Reasoning Analytically About Password-Cracking Software

<u>Enze "Alex" Liu</u>, Amanda Nakanishi, Maximilian Golla, David Cash, Blase Ur

THE UNIVERSITY OF CHICAGO

RUB

Chic4go

Attack Model

80d561388725fa74f2d03cd16e1d687c

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e
2. h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e
2. h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99
3. h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e
2. h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99
3. h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885
4. h(“letmein”) = 0d107d09f5bbe40cade3de5c71e9e9b7

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e
2. h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99
3. h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885
4. h(“letmein”) = 0d107d09f5bbe40cade3de5c71e9e9b7
5. h(“p@ssw0rd”) = 0f359740bd1cda994f8b55330c86d845

Attack Model

80d561388725fa74f2d03cd16e1d687c

1. h("123456") = e10adc3949ba59abbe56e057f20f883e
2. h("password") = 5f4dcc3b5aa765d61d8327deb882cf99
3. h("monkey") = d0763edaa9d9bd2a9516280e9044d885
4. h("letmein") = 0d107d09f5bbe40cade3de5c71e9e9b7
5. h("p@ssw0rd") = 0f359740bd1cda994f8b55330c86d845
6. h("Chic4go") = **80d561388725fa74f2d03cd16e1d687c**

Chic4go

```
Chic4go
```

Guess # 6

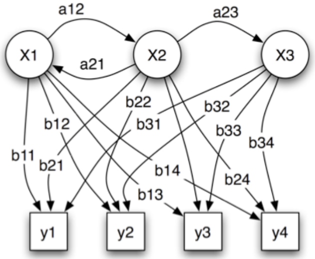
`Chic4go`

~~Guess # 6~~

Guess # 13,545,239,432

Password-Cracking Methods

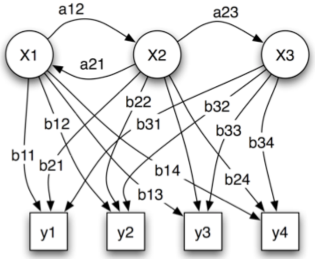
Probabilistic Models



Software Tools

Password-Cracking Methods

Probabilistic Models



Software Tools

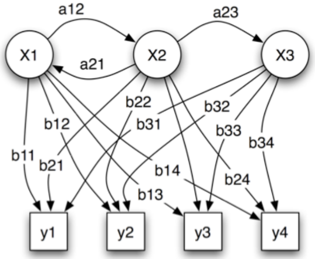
John the Ripper

Hashcat

Chic4go → Guess #

Password-Cracking Methods

Probabilistic Models



Software Tools

John the Ripper

Hashcat

`Chic4go` → Guess #

Guess Number by Enumeration

1. 123456

Does Not Scale !!!

Our Analysis Goals

1. Compute guess numbers efficiently

2. Configure guessing method systematically

Outline

- State of the art
- How software password-cracking tools work
- Our efficient techniques for guess numbers
- Our techniques for systematic configuration

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

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

Monte Carlo Strength Evaluation:
Fast and Reliable Password Checking

[CCS 2015]

Matteo Dell'Amico
Symantec Research Labs, France
matteo_dellamico@symantec.com

Maurizio Filippone*
University of Glasgow, UK
maurizio.filippone@eurecom.fr

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

Monte Carlo Strength Evaluation:
Fast and Reliable Password Checking

[CCS 2015]

Matteo Dell'Amico
Symantec Research Labs, France
matteo_dellamico@symantec.com

Maurizio Filippone*
University of Glasgow, UK
maurizio.filippone@eurecom.fr

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

Monte Carlo Strength Evaluation:
Fast and Reliable Password Checking

[CCS 2015]

Matteo Dell'Amico
Symantec Research Labs, France
matteo_dellamico@symantec.com

Maurizio Filippone*
University of Glasgow, UK
maurizio.filippone@eurecom.fr

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-Efficient

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-Efficient

Wall-Clock Time Slow

Software Tools

John the Ripper

Hashcat

Software Tools

chicdog

chicago1

chicagos

CHICAG

chicago2

chicago

chicaga

chicago3

Chicago

chicago6

CHICAGO

chicago9

CHIcago

Software Tools

John the Ripper

Hashcat

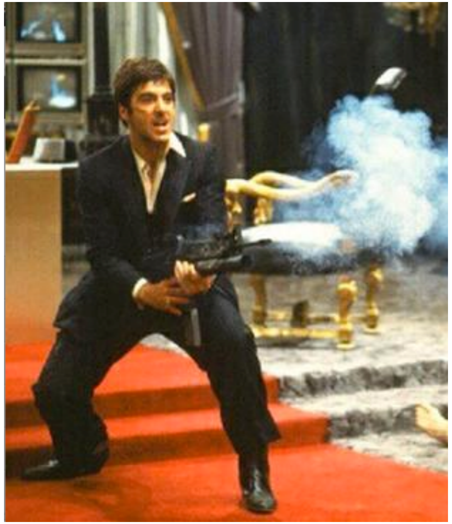
Guess-Inefficient

Wall-Clock Time Fast

Software Tools

John the Ripper

Hashcat



Guess-Inefficient

Wall-Clock Time Fast

Software Tools

John the Ripper

Hashcat

Guess #

Configuration

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

- State of the art
- How software password-cracking tools work
- Our efficient techniques for guess numbers
- Our techniques for systematic configuration

Mangled Wordlist Attack

Mangled Wordlist Attack

Wordlist

Super
Password
Chicago

Mangled Wordlist Attack

Wordlist

Super
Password
Chicago

Rulelist

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

Mangled Wordlist Attack

Wordlist

Super
Password
Chicago

Rulelist

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

Guesses

Super1

Mangled Wordlist Attack

Wordlist

Super
Password
Chicago

Rulelist

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

Guesses

Super1
Password1

Mangled Wordlist Attack

Wordlist

Super
Password
Chicago

Rulelist

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

Guesses

Super1
Password1
Chicago1

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

PGS (≈ 20,000,000)

Linkedin (≈ 60,000,000)

HIBP (≈ 500,000,000)

Example Wordlists and Rulelists

Wordlist

PGS (≈ 20,000,000)

Linkedin (≈ 60,000,000)

HIBP (≈ 500,000,000)

Rulelist

Korelogic (≈ 5,000)

Megatron (≈ 15,000)

Generated2 (≈ 65,000)

Example Wordlists and Rulelists

Wordlist

PGS (≈ 20,000,000)

Linkedin (≈ 60,000,000)

HIBP (≈ 500,000,000)

Rulelist

Korelogic (≈ 5,000)

Megatron (≈ 15,000)

Generated2 (≈ 65,000)

→ 10^9 - 10^{15} guesses

Example Wordlists and Rulelists

Wordlist	Rulelist
PGS (≈ 20,000,000)	Korelogic (≈ 5,000)
Linkedin (≈ 60,000,000)	Megatron (≈ 15,000)
HIBP (≈ 500,000,000)	Generated2 (≈ 65,000)

→ 10^9 - 10^{15} guesses

+ Hackers' private word/rule lists

Outline

- State of the art
- How software password-cracking tools work
- Our efficient techniques for guess numbers
- 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: Invert Rules

Password

Chic4go

Insight: Invert Rules

Rulelist

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

Password

Chic4go

Insight: Invert Rules

Rulelist

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

Password

Chic4go

Insight: Invert Rules

Preimages

Chicago
Chic4go

Rulelist

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

Password

Chic4go

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	+

*05 O03 d '7

Switch the first and the sixth char;

Delete the first three chars;

Duplicate the whole word;

Truncate the word to length 7;

Preimages? ← Chic4go

Where in the Stream?

Wordlist

Super
Password
Chicago

Rulelist

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

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go

Where in the Stream?

Wordlist

Super
Password
Chicago

Rulelist

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

Guesses

Super1
Password1
Chicago1
Super
P4ssword
Chic4go

Counting Guesses For Each Rule

Wordlist		Rule		Guesses
Super Password Chicago	×	Reject if no “a”; Replace a→ 4	=	2

Our First Contribution

- Fast Guess Number Estimation

Fast Guess Number Estimation

Linkedin + SpiderLab

Fast Guess Number Estimation

Linkedin + SpiderLab $= 3.01 \times 10^{14}$ Guesses

Fast Guess Number Estimation

Linkedin + SpiderLab $= 3.01 \times 10^{14}$ Guesses

	Enumeration	Our Approach
Size	~ 3 PB	~ 10 GB

Fast Guess Number Estimation

Linkedin + SpiderLab $= 3.01 \times 10^{14}$ Guesses

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

Fast Guess Number Estimation

Linkedin + SpiderLab $= 3.01 \times 10^{14}$ Guesses

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

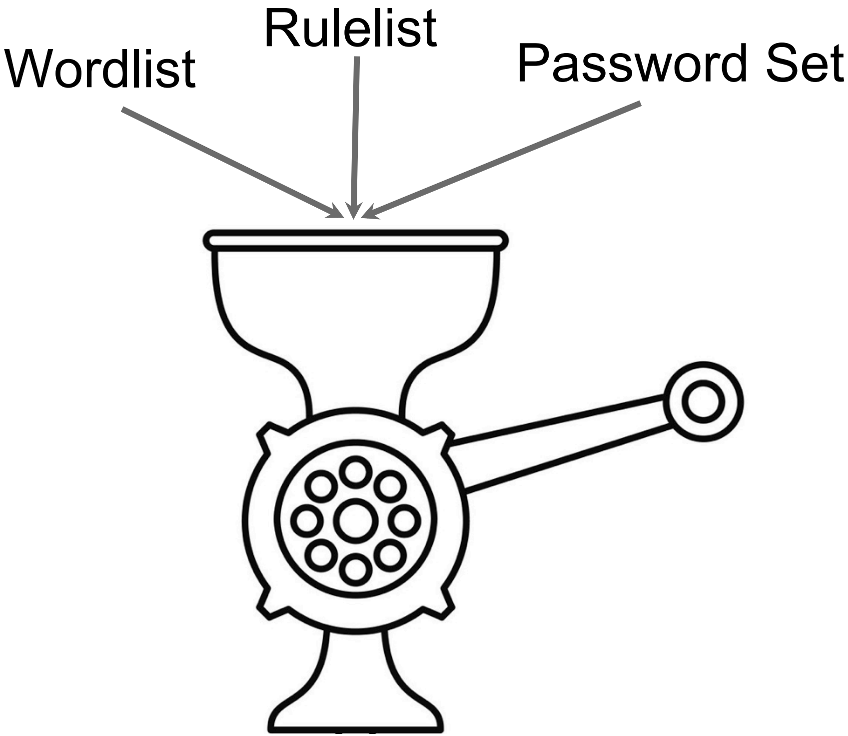
Outline

- State of the art
- How software password-cracking tools work
- Our efficient techniques for guess numbers
- Our techniques for systematic configuration

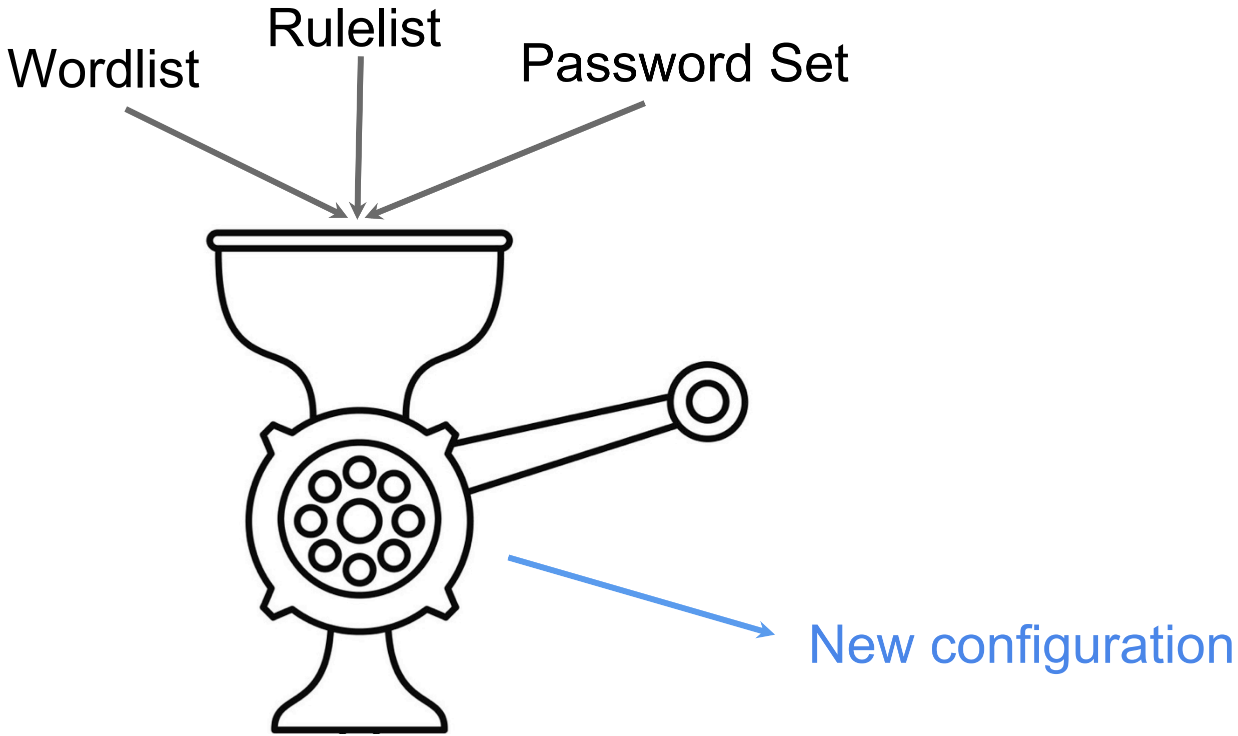
Software Tools Depend On

- Order of rules
- Contents of the rulelist
- Order of words
- Contents of the wordlist

Insight: Data-Driven Configuration



Insight: Data-Driven Configuration



Data-Driven Configuration

- Order of rules
- Contents of the rulelist
- Order of words
- Contents of the wordlist

Rule Ordering

- Should the rules be in a different order?

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

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

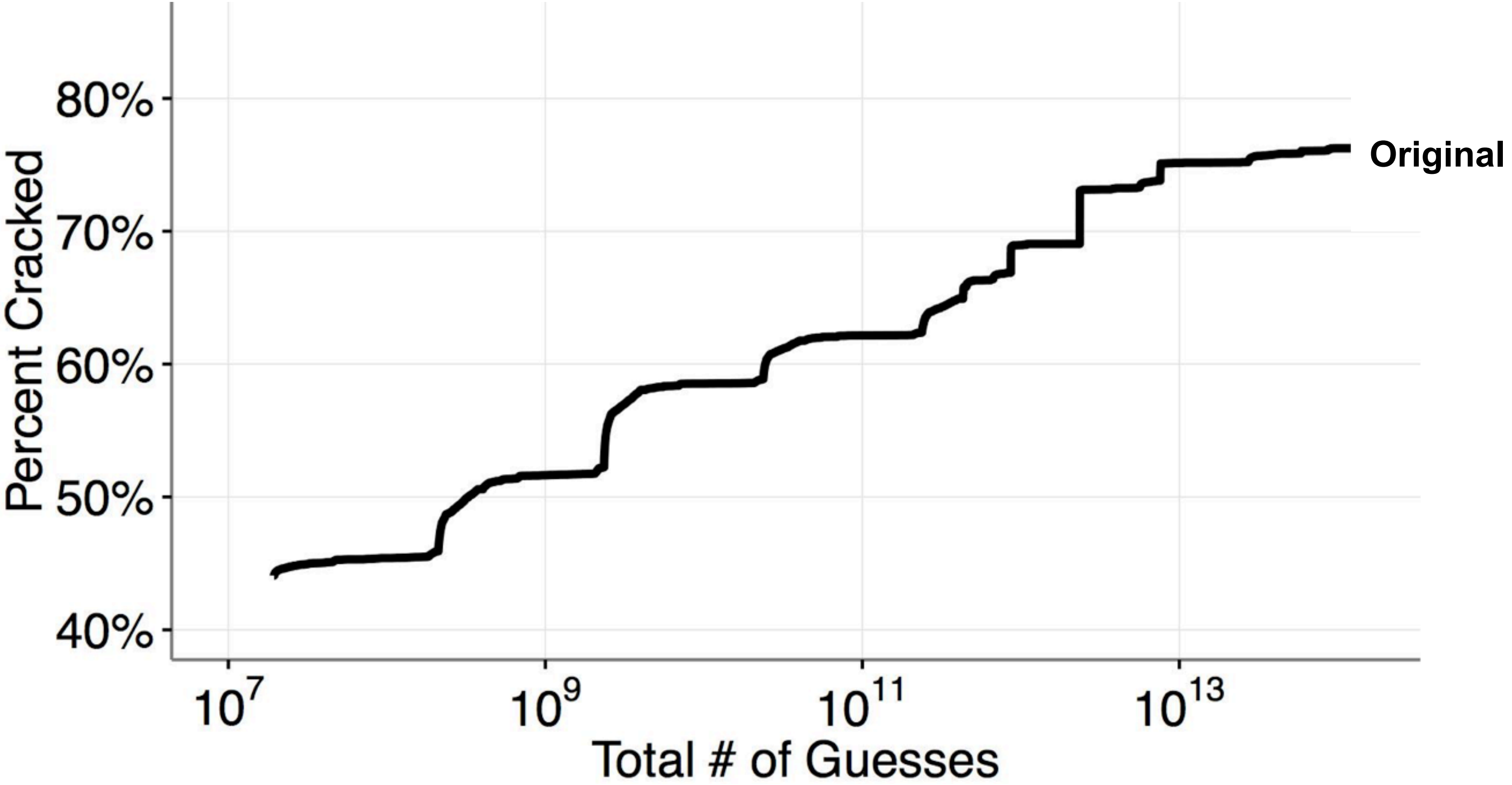
Rule Ordering

- Should the rules be in a different order?
- Key idea: Order by # cracks per guess

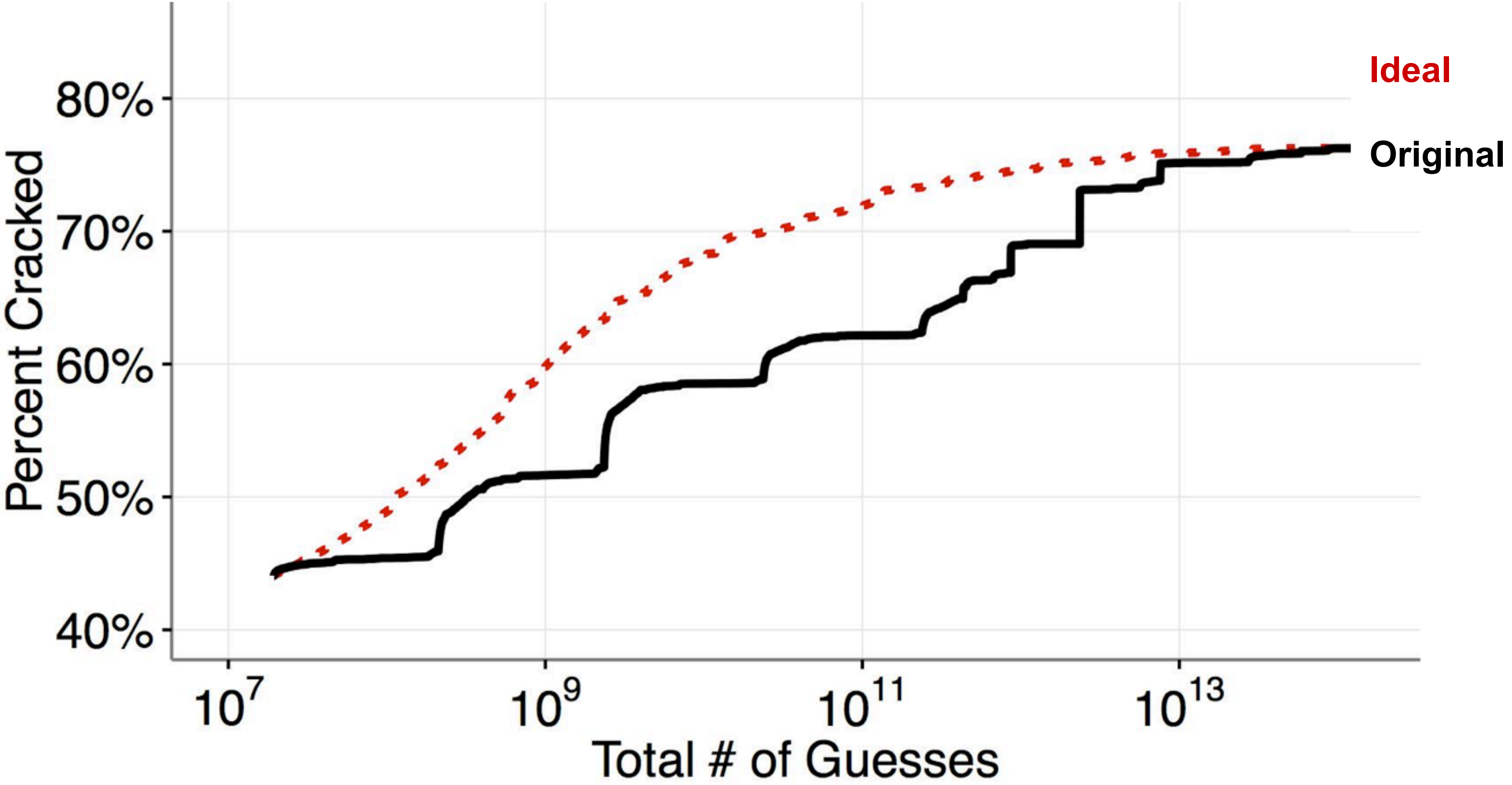
1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

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

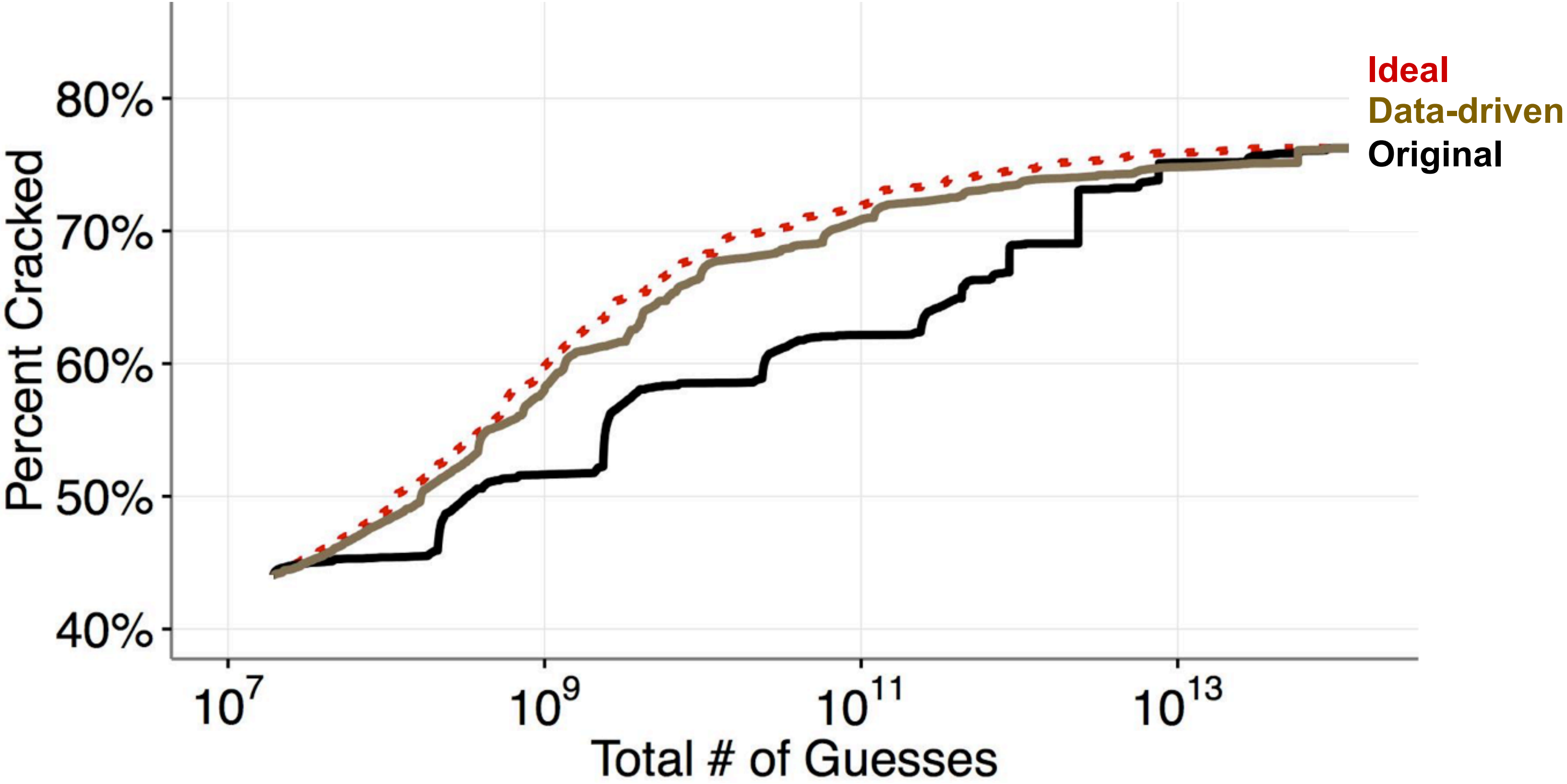
Rule Ordering Results



Rule Ordering Results



Rule Ordering Results



Word Completeness

- Should other words be in the wordlist?
- Key idea: Add frequent preimage "misses"

IEEE S&P

Preimages	Rulelist	
Oakland	1. Append "1"	Oakland1
	2. Replace "a" → "4"	O@kl@nd
	3. Lowercase all	oakland

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

Word Completeness (Sample Results)

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

Takeaway

Analytical Tools

Takeaway

Guess Number

Analytical Tools

Takeaway

Guess Number	Configuration Tools
Analytical Tools	

Takeaway

https://github.com/UChicagoSUPERgroup/

Guess Number	Configuration Tools
Analytical Tools	

Takeaway

https://github.com/UChicagoSUPERgroup/

Guess Number	Configuration Tools
Analytical Tools	

Reasoning Analytically About Password-Cracking Software

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