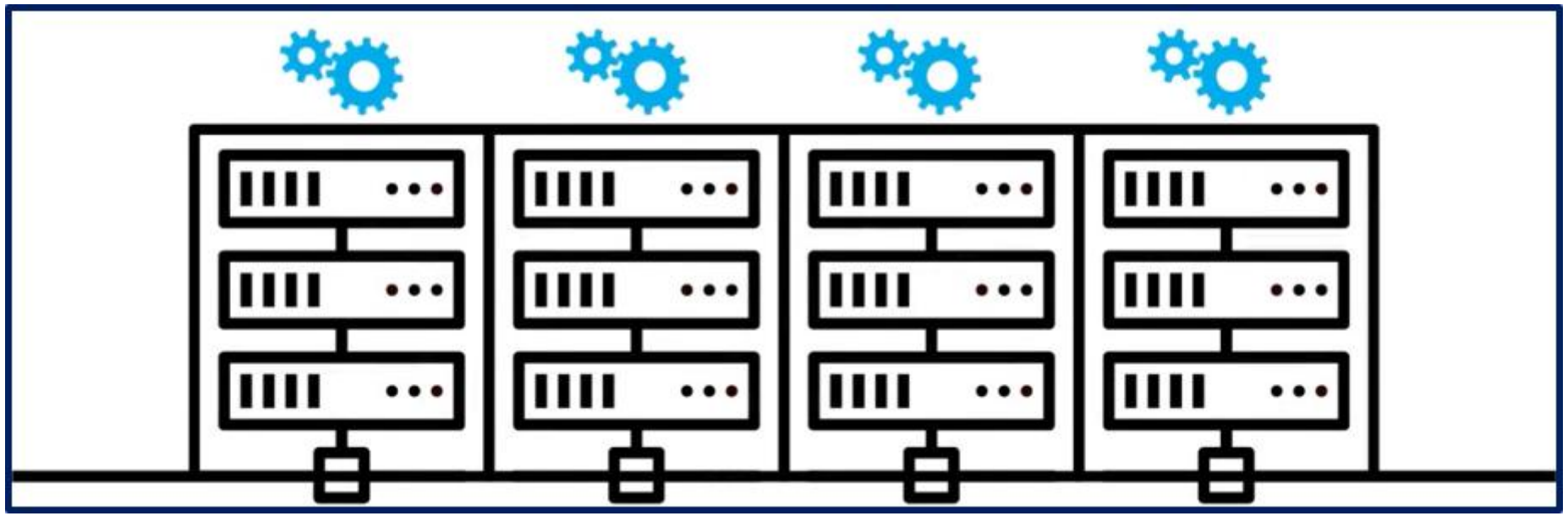
Authentication
Part 2

Blase Ur (and David Cash and Grant Ho)
May 14, 2026
CMSC 23200

THE UNIVERSITY OF CHICAGO

Password Guessing (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 (no longer used): Shannon entropy
- Disadvantages of statistical approaches
  - Entropy does not consider human tendencies
  - Usually no per-password estimates
  - Huge sample required for accuracy (since many passwords are related to each other)
  - Does not model real-world attacks

Parameterized Guessability

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

Parameterized Guessability

`letmein!`

Guess # 6

Parameterized Guessability

`Dav!Dc@SHHH182`

Guess # 366,163,847,194

Parameterized Guessability

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

Guess # past cutoff

Guessing Model

80d561388725fa74f2d03cd16e1d687c

Guessing Model

80d561388725fa74f2d03cd16e1d687c

1. h(“123456”) = e10adc3949ba59abbe56e057f20f883e

Guessing Model

80d561388725fa74f2d03cd16e1d687c

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

Guessing Model

80d561388725fa74f2d03cd16e1d687c

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

Guessing Model

80d561388725fa74f2d03cd16e1d687c

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

Guessing Model

80d561388725fa74f2d03cd16e1d687c

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

Guessing 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**

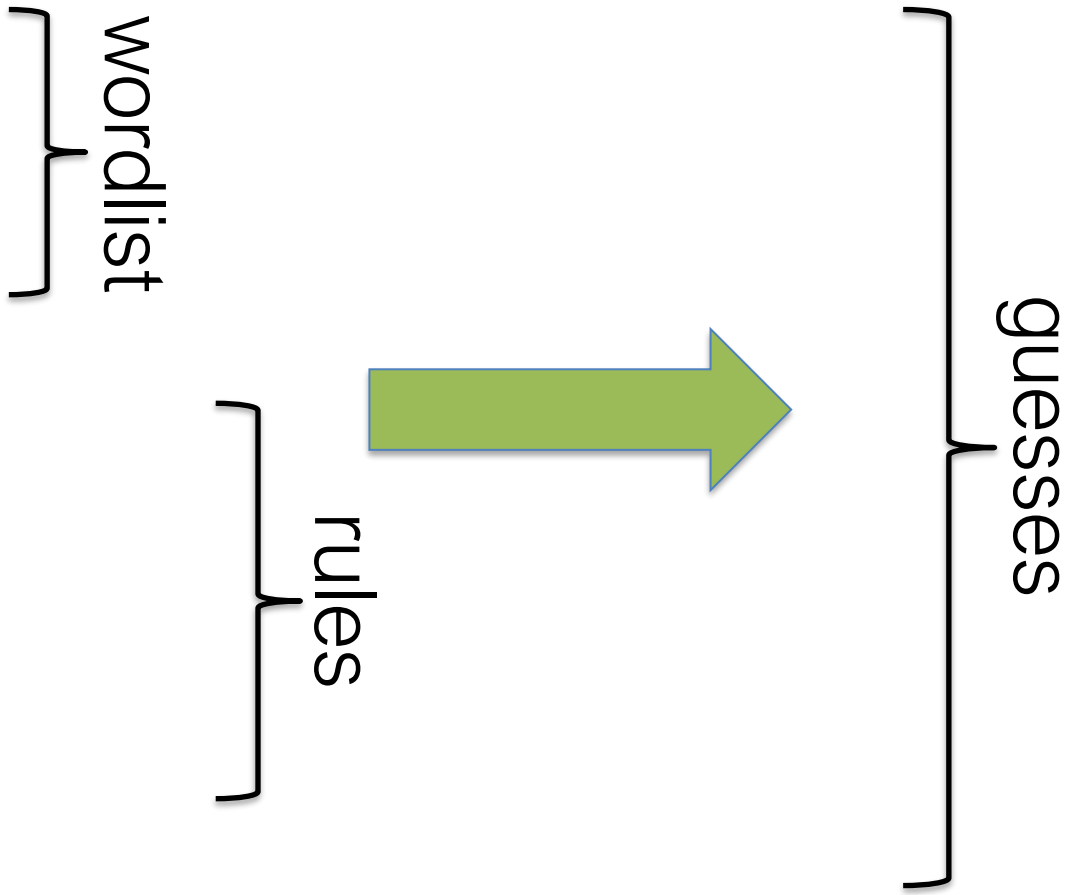
Some Key Password-Cracking Approaches

- Brute force (or selective brute force): most comprehensive
- Wordlist: highest success rate per guess
- Mangled wordlist
  - Hashcat and John the Ripper
- Markov models
- Probabilistic Context-Free Grammar
- Deep learning (e.g., RNNs)
- In practice: manual, iterative updates

John the Ripper

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

John the Ripper



John the Ripper

uchicago

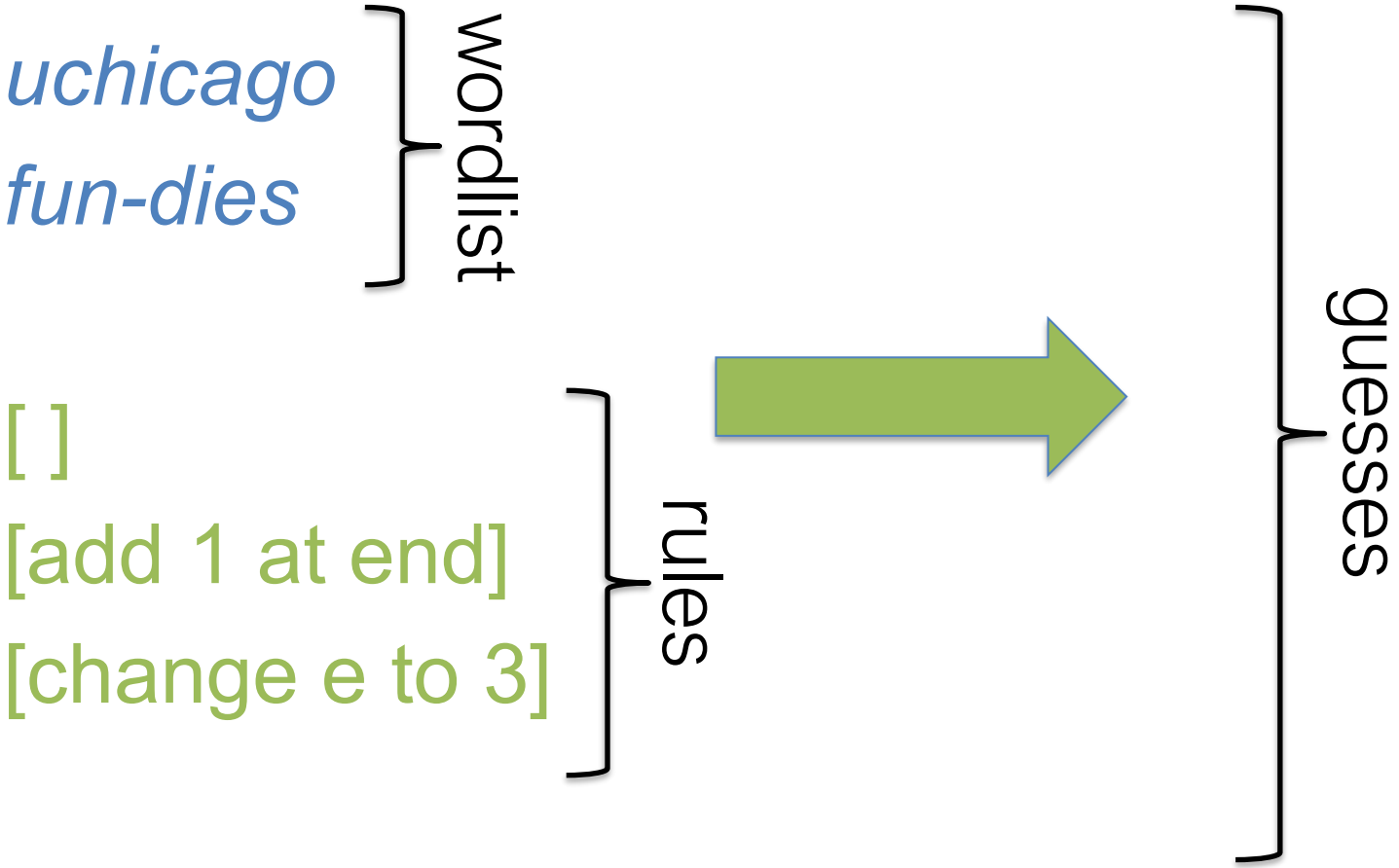
fun-dies

wordlist

rules

guesses

John the Ripper



John the Ripper

uchicago

fun-dies

wordlist

[ ]

[add 1 at end]

[change e to 3]

rules

uchicago

fun-dies

guesses

John the Ripper

uchicago
fun-dies

wordlist

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

rules

uchicago
fun-dies
uchicago1
fun-dies1

guesses

John the Ripper

uchicago
fun-dies

wordlist

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

rules

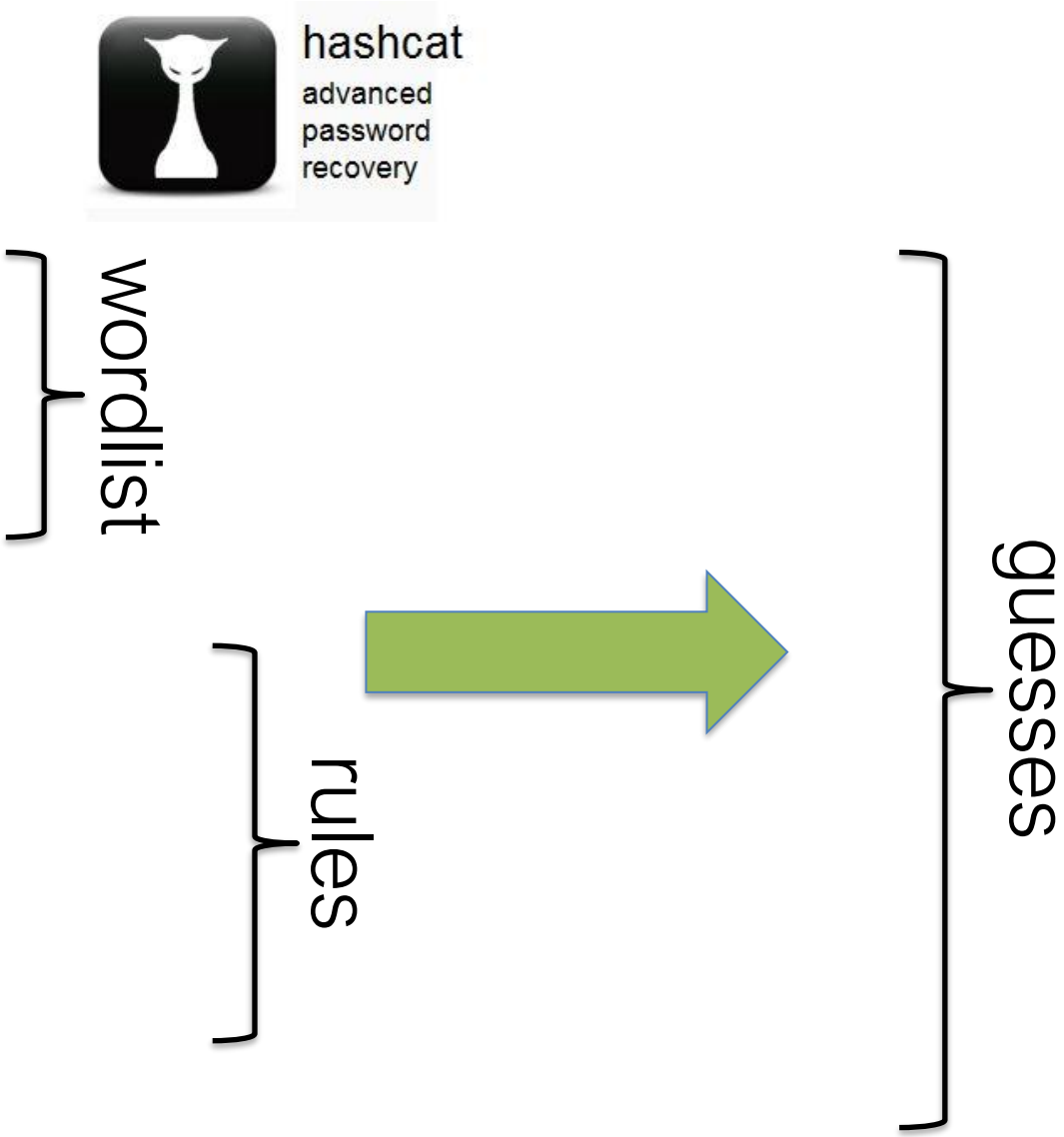
uchicago
fun-dies
uchicago1
fun-dies1
uchicago
fun-di3s

guesses

Hashcat

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

Hashcat


hashcat
advanced
password
recovery
wordlist
rules
guesses

Hashcat

uchicago

fun-dies

wordlist

[ ]

[add 1 at end]

[change e to 3]

rules

guesses

Hashcat

uchicago

fun-dies

wordlist

[ ]

[add 1 at end]

[change e to 3]

rules

uchicago

uchicago1

uchicago

guesses

Hashcat

uchicago

fun-dies

wordlist

[ ]

[add 1 at end]

[change e to 3]

rules

uchicago

uchicago1

uchicago

fun-dies

fun-dies1

fun-di3s

guesses

Hashcat Mangling-Rule Language

Name	Function	Description	Example Rule	Input Word	Output Word	Note
Nothing	:	do nothing	:	p@ssW0rd	p@ssW0rd	
Lowercase	l	Lowercase all letters	l	p@ssW0rd	p@ssw0rd	
Uppercase	u	Uppercase all letters	u	p@ssW0rd	P@SSW0RD	
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@SSW0RD	
Toggle Case	t	Toggle the case of all characters in word.	t	p@ssW0rd	P@SSw0RD	
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	$1	p@ssW0rd	p@ssW0rd1	
Prepend Character	^X	Prepend character X to front	^1	p@ssW0rd	1p@ssW0rd	
Truncate left	[	Deletes first character	[	p@ssW0rd	@ssW0rd	
Trucate right	]	Deletes last character	]	p@ssW0rd	p@assW0r	
Delete @ N	DN	Deletes character at position N	D3	p@ssW0rd	p@sW0rd	*
Extract range	xNM	Extracts M characters, starting at position N	x04	p@ssW0rd	p@ss	* #
Omit range	ONM	Deletes M characters, starting at position N	O12	p@ssW0rd	psW0rd	*
Insert @ N	iNX	Inserts character X at position N	i4!	p@ssW0rd	p@ss!W0rd	*
Overwrite @ N	oNX	Overwrites character at position N with X	o3$	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@$$W0rd	
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@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	+

Hashcat Mangling-Rule Language

*05 O03 d '7

Hashcat Mangling-Rule Language

*05 O03 d '7

Switch the first and the sixth char;

Delete the first three chars;

Duplicate the whole word;

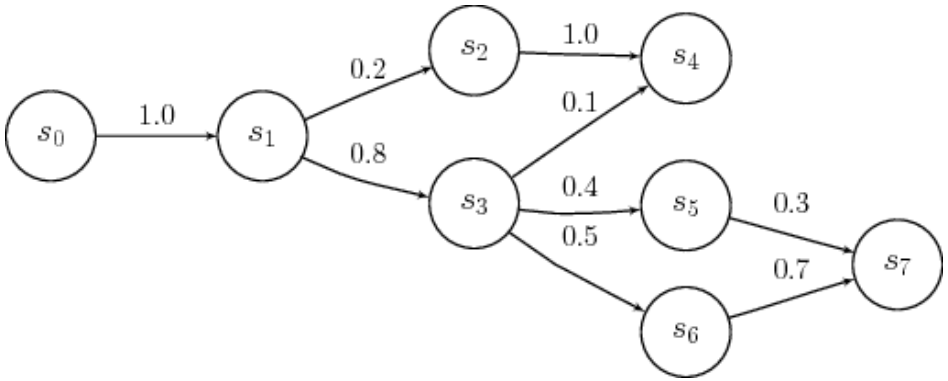
Truncate the word to length 7;

Hashcat (Other Modes)

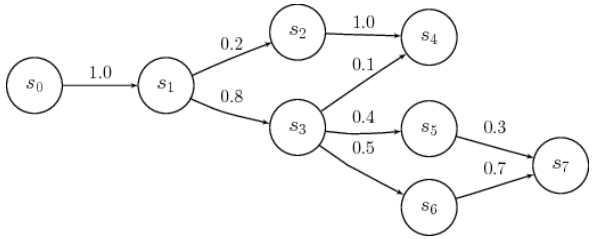
- Mask attack (brute force within a specified character-class structure)
- Combinator attacks
- Hybrid attacks
- Many more!

Markov Models

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



Markov Models



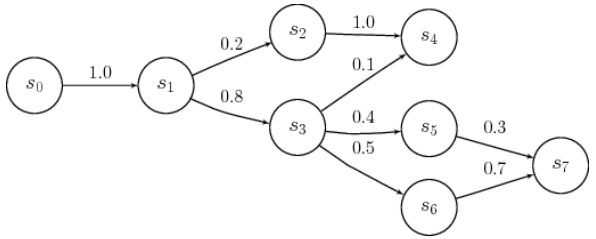
chic4gooo

2-gram model (1 character of context):

[start] ➔ **c** (1.0)

4 ➔ **g** (1.0)

Markov Models



chic4gooo

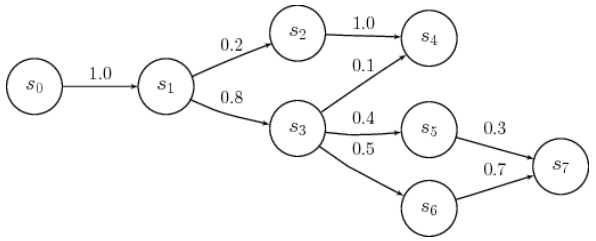
2-gram model (1 character of context):

[start] ➔ **c** (1.0)

4 ➔ **g** (1.0)

c ➔ **h** (0.5), **4** (0.5)

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)

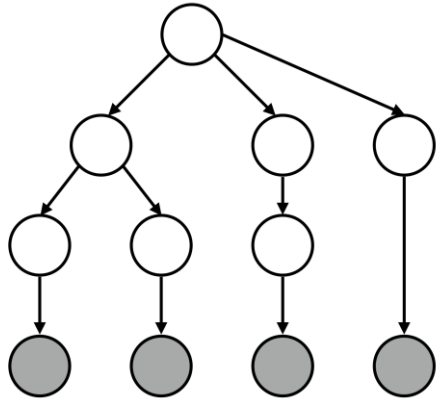
h ➔ **i** (1.0)

i ➔ **c** (1.0)

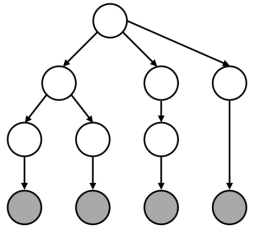
o ➔ **o** (0.67) **[end]** (0.33)

Probabilistic Context-Free Grammar

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



PCFG



passwordpassword

password123

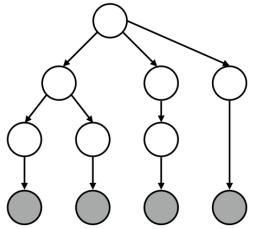
usenix3

5ecurity

iloveyou

nirvanaa123

PCFG



*password**password*

*password**123*

*usenix**3*

*5**ecurity*

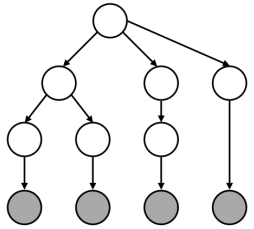
iloveyou

*nirvanaa**123*

Structure Model:

$\mathbf{L_{16}}$	(1/6)
$\mathbf{L_8D_3}$	(2/6)
$\mathbf{L_6D_1}$	(1/6)
$\mathbf{D_1L_7}$	(1/6)
$\mathbf{L_8}$	(1/6)

PCFG



passwordpassword

password123

usenix3

5ecurity

iloveyou

nirvanaa123

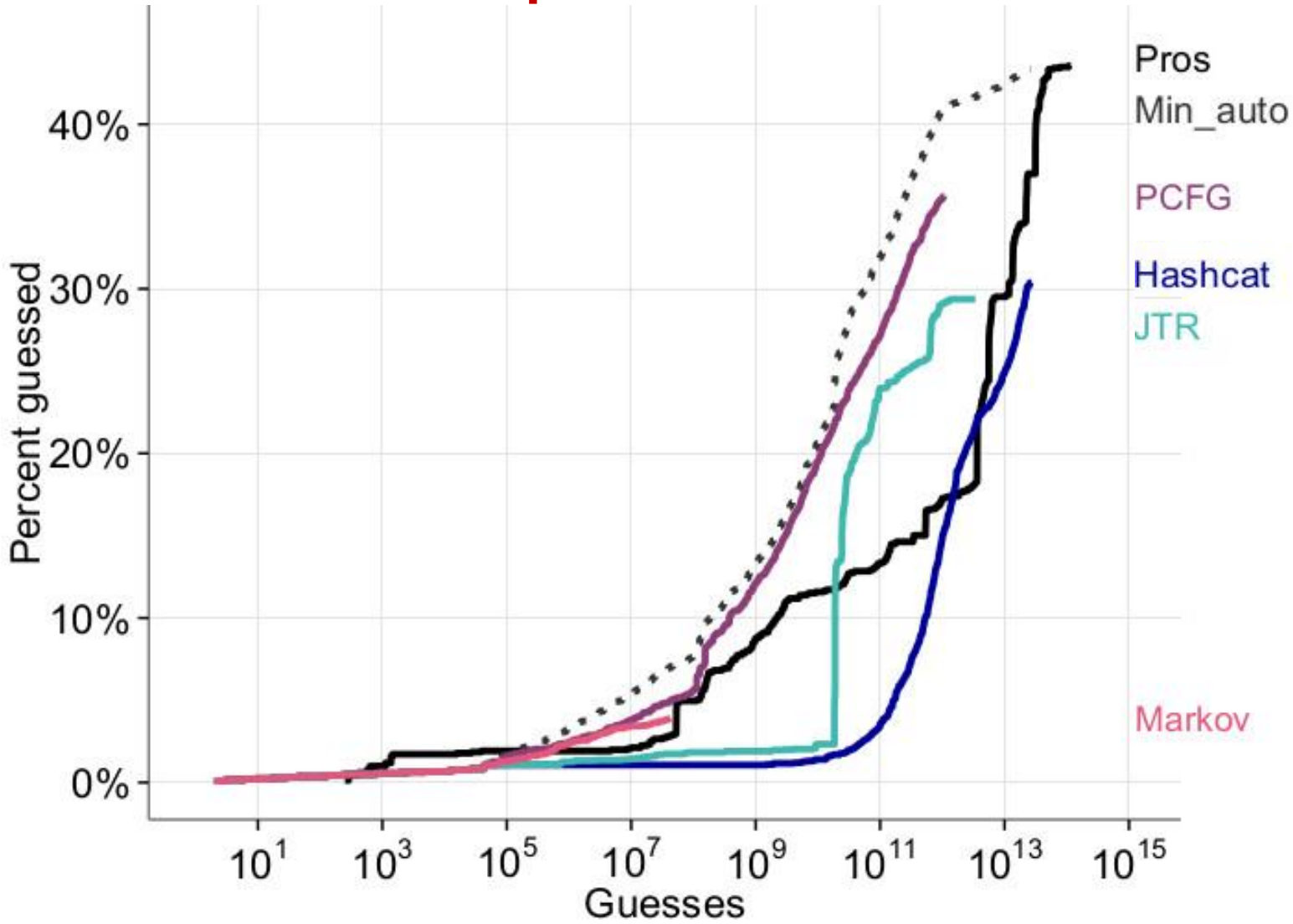
Digit Model:

D_1 → 3 (0.5) **5** (0.5)

D_3 → 123 (1.0)

Repeat for letters, etc.

Comparison for Complex Passwords



Per-Password Highly Impacted By Approach

P@ssw0rd!

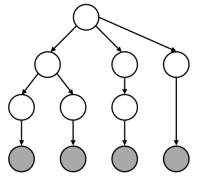
Per-Password Highly Impacted By Approach

- JTR guess # 801 

`P@ssw0rd!`

Per-Password Highly Impacted By Approach

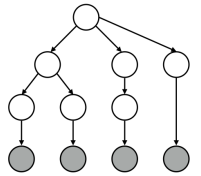
- JTR guess # 801 

- Not guessed in 10^{14} PCFG guesses 

P@ssw0rd!

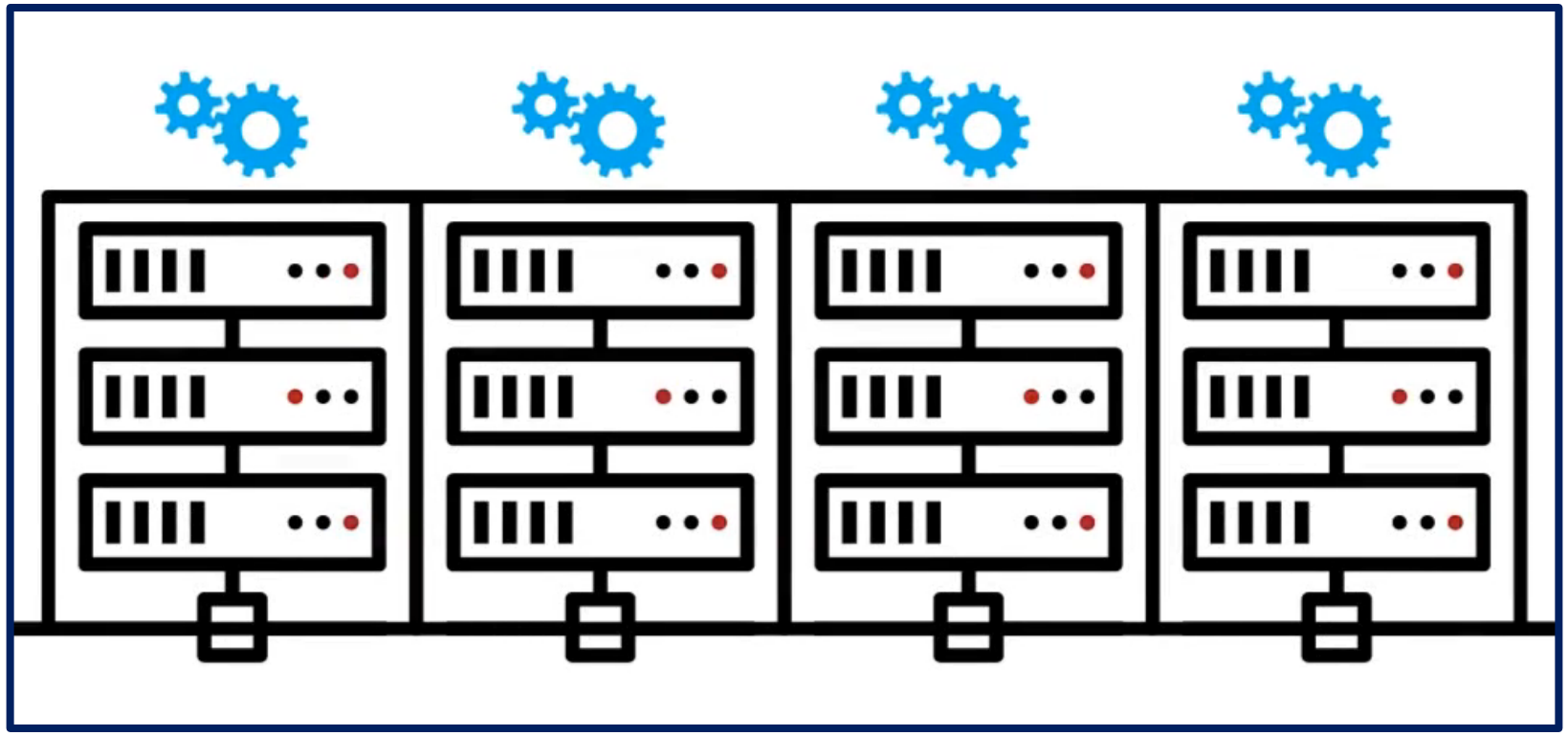
Per-Password Highly Impacted By Approach

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



P@ssw0rd!

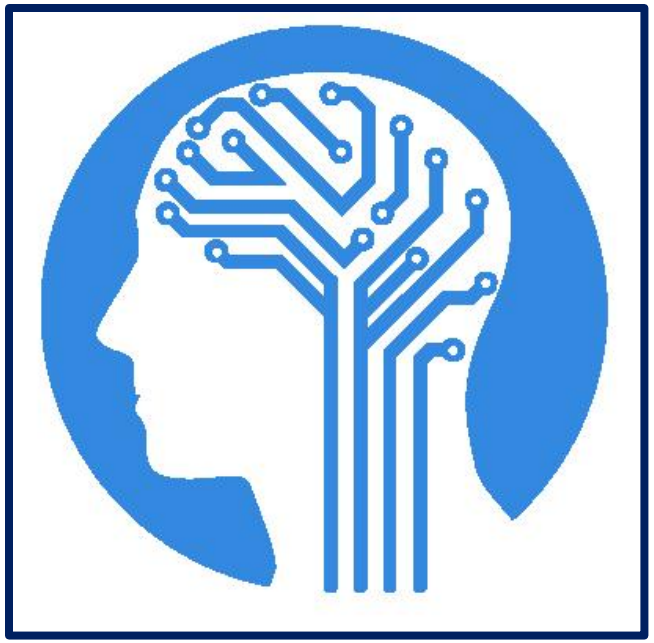
Neural Networks For Passwords



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: …

Design Space

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

Choosing a Model For Guessing

- Need to consider the computational cost carefully!
  - How do you produce an **ordered list** of guesses
  - For many probabilistic models, produce all candidates in a given probability range and then sort
- For defense purposes (strength estimation), probabilistic models can produce estimates via Monte Carlo sampling

Authentication in Practice: Password Managers

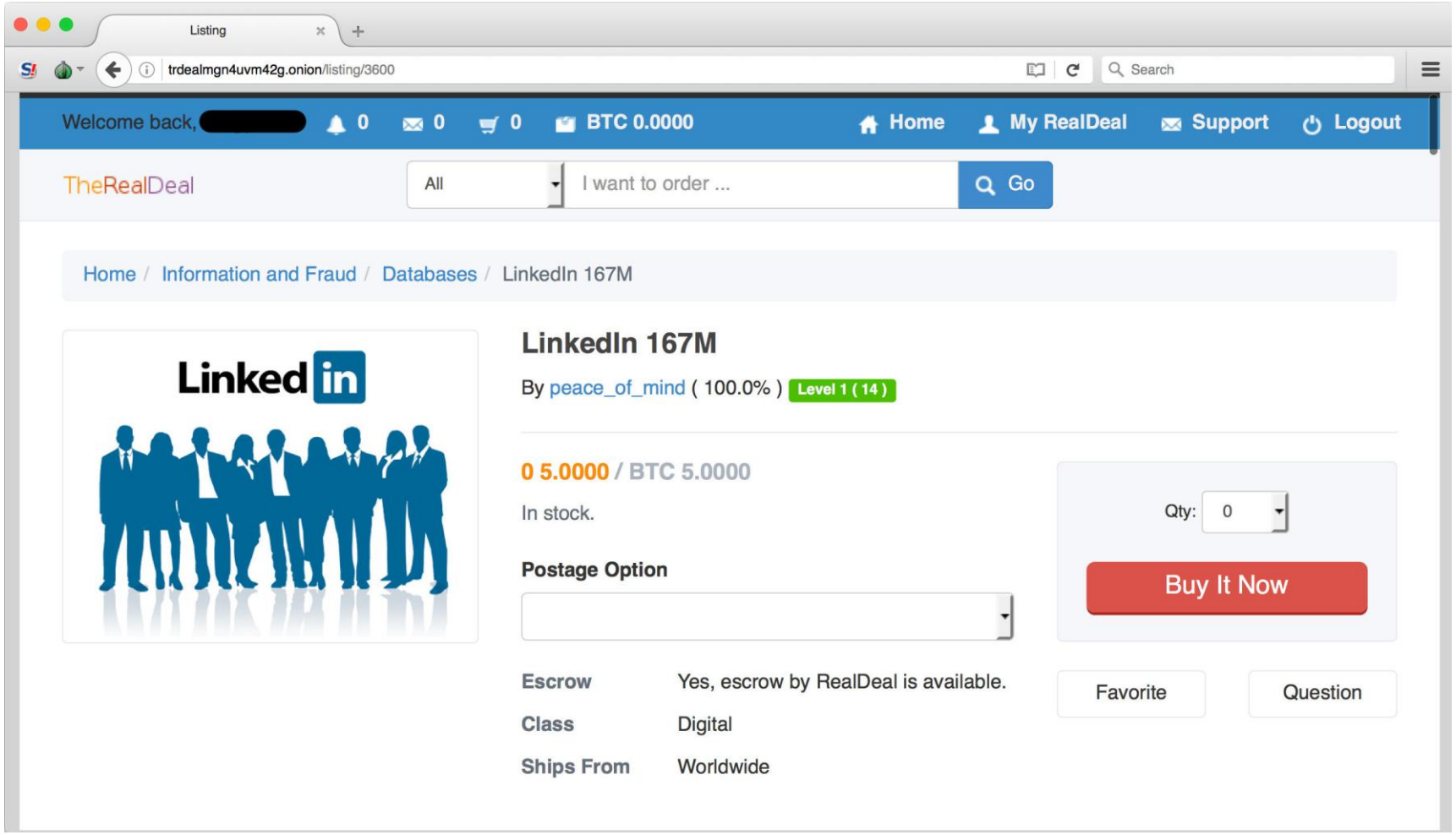
Password Managers

- Trust all passwords to a single master password (still a good idea in most cases)
  - Also trust software
  - Centralized vs. decentralized architectures

1Password

Authentication in Practice: Password Reuse ☹

Monitoring the Black Market



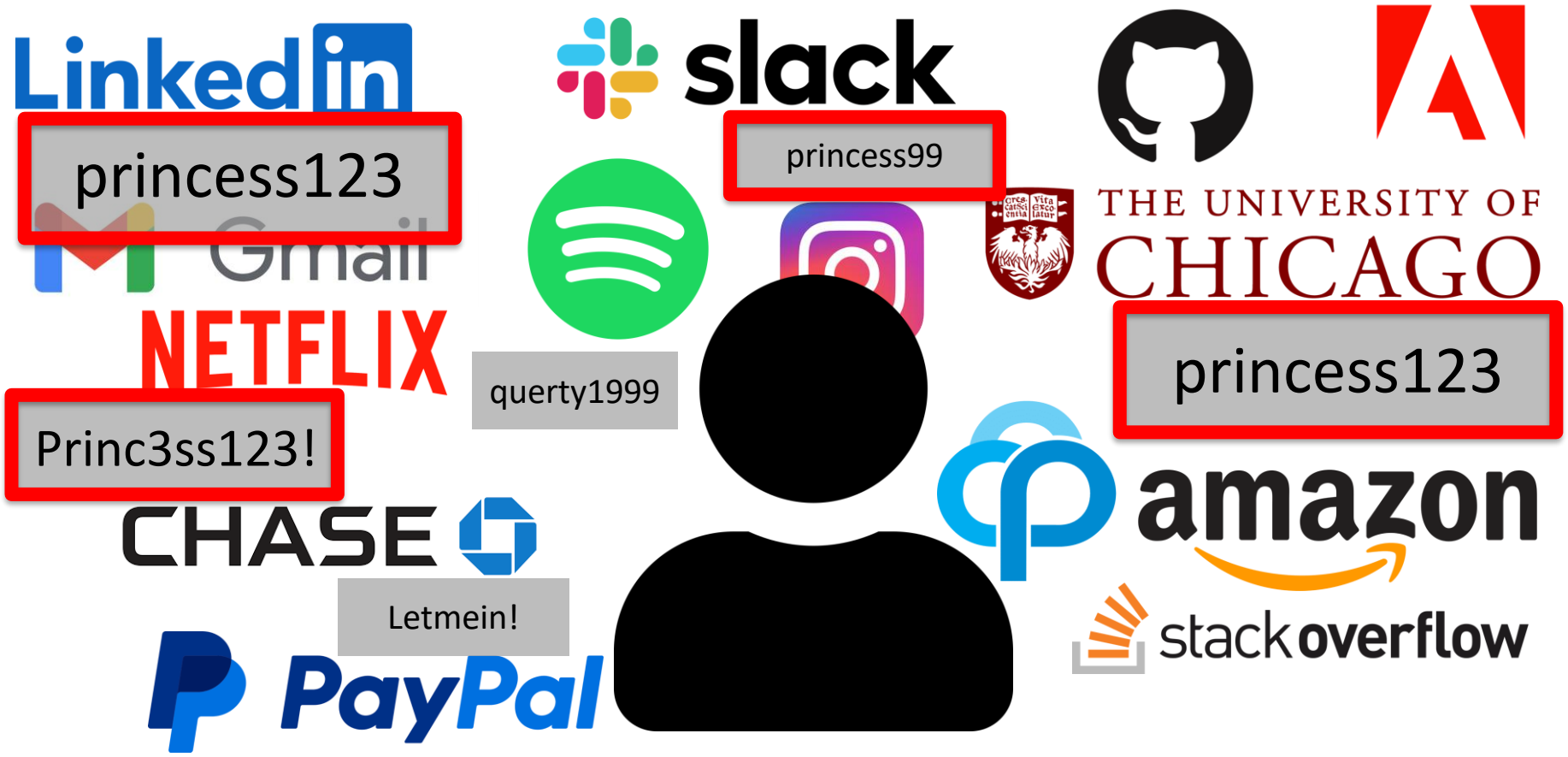
SECURITY

Facebook buys black market passwords to keep your account safe

The company's security chief says account safety is about more than just building secure software.

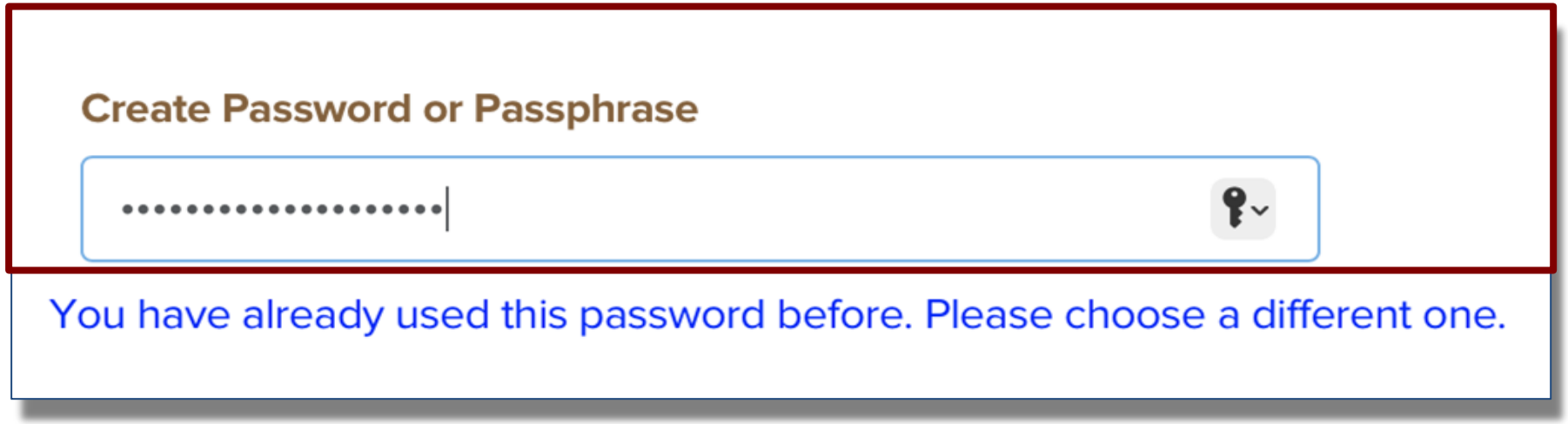
BY KATIE COLLINS | NOVEMBER 9, 2016 12:56 PM PST

Measuring Vulnerability to Password Reuse



Alexandra Nisenoff, Maximilian Golla, Miranda Wei, Juliette Hainline, Hayley Szymanek, Annika Braun, Annika Hildebrandt, Blair Christensen, David Langenberg, Blase Ur. A Two-Decade Retrospective Analysis of a University's Vulnerability to Attacks Exploiting Reused Passwords. In *Proc. USENIX Security Symposium*, 2023.

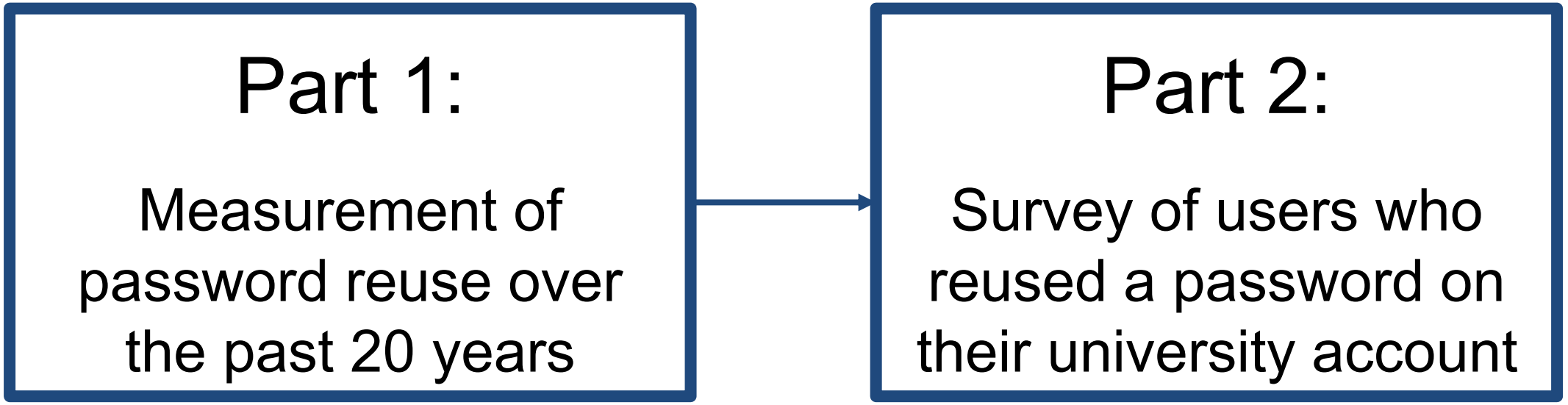
UChicago Password History Database



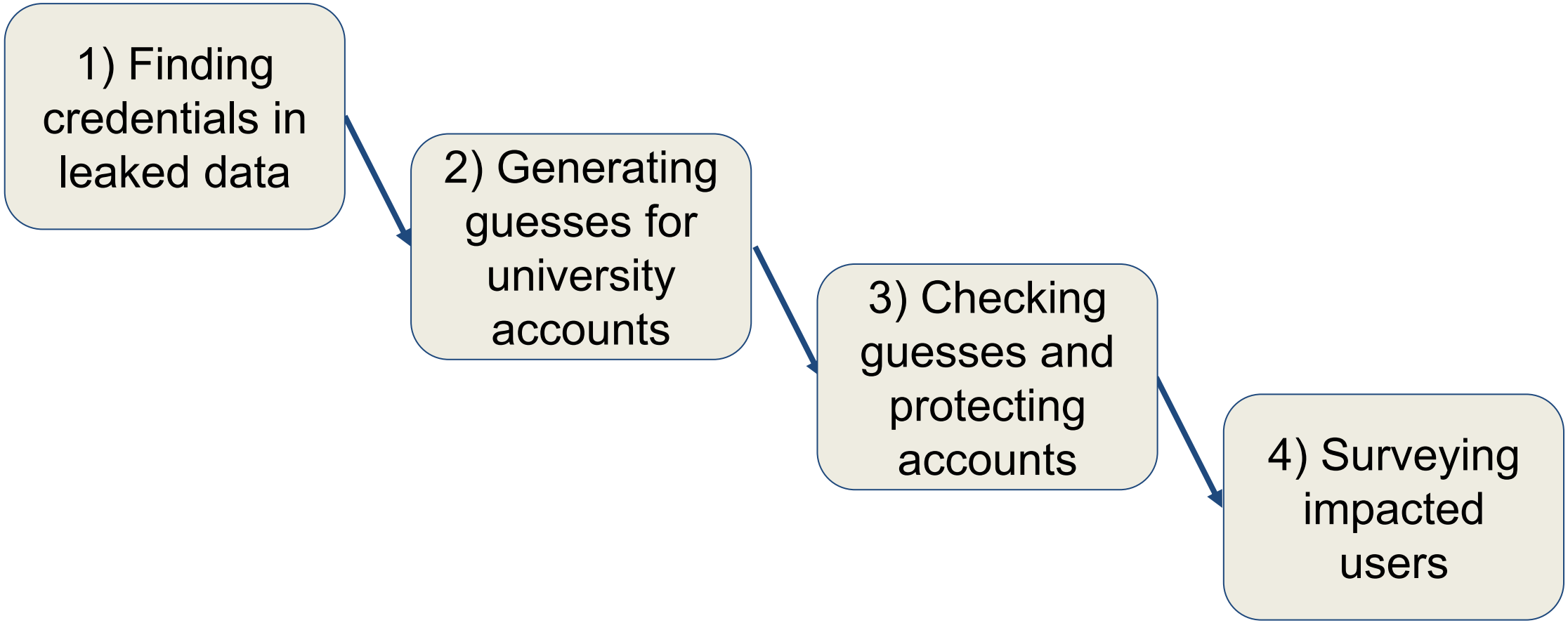
UChicago Password History Database

Username	Hash of Password	Created	Changed	
weimf	**hash(**i<3cats1234**)**	Sep 17, 2016	Jul 1, 2019	…
weimf	**hash(**i<3cats2019!**)**	Jul 1, 2019	present	…
hszym	**hash(**p@nc@kes99**)**	Aug 15, 2018	present	…
blase	**hash(**cyb#rS3curity**)**	Nov 10, 2017	Aug 23, 2019	…
…	…	…	…	…

Study Flow


Part 1:
Measurement of password reuse over the past 20 years
Part 2:
Survey of users who reused a password on their university account

Study Flow


1) Finding credentials in leaked data
2) Generating guesses for university accounts
3) Checking guesses and protecting accounts
4) Surveying impacted users

Study Flow

1) Finding credentials in leaked data

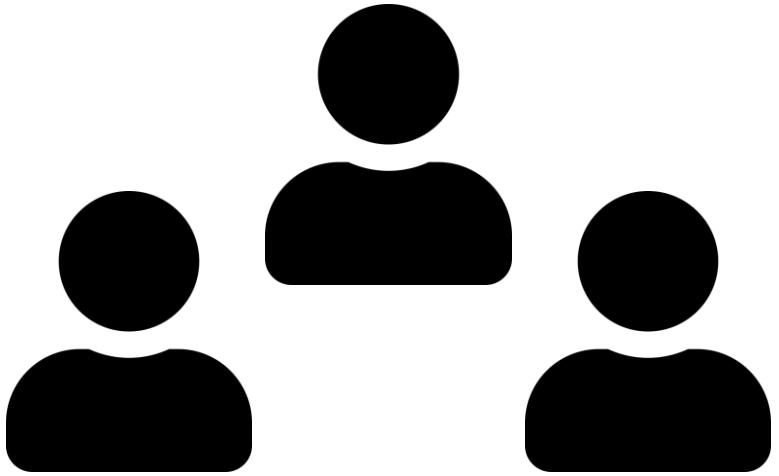
2) Generating guesses for university accounts

3) Checking guesses and protecting accounts

4) Surveying impacted users

Starting Point: UChicago Usernames

1) Finding credentials in leaked data



227,976 Usernames

Data Source 1: Data Breaches

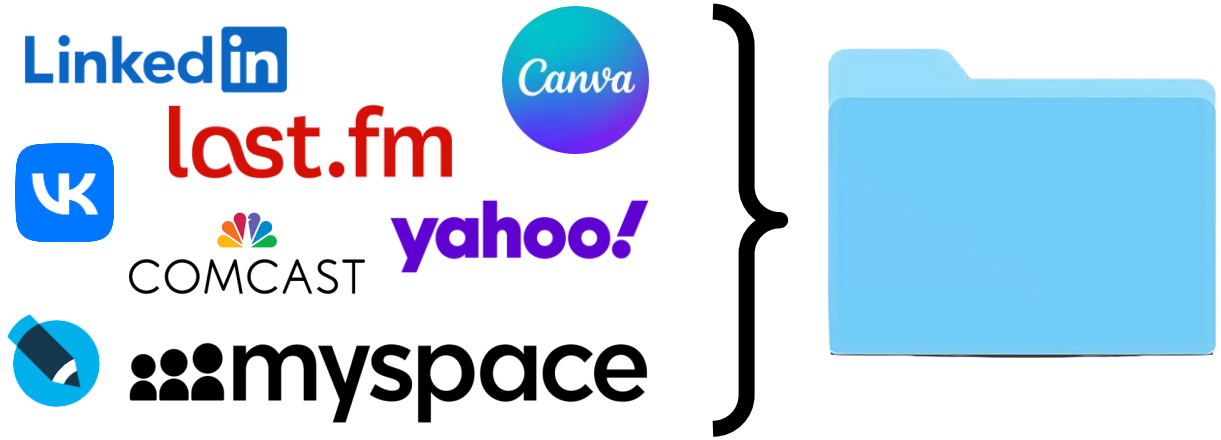
1) Finding credentials in leaked data

- 450 individual service breaches

Data Source 2: Breach Compilations

1) Finding credentials in leaked data

- 12 large breach compilations
  - Collection #1, Anti Public, etc.



Matching Strategies

1) Finding credentials in leaked data

username: **blase**

blase@uchicago.edu

blase@cmu.edu

blase

blase99@gmail.com

Study Flow

1) Finding credentials in leaked data

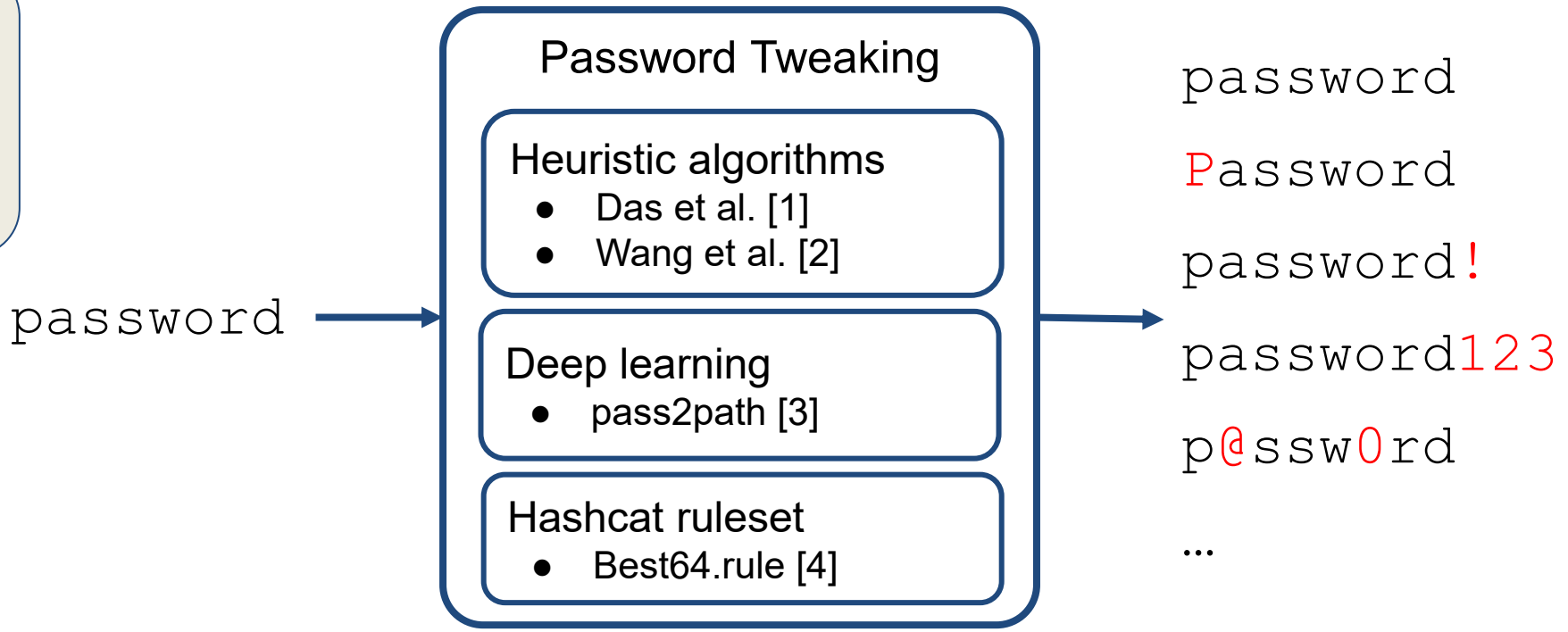
2) Generating guesses for university accounts

3) Checking guesses and protecting accounts

4) Surveying impacted users

Tweaking Guesses (Similar Passwords)

2) Generating guesses for university accounts



[1] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The Tangled Web of Password Reuse. In *Symposium on Network and Distributed System Security*, NDSS, 2014.
[2] C. Wang, S.Jan, H. Hu, D. Bossart, and G. Wang. The Next Domino to Fall: Empirical Analysis of User Passwords across Online Services. CODASPY, 2018.
[3] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart. Beyond Credential Stuffing: Password Similarity Models using Neural Networks. IEEE SP , 2019.
[4] J. Steube ("atom") and Community. Official Best64 Challenge Thread, 2012. https://hashcat.net/forum/thread-1002- post-5284.html#pid5284

Study Flow

1) Finding credentials in leaked data

2) Generating guesses for university accounts

3) Checking guesses and protecting accounts

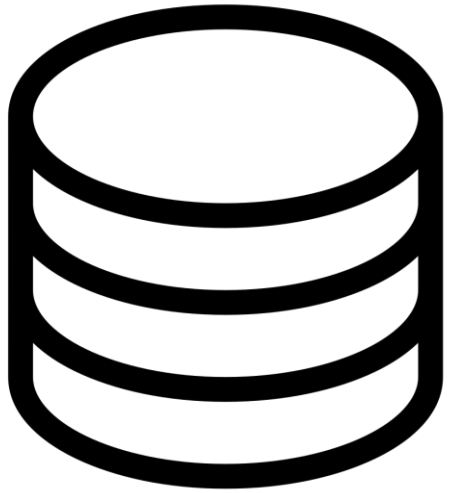
4) Surveying impacted users

Transferring Guesses

3) Checking guesses and protecting accounts

Username	Password	…
nisenoff	letmein123	…
blase	qwerty123	…
mgolla	Monkey<3	…
…	…	…

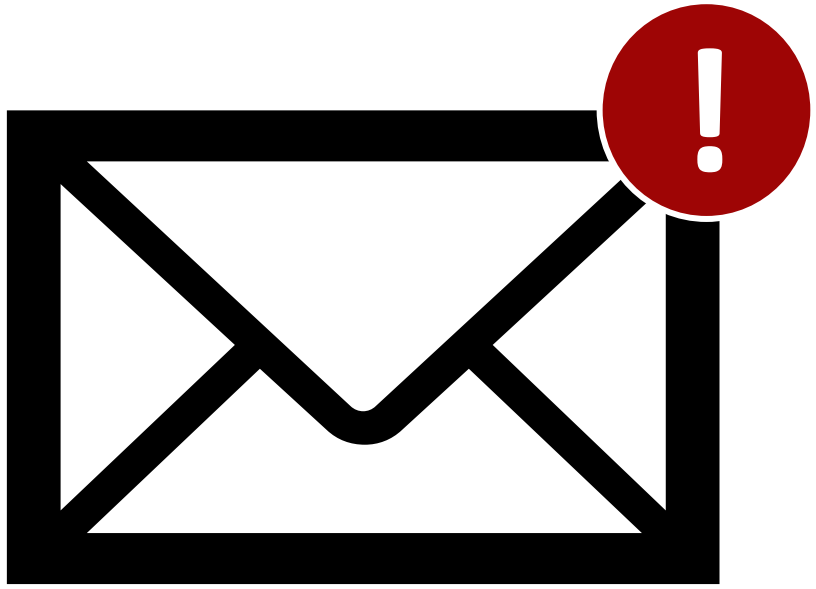
Credential Guesses



UChicago Password History Database

Notifying Vulnerable Users

3) Checking guesses and protecting accounts



Study Flow

1) Finding credentials in leaked data

2) Generating guesses for university accounts

3) Checking guesses and protecting accounts

4) Surveying impacted users

Ethical Considerations

- Approved by IRB
- Study design informed by discussions with:
  - IT Leadership
  - Provost's office
  - Communications team
  - Alumni association
- Minimizing access to password history database
- Password resets to protect UChicago users

Results

12,247 correct guesses
based on password reuse

Results

We guessed at least one password for:

Results

We guessed at least one password for:

- **4.5%** of all users in the password history database

Results

We guessed at least one password for:

- **4.5%** of all users in the password history database
- **6.5%** of users for whom we made a guess

Results

We guessed at least one password for:

- **4.5%** of all users in the password history database
- **6.5%** of users for whom we made a guess
- **32.0%** of users with a uchicago.edu email in a data breach

Results

We guessed the current password for

3,618 accounts

Sources of Correct Guesses

71 individual service breaches

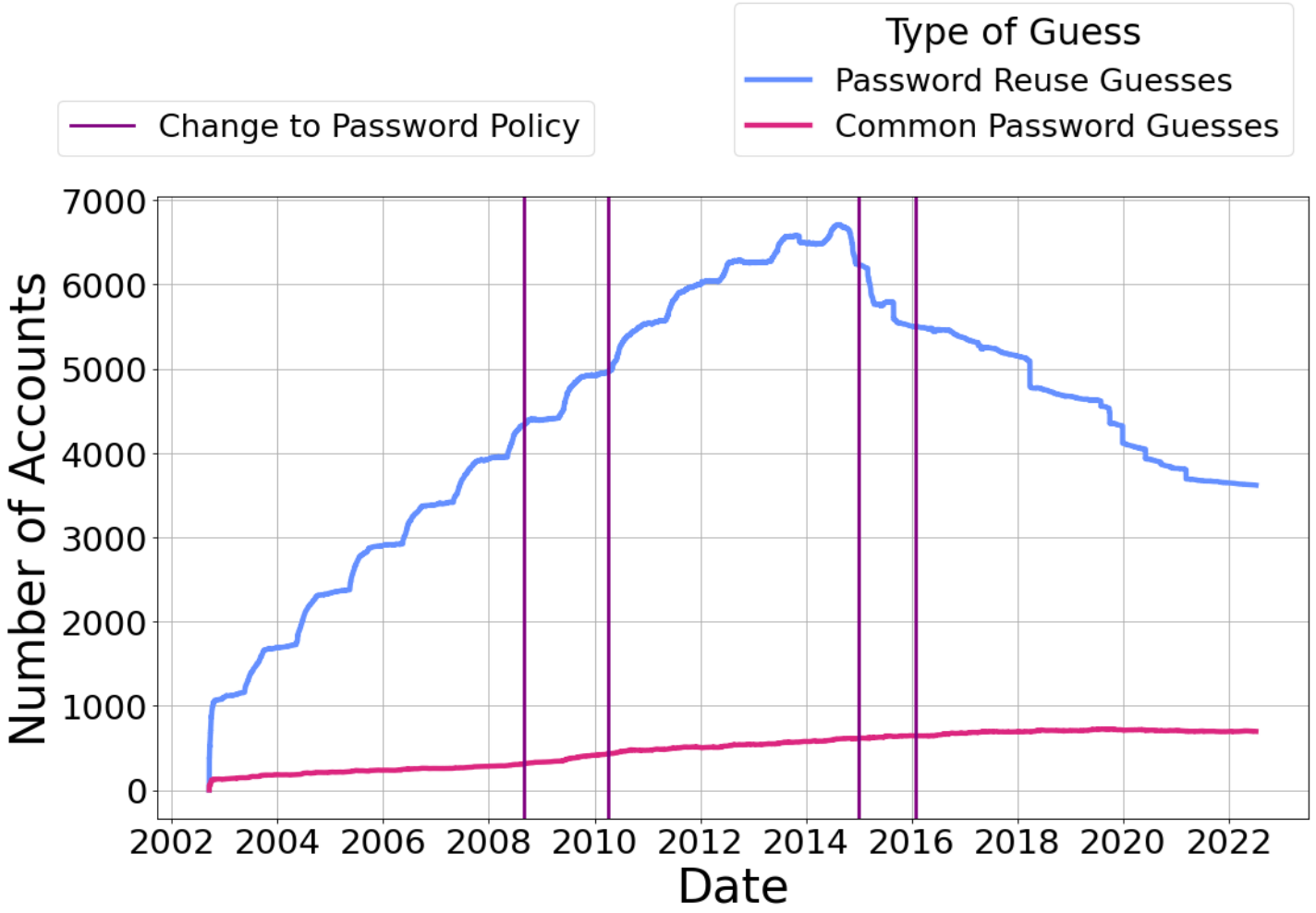
Sources of Correct Guesses

71 individual service breaches
…and all 12 breach compilations

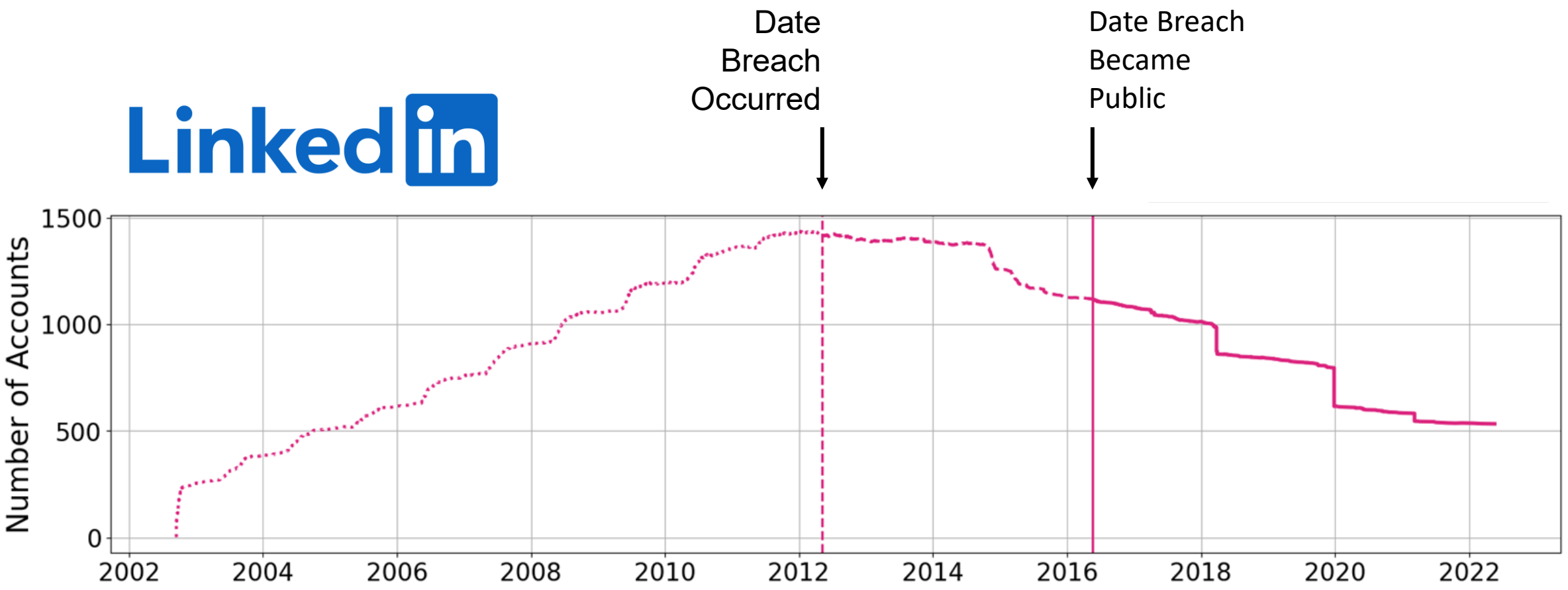
Impact on Specific User Groups

	LinkedIn	Chegg
Students	11.2%	41.4%
Faculty	54.3%	2.2%

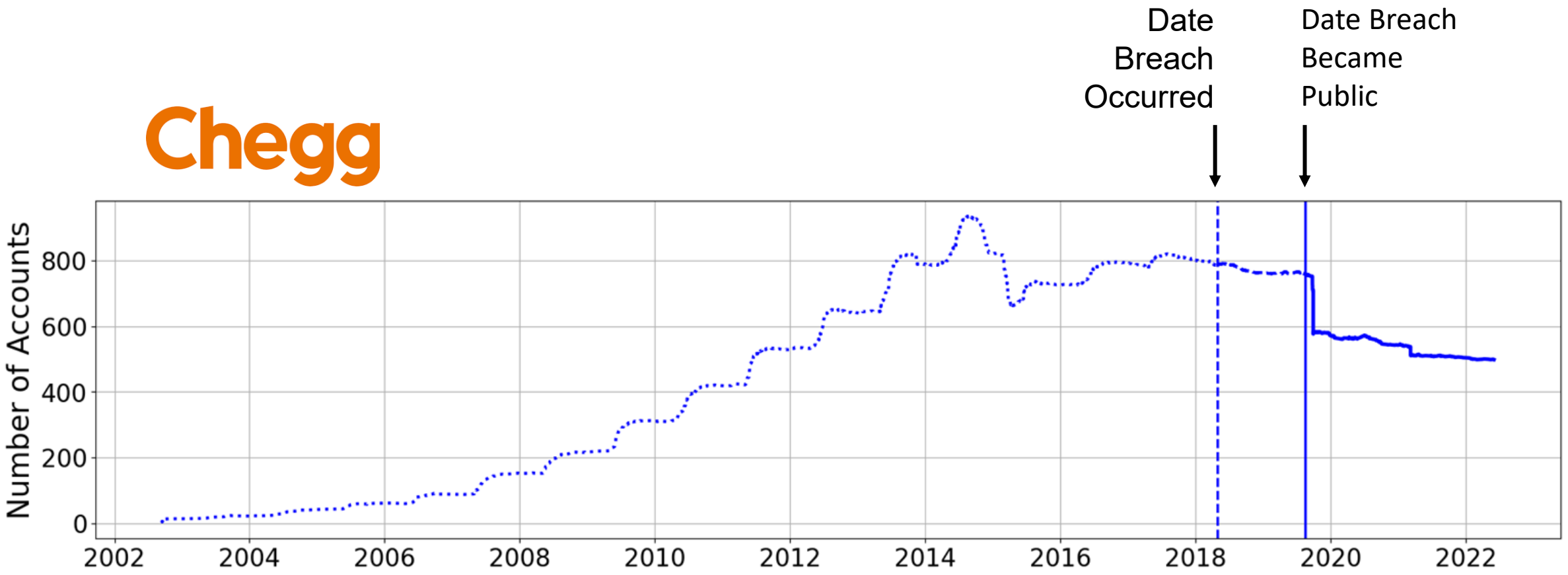
Number of Vulnerable Accounts Over Time



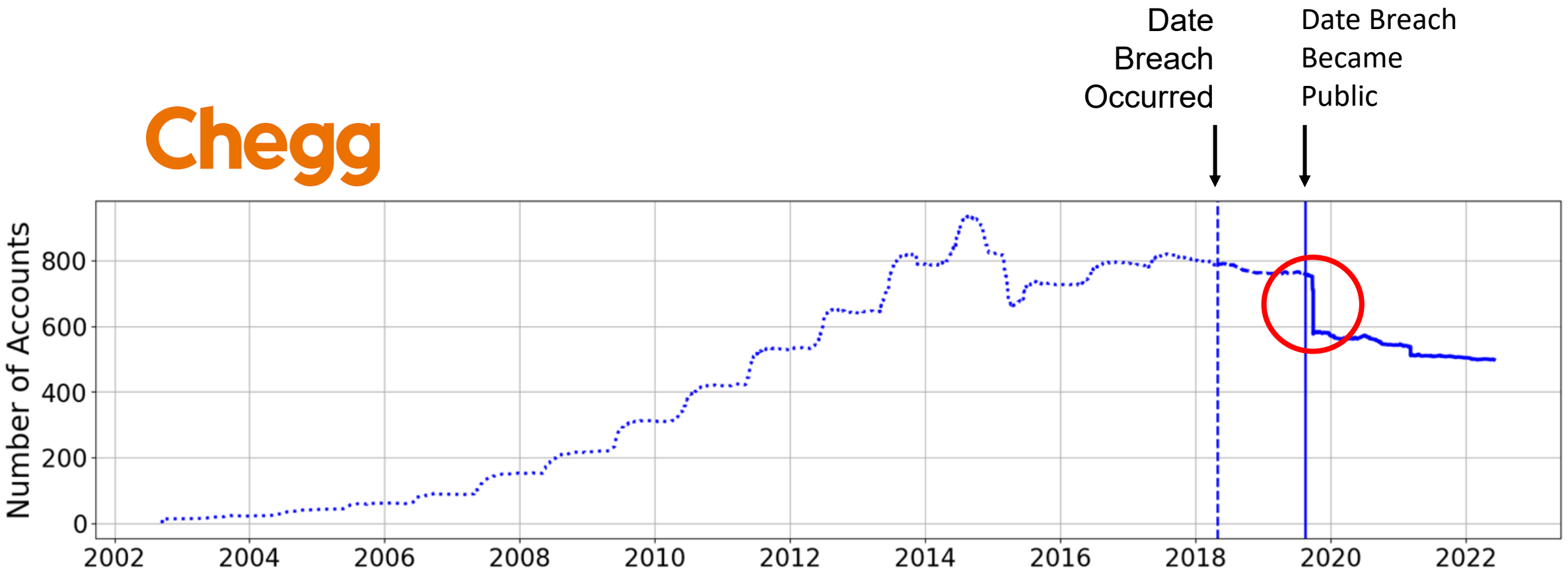
Some Accounts Remained Vulnerable For Years



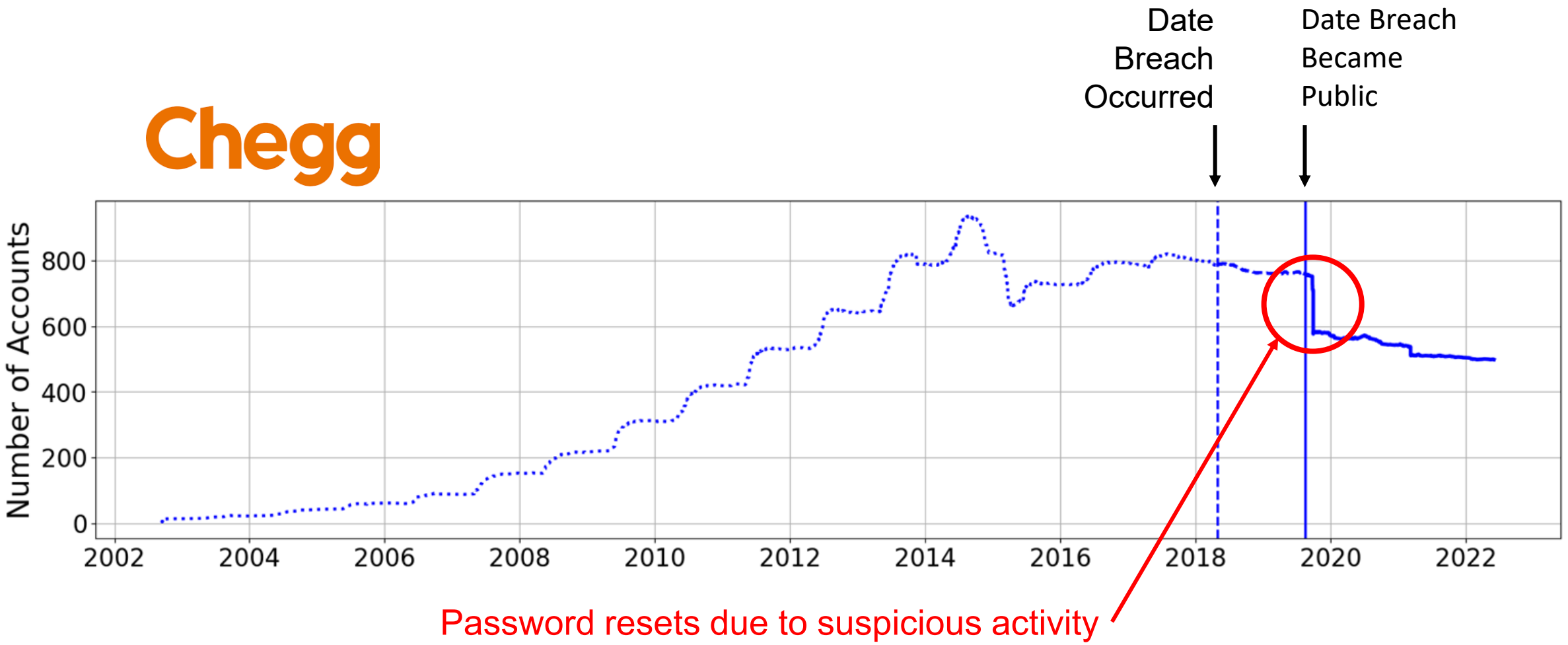
Some Accounts Were Quickly Exploited



Some Accounts Were Quickly Exploited



Some Accounts Were Quickly Exploited

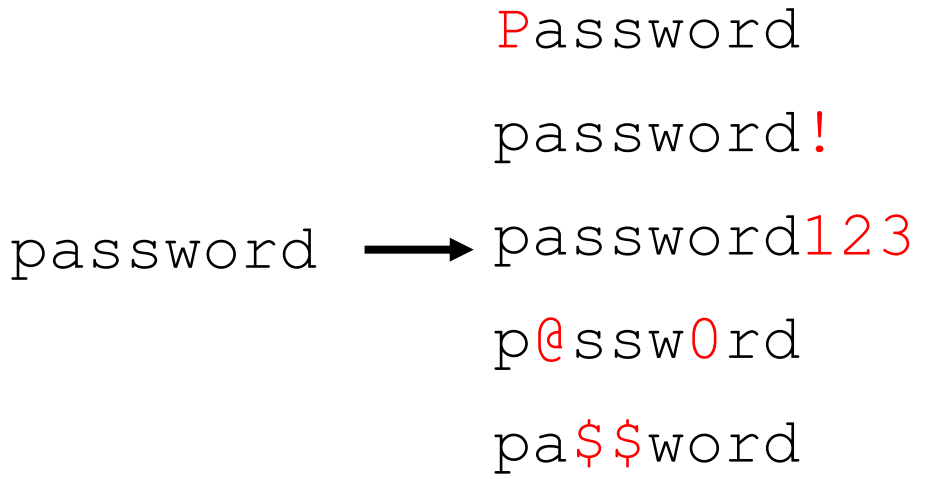


Importance of Cracking Hashes

Plaintext	Hashed
85.3%	14.7%
Sunshine!	5F4DCC3B5AA765D61D8327DEB882CF99
correctbatteryhorsestaple	482C811DA5D5B4BC6D497FFA98491E38
i@mforg3tful!	62099D23A9D9910879D67449D9E084ED
ineedapassword	1C8F93D67A694EE1DE6363D20228DAC8

Importance of Tweaking Guesses

Verbatim Reuse 54.7%



Tweaked Passwords 45.3%

Authentication in Practice: Moving Towards A Passwordless World

The following slides borrow heavily from slides originally made by Maximilian Golla (@CISPA, Germany).

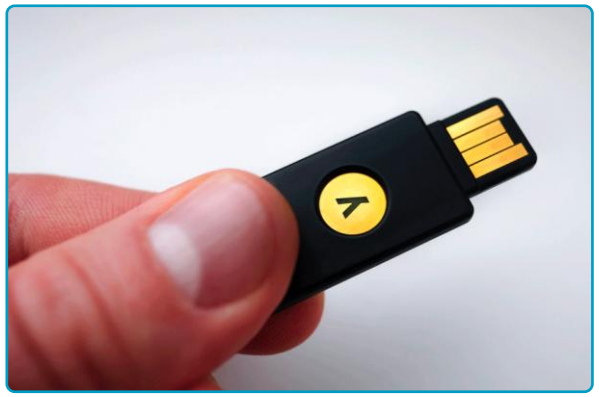
The FIDO Alliance

July 2012 – Founded by **PayPal**, **Lenovo**, **Nok Nok Labs**, **Validity Sensors**, **Infineon**, and **Agnitio.**
April 2013 – **Google**, **Yubico**, and **NXP** joined, in December **Microsoft** followed.
January 2020 – **Apple** joined, passkeys are born.

2012 — 2015 — 2018 — 2021 — 2024

A Stack of Standards

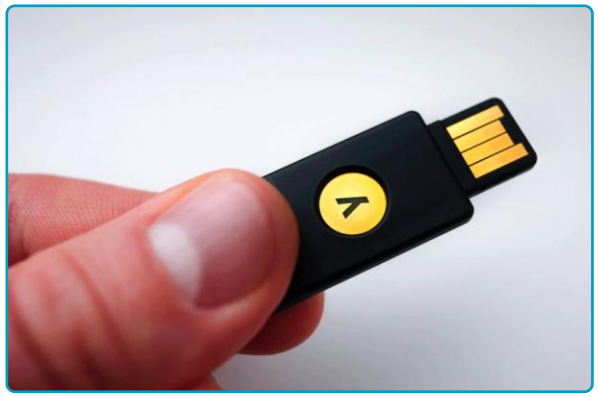
U2F



Security Keys
(2014)

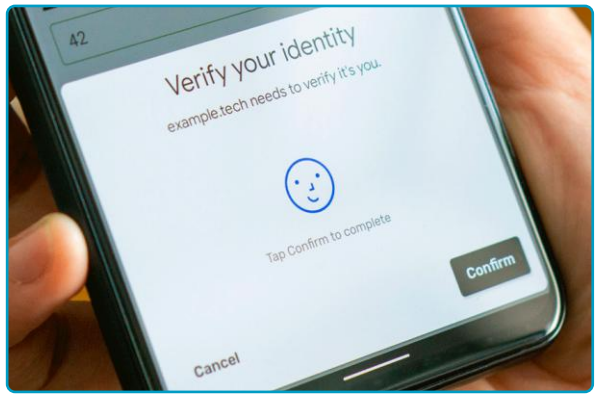
A Stack of Standards

U2F



Security Keys
(2014)

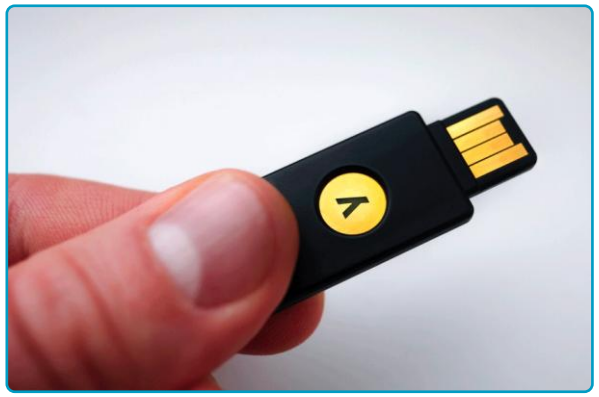
FIDO2



Device-Bound Credentials
(2019)

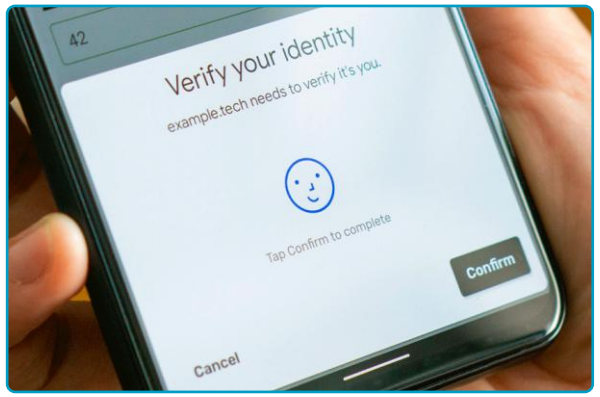
A Stack of Standards

U2F



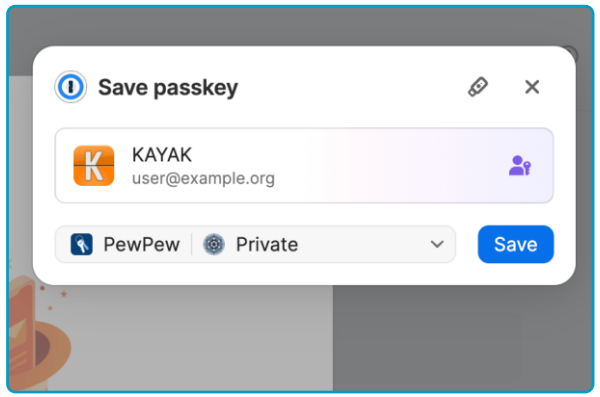
Security Keys
(2014)

FIDO2



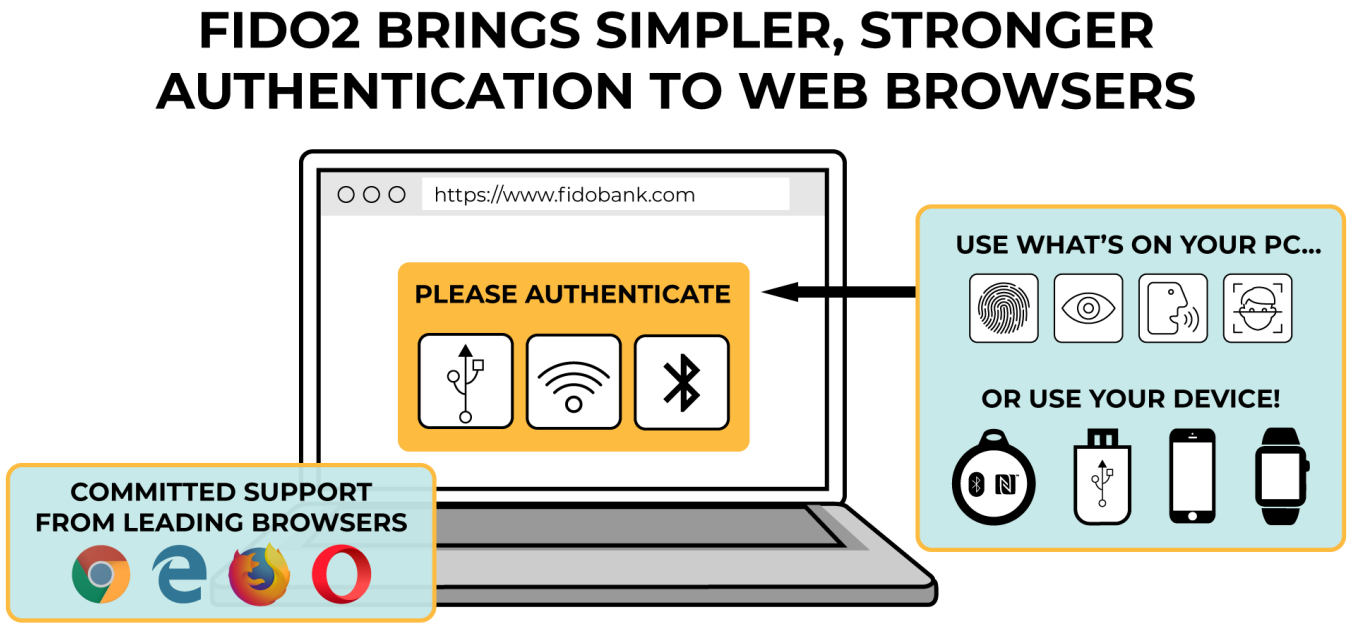
Device-Bound Credentials
(2019)

Passkeys



Multi-Device Credentials
(2022)

Passwordless FIDO2



FIDO AUTHENTICATION: THE NEW GOLD STANDARD

Protects against phishing, man-in-the-middle and attacks using stolen credentials

Log in with a single gesture – HASSLE FREE!

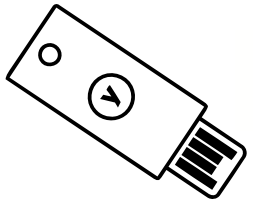
Already supported in market by top online services

FIDO2 Standard

= CTAP2 + WebAuthn

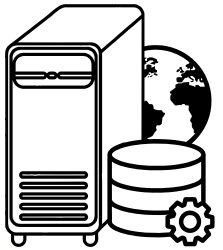
FIDO Protocols

Authenticator
(Security Key)



Client
(Browser)

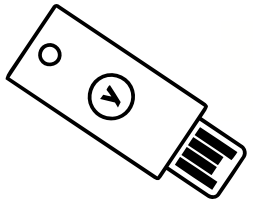
Server
(Relying Party)



Farke et al. **Exploring FIDO2 Security Keys in a Small Company**. USENIX SOUPS. 2021.

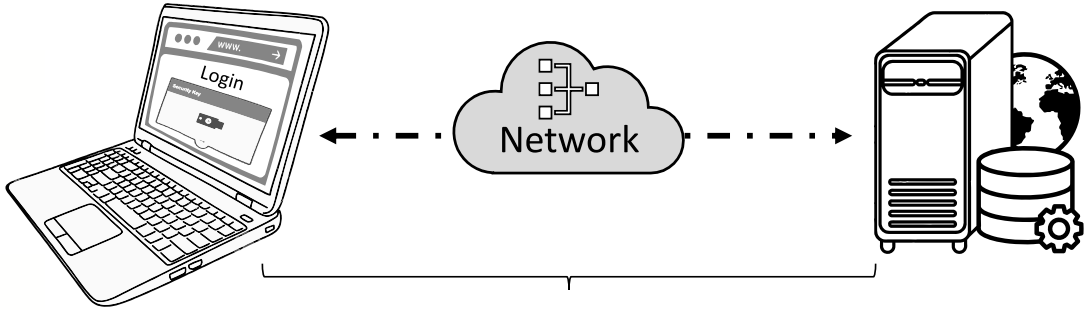
FIDO Protocols

Authenticator
(Security Key)



Client
(Browser)

Server
(Relying Party)

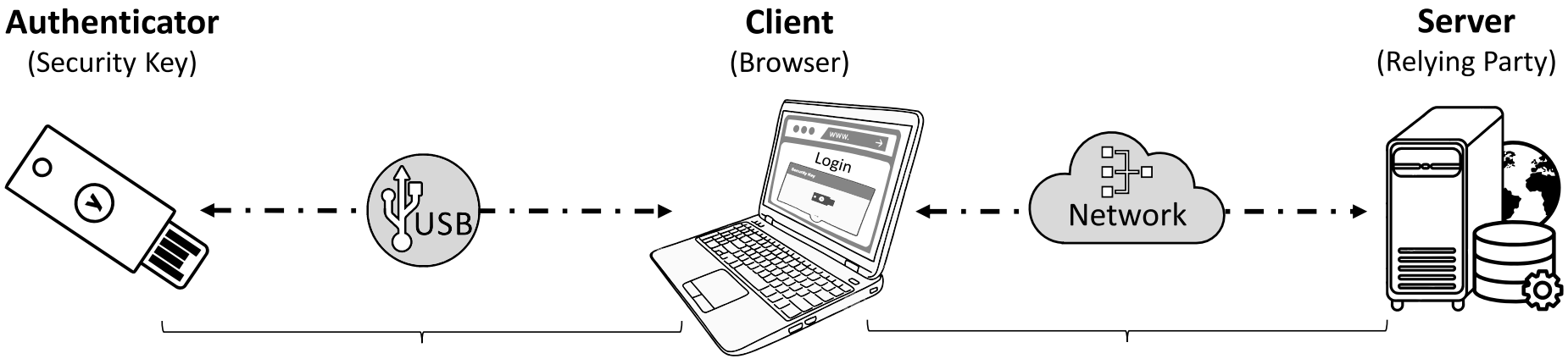


Web Authentication
(WebAuthn)

W3C Standardized JavaScript API

Farke et al. **Exploring FIDO2 Security Keys in a Small Company**. USENIX SOUPS. 2021.

FIDO Protocols



Client to Authenticator Protocol (CTAP)

Built-In, USB, NFC, Bluetooth

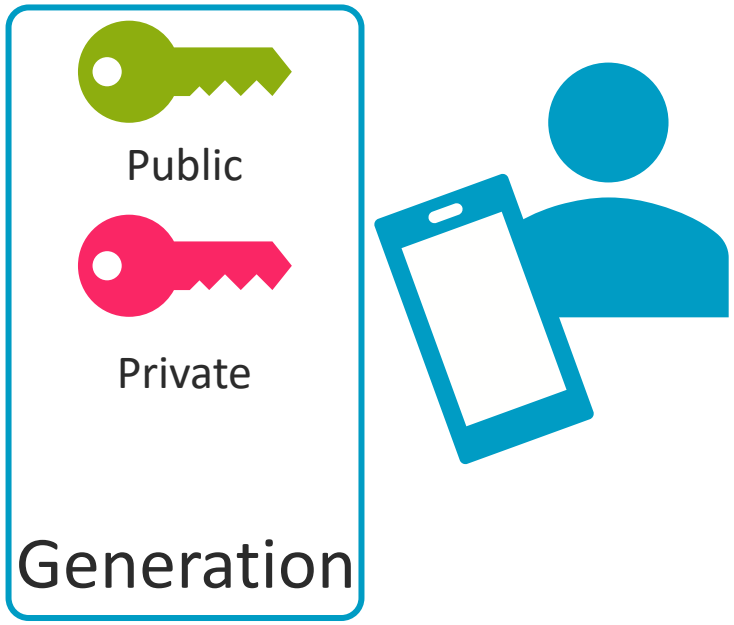
Web Authentication (WebAuthn)

W3C Standardized JavaScript API

Farke et al. **Exploring FIDO2 Security Keys in a Small Company**. USENIX SOUPS. 2021.

Registration

Simplified



Registration

Simplified

Authentication

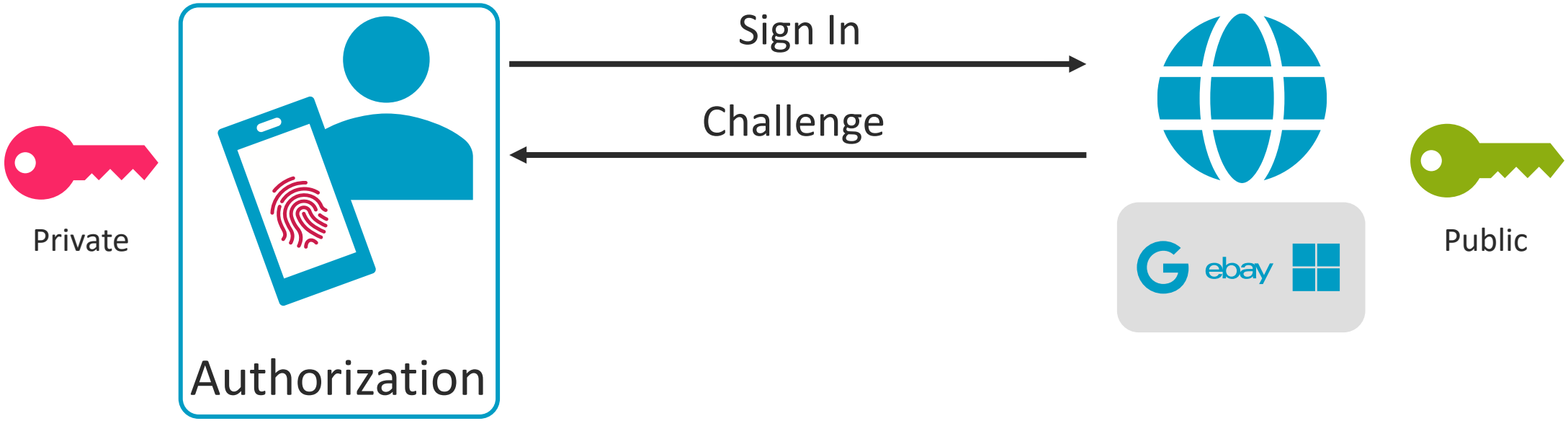
Simplified

Authentication

Simplified

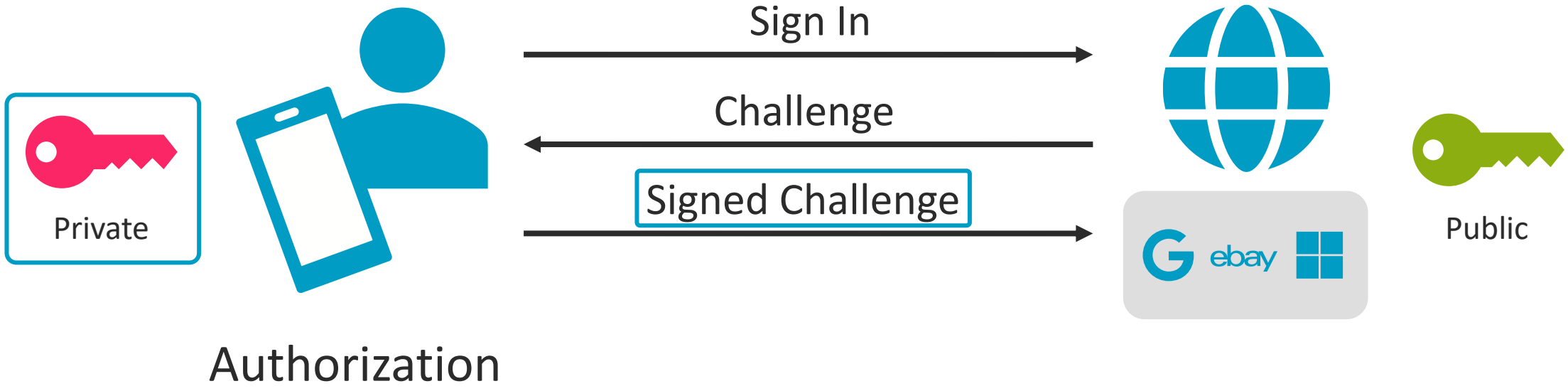
Authentication

Simplified



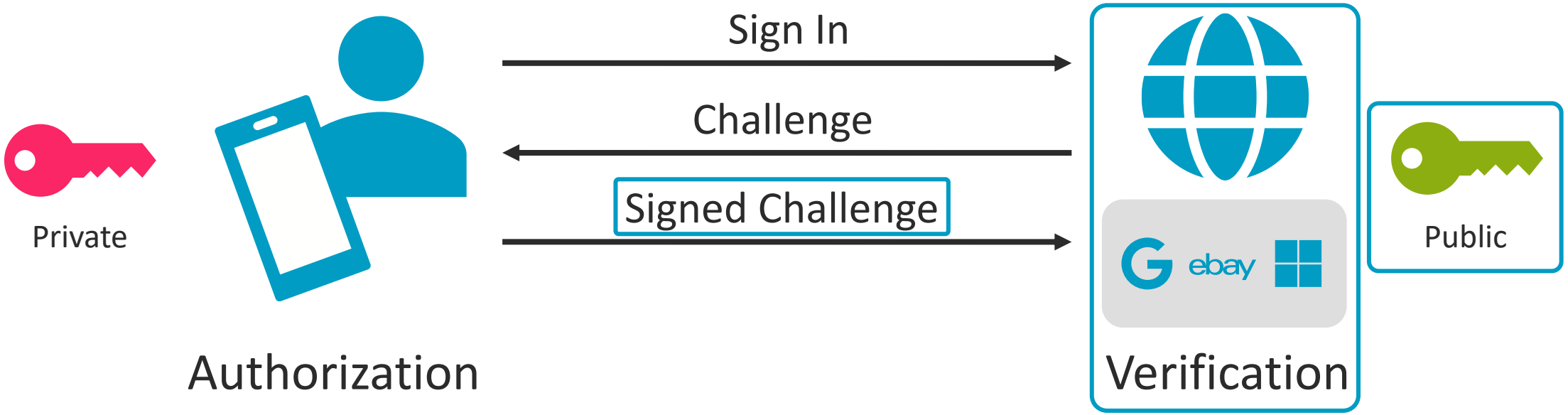
Authentication

Simplified



Authentication

Simplified



FIDO Advantages

fido™ ALLIANCE

- Fast
- Nothing to remember
- Phishing resistant
- Cryptographic keys
- No secrets on server

28 Interviews with Three Types of Stakeholders

FIDO Experts

6x

IAM Vendors

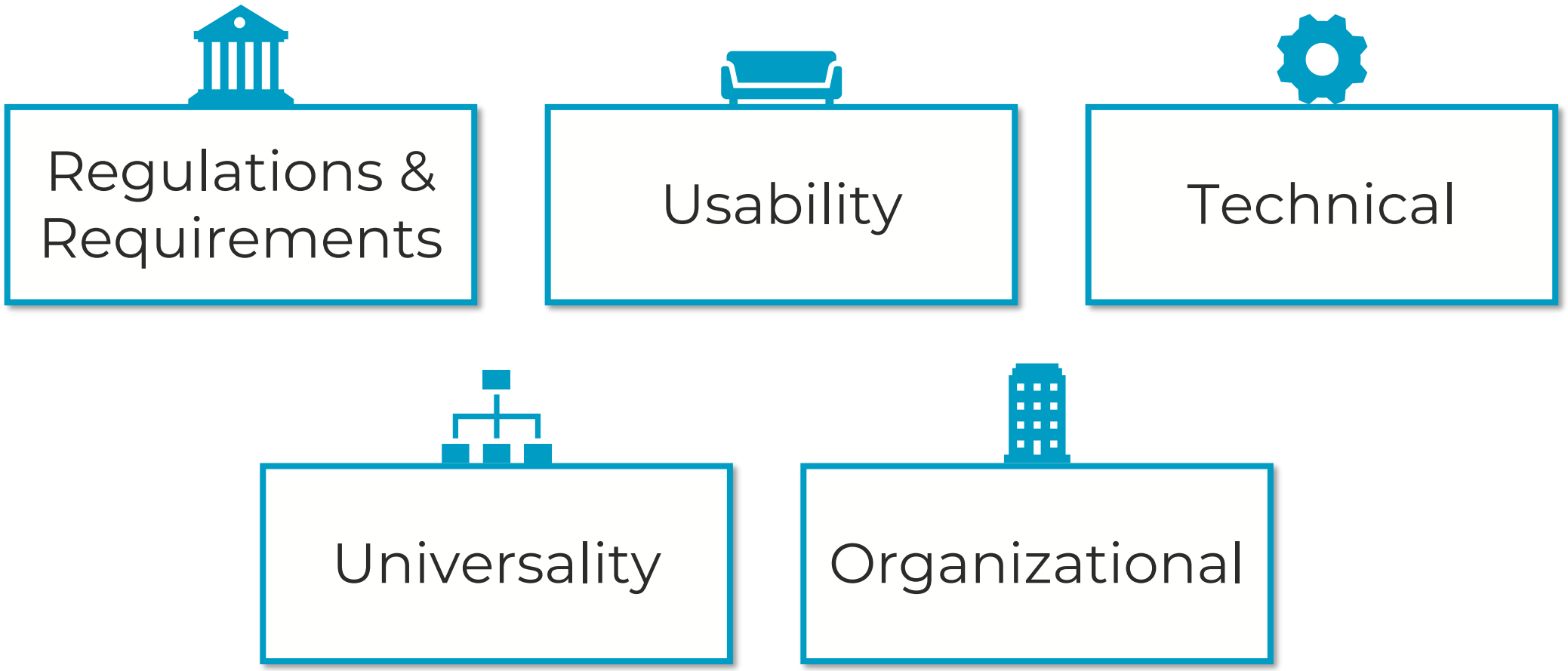
4x

Organization

18x

Lassak et al. Why Aren't We Using Passkeys? **Obstacles Companies Face Deploying FIDO2 Passwordless Authentication**. USENIX Security. 2024.

Identified Obstacles



Lassak et al. Why Aren't We Using Passkeys? **Obstacles Companies Face Deploying FIDO2 Passwordless Authentication**. USENIX Security. 2024.