

# 10. Authentication

## Part 2



Blase Ur and David Cash  
February 2<sup>nd</sup>, 2022  
CMSC 23200 / 33250



THE UNIVERSITY OF  
CHICAGO

# Credential Breaches

# Some Breached Companies

LinkedIn



Adobe

SONY






GAWKER

000webhost.com  
better than paid hosting

YAHOO!

STRATFOR  
GLOBAL INTELLIGENCE

# Data-Driven Statistical Attacks








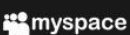


- (2009) 32 million passwords: 
- (2016) 117 million passwords: 
- (2017) 3 billion passwords: 
- Total: >10 billion passwords stolen from >500 services

# Have I Been Pwned (HIBP)


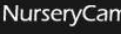








https://haveibeenpwned.com

511 pwned websites      10,599,375,985 pwned accounts      113,991 pastes      199,574,641 paste accounts

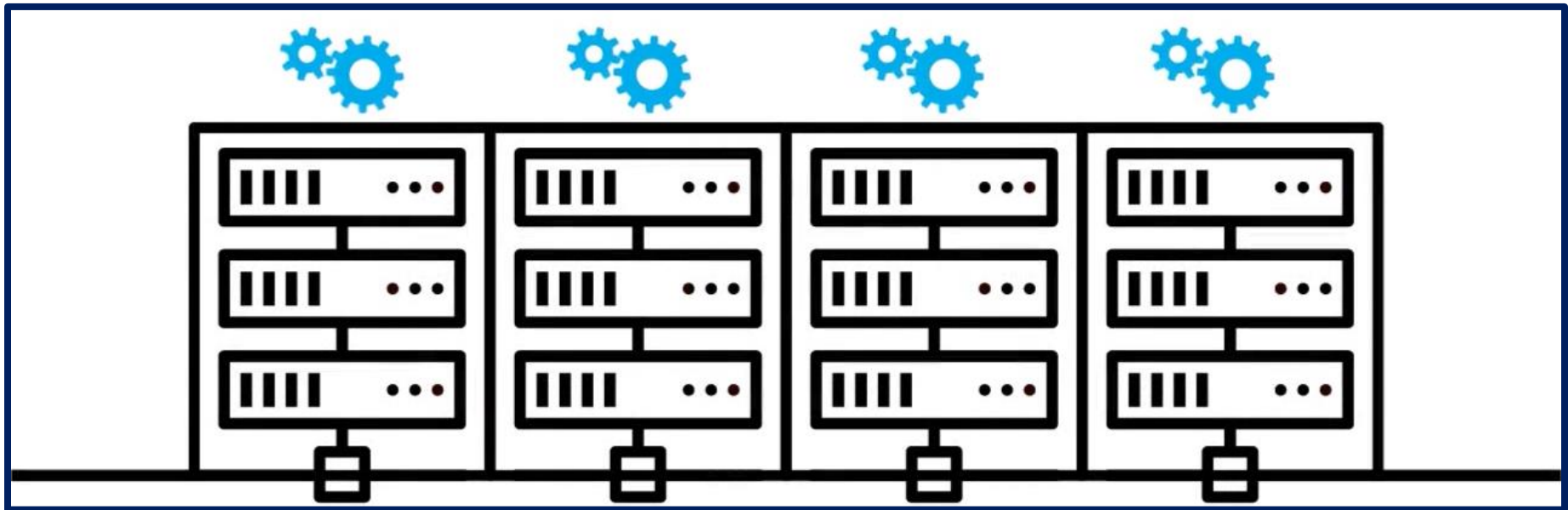
### Largest breaches

	772,904,991 <a href="#">Collection #1 accounts</a>
	763,117,241 <a href="#">Verifications.io accounts</a>
	711,477,622 <a href="#">Onliner Spambot accounts</a>
	622,161,052 <a href="#">Data Enrichment Exposure From PDL Customer accounts</a>
	593,427,119 <a href="#">Exploit.In accounts</a>
	457,962,538 <a href="#">Anti Public Combo List accounts</a>
	393,430,309 <a href="#">River City Media Spam List accounts</a>
	359,420,698 <a href="#">MySpace accounts</a>
	268,765,495 <a href="#">Wattpad accounts</a>
	234,842,089 <a href="#">NetEase accounts</a>

### Recently added breaches

	645,786 <a href="#">Filmai.in accounts</a>
	10,585 <a href="#">NurseryCam accounts</a>
	358,822 <a href="#">People's Energy accounts</a>
	1,436,435 <a href="#">NetGalley accounts</a>
	110,156 <a href="#">CityBee accounts</a>
	2,481,121 <a href="#">Ge.tt accounts</a>
	1,047,200 <a href="#">StoryBird accounts</a>
	1,906,808 <a href="#">Pixlr accounts</a>
	1,422,717 <a href="#">MeetMindful accounts</a>
	2,811,929 <a href="#">Bonobos accounts</a>

# Password Cracking



Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, Richard Shay. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In *Proc. USENIX Security Symposium*, 2015.

# Statistical Metrics For Passwords

- Traditionally: Shannon entropy
- Recently:  $\alpha$ -guesswork
- Disadvantages of statistical approaches
  - Entropy does not consider human tendencies
  - Usually no per-password estimates
  - Huge sample required
  - Not real-world attacks

# Parameterized Guessability

- How many guesses a particular cracking algorithm with particular training data would take to guess a password



# 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.  $h("Chic4go") = \mathbf{80d561388725fa74f2d03cd16e1d687c}$



Chic4go

**Guess # 6**



j@mesb0nd007!

**Guess # 366,163,847,194**

n (c\$JZX! zKc^bIAX^N

Guess # past cutoff

# Broad Cracking Approaches

- Brute force (or selective brute force)
- Wordlist
- Mangled wordlist
  - Hashcat and John the Ripper
- Markov models
- Probabilistic Context-Free Grammar
- Neural networks
- In practice: manual, iterative updates

# 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 ( $\approx 20,000,000$ )

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

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

# Example Wordlists and Rulelists

## Wordlist

## Rulelist

PGS ( $\approx 20,000,000$ )

Korelogic ( $\approx 5,000$ )

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

Megatron ( $\approx 15,000$ )

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

Generated2 ( $\approx 65,000$ )

# Example Wordlists and Rulelists

## Wordlist

PGS ( $\approx 20,000,000$ )

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

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

## Rulelist

Korelogic ( $\approx 5,000$ )

Megatron ( $\approx 15,000$ )

Generated2 ( $\approx 65,000$ )



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

# Example Wordlists and Rulelists

## Wordlist

## Rulelist

PGS ( $\approx 20,000,000$ )

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

+ Hackers' private word/rule lists

# John the Ripper

- Wordlist mode requires:
  - Wordlist (passwords and dictionary entries)
  - Mangling rules
- Guesses variants of input wordlist
- Speed: Fast
- “JTR”



# John the Ripper



wordlist

rules



guesses



# John the Ripper



*usenix*  
*security*

}  
wordlist

}  
rules



}  
guesses

# John the Ripper



*usenix*  
*security*

}  
wordlist

[ ]  
[add 1 at end]  
[change e to 3]

}  
rules



}  
guesses

# John the Ripper



*usenix*  
*security*

} wordlist

[ ]

[add 1 at end]  
[change e to 3]

} rules

usenix  
security

usenix1

security1

us3nix

s3curity

} guesses

# John the Ripper



*unix*  
*security*

}  
wordlist

unix  
security

unix1  
security1

us3nix

s3curity

}  
guesses

[ ]

[add 1 at end]

[change e to 3]

}  
rules

# John the Ripper



*unix*  
*security*

}  
wordlist

[ ]  
[add 1 at end]

[change e to 3]

}  
rules

unix  
security

unix1

security1

us3nix

s3curity

}  
guesses

# Hashcat

- Wordlist mode requires:
  - Wordlist (passwords and dictionary entries)
  - Mangling rules
- Guesses variants of input wordlist
- Speed: Fast



hashcat  
advanced  
password  
recovery

# Hashcat



hashcat  
advanced  
password  
recovery

wordlist

rules



guesses

# Hashcat



hashcat  
advanced  
password  
recovery

*unix*  
*security*

Wordlist

[ ]  
[add 1 at end]  
[change e to 3]

rules



guesses



# Hashcat



hashcat  
advanced  
password  
recovery

**usenix**

security

wordlist

[ ]

[add 1 at end]

[change e to 3]

rules

usenix

usenix1

us3nix

security

security1

s3curity

guesses

# Hashcat



hashcat  
advanced  
password  
recovery

*usenix*

*security*

wordlist

usenix

usenix1

us3nix

security

security1

s3curity

guesses

[ ]

[add 1 at end]

[change e to 3]

rules

# Hashcat: Rule Language

Name	Function	Description	Example Rule	Input Word	Output Word	Note
Nothing	:	do nothing	:	p@ss-W0rd	p@ssW0rd	
Lowercase	l	Lowercase all letters	l	p@ss-W0rd	p@ssw0rd	
Uppercase	u	Uppercase all letters	u	p@ss-W0rd	P@SSWORD	
Capitalize	c	Capitalize the first letter and lower the rest	c	p@ss-W0rd	P@ssw0rd	
Invert Capitalize	C	Lowercase first found character, uppercase the rest	C	p@ss-W0rd	p@SSWORD	
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	
Duplicate	d	Duplicate entire word	d	p@ss-W0rd	p@ssW0rdp@ssW0rd	
Duplicate 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 Character	\$X	Append character X to end	\$1	p@ss-W0rd	p@ssW0rd1	
Prepend Character	^X	Prepend character X to front	^1	p@ss-W0rd	1p@ssW0rd	
Truncate left	[	Deletes first character	[	p@ss-W0rd	@ssW0rd	
Truncate right	]	Deletes last character	]	p@ss-W0rd	p@assW0r	
Delete @ N	DN	Deletes character at position N	D3	p@ss-W0rd	p@ssW0rd	*
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@ssiW0rd	*
Overwrite @ N	oNX	Overwrites character at position N with X	o3\$	p@ss-W0rd	p@ssW0rd	*
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@ssW0rd	
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?sW0rd	*
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@ssW0rd-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	+

# Hashcat: Rule Language

```
*05 003 d '7
```

Switch the first and the sixth char;

Delete the first three chars;

Duplicate the whole word;

Truncate the word to length 7;

# Hashcat (Other Modes)

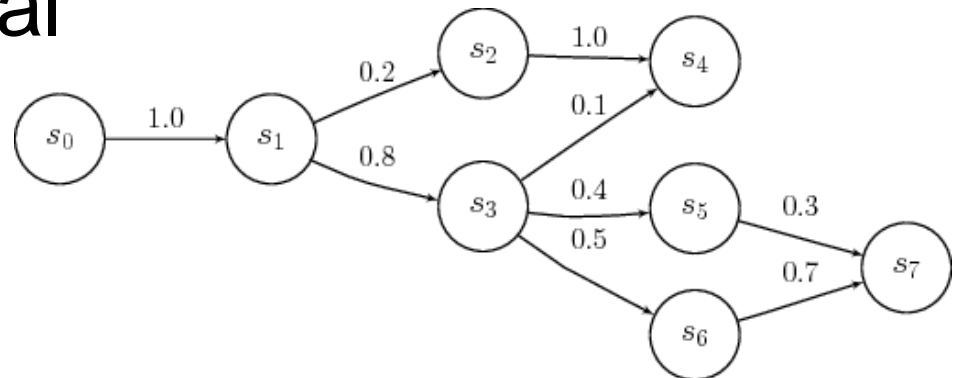
- Mask attack (brute force within a specified character-class structure)
- Combinator attacks
- Hybrid attacks
- Many more!



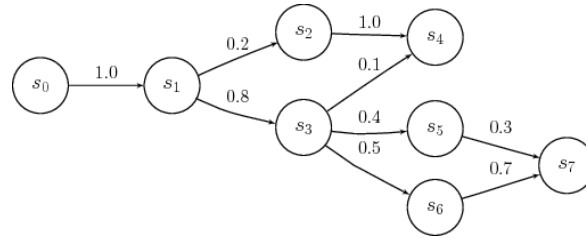
hashcat  
advanced  
password  
recovery

# Markov Models

- Predicts future characters from previous
- Approach requires weighted data:
  - Passwords
  - Dictionaries
- Speed: Slow
- Smoothing is critical



# Markov Models



chic4gooo

2-gram model (1 character of context):

**[start] → c (1.0)**

**4 → g (1.0)**

**c → h (0.5), 4 (0.5)**

**g → o (1.0)**

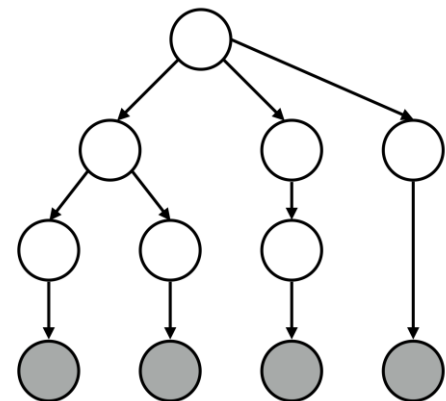
**h → i (1.0)**

**i → c (1.0)**

**o → o (0.67) [end] (0.33)**

# Probabilistic Context-Free Grammar

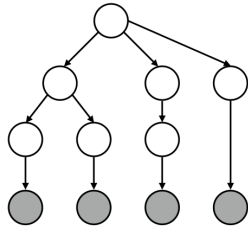
- Generate password grammar
  - Structures
  - Terminals
- OG: Weir et al. IEEE S&P 2009
- Speed: Slow
- “PCFG”







# PCFG



*passwordpassword*

*password*123

*usenix*3

5*ecurity*

*iloveyou*

*nirvanaa*123

Structure Model:

$L_{16}$  (1/6)

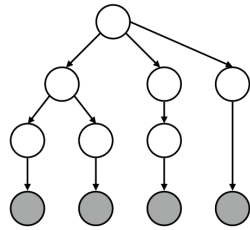
$L_8 D_3$  (2/6)

$L_6 D_1$  (1/6)

$D_1 L_7$  (1/6)

$L_8$  (1/6)

# PCFG



*passwordpassword*

*password*123

*usenix*3

5*security*

*iloveyou*

*nirvanaa*123

Digit Model:

$D_1 \rightarrow 3 (0.5) 5 (0.5)$

$D_3 \rightarrow 123 (1.0)$

Repeat for letters, etc.

# Professionals (“Pros”)

- Proprietary wordlists and configurations
  - $10^{14}$  guesses
  - Manually tuned, updated
- For example: KoreLogic
  - Password audits for Fortune 500 companies
  - Run DEF CON “Crack Me If You Can”

**KoreLogic**  
S E C U R I T Y



# Approach

## 4 password sets

```
password
iloveyou
team0123
...
```

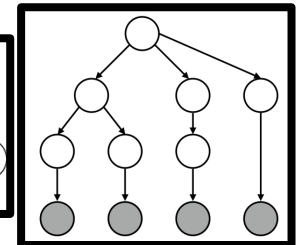
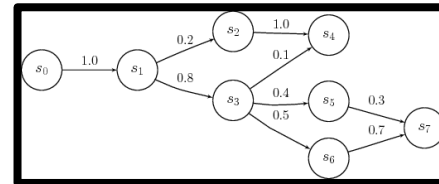
```
passwordpassword
1234567812345678
!1@2#3$4%5^6&7*8
...
```

```
Pa$$w0rd
iLov3you!
1QaZ2W@x
...
```

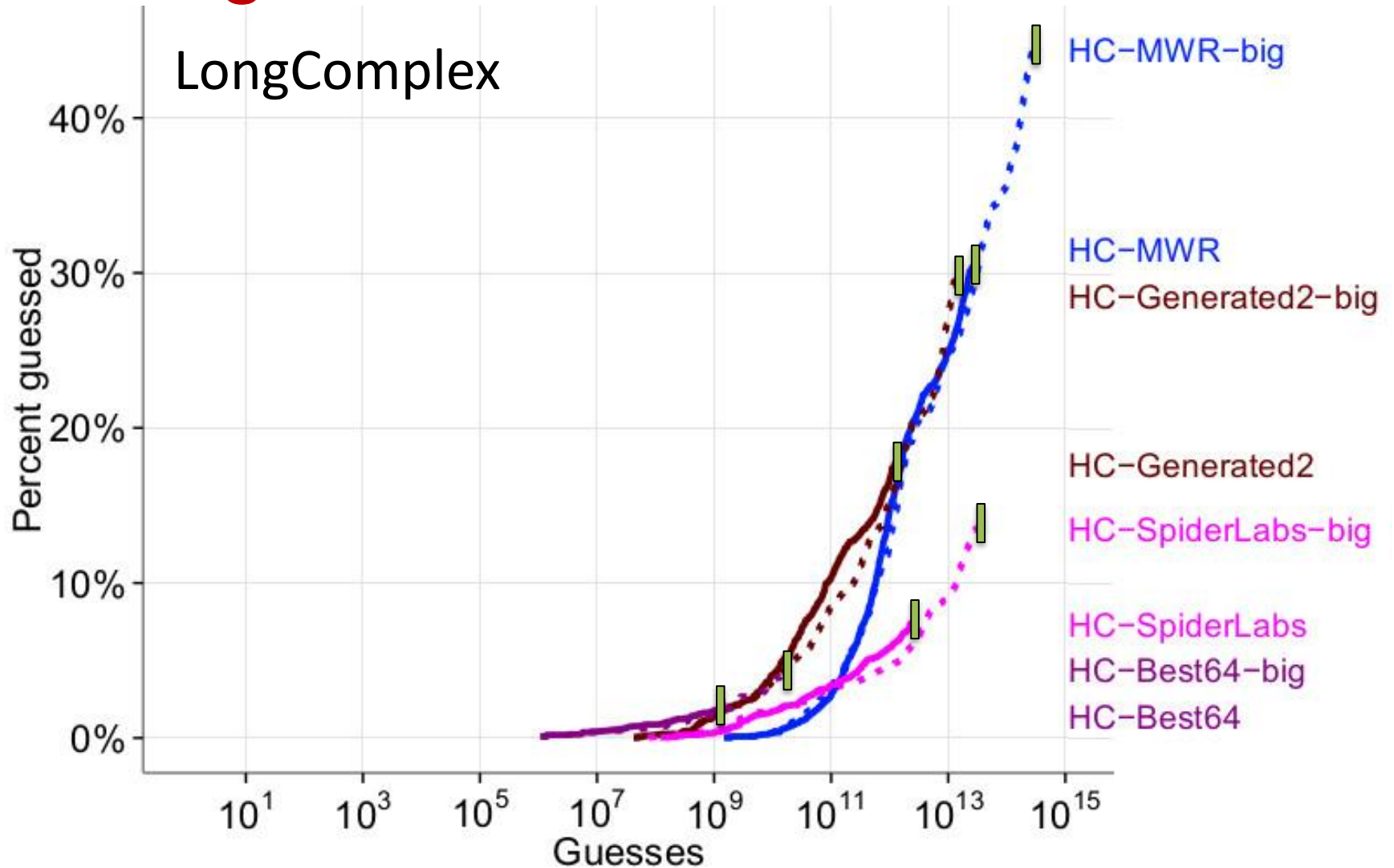
```
pa$$word1234
12345678asDF
!q1q!q1q!q1q
...
```



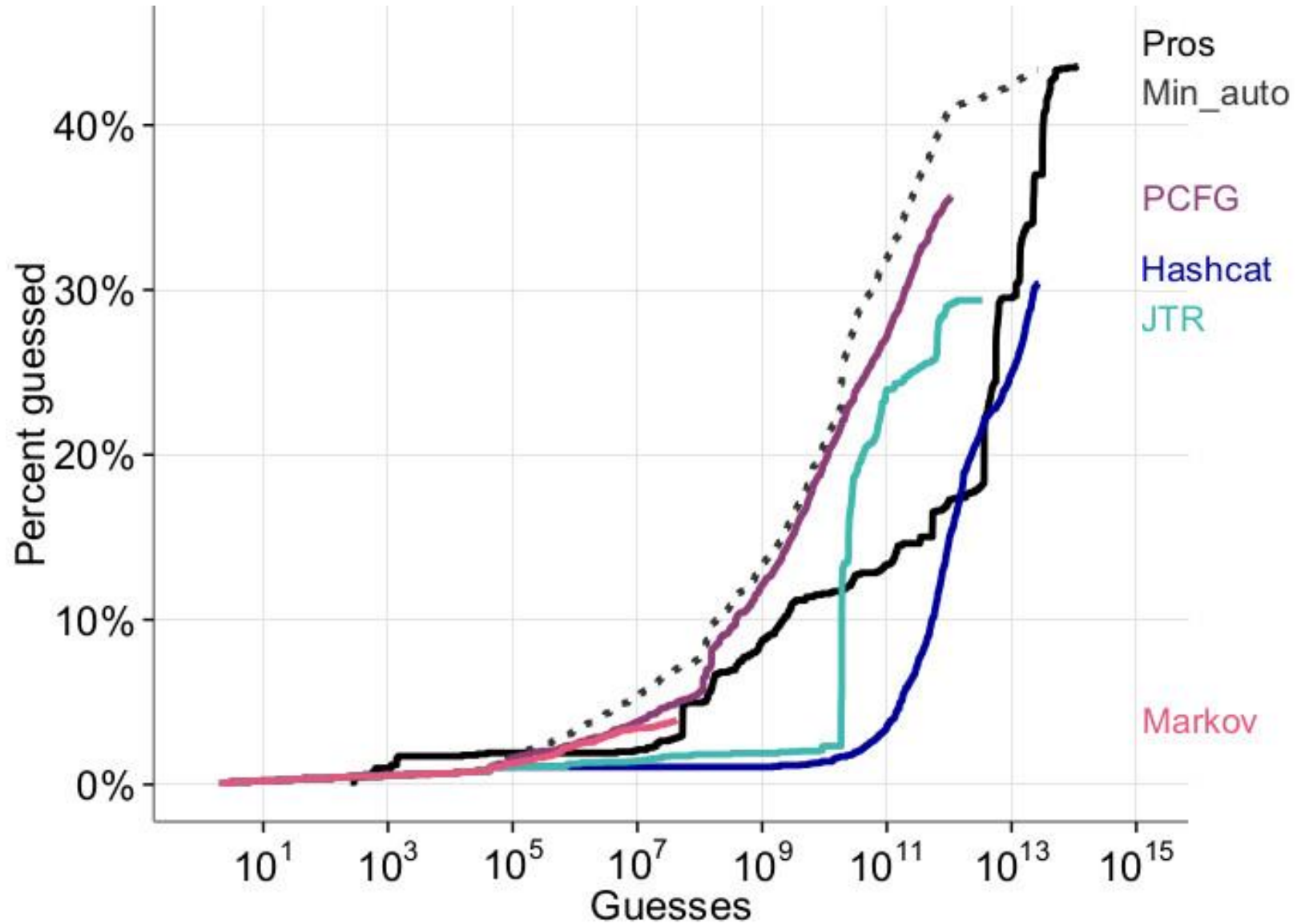
## 5 approaches



# Configuration Is Crucial



# Comparison for Complex Passwords



# Per-Password Highly Impacted

P@ssw0rd!



# Per-Password Highly Impacted

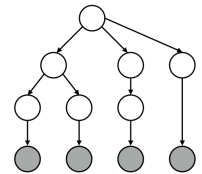
- JTR guess # 801



P@ssw0rd!

# Per-Password Highly Impacted

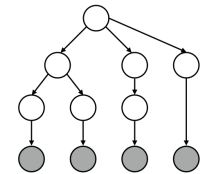
- JTR guess # 801
- Not guessed in  $10^{14}$  PCFG guesses



P@ssw0rd!

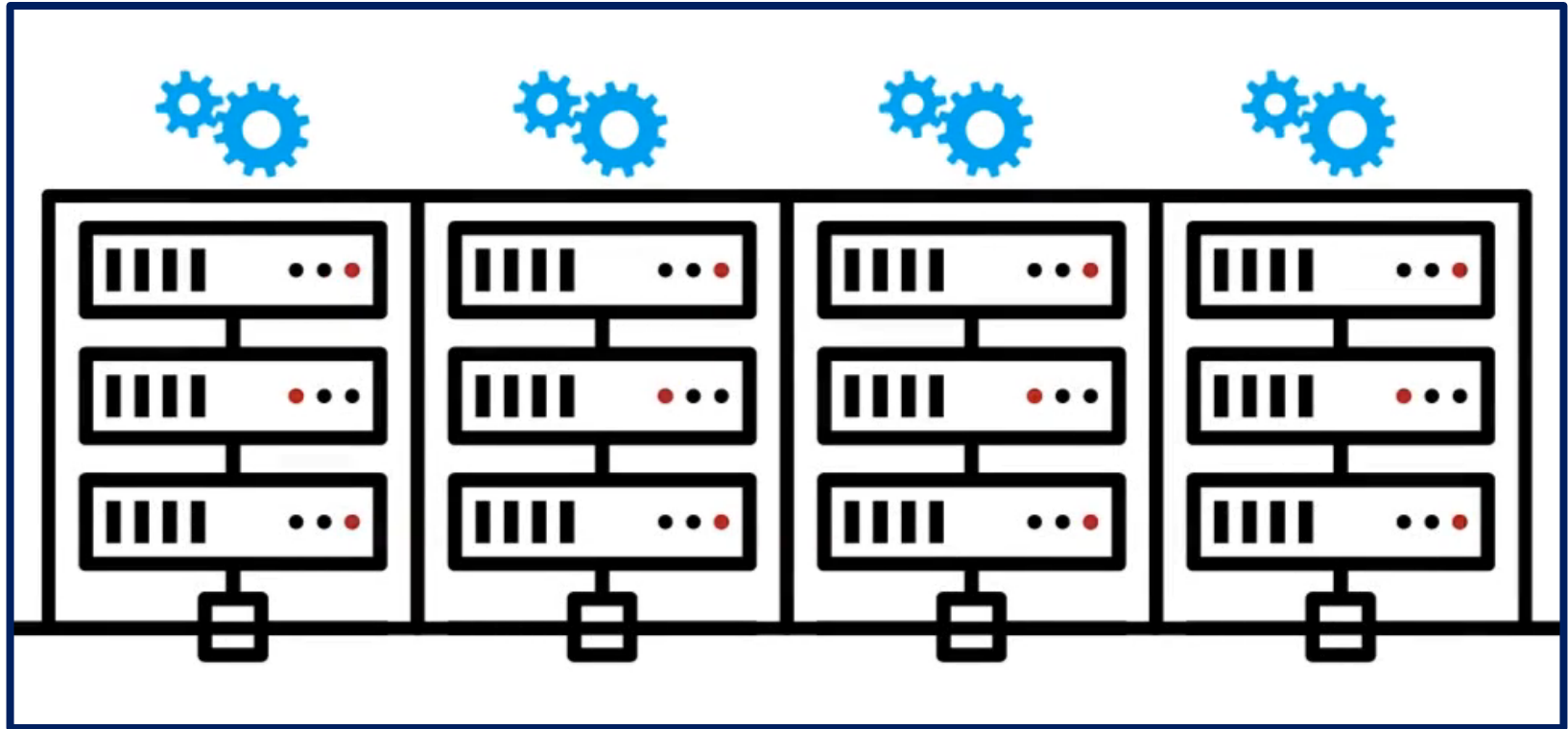
# Per-Password Highly Impacted

- JTR guess # 801
- Not guessed in  $10^{14}$  PCFG guesses



P@ssw0rd!

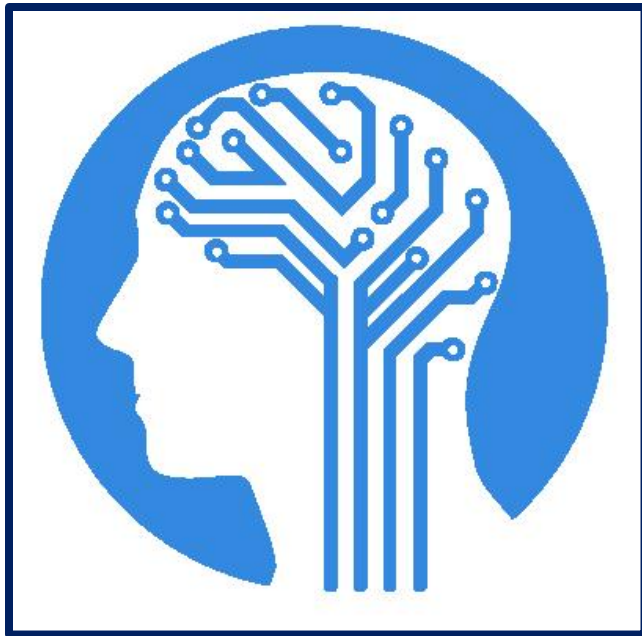
# Neural Networks For Passwords



William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks. In *Proc. USENIX Security Symposium*, 2016.

# Better Password Scoring

- Real-time feedback
- Runs entirely client-side
- Accurately models password guessability



**Recurrent Neural  
Networks (RNNs)**

**LSTM Architecture**

# Generating Passwords

# Generating Passwords

passw  o or maybe 0 or O or ...

# Generating Passwords

passw



Next char is:

A: 3%

B: 1%

C: 0.6%

...

O: 55%

...

Z: 0.01%

0: 20%

1: ...



# Generating Passwords

""

Prob: 100%



Next char is:

A: 3%

B: 2%

C: 5%

...

O: 2%

...

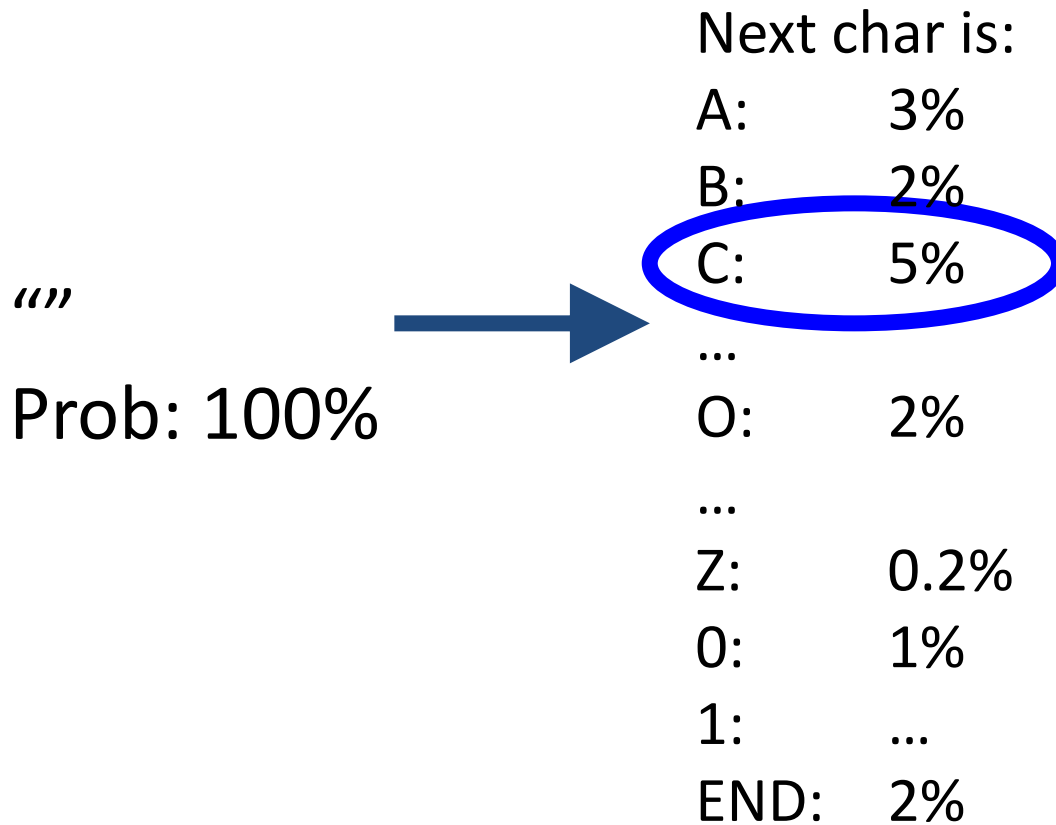
Z: 0.2%

0: 1%

1: ...

END: 2%

# Generating Passwords



# Generating Passwords

“C”

Prob: 5%



# Generating Passwords

“C”  
Prob: 5%



Next char is:

A: 10%

B: 1%

C: 4%

...

O: 8%

...

Z: 0.02%

0: 3%

1: ...

END: 6%

# Generating Passwords

“C”  
Prob: 5%



Next char is:

A: 10%

B: 1%

C: 4%

...

O: 8%

...

Z: 0.02%

0: 3%

1: ...

END: 6%

# Generating Passwords

“CA”

Prob: 0.5%



Next char is:

A: 3%

B: 10%

C: 7%

...

O: 1%

...

Z: 0.03%

0: 2%

1: ...

END: 12%

# Generating Passwords

“CAB”

Prob: 0.05%



Next char is:

A: 3%

B: 10%

C: 7%

...

O: 1%

...

Z: 0.03%

0: 2%

1: ...

END: 3%

# Generating Passwords

“CAB”

Prob: 0.05%



Next char is:

A: 4%

B: 3%

C: 1%

...

O: 2%

...

Z: 0.01%

0: 4%

1: ...

END: 12%



# Generating Passwords

“CAB”

Prob: 0.05%



Next char is:

A: 4%

B: 3%

C: 1%

...

O: 2%

...

Z: 0.01%

0: 4%

1: ...

END: 12%

# Generating Passwords

“CAB”

Prob: 0.006%

# Descending Probability Order

CAB - 0.006%  
CAC - 0.0042%  
ADD1 - 0.002%  
CODE - 0.0013%  
...

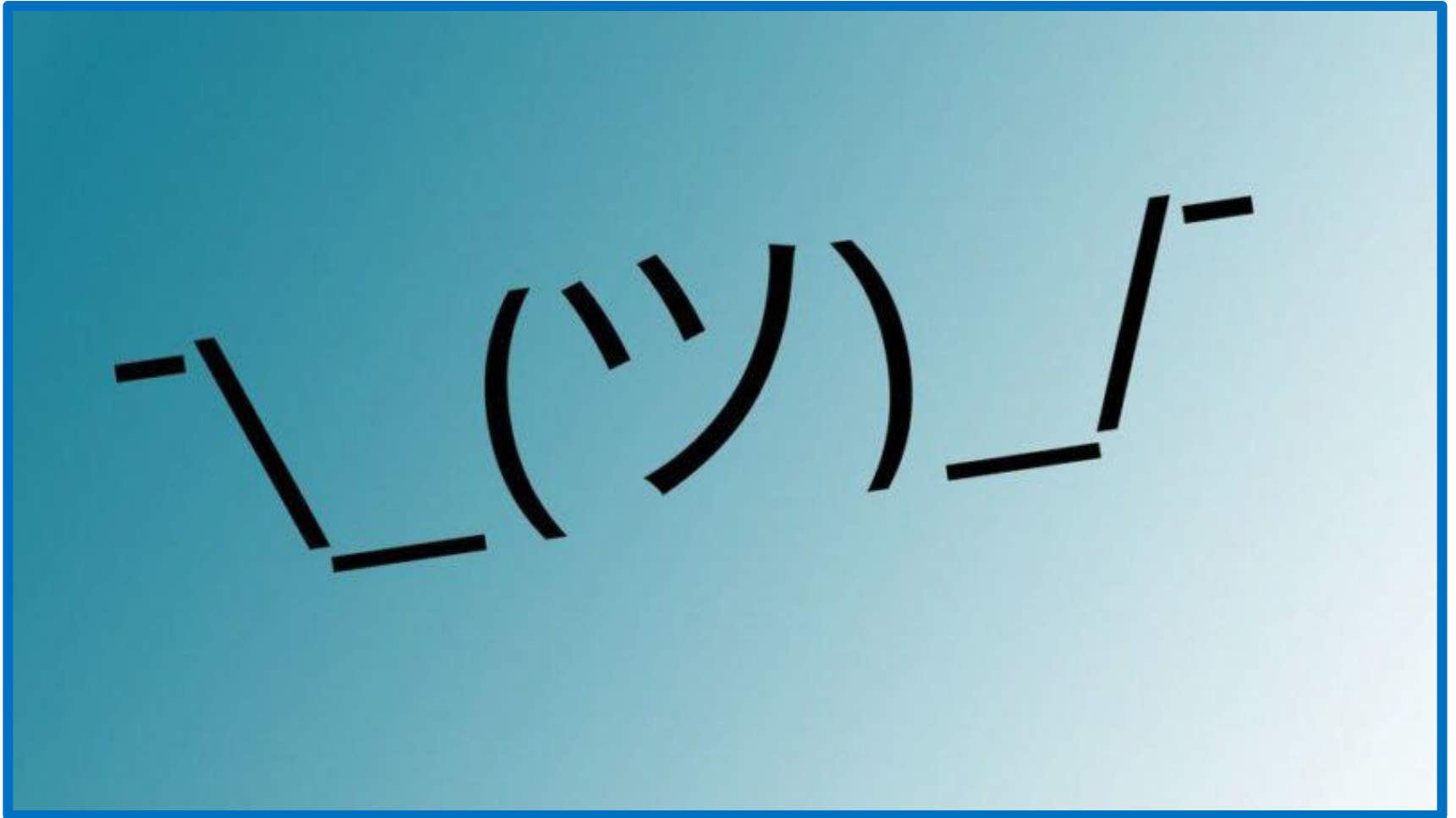
# Design Space

- Model size: 3mb (browser) vs. 60mb (GPU)
- Transference learning
  - Novel password-composition policies
- Training data
  - Natural language
- (Many others)

# Key Results

- Neural networks produce better guesses than previous methods
- Larger model not a major advantage
- Browser implementation in Javascript

# Intelligibility (Explanations)



# Building a Data-Driven Meter

The screenshot shows a web form titled "Create Your Password". It contains three input fields: "Username", "Password", and "Confirm Password". The "Password" field contains the text "Mypassword123" and has a red progress bar below it. A checkbox labeled "Show Password & Detailed Feedback" is checked. A blue "Continue" button is located at the bottom right of the form. A grey feedback box on the right side of the form displays the following content:

Your password is very easy to guess.

- Don't use dictionary words ([password](#))
- Capitalize a letter in the middle, rather than the first character ([Why?](#))
- Consider inserting digits into the middle, not just at the end ([Why?](#))

A better choice: **My123passwoRzd**

[How to make strong passwords](#)

Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, William Melicher. Development and Evaluation of a Data-Driven Password Meter. In *Proc. CHI*, 2017.



We designed & tested a meter with:

- 1) Principled strength estimates
- 2) Data-driven feedback to users









- We designed & tested a meter with:
- 1) Principled strength estimates
  - 2) Data-driven feedback to users



# Provide Intelligent Explanations

Unic0rns

Don't use simple transformations of words or phrases (**unicorns** → **Unic0rns**)

Capitalize a letter in the middle, rather than the first character

- 21 characteristics
- Weightings determined with regression

# After Requirements Are Met...

## Create Your Password

Username  
blase

Password  
.....

Show Password & Detailed Feedback

Confirm Password

[Continue](#)

**Your password could be better.**

- Don't use dictionary words or words used on Wikipedia [\(Why?\)](#)
- Consider inserting digits into the middle [\(Why?\)](#)
- Consider making your password longer [\(Why?\)](#)

[See Your Password With Our Improvements](#)

[How to make strong passwords](#)

# ...Displays Score Visually

## Create Your Password

Username  
blase

Password  
.....

**Progress Bar:** [Yellow bar indicating score]

Show Password & Detailed Feedback

Confirm Password

[Continue](#)

**Your password could be better.**

- Don't use dictionary words or words used on Wikipedia [\(Why?\)](#)
- Consider inserting digits into the middle [\(Why?\)](#)
- Consider making your password longer [\(Why?\)](#)

[See Your Password With Our Improvements](#)

[How to make strong passwords](#)

# ...Provides Text Feedback

## Create Your Password

Username  
blase

Password  
.....

Show Password & Detailed Feedback

Confirm Password

[Continue](#)

Your password could be better.

- Don't use dictionary words or words used on Wikipedia [\(Why?\)](#)
- Consider inserting digits into the middle [\(Why?\)](#)
- Consider making your password longer [\(Why?\)](#)

[See Your Password With Our Improvements](#)

[How to make strong passwords](#)



# ...Gives Detail (Password Shown)

## Create Your Password

Username  
blase

Password  
CryptoUnicorn3|  
 Show Password & Detailed Feedback

Confirm Password

[Continue](#)

Your password could be better.

- Don't use dictionary words (Unicorn) or words used on Wikipedia (Crypto) [\(Why?\)](#)
- Consider inserting digits into the middle, not just at the end [\(Why?\)](#)
- Consider making your password longer than 14 characters [\(Why?\)](#)

A better choice: C3ryptoUniCORN@

[How to make strong passwords](#)

# ...Offers Explanations

## Create Your Password

Username  
blase

Password  
CryptoUnicorn3|  
 Show Password & Detailed Feedback

Confirm Password

[Continue](#)

Your password could be better.

- Don't use dictionary words (Unicorn) or words used on Wikipedia (Crypto) [\(Why?\)](#)
- Consider inserting digits into the middle, not just at the end [\(Why?\)](#)
- Consider making your password longer than 14 characters [\(Why?\)](#)

A better choice: C3ryptoUniCORN@  
[How to make strong passwords](#)



# Explanations Shown in Modal



The screenshot shows a modal window with a dark red header containing the title "Ways to Improve Your Password" and a close button. Below the header is a text input field containing the password "CryptoUnicorn3". A progress bar below the input field is partially filled with yellow, indicating a moderate password strength. Below the progress bar is a checkbox labeled "Show Password & Detailed Feedback" which is checked. The main content of the modal includes a suggestion for a better password: "A better choice: C3ryptoUniC0rn@". Below this is the text "Your password could be better." followed by a list of three bullet points providing feedback on the password's weaknesses. At the bottom of the modal is a blue "OK" button.

Ways to Improve Your Password

CryptoUnicorn3

Show Password & Detailed Feedback

A better choice: C3ryptoUniC0rn@

Your password could be better.

- Don't use dictionary words (Unicorn) or words used on Wikipedia (Crypto)  
Attackers use software that automatically guesses millions of words commonly found in dictionaries, wordlists, or other people's passwords
- Consider inserting digits into the middle, not just at the end  
38% of people also put digits at the end of the password
- Consider making your password longer than 14 characters  
In recent years, attackers have gotten much better at guessing passwords under 16 characters

[How to make strong passwords](#)

OK

# Standard Feedback

Create Your Password

Username  
blase

Password

Confirm Password

Continue

Your password could be better.

- Don't use dictionary words (Unicorn) or words used on Wikipedia (Crypto) [\(Why?\)](#)
- Consider making your password longer than 14 characters [\(Why?\)](#)

A better choice: C3ryptoUniCorn@

A better choice: C3ryptoUniCorn@

[How to make strong passwords](#)