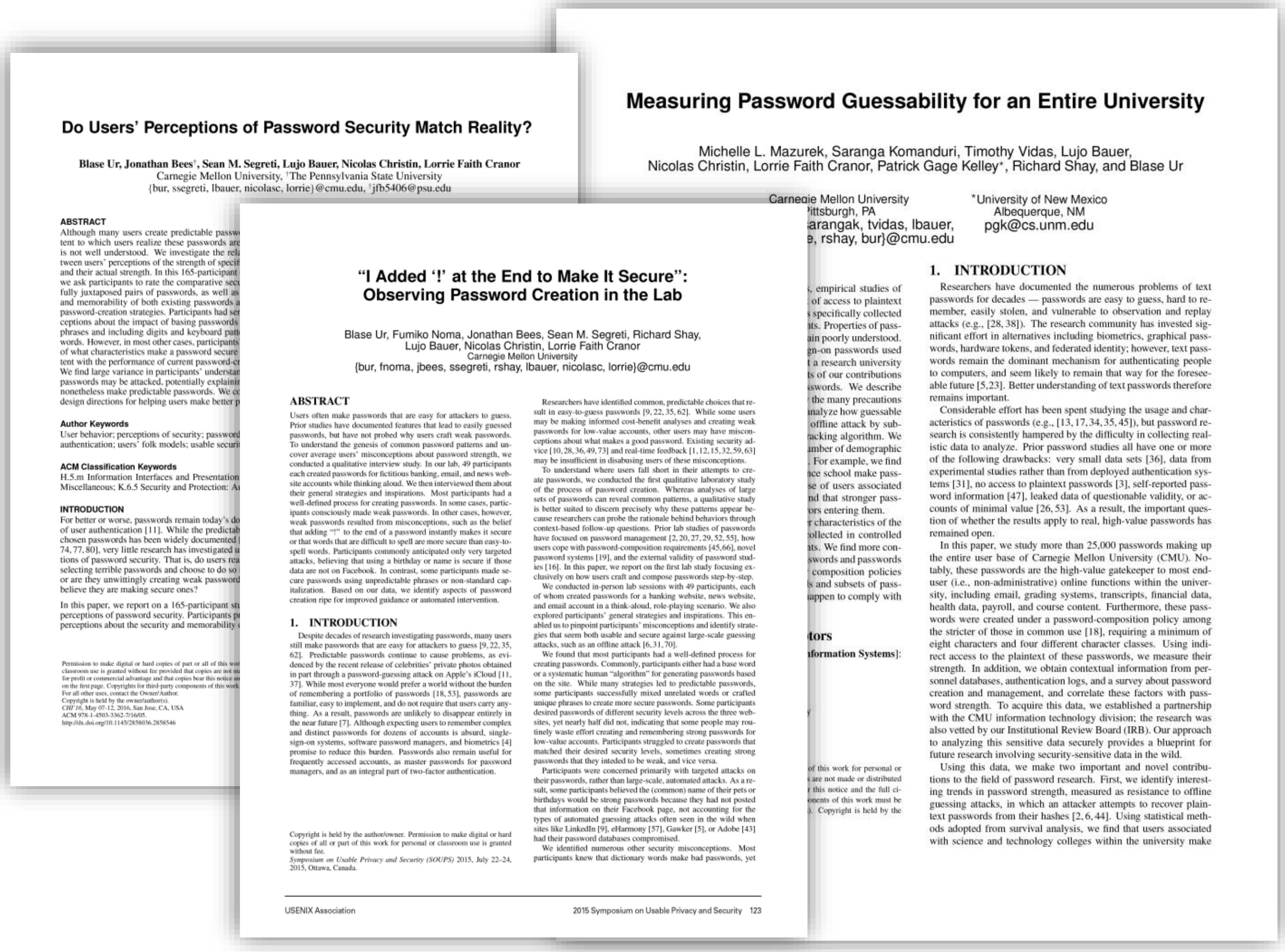
Application 2: Proactive Checking

Password123!

Application 2: Proactive Checking

Per Thorsheim, founder of PasswordsCon

Application 3: Academic Research



Guess # Depends on Model

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

Hashcat / LinkedIn	!=	Dropbox / John the Ripper	!=	Yahoo! / PCFG	!=	RNN / last.fm
Guess #: **1,928,730,033**		Guess #: **8,346,290,721**		Guess #: **inf.**		Guess #: **390,000**

Password Cracking:

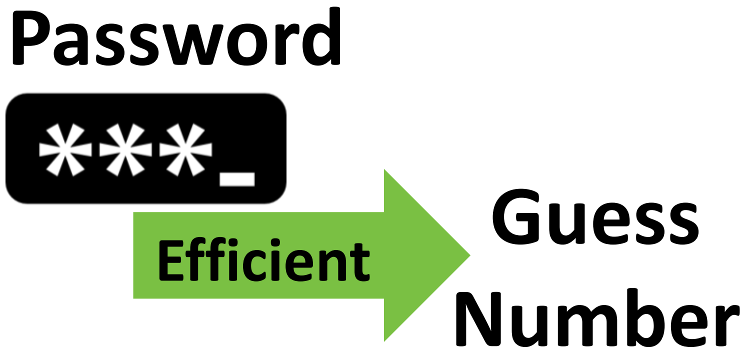
Johnny14! - **cracked**

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



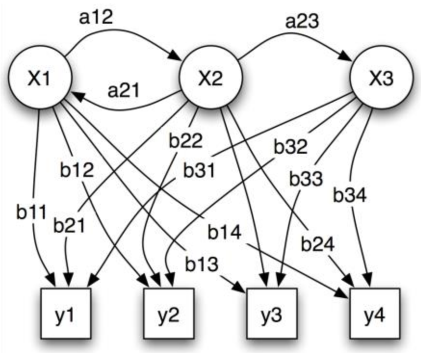
Real-World

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]

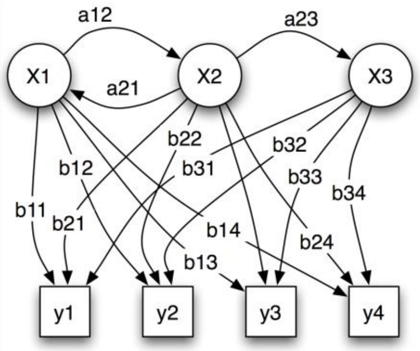
Guess # ✓

Configuration ✓

Real ☹

Password-Cracking Methods

Probabilistic Models



Software Tools

John the Ripper

Hashcat

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

Linkedin (≈ 60,000,000)

HIBP (≈ 500,000,000)

Rulelist

Korelogic (≈ 5,000)

Megatron (≈ 15,000)

Generated2 (≈ 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!

Insight

“Rule Reversal”

Marechal (PasswordsCon 2012)

Kacherginsky (PasswordsCon 2013)

and many others

Inversion Process

Rulelist | Password

✖

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

← Chic4go

Inversion Process

Preimages | Rulelist | Password

Preimages

Chicago
Chic4go

Rulelist

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

Password

Chic4go

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

3

Rulelist

1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

Guesses

Supe
Pass
Chica

3

Super
P4ssword
Chic4go
super
password
chicago

Count Guesses

Wordlist	Rulelist	Guesses
Super Password Chicago	1. Append “1” 2. Replace “a” → “4” 3. Lowercase all	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@ss-W0rd	p@ssW0rd	
Lower-case	l	Lowercase all letters	l	p@ss-W0rd	p@ssw0rd	
Upper-case	u	Uppercase all letters	u	p@ss-W0rd	P@SSW0RD	
Capital-ize	c	Capitalize the first letter and lower the rest	c	p@ss-W0rd	P@ssw0rd	
Invert Capital-ize	C	Lowercase first found character, uppercase the rest	C	p@ss-W0rd	p@SSW0RD	
Toggle Case	t	Toggle the case of all characters in word.	t	p@ss-W0rd	P@SSw0RD	
Toggle @	TN	Toggle the case of characters at position N	T3	p@ss-W0rd	p@sSW0rd	*
Reverse	r	Reverse the entire word	r	p@ss-W0rd	dr0Wss@p	
Dupli-cate	d	Duplicate entire word	d	p@ss-W0rd	p@ssW0rdp@ssW0rd	
Dupli-cate N	pN	Append duplicated word N times	p2	p@ss-W0rd	p@ssW0rdp@ssW0rdp@ssW0rd	
Reflect	f	Duplicate word reversed	f	p@ss-W0rd	p@ssW0rd-dr0Wss@p	
Rotate Left	{	Rotates the word left.	{	p@ss-W0rd	@ssW0rdp	
Rotate Right	}	Rotates the word right	}	p@ss-W0rd	dp@ssW0r	
Append Charac-ter	$X	Append character X to end	$1	p@ss-W0rd	p@ssW0rd1	
Prepend Charac-ter	^X	Prepend character X to front	^1	p@ss-W0rd	1p@ssW0rd	
Truncate left	[	Deletes first character	[	p@ss-W0rd	@ssW0rd	
Trucate right	]	Deletes last character	]	p@ss-W0rd	p@assW0r	
Delete @ N	DN	Deletes character at position N	D3	p@ss-W0rd	p@sW0rd	*
Extract range	xNM	Extracts M characters, starting at position N	x04	p@ss-W0rd	p@ss	* #
Omit range	ONM	Deletes M characters, starting at position N	O12	p@ss-W0rd	psW0rd	*
Insert @ N	iNX	Inserts character X at position N	i4!	p@ss-W0rd	p@ss!W0rd	*
Over-write @ N	oNX	Overwrites character at position N with X	o3$	p@ss-W0rd	p@s$W0rd	*
Truncate @ N	'N	Truncate word at position N	'6	p@ss-W0rd	p@ssW0	*
Replace	sXY	Replace all instances of X with Y	ss$	p@ss-W0rd	p@$$W0rd	
Purge	@X	Purge all instances of X	@s	p@ss-W0rd	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@æsW0rd	*
Bitwise shift right	RN	Bitwise shift right character @ N	R2	p@ssW0rd	p@9sW0rd	*
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@ss-W0rd	*
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@ssW0rd w0rld	P@ssw0rd W0rld	+
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-w0rld	P@ssw0rd-W0rld	+

Approach to Inverting Passwords

`Chic4go`

- Represent preimages as ≈ regex
- Few: [{C} {h} {i} {c} {a,4} {g} {o}]
- Many: 4444 → [{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 $= 3.01 \times 10^{14}$ 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

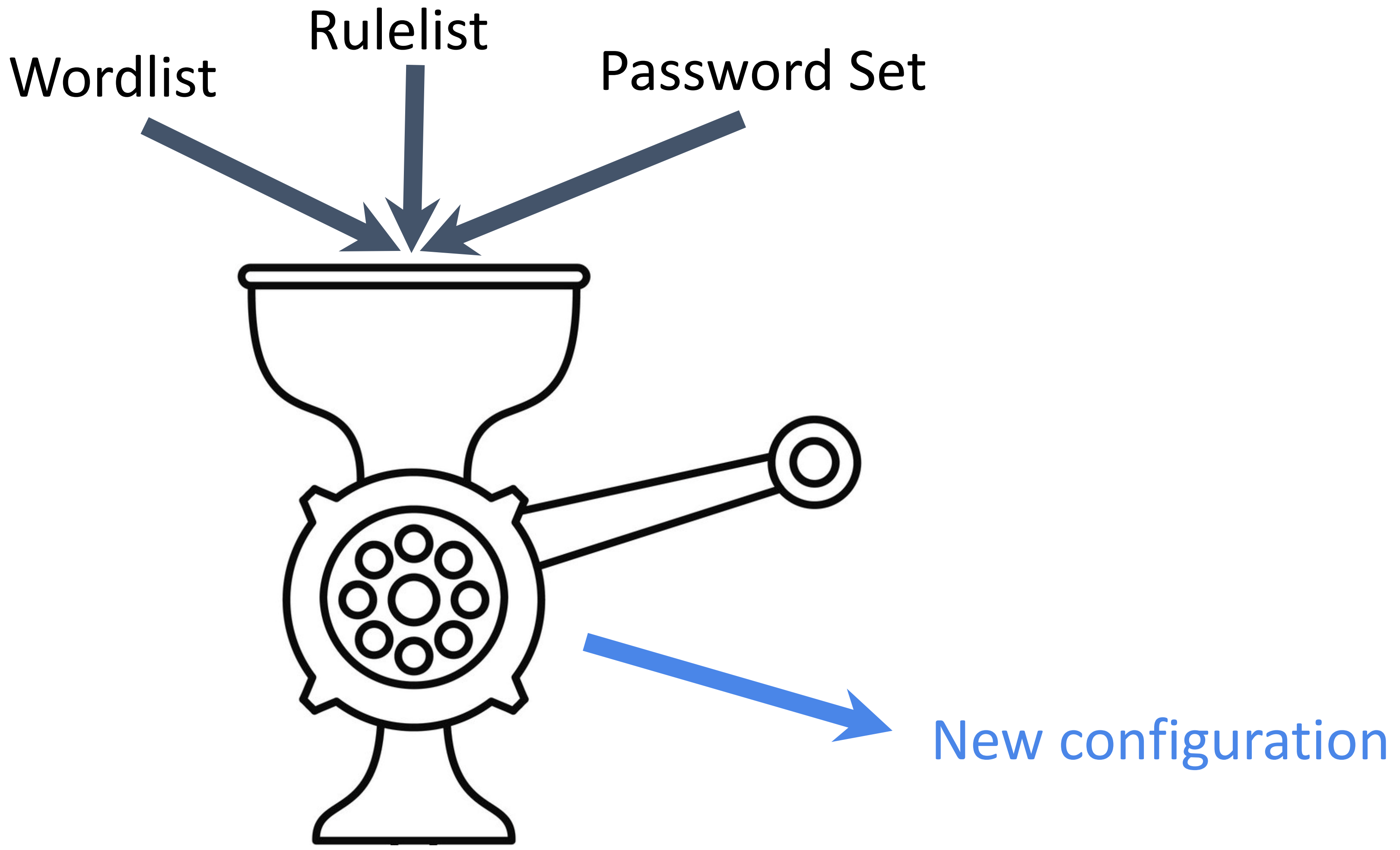
Contents of the wordlist

Order of words

Contents of the rulelist

Order of rules

Insight: Data-Driven Configuration



Data-Driven Configuration

Contents of the wordlist

Order of words

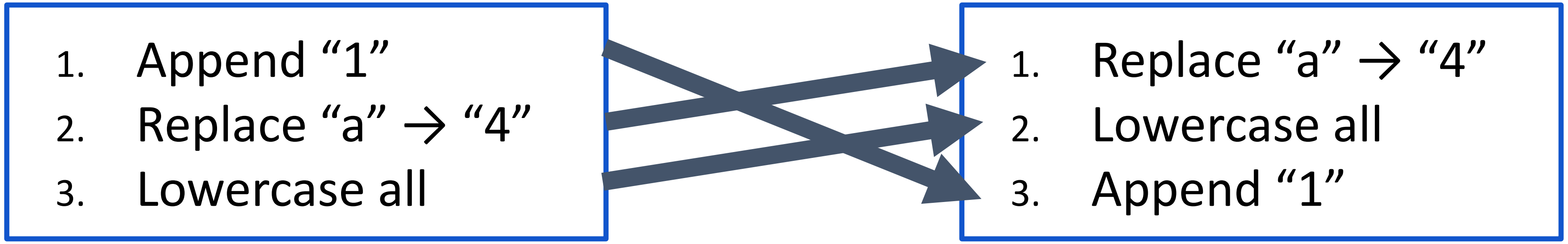
Contents of the rulelist

Order of rules

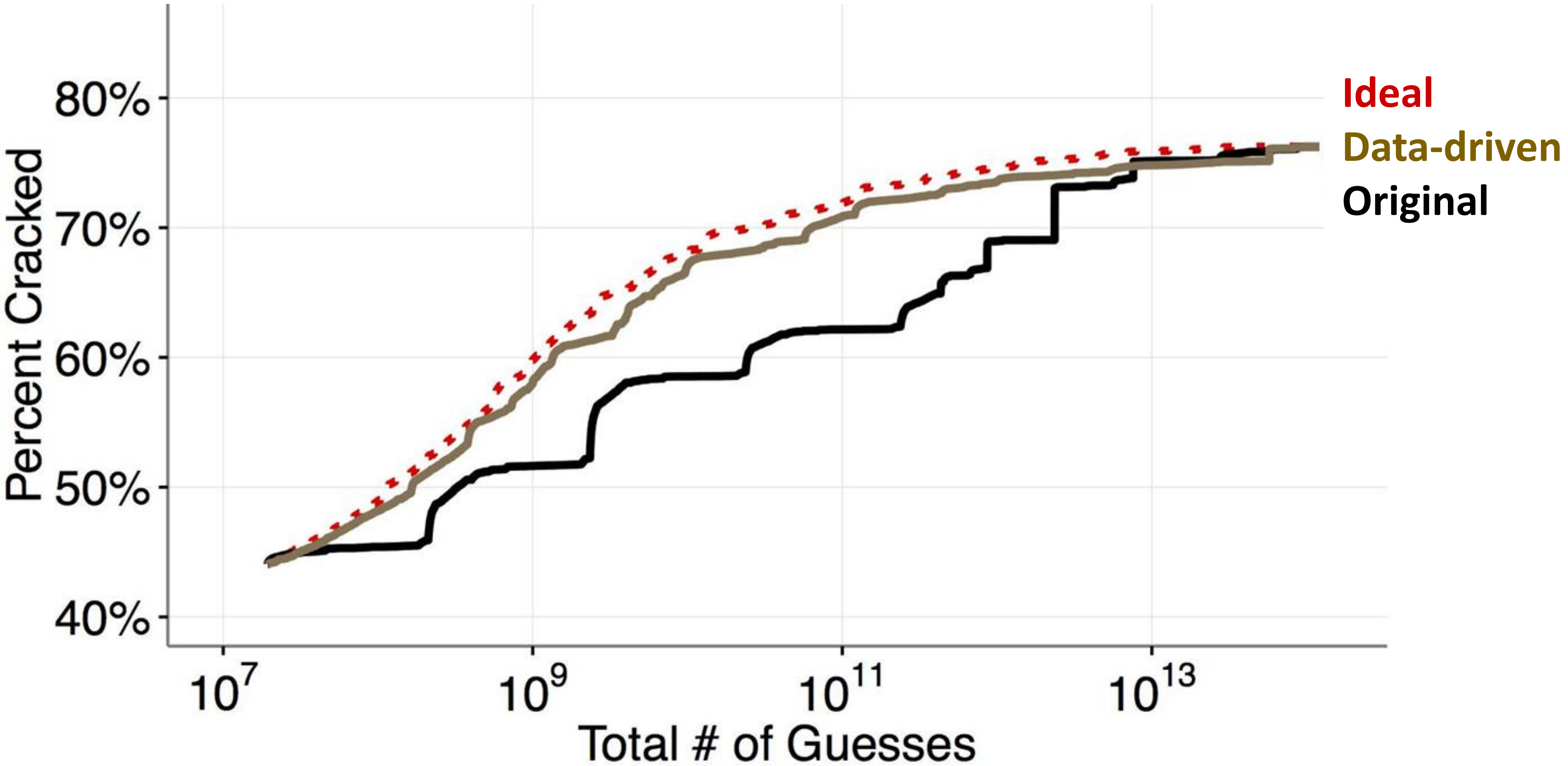
Rule Ordering

Should the rules be in a different order?

Key idea: Order by # cracks per guess



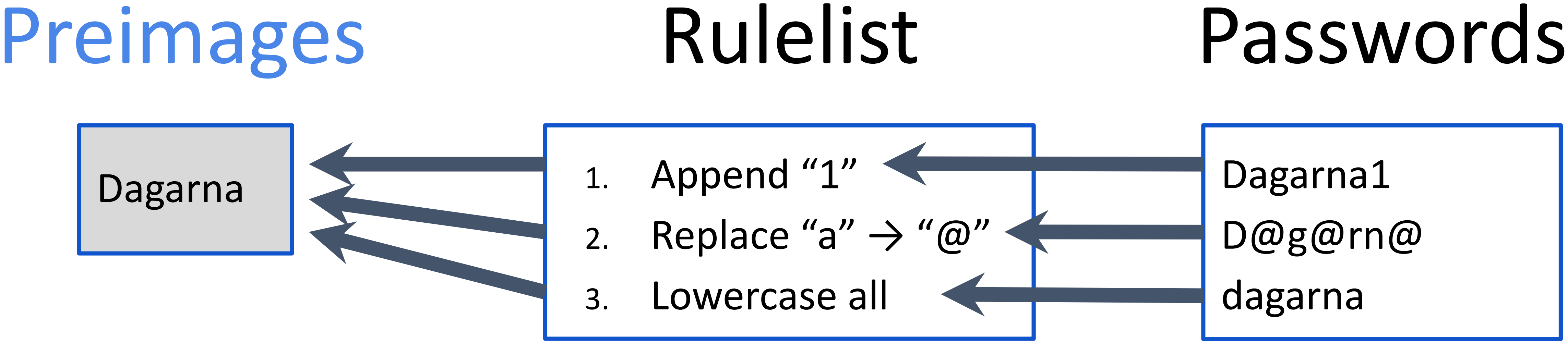
Rule Ordering Results



Word Completeness

Should other words be in the wordlist?

Key idea: Add frequent preimage “misses”



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