When Research Comes Full Circle: A Missed Opportunity and What to Learn From It

Michael Reiter

James B. Duke Distinguished Professor

Computer Science and Electrical & Computer Engineering, Duke University

and

Researcher, Chainlink Labs

Duke

Who Is This Person? Dr. Li Gong

- Best Paper Award, IEEE S&P 1989

- IEEE CSF PC Chair, 1994-5
- ACM CCS PC Co-Chair, 1996-7
- ACM CCS General Chair, 1998
- IEEE S&P PC Co-Chair, 1998-9
- IEEE S&P General Chair, 2001

Duke

Who Is This Person? Dr. Li Gong

- Co-winner of the 1994 IEEE ComSoc Leonard G. Abraham Prize

648 IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 11, NO. 5, JUNE 1993

Protecting Poorly Chosen Secrets from Guessing Attacks

Li Gong, Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer, *Fellow, IEEE*

I'll return to Li later …

Duke

Passwords are Dead (2004)

News > Privacy

Gates predicts death of the password

Traditional password-based security is headed for extinction, says Microsoft's chairman, because it cannot "meet the challenge" of keeping critical information secure.

Munir Kotadia

Feb. 25, 2004 1:27 p.m. PT

3 min read

Long Live Passwords!

A Research Agenda Acknowledging the Persistence of Passwords

Cormac Herley | Microsoft Research
Paul van Oorschot | Carleton University

> **The incorrect assumption that passwords are dead has been harmful, discouraging research on how to improve the lot of close to 2 billion people who use them. Every effort should be made to correct this.**

IEEE Security & Privacy, Jan/Feb 2012

Here's Why [Insert Thing Here] Is Not a Password Killer

05 NOVEMBER 2018

Troy Hunt

Hi, I'm Troy Hunt, I write this blog, run "Have I Been Pwned" and am a Microsoft Regional Director and MVP who travels the world speaking at events and training technology professionals →

Multifactor Authentication (MFA)?

- Can be very effective where its adoption can be enforced
- But many sites requiring a low-friction user experience will not
- "People significantly preferred passwords over MFA and were willing to pay about a $3 premium (on a $60 smart speaker) to have the password compared to MFA."

Prof. Pardis Emami-Naeini, based on Emami-Naeini et al., "Are Consumers Willing to Pay for Security and Privacy of IoT Devices?", USENIX Security 2023.

2020:

Only 9.27% of all npm developers use 2FA

Two-factor authentication not widely adopted on npm, the de-facto JavaScript package manager, and the largest package repository on the internet.

Written by **Catalin Cimpanu,** Contributor on Jan. 6, 2020

2021:

Twitter reveals surprisingly low two-factor auth (2FA) adoption rate

By **Sergiu Gatlan**

July 23, 2021 08:06 AM 6

2022:

Catalin Cimpanu | February 4, 2022

Microsoft says MFA adoption remains low, only 22% among enterprise customers

Duke

PassKeys!

Duke

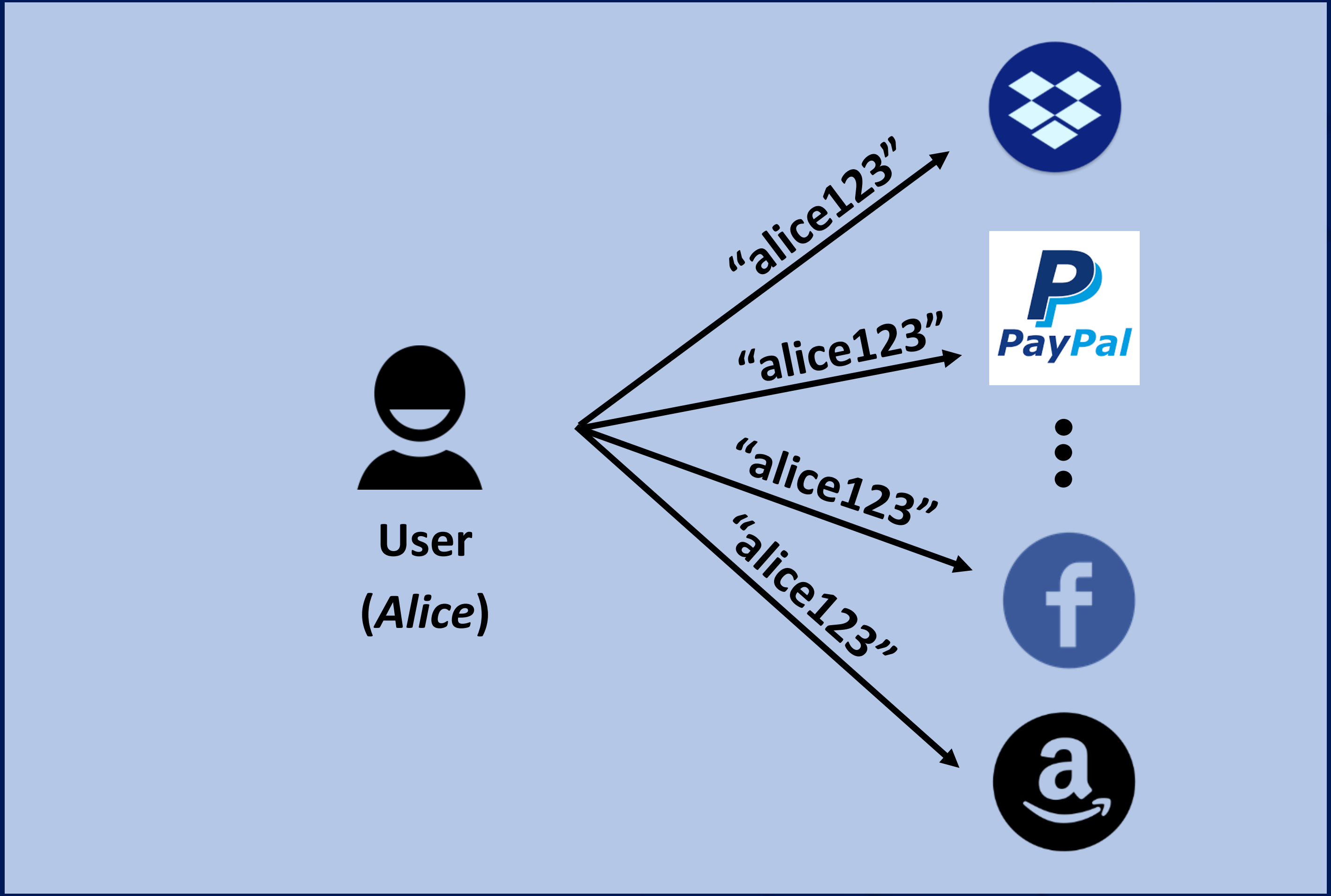
Credential Abuse across Sites

Password reuse

Duke

Credential Abuse across Sites

Password reuse



Duke

Credential Abuse across Sites

Password reuse

The reuse of passwords is the No. 1 cause of harm on the internet.

--- Alex Stamos (former CSO, Facebook) [2016]

99% of compromised user accounts come from password reuse.

--- Patrick Heim (Head of Trust & Security, Dropbox) [2016]

Credential stuffing is enormously effective due to the password reuse problem.

--- Troy Hunt (Regional Director, Microsoft) [2017]

Duke

Credential Abuse across Sites

Password reuse

Database breaches

Duke

Credential Abuse across Sites

Password reuse

Database breaches

The 15 Biggest Data Breaches of the 21st Century – CSO Online (Jan 24, 2021)

Time ↓

Year	Site	Users (M)	Usernames	Passwords	Email addrs	Other
2008	Heartland Payment	134	○	○	○	●
2012	LinkedIn	165	○	◑	●	○
2013	Adobe	153	●	●	○	●
	MySpace	360	●	◑	●	●
	Yahoo!	3000	○	●	●	●
2014	eBay	145	○	◑	○	●
	Marriott	500	○	○	○	●
2015	NetEase	235	○	●	●	○
2016	Adult Friend Finder	412	○	●	●	●
2017	Equifax	150	○	○	○	●
2018	Dubsmash	162	●	◑	●	○
	My Fitness Pal	150	●	◑	●	○
2019	Canva	61	●	◑	●	●
	Zynga	218	●	◑	●	●
2020	Sina Weibo	538	●	○	○	●

Credential Abuse across Sites

Password reuse

Database breaches

The 15 Biggest Data Breaches of the 21st Century – CSO Online (Jan 24, 2021)

Time ↓

Year	Site	Users (M)	Usernames	Passwords	Email addrs	Other
2008	Heartland Payment	134	○	○	○	●
2012	LinkedIn	165	○	◑	●	○
2013	Adobe	153	●	●	○	●
	MySpace	360	●	◑	●	●
	Yahoo!	3000	○	●	●	●
2014	eBay	145	○	◑	○	●
	Marriott	500	○	○	○	●
2015	NetEase	235	○	●	●	○
2016	Adult Friend Finder	412	○	●	●	●
2017	Equifax	150	○	○	○	●
2018	Dubsmash	162	●	◑	●	○
	My Fitness Pal	150	●	◑	●	○
2019	Canva	61	●	◑	●	●
	Zynga	218	●	◑	●	●
2020	Sina Weibo	538	●	○	○	●

Duke

Credential Abuse across Sites

Password reuse

Database breaches

The 15 Biggest Data Breaches of the 21st Century – CSO Online (Jan 24, 2021)

Time

Year	Site	Users (M)	Usernames	Passwords	Email addrs	Other
2008	Heartland Payment	134	○	○	○	●
2012	LinkedIn	165	○	◑	●	○
2013	Adobe	153	●	●	○	●
	MySpace	360	●	◑	●	●
	Yahoo!	3000	○	●	●	●
2014	eBay	145	○	◑	○	●
	Marriott	500	○	○	○	●
2015	NetEase	235	○	●	●	○
2016	Adult Friend Finder	412	○	●	●	●
2017	Equifax	150	○	○	○	●
2018	Dubsmash	162	●	◑	●	○
	My Fitness Pal	150	●	◑	●	○
2019	Canva	61	●	◑	●	●
	Zynga	218	●	◑	●	●
2020	Sina Weibo	538	●	○	○	●

Credential Abuse across Sites

Password reuse

Database breaches

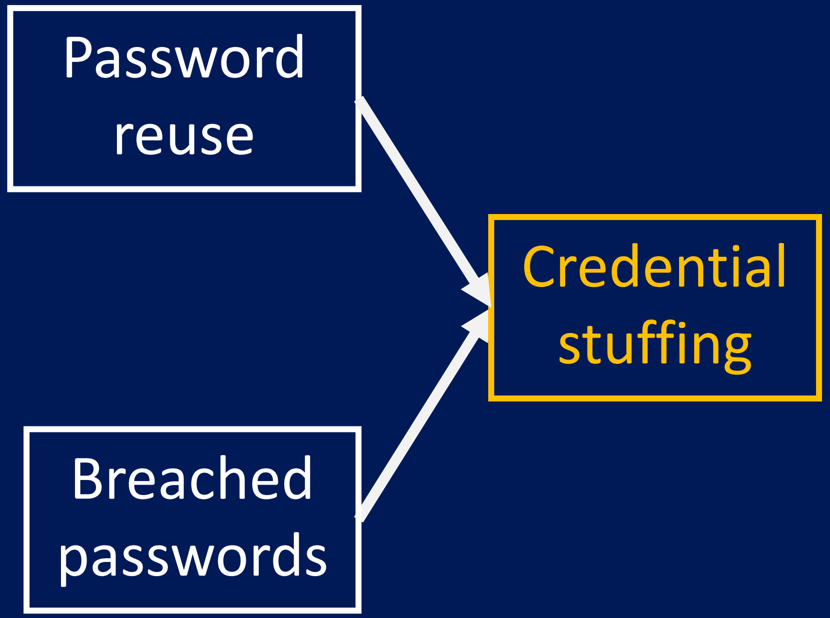
Among 1665 database breaches identified between Nov. 2018 and Oct. 2019, 60% leaked credentials.

--- Verizon [2020]

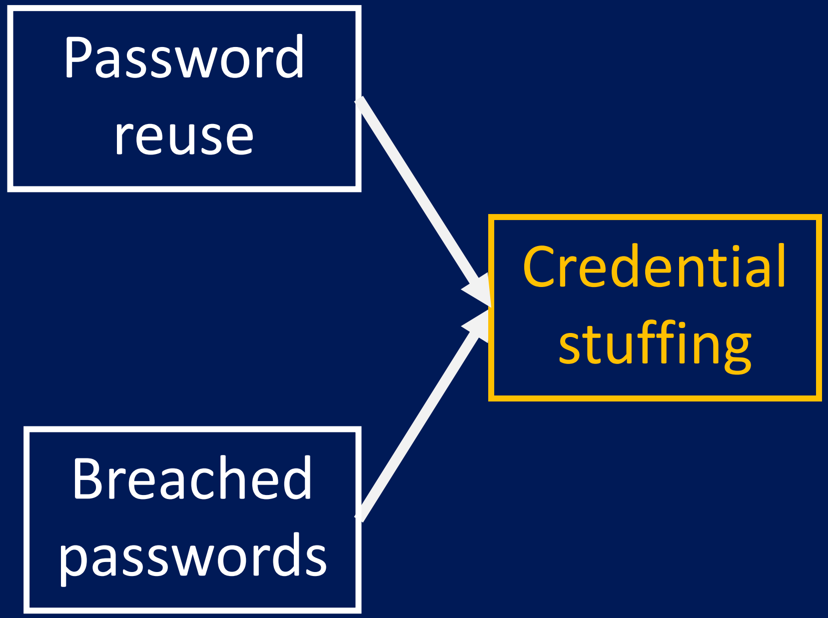
The estimated average delay between when a breach occurs and when the breach is discovered ranges from 7 to 15 months.

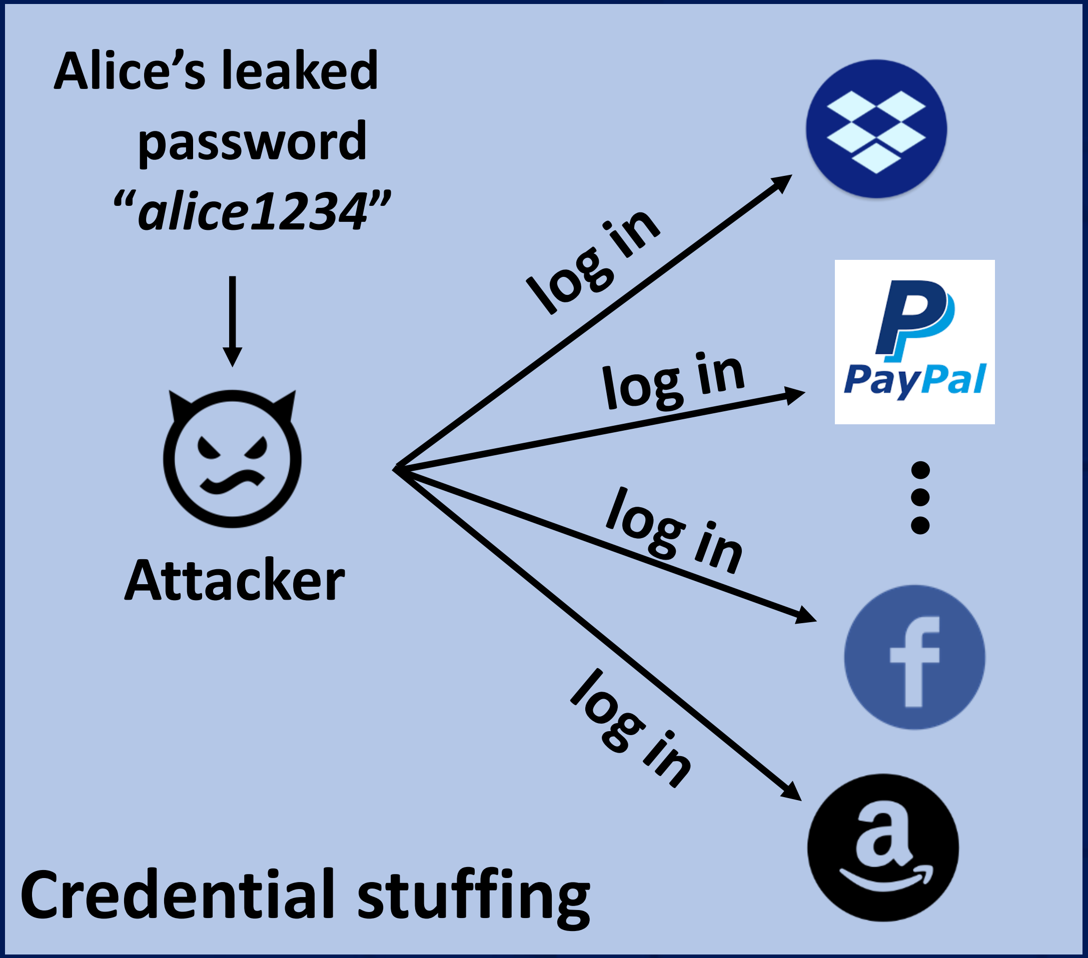
--- IBM [2020] and Shape Security [2018]

Credential Abuse across Sites

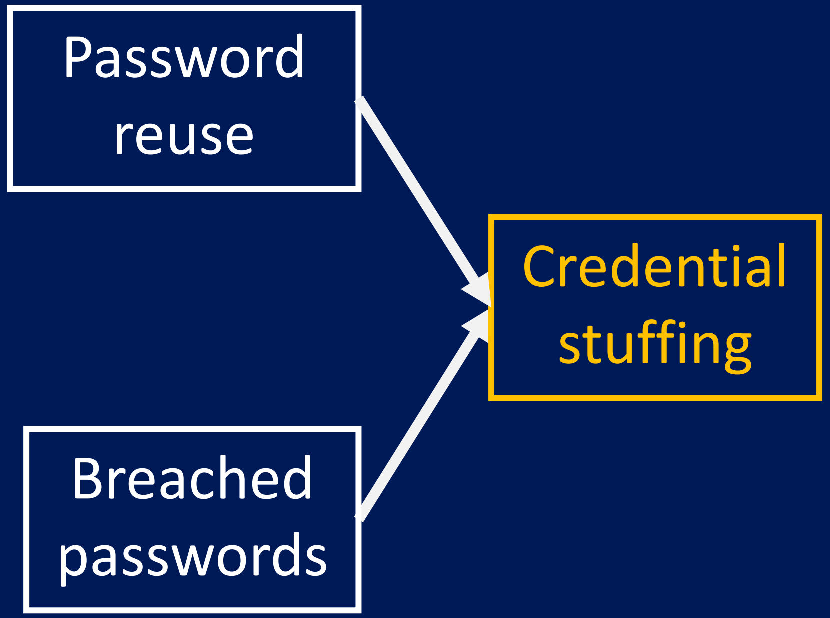


Credential Abuse across Sites





Credential Abuse across Sites



Akamai observed 193 billion credential stuffing attempts in 2020 alone.

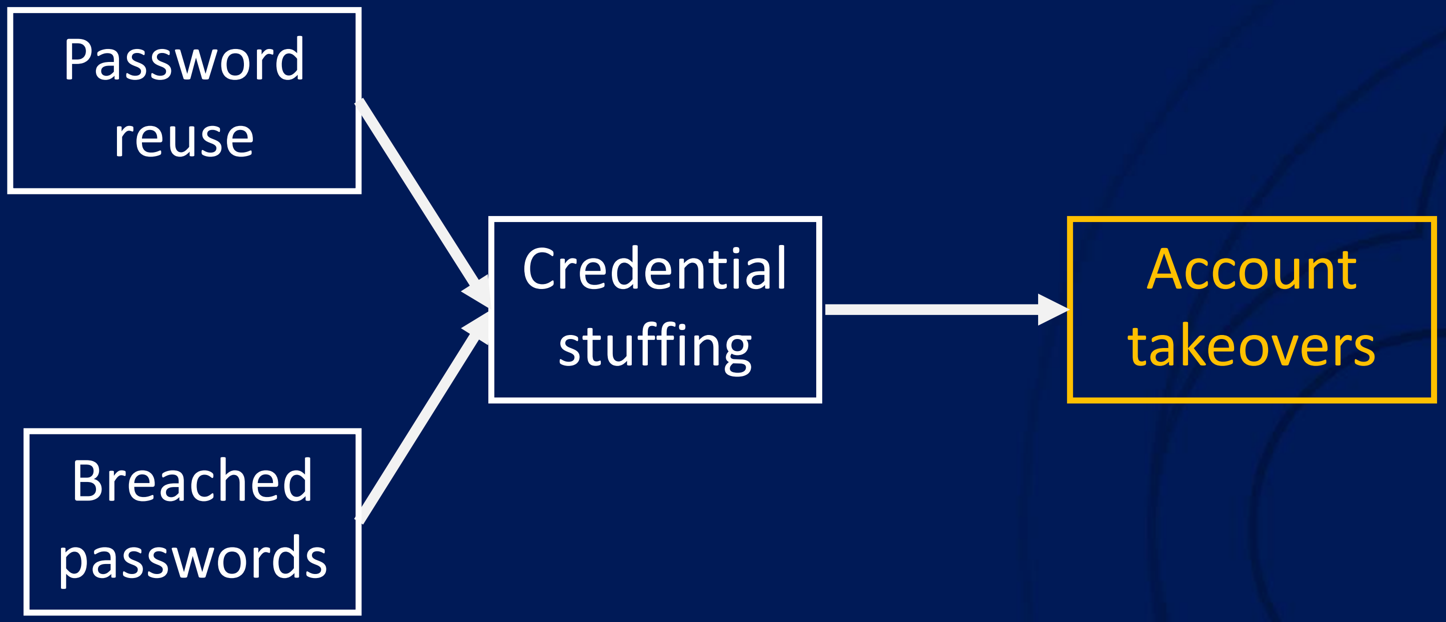
--- Akamai [2021]

Credential stuffing imposes actual losses estimated at $300M, $400M, $1.7B, and $6B on the hotel, airline, consumer banking, and retail industries, per year.

--- Shape Security [2018]

Duke

Credential Abuse across Sites



Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

Password reuse

An employee from a company ***reused*** *a* ***complicated*** *password across his/her company VPN account and an account at a different website.*

Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

Password reuse

An employee from a company ***reused*** *a* ***complicated*** *password across his/her company VPN account and an account at a different website.*

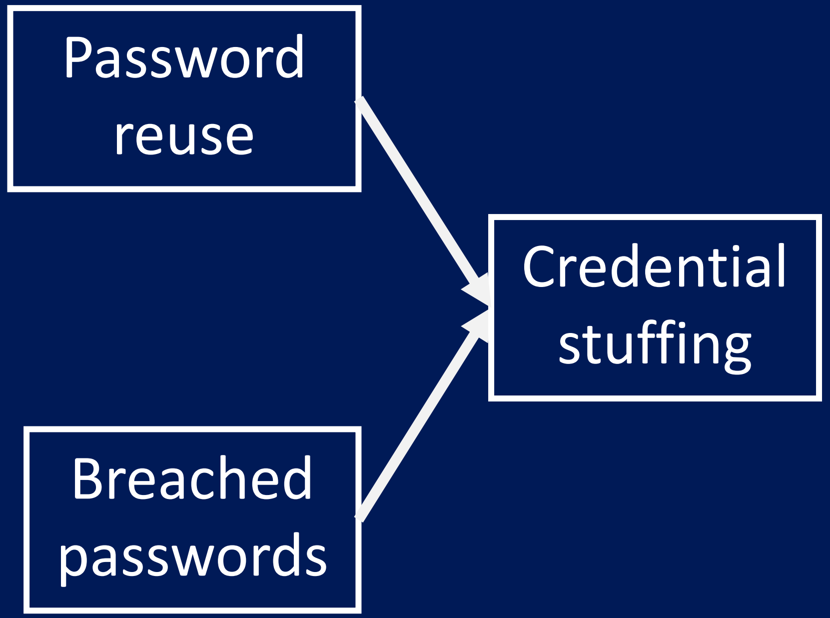
Breached passwords

The password got ***leaked*** *when the other website was* ***breached****.*

Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

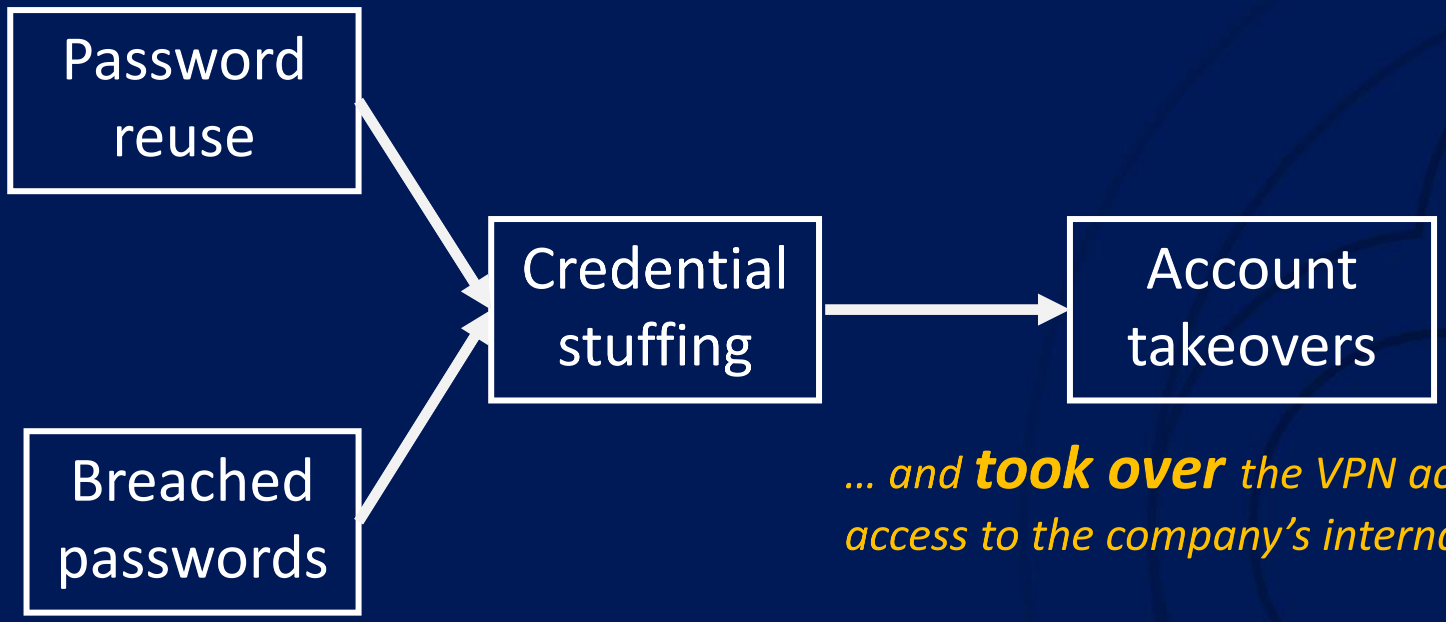


An attacker ***stuffed*** *the leaked password at the employee's VPN account …*

Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)



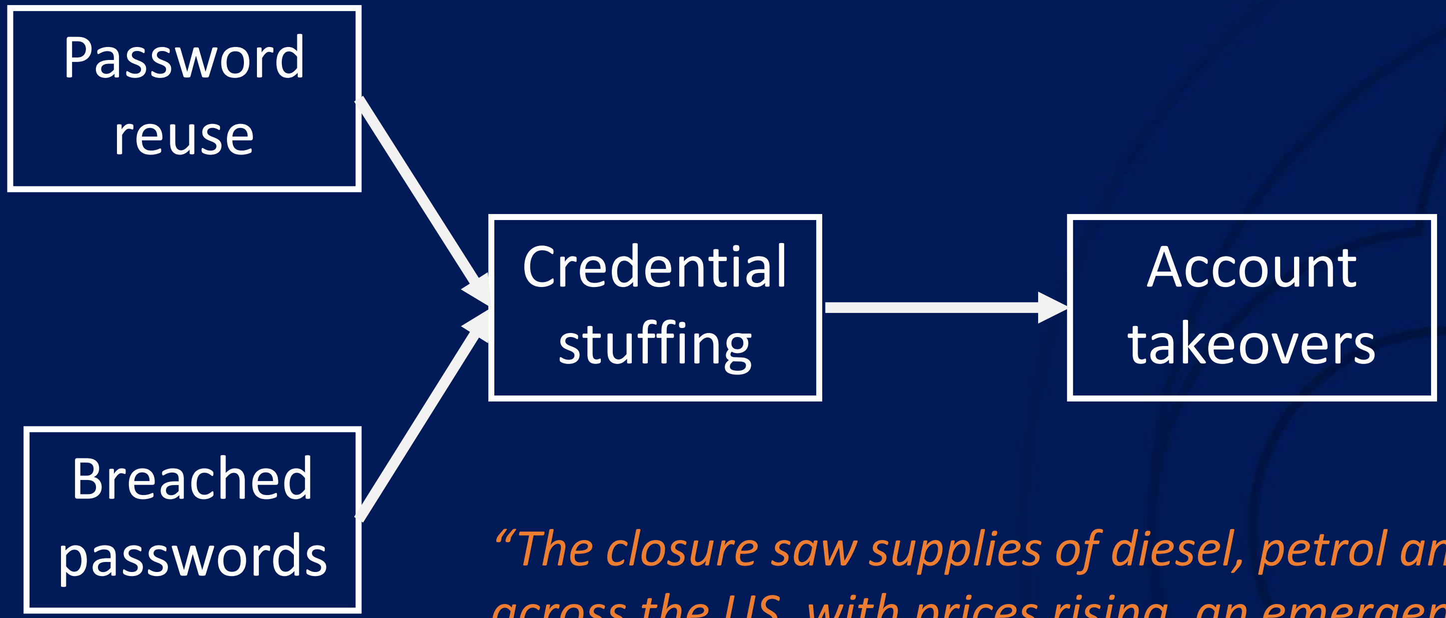
… and ***took over*** *the VPN account, getting access to the company's internal network.*

The attacker disabled part of the company's network and asked for $5M in ransom to recover it.

Duke

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

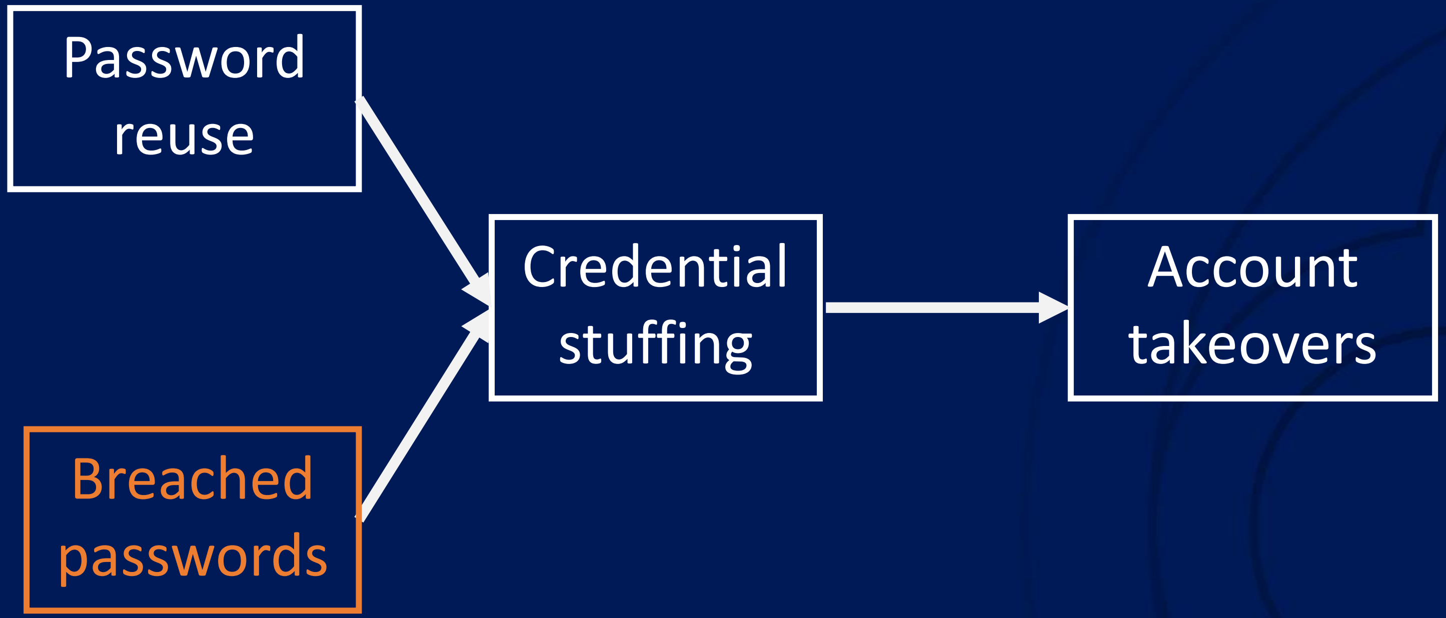


"The closure saw supplies of diesel, petrol and jet fuel tighten across the US, with prices rising, an emergency waiver passed on Monday and a number of states declaring an emergency."

-- BBC

Duke

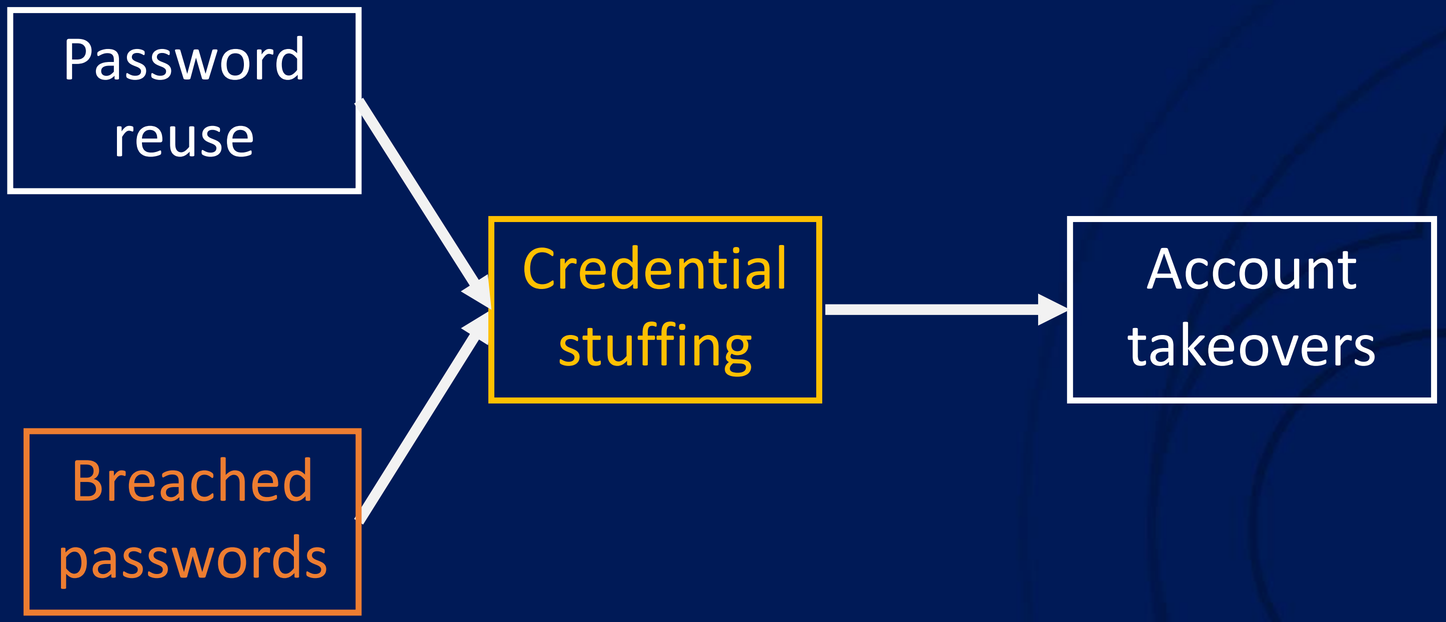
Where to Tackle this Problem?



K. C. Wang and M. K. Reiter, “*Using Amnesia to detect credential database breaches* ”, USENIX Security Symposium, 2021.

K. C. Wang and M. K. Reiter, “*Bernoulli honeywords*”, ISOC Network and Distributed System Security Symposium, 2024.

Where to Tackle this Problem?



K. C. Wang and M. K. Reiter, "*Using Amnesia to detect credential database breaches* ", USENIX Security Symposium, 2021.

K. C. Wang and M. K. Reiter, "*Bernoulli honeywords*", ISOC Network and Distributed System Security Symposium, 2024.

Duke

Honeywords
(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password*:

password1
password2
password3
password4
password5

Decoy passwords (honeywords) are generated based on the real one.

Real user password

Web Server
Credential Database

Duke

** Assuming that attacker can reverse all leaked password (salted) hashes offline,*
Here we ignore the use of hashing (and salting) for simplicity.

Honeywords
(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

???

Web Server
Credential Database

Honeywords

(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

2

The index of the real user password

Web Server
Credential Database

Duke

Honeywords
(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

2

~~???~~

Web Server
Credential Database

Duke

Honeywords
(Juels & Rivest 2013)

Honeychecker

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

Web Server
Credential Database

UID: *alice@gmail.com*

Password index:

2

Use a 2nd secure component to store the index of the real passwords

Duke

Honeywords
(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

???

BREACHED Web Server
Credential Database

Honeychecker

UID: *alice@gmail.com*

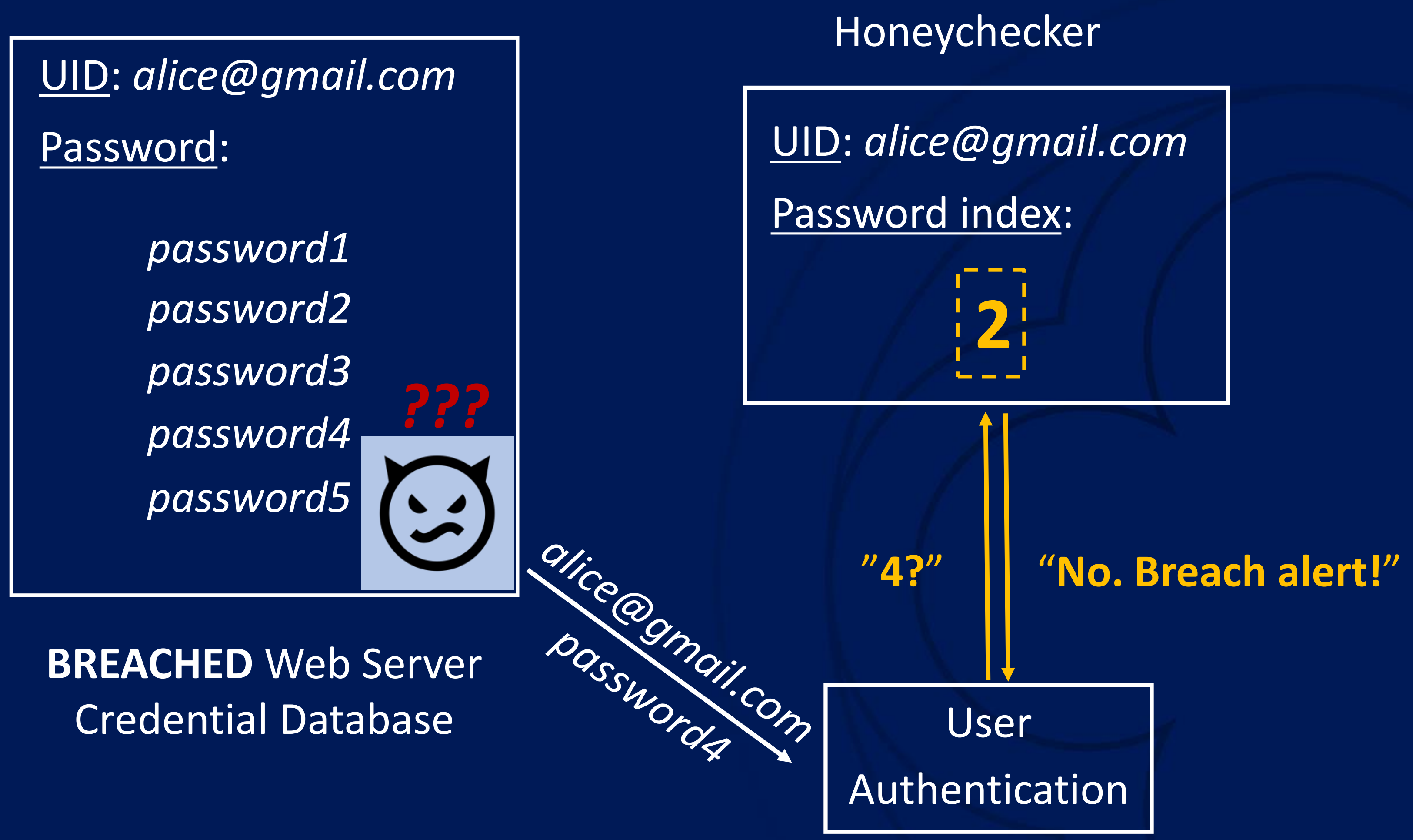
Password index:

2

Duke

Honeywords

(Juels & Rivest 2013)



Duke

Honeywords
(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

???

BREACHED Web Server
Credential Database

Honeychecker

UID: *alice@gmail.com*

Password index:

2

Juels & Rivest's proposal relies on the secret (indices) persistently stored at 2nd SECURE component.

Duke

Honeywords

(Juels & Rivest 2013)

Honeychecker

UID: *alice@gmail.com*

Password:

UID: *alice@gmail.com*

Can we still use honeywords to detect credential database breaches ***without assuming the security of any persistently stored secrets****?*

password4

password5

BREACHED Web Server Credential Database

Juels & Rivest's proposal relies on the secret (indices) persistently stored at 2nd SECURE component.

Duke

Honeywords

(Juels & Rivest 2013)

Honeychecker

UID: *alice@gmail.com*

Password:

UID: *alice@gmail.com*

Can we still use honeywords to detect credential database breaches ***without assuming the security of any persistently stored secrets****?*

YES!!

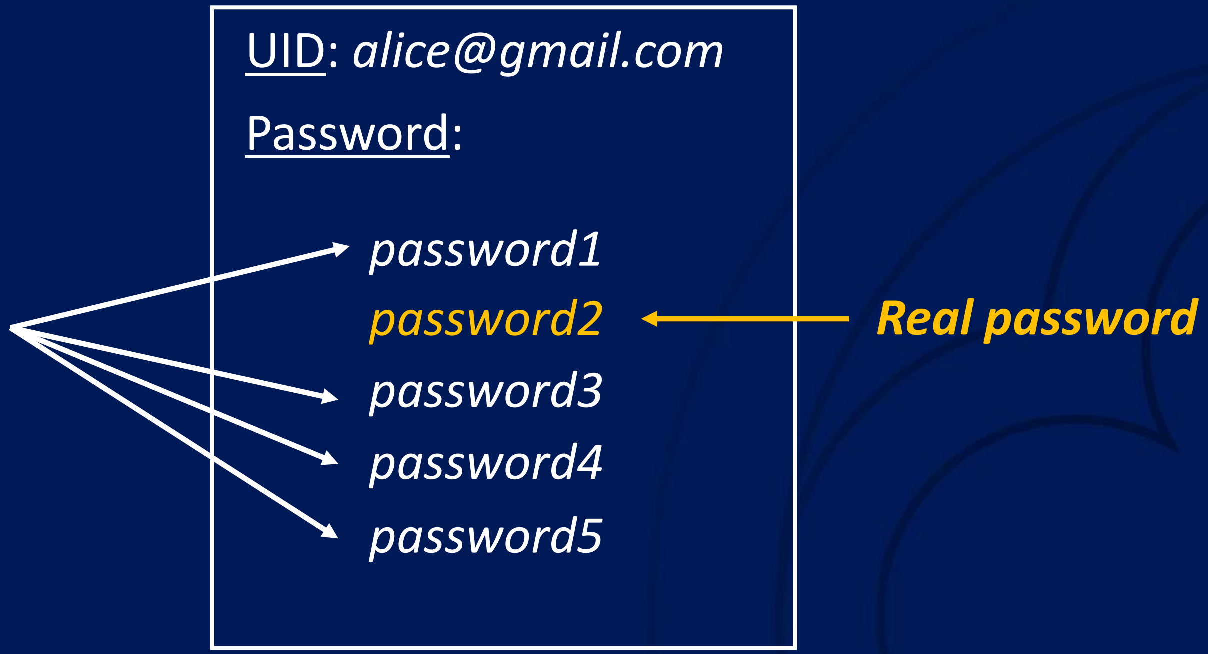
BREACHED Web Server Credential Database

the secret (indices) persistently stored at 2nd SECURE component.

Duke

Amnesia

Decoy passwords (honeywords) are generated based on the real one.



Web Server
Credential Database

Amnesia

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

Web Server
Credential Database

After a successful login:

Duke

Amnesia

UID: *alice@gmail.com*

Password:

password1
password2*
password3
password4
password5

Web Server
Credential Database

After a successful login:

1. ***Mark the last submitted password***

Duke

Amnesia

UID: *alice@gmail.com*

Password:

password1*
*password2**
password3
password4*
password5

Web Server
Credential Database

After a successful login:

1. Mark the last submitted password
2. ***Mark each of other passwords with a preset probability***

Duke

Amnesia

During a login attempt:

If the submitted password is one of the **marked** passwords:

Successful login &
No breach alert.

UID: *alice@gmail.com*

Password:

- *password1**
- *password2**
- *password3*
- *password4**
- *password5*

Web Server
Credential Database

Duke

Amnesia

During a login attempt:

If the submitted password is one of the **unmarked** passwords:

Breach alert!

UID: *alice@gmail.com*

Password:

*password1**
*password2**
password3
*password4**
password5

Web Server
Credential Database

Amnesia

User password: *password2*

Submitted password: **password2**

During login:

*password1**
*password2**
password3
*password4**
password5

After login:

password1
*password2**
*password3**
password4
*password5**

The real password remains marked.

Duke

Amnesia

User password: *password2*

Submitted password: ***password4***

During login:

password1 *
password2 *
password3
password4 *
password5

After login:

password1
password2
password3 *
password4 *
password5 *

Amnesia

User password: *password2*

Submitted password: **password4**

During login	After login
*password1**	*password1*
*password2**	*password2*
password3	*password3**
*password4**	*password4**
password5	*password5**

The submitted honeyword will remain ***marked****.*

Duke

Amnesia

User password: *password2*

Submitted password: ***password4***

During login	After login
*password1**	*password1*
*password2**	*password2*
password3	*password3**
*password4**	*password4**
password5	*password5**

*It's possible that the real user password will be **unmarked**.*

Duke

Amnesia

User password: *password2*

Submitted password: ***password4***

During login	After login
*password1**	*password1*
*password2**	*password2*
password3	*password3**
*password4**	*password4**
password5	*password5**

User's next login with the real password would trigger a breach detection.

It's possible that the real user password will be **unmarked**.

Stuffing Honeywords to Avoid Detection

alice@gmail.com:

password1
password2
password3
password4

Site A

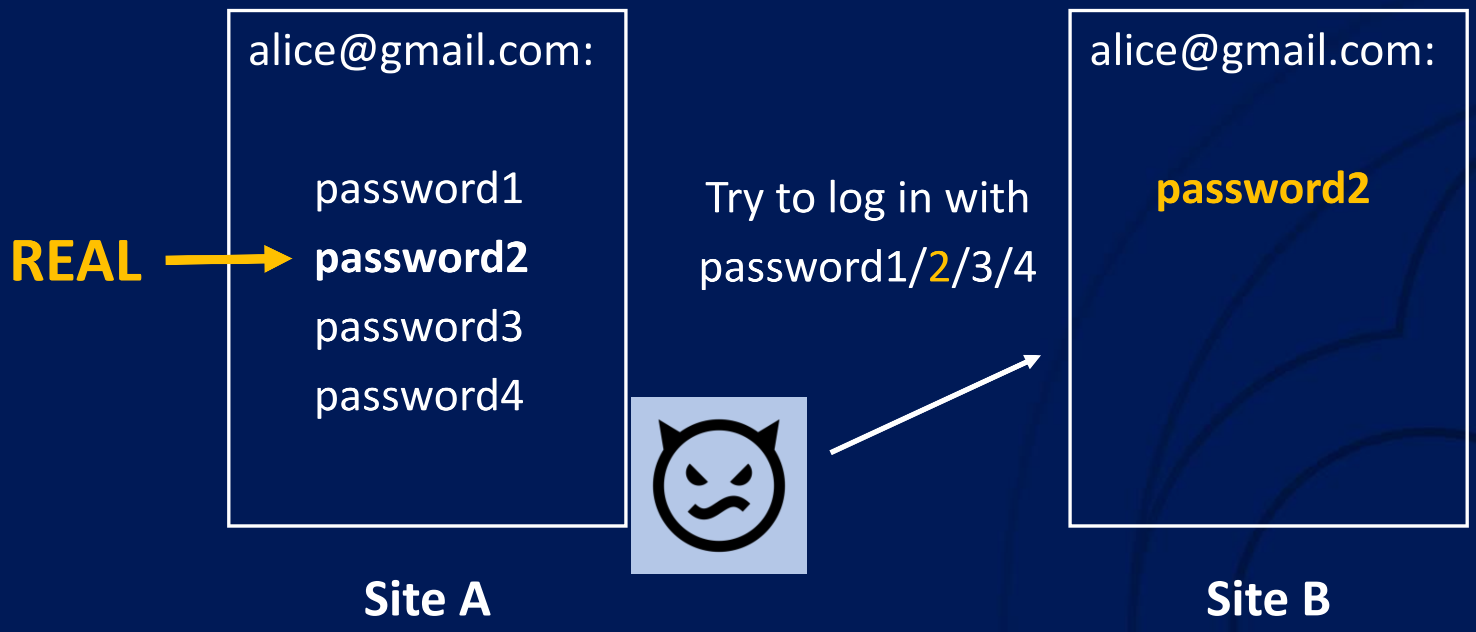
alice@gmail.com:

password2

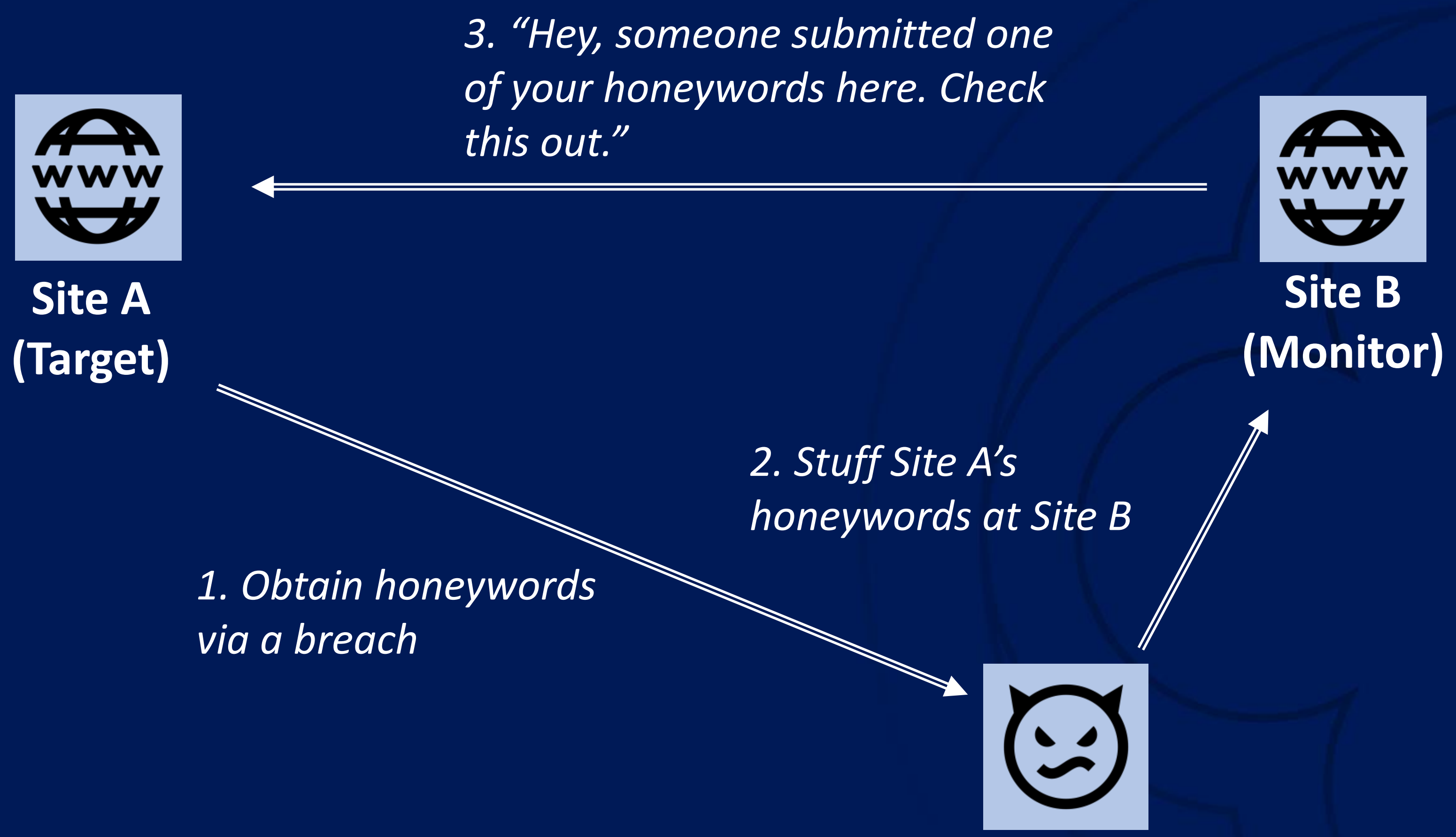
Site B

Duke

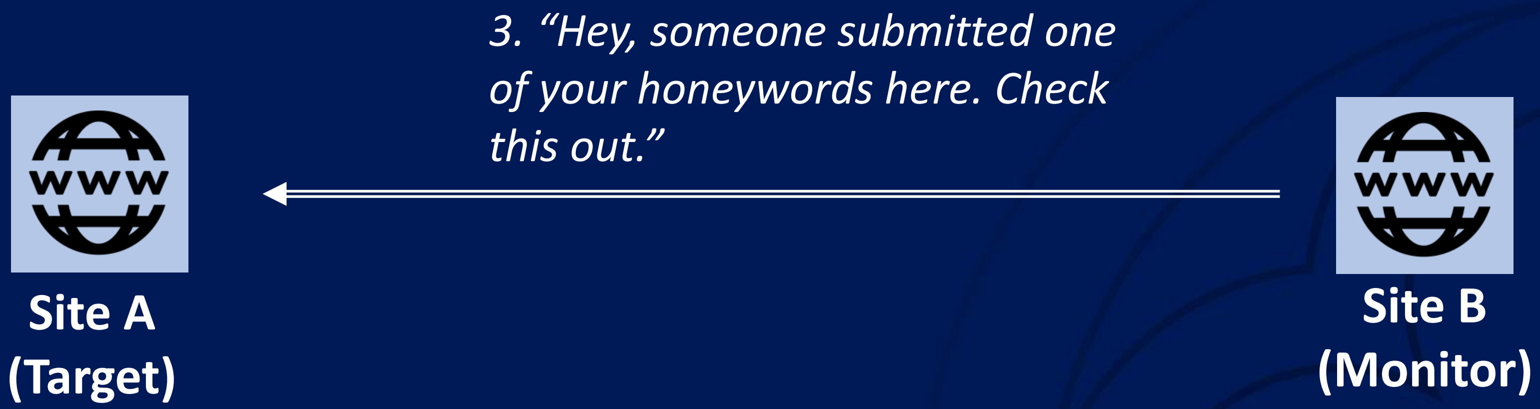
Stuffing Honeywords to Avoid Detection



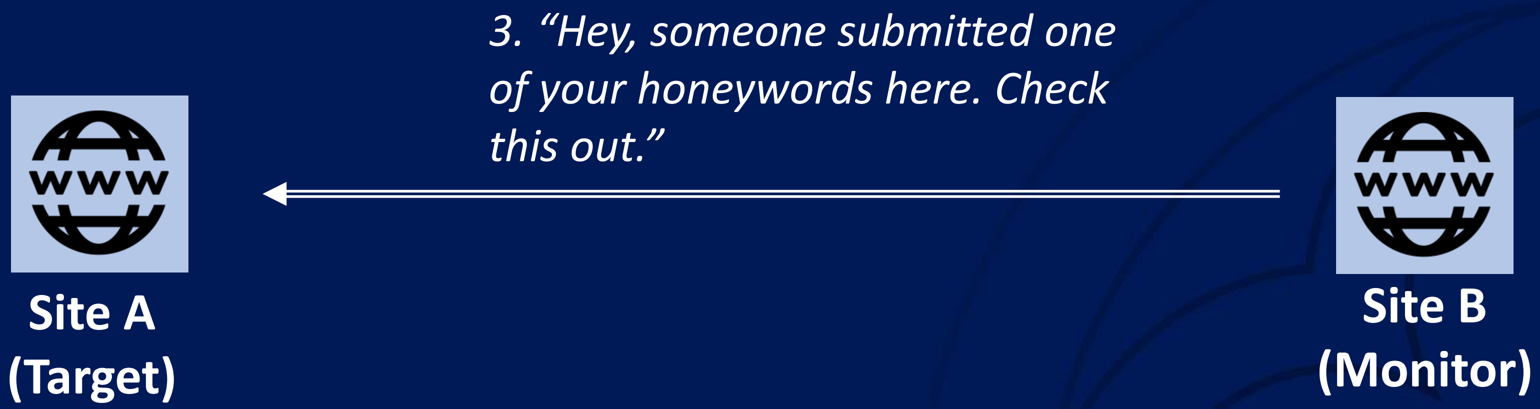
Detecting Remotely Stuffed Honeywords



Detecting Remotely Stuffed Honeywords



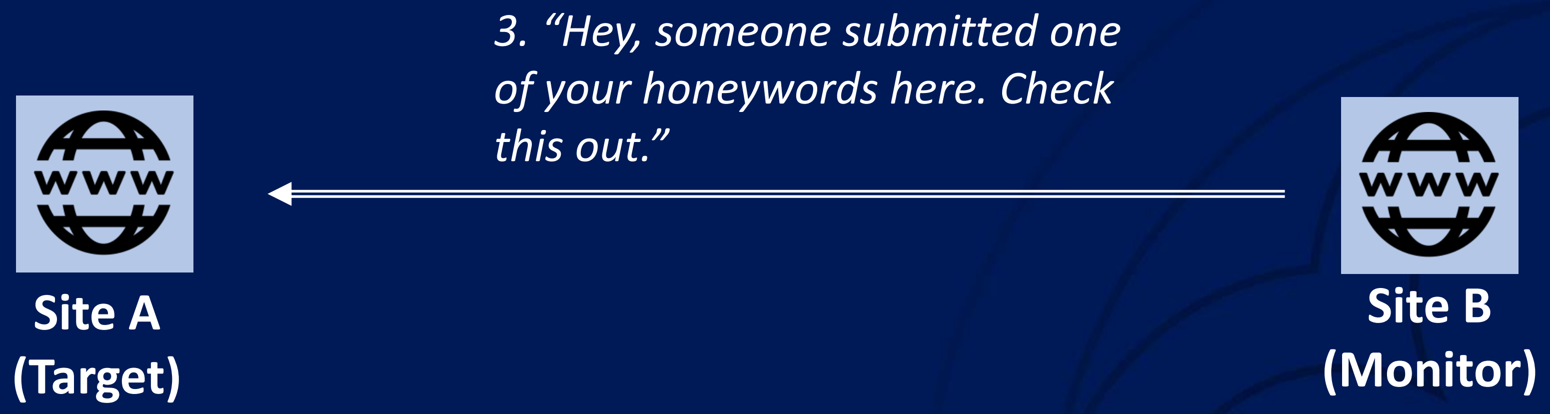
Detecting Remotely Stuffed Honeywords



- **Should not leak Target’s stored passwords to Monitor**

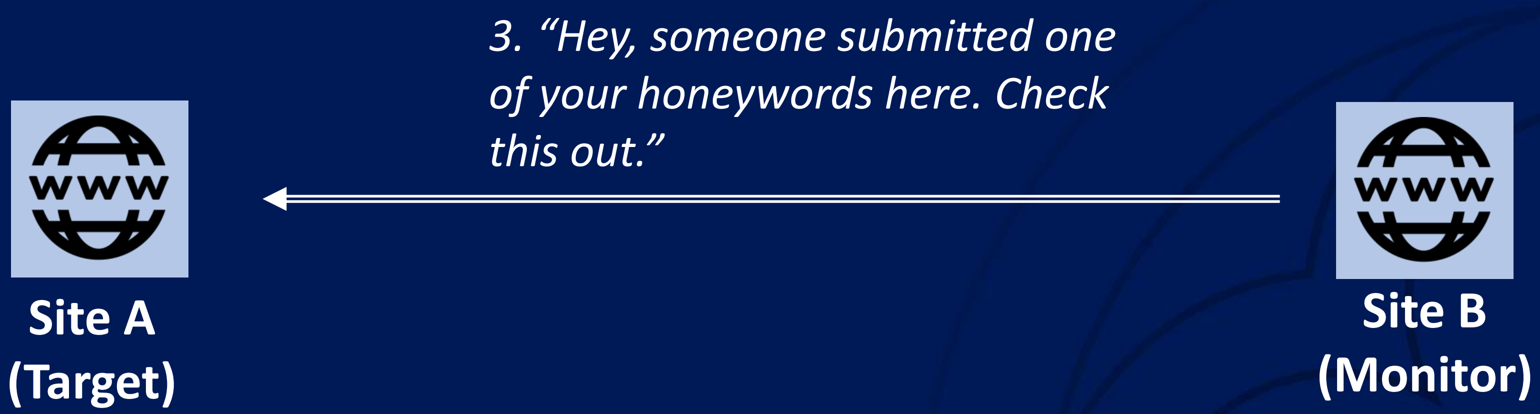
Duke

Detecting Remotely Stuffed Honeywords



- Should not leak Target's stored passwords to Monitor
- **Should not leak the submitted password at Monitor to Target if the password is not one of Target's stored passwords**

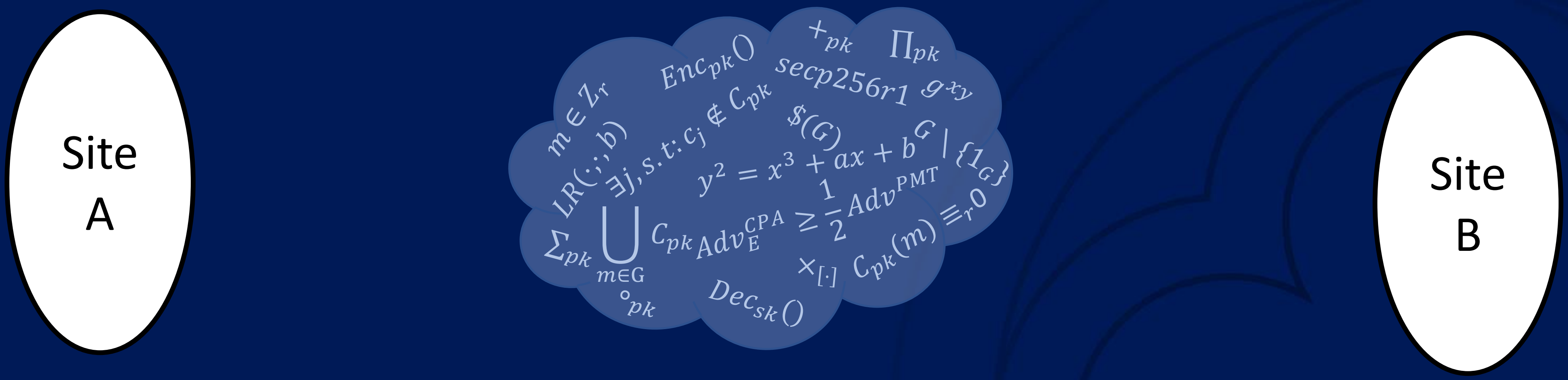
Detecting Remotely Stuffed Honeywords



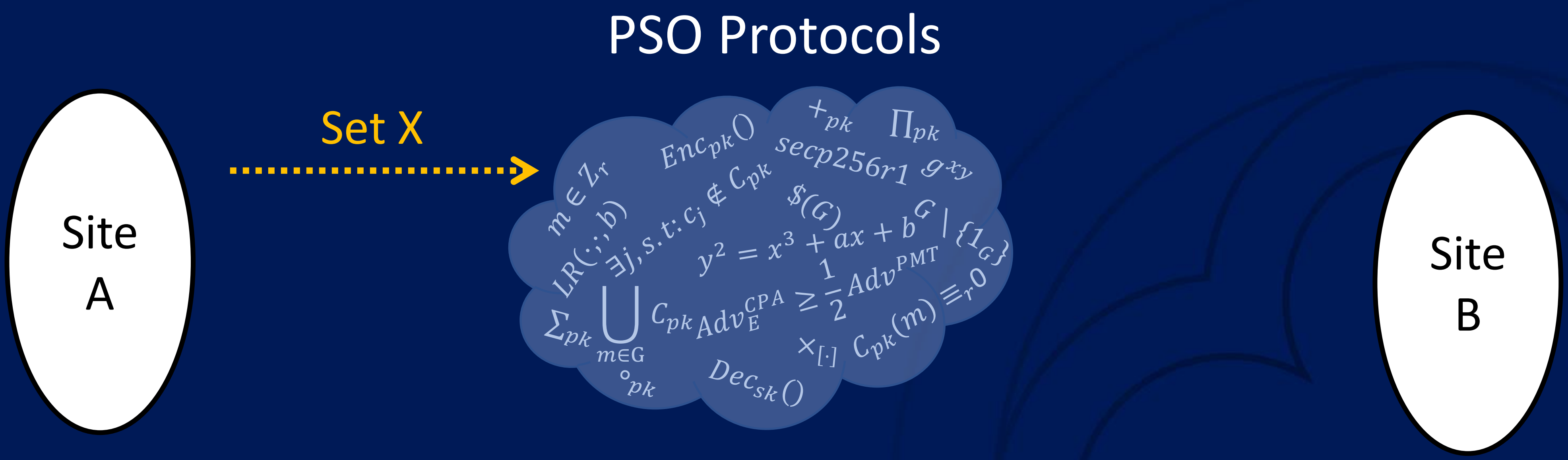
- Should not leak Target's stored passwords to Monitor
- Should not leak the submitted password at Monitor to Target if the password is not one of Target's stored passwords
- **Should not allow the monitor to trigger a false detection if no breach has happened to Target**

Private Set Operation (PSO) Protocols

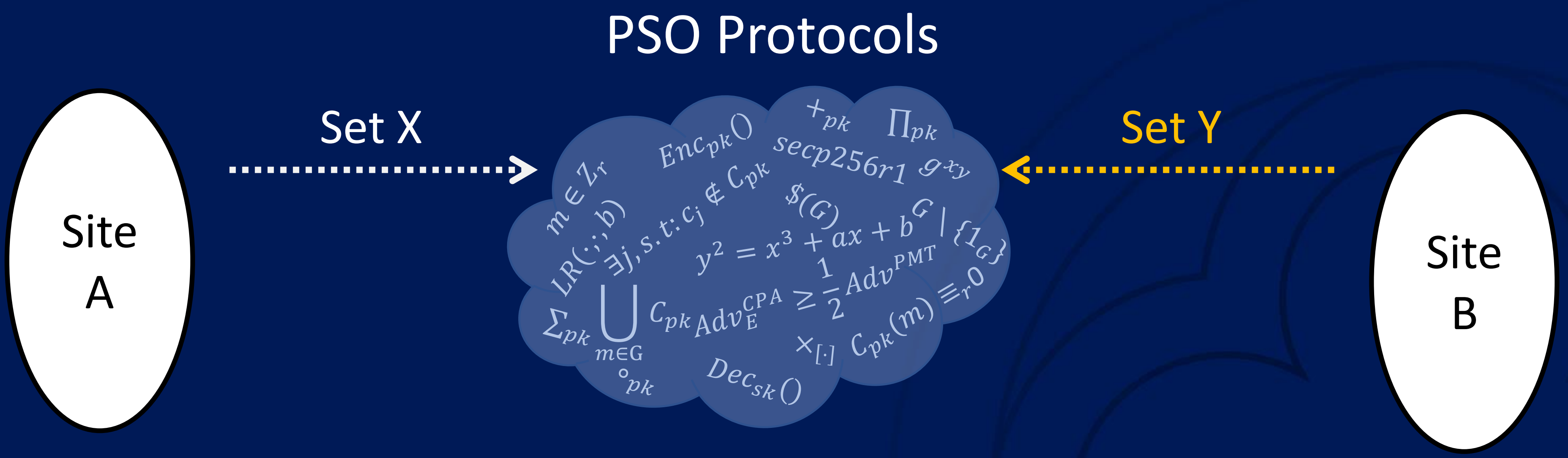
PSO Protocols



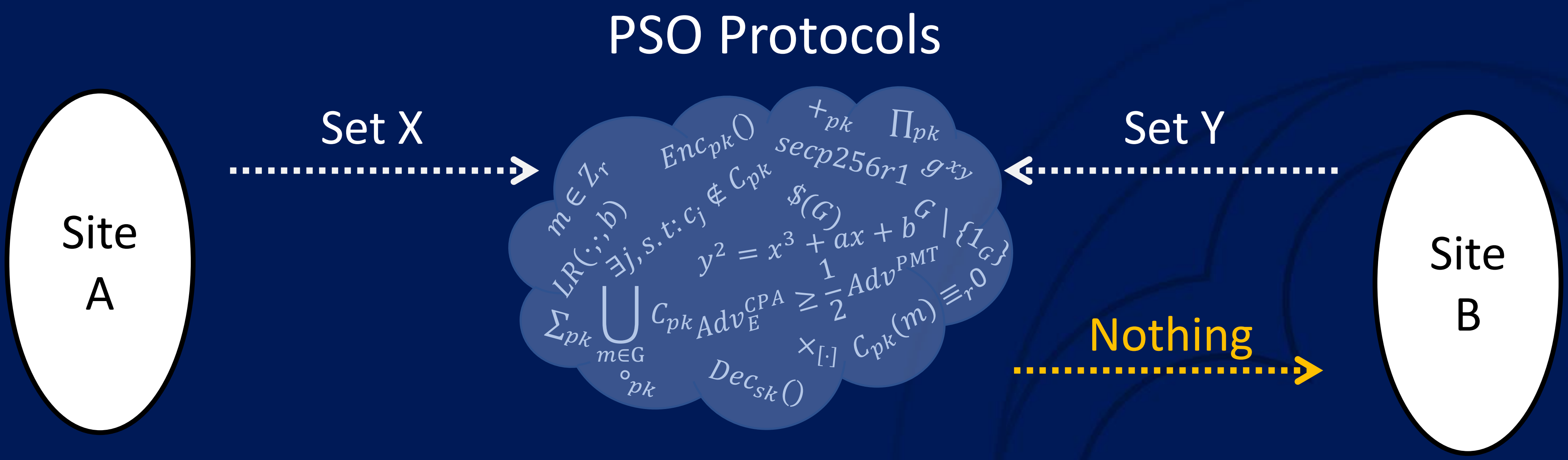
Private Set Operation (PSO) Protocols



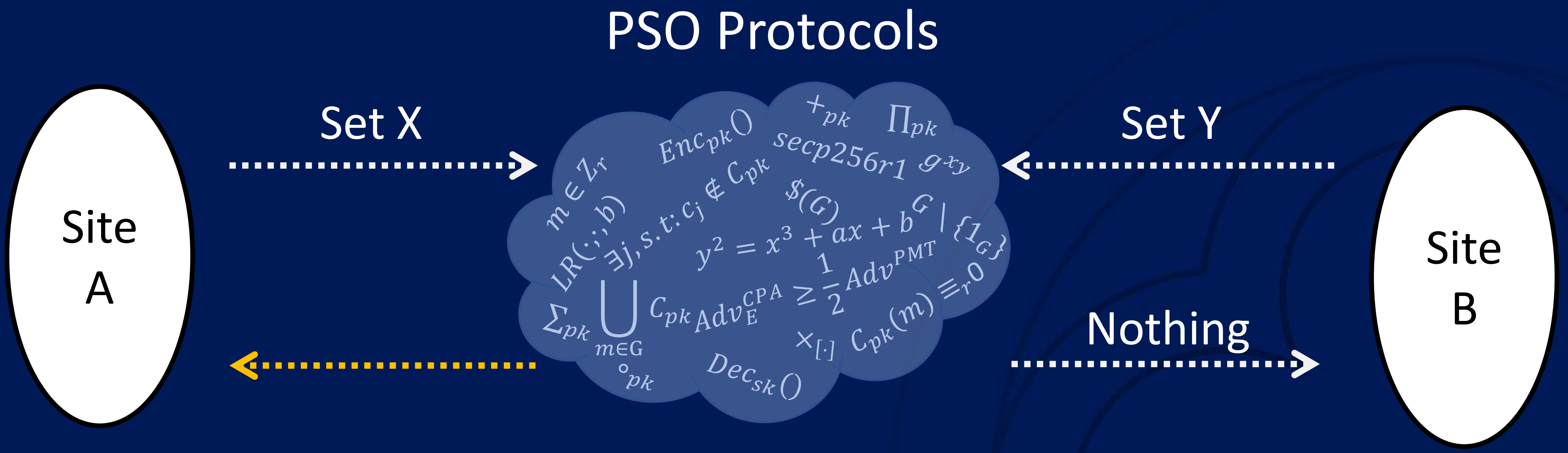
Private Set Operation (PSO) Protocols



Private Set Operation (PSO) Protocols



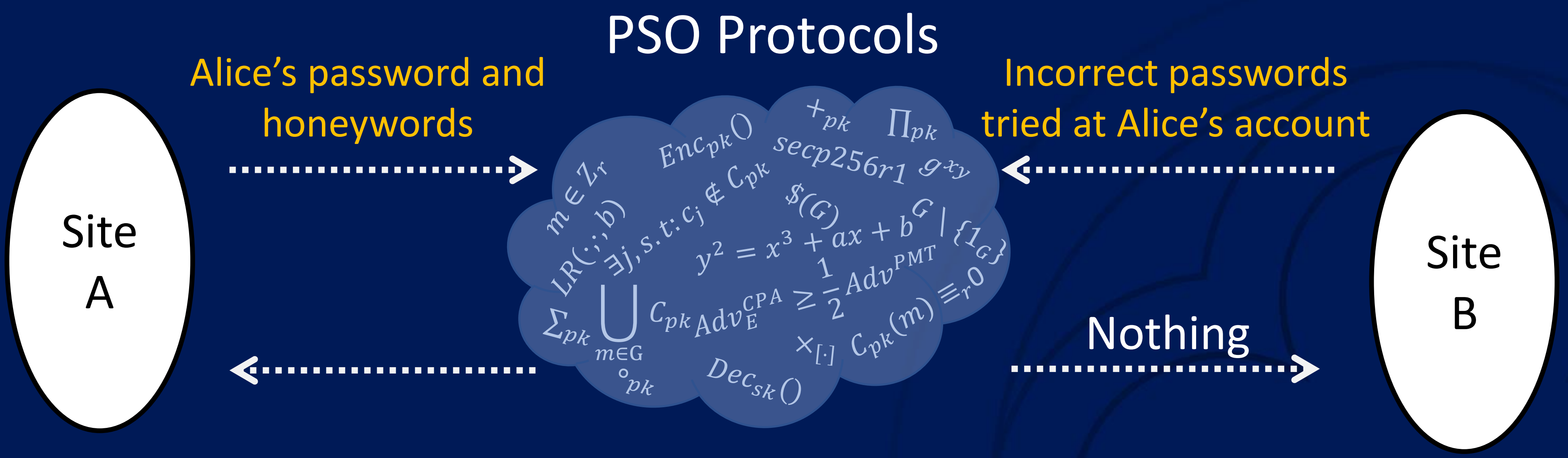
Private Set Operation (PSO) Protocols



Needed information *only*, e.g.:

- Set intersection
- Set intersection size
- …

PSO for Password Database Breach Detection



Needed information:

- Set intersection including >= 1 honeyword: password database breach

Bloom Filters

(Bloom 1970)

k uniform hash functions: $f_1(\,), \ldots, f_k(\,)$

A password hashing function: $h(\,)$

Slot

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	1	0	1	1	1	0	0

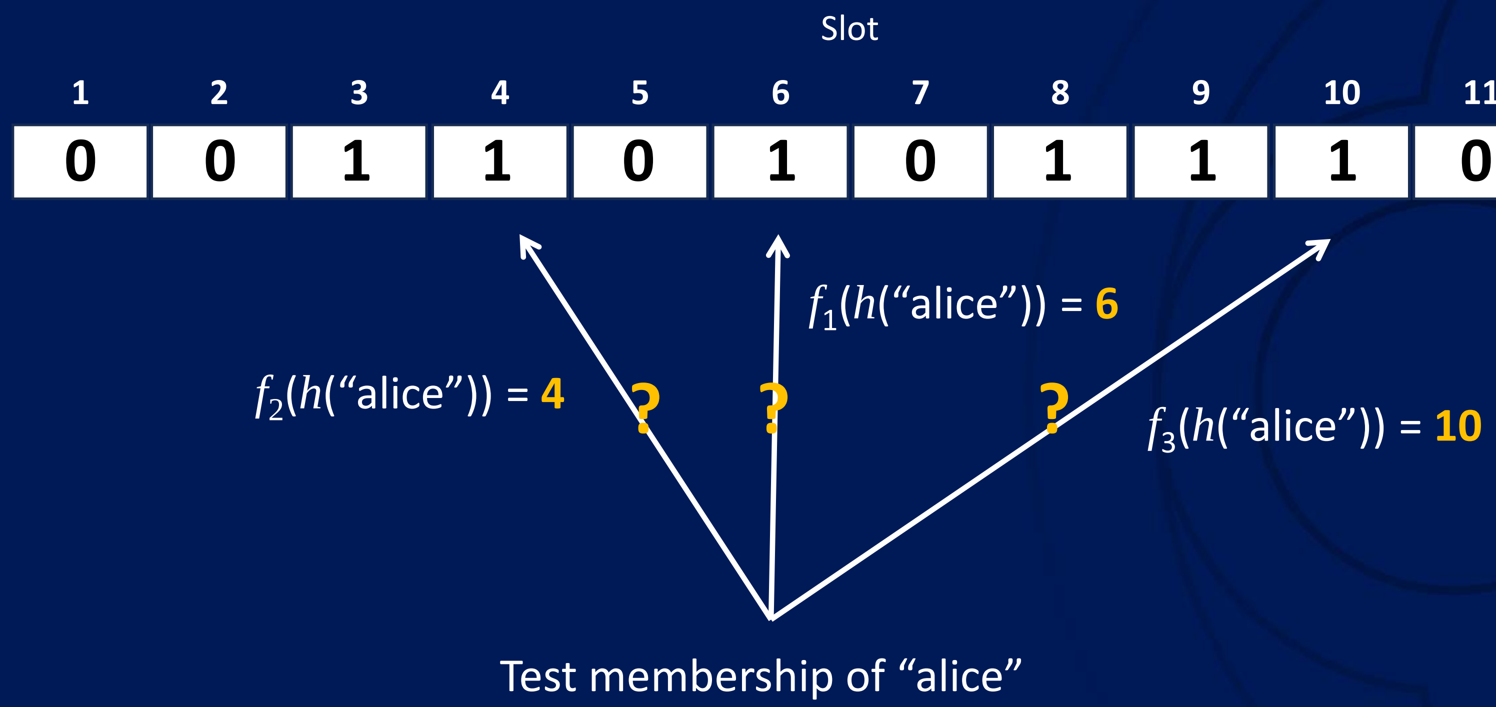
Duke

Bloom Filters

(Bloom 1970)

k uniform hash functions: $f_1(\,), \ldots, f_k(\,)$

A password hashing function: $h(\,)$



Duke

Bloom Filters

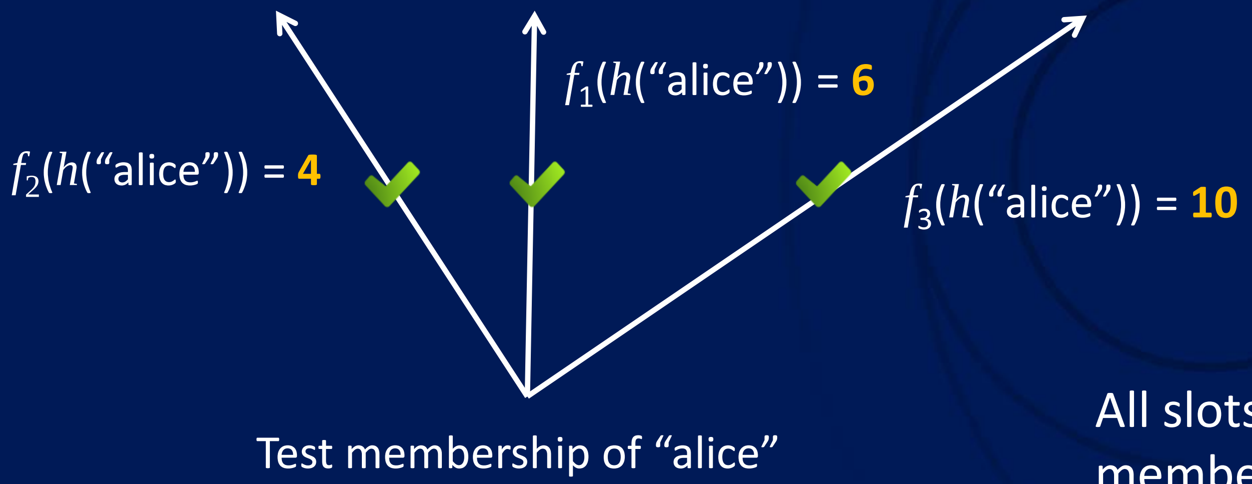
(Bloom 1970)

<u>*k* uniform hash functions</u>: $f_1(\,), \ldots, f_k(\,)$

<u>A password hashing function</u>: $h(\,)$

Slot

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	1	0	1	1	1	0	0



All slots $f_i(h(\text{"alice"})) = 1$, and so membership is *confirmed*

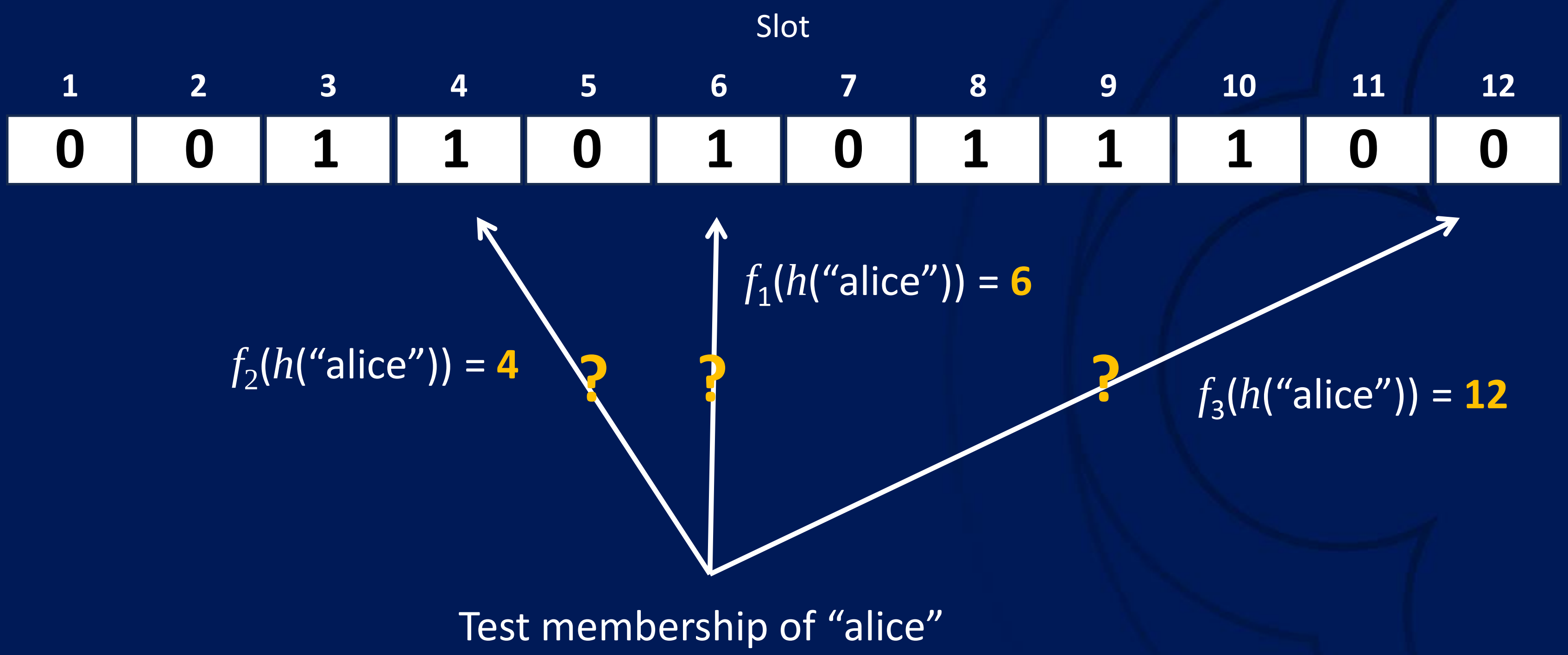
Duke

Bloom Filters

(Bloom 1970)

k uniform hash functions: $f_1(\,), \ldots, f_k(\,)$

A password hashing function: $h(\,)$

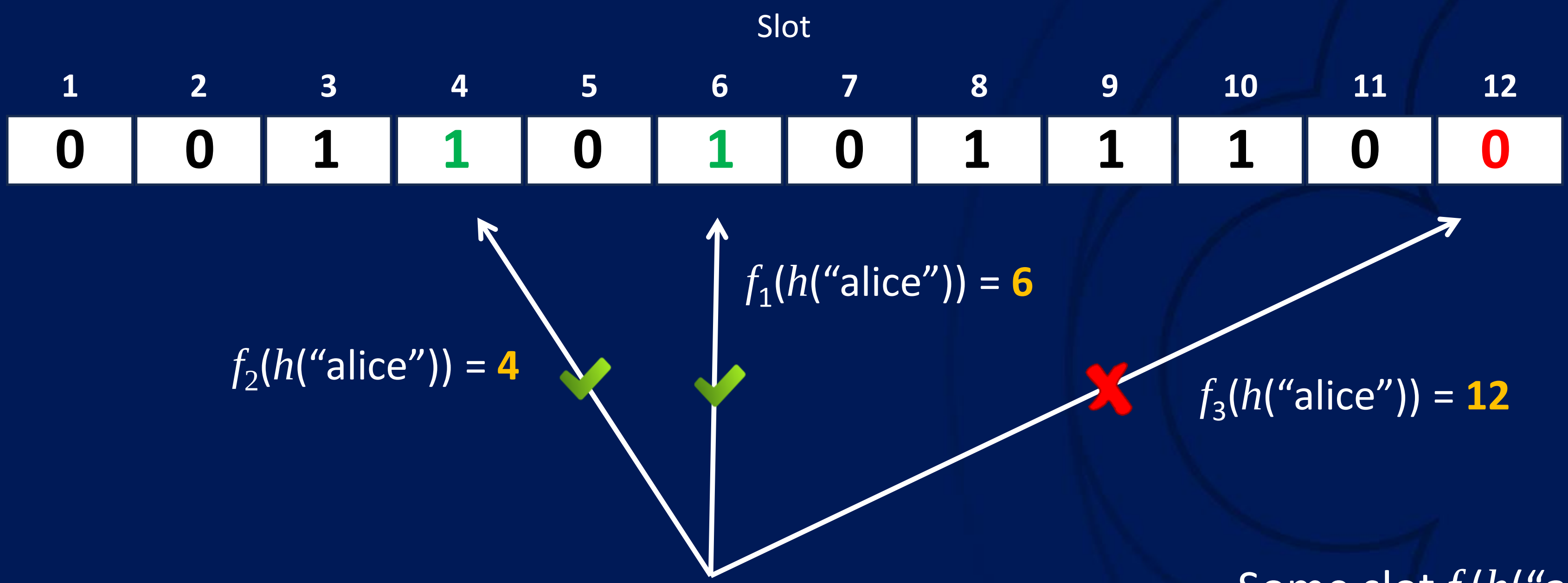


Bloom Filters

(Bloom 1970)

k uniform hash functions: $f_1(\,), \ldots, f_k(\,)$

A password hashing function: $h(\,)$



Test membership of “alice”

Some slot $f_i(h(\text{“alice”})) = 0$, and so membership is *refuted*

Duke

High-Level Structure

UID: *alice@gmail.com*

Password:

password1 *

password2 *

password3

password4 *

password5

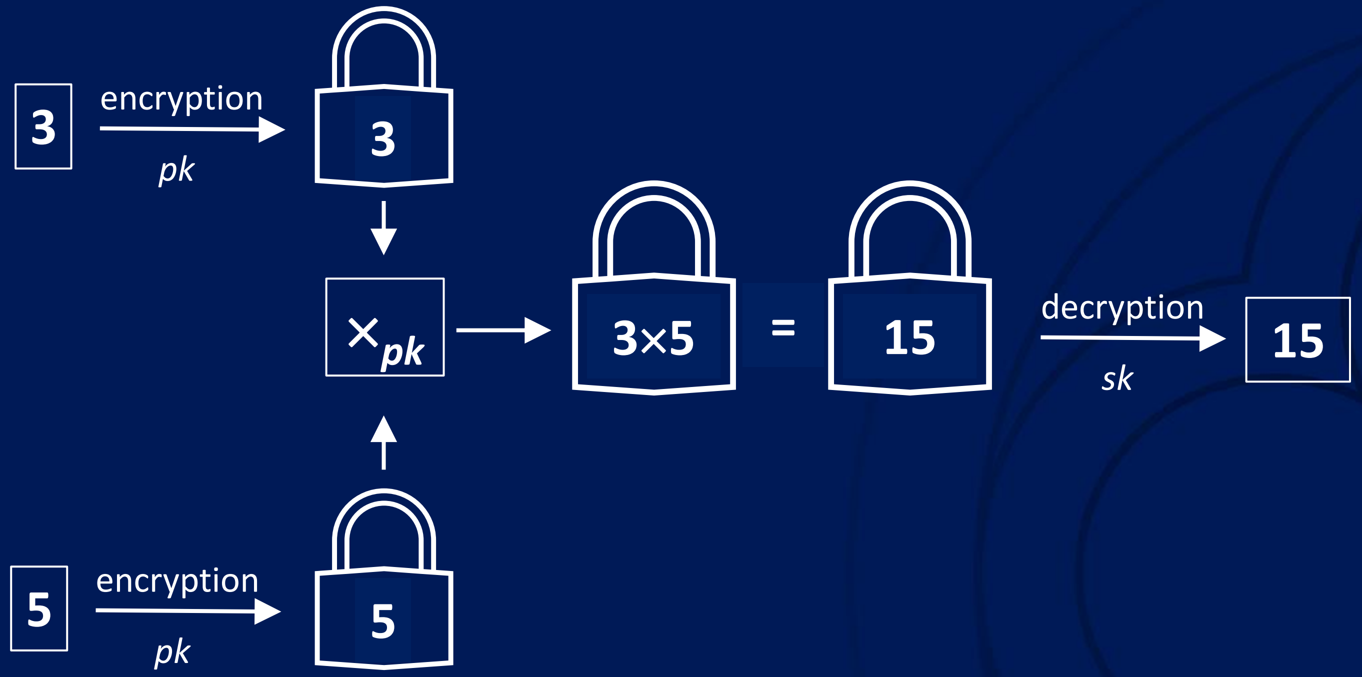
Web Server
Credential Database

1
0
1
1
…
0

→ Deploy monitor requests

Duke

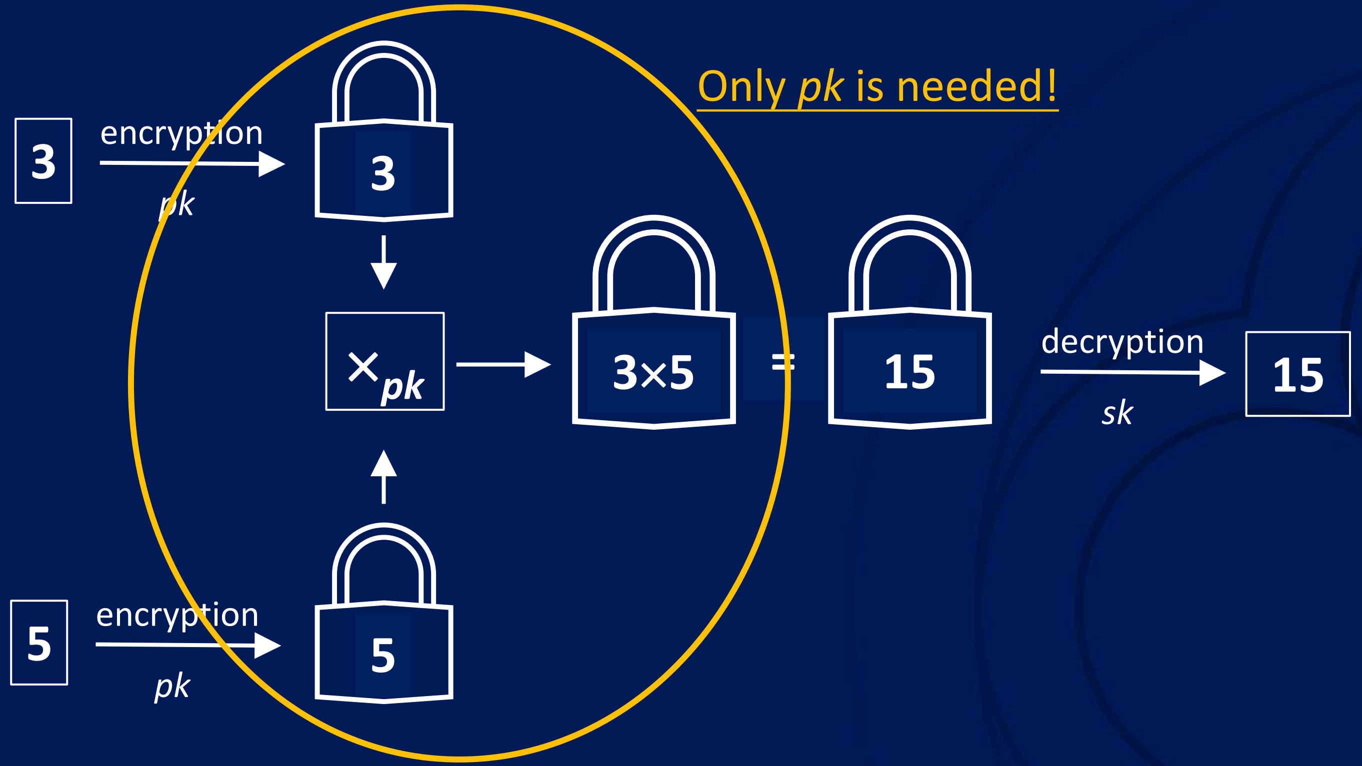
Partially Homomorphic Encryption



$\times_{pk}$: homomorphic multiplication (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

Duke

Partially Homomorphic Encryption



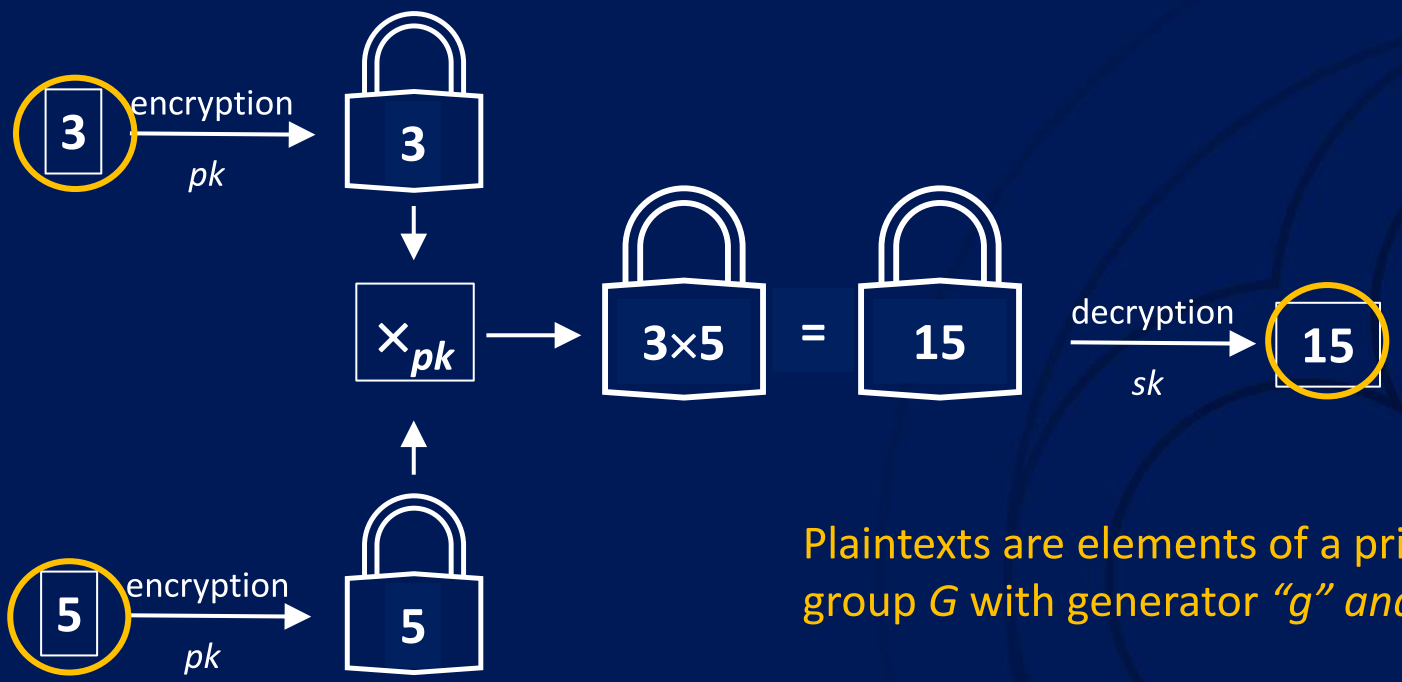
$\times_{pk}$: homomorphic multiplication (only *pk* is needed)

pk : public key (or “encryption key”)

sk : private key (or “decryption key”)

Duke

Partially Homomorphic Encryption



Plaintexts are elements of a prime-order group *G* with generator *"g" and identity "1"*

$\times_{pk}$: homomorphic multiplication (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$$b' = \sum_i slot[i]$$

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

b'

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$b', \ \{f_j\}_{j=1}^{k}$

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$b',\ \{f_j\}_{j=1}^{k},\ pk$

Duke

ElGamal 1985

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$E_{pk}(g^{-1})$ → c_1

$E_{pk}(g)$ → c_4

$b', \{f_j\}_{j=1}^{k}, pk$

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10	…	b
0	0	1	1	0	1	0	1	1	1	…	0

$E_{pk}(g^{-1})$ $E_{pk}(g)$

↓ ↓

c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} $\cdots$ c_b

$b', \ \{f_j\}_{j=1}^{k}, \ pk$

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10	…	b
0	0	1	1	0	1	0	1	1	1	…	0

$E_{pk}(g^{-1})$ $\qquad$ $E_{pk}(g)$

$\downarrow$ $\qquad$ $\downarrow$

$c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5 \quad c_6 \quad c_7 \quad c_8 \quad c_9 \quad c_{10} \quad \cdots \quad c_b$

$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b}$

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$E_{pk}(g^{-1})$ $E_{pk}(g)$

c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} $\cdots$ c_b

$$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b},\ \theta = zkp\left[c_i \in C_{pk}(g) \cup C_{pk}(g^{-1})\right]$$

Chaum and Pedersen 1993

Cramer, Damgård, and Schoenmakers 1994

Duke

PSO Protocol (Bloom Filter)

Slot

1	2	3	4	5	6	7	8	9	10		b
0	0	1	1	0	1	0	1	1	1	…	0

$$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b},\ \theta$$

Target sends this to the monitor.

Duke

PSO Protocol (Bloom Filter)

Monitor receives

$$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b},\ \theta$$

Duke

PSO Protocol (Bloom Filter)

Monitor receives

$$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b},\ \theta$$

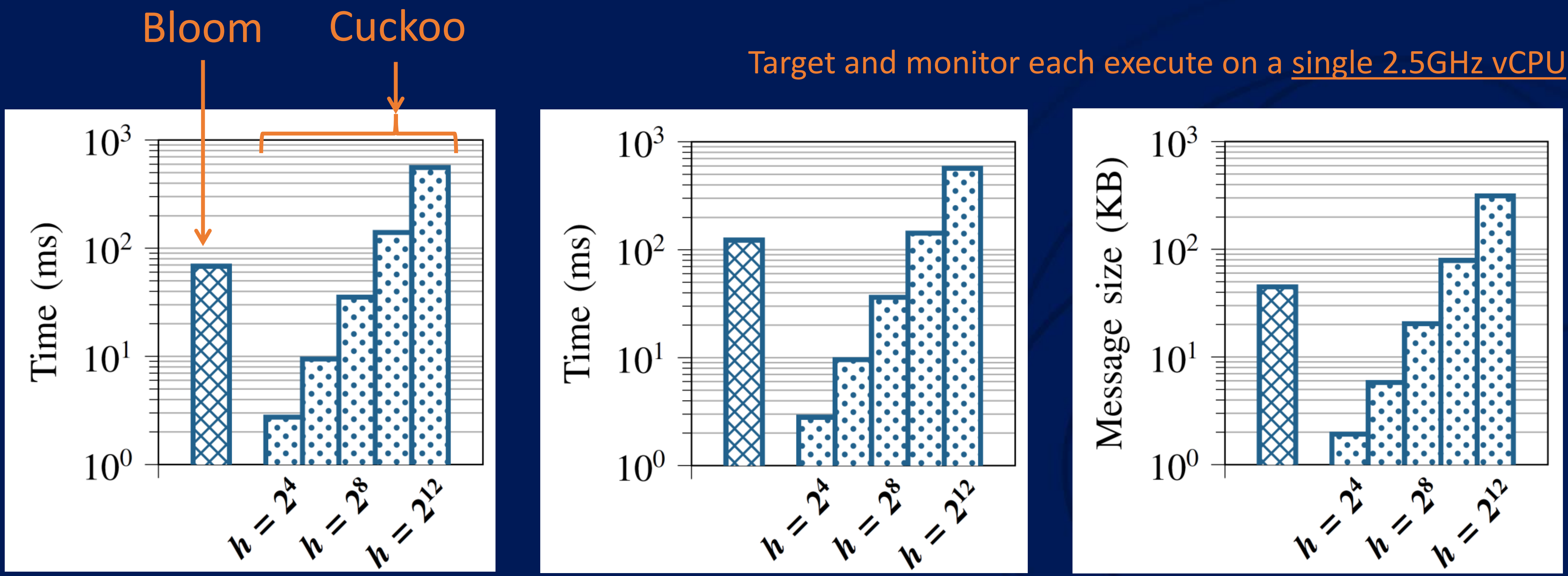
Monitor stores

$$b',\ \{f_j\}_{j=1}^{k},\ pk,\ \{c_i\}_{i=1}^{b},\ \underbrace{d_0 = c_1 \times_{pk} \cdots \times_{pk} c_b \times_{pk} E_{pk}\left(g^{b-2b'}\right)}_{d_0 \in C_{pk}(1) \text{ if } b' \text{ is truthful}}$$

Duke

Monitor Deployment Costs (Infrequent)

Target and monitor each execute on a single 2.5GHz vCPU



Request generation by target

Request validation by monitor

Request size

Duke

PSO Protocol (Bloom Filter)

For login attempt at Monitor with an <u>incorrect</u> password p where $i_j = f_j(\mathrm{h}(p))$ …

$$d_1 = \underbrace{\left(c_{i_1} \times_{pk} E_{pk}(g^{-1})\right) \times_{pk} \cdots \times_{pk} \left(c_{i_k} \times_{pk} E_{pk}(g^{-1})\right)}$$

$d_1 \in C_{pk}(1)$ if p is in the Bloom filter

Duke

PSO Protocol (Bloom Filter)

For login attempt at Monitor with an <u>incorrect</u> password p where $i_j = f_j(\mathrm{h}(p))$ …

$$d_1 = \left(c_{i_1} \times_{pk} E_{pk}(g^{-1})\right) \times_{pk} \cdots \times_{pk} \left(c_{i_k} \times_{pk} E_{pk}(g^{-1})\right)$$

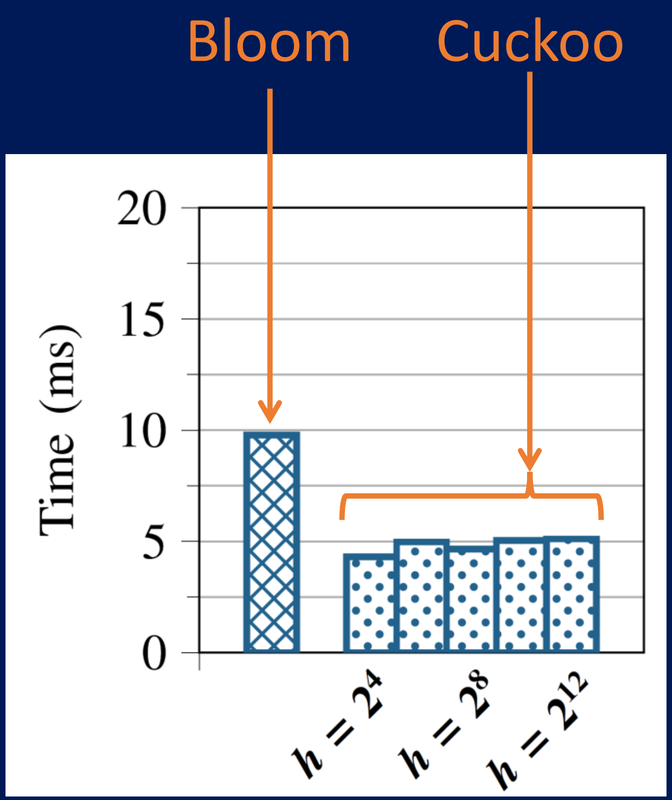
$$\hat{c}_0 = \$_{pk}(d_0) \times_{pk} \$_{pk}(d_1)$$
$$\hat{c}_1 = \$_{pk}(\hat{c}_0) \times_{pk} E_{pk}(p)$$

} Monitor returns $\hat{c}_0, \hat{c}_1$

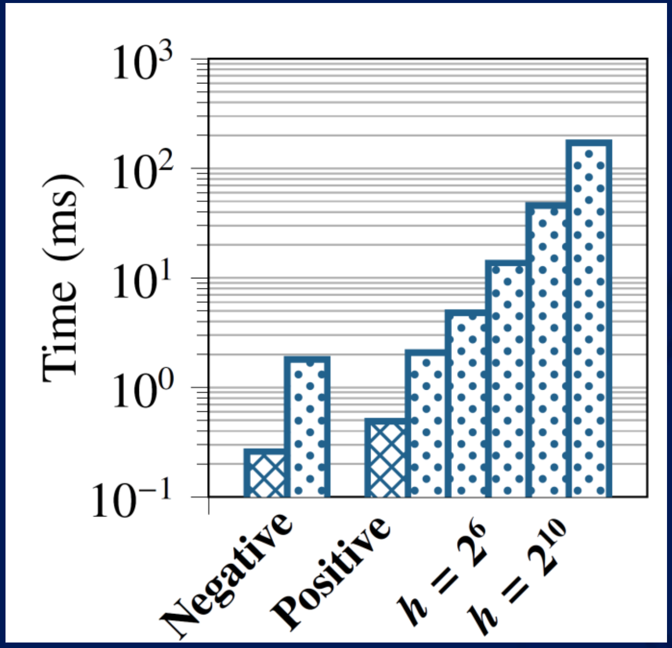
Duke

Response Generation Costs (Frequent)

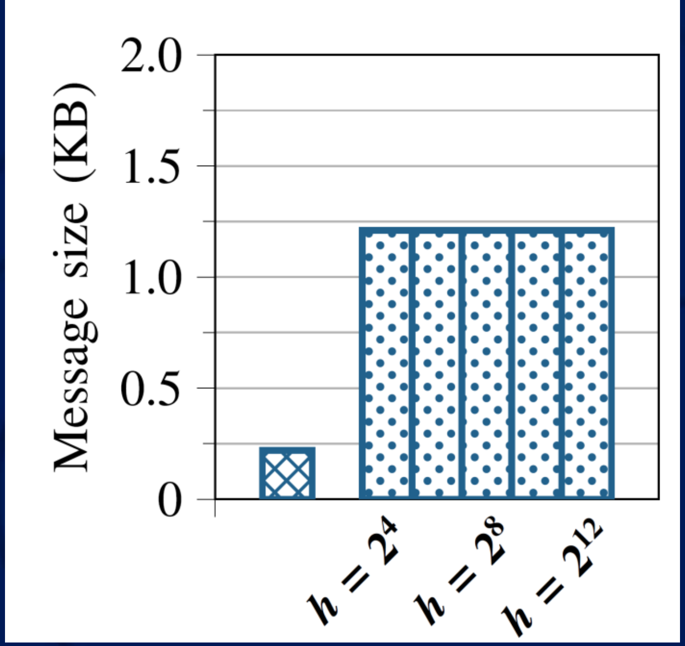
Target and monitor each execute on a single 2.5GHz vCPU



Response generation by monitor



Response processing by target



Response size

Duke

STRONTIUM Credential Stuffing Campaign
(Sep 2019 – Jun 2020)

- Most aggressive attacks averaged 335 login attempts per hour per account for hours or days at a time
- Over 200 organizations were targeted, seeing login attempts on an average of 20% of their total accounts

- The number of monitor requests for which induced monitor-response load could be maintained with one single-core 2.5GHz computer and no per-account login-attempt limit would have been an average of …
  - ~5,373 monitoring requests per monitor, or
  - ~26,865 monitoring requests per target

Duke

To Summarize

UID: *alice@gmail.com*

Password:

*password1**
*password2**
password3
*password4**
password5

Web Server
Credential Database

1
0
1
1
…
0

Deploy monitor requests

Duke

But Wait … What if Instead …

UID: *alice@gmail.com*

Password:

password1 *
password2 *
password3
password4 *
password5

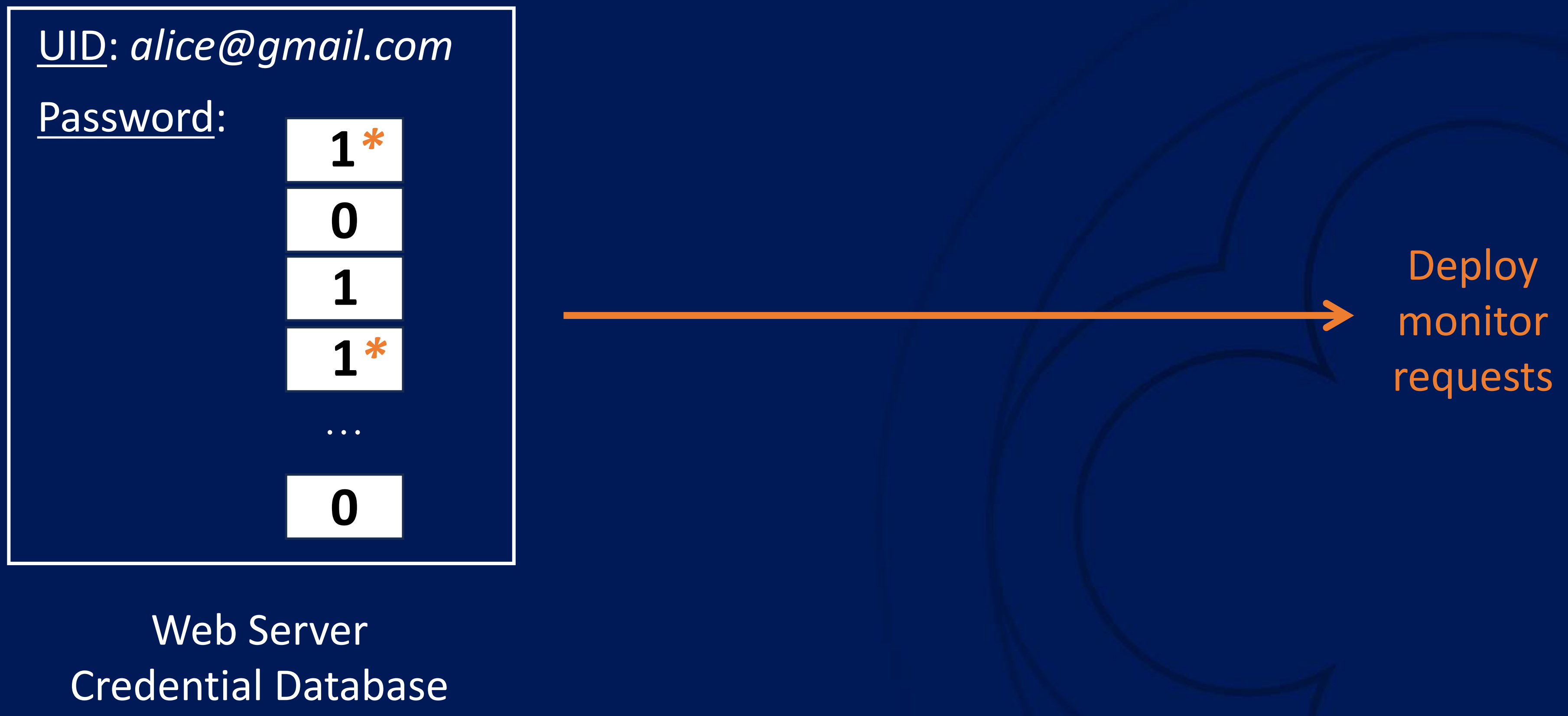
Web Server
Credential Database

1
0
1
1
…
0

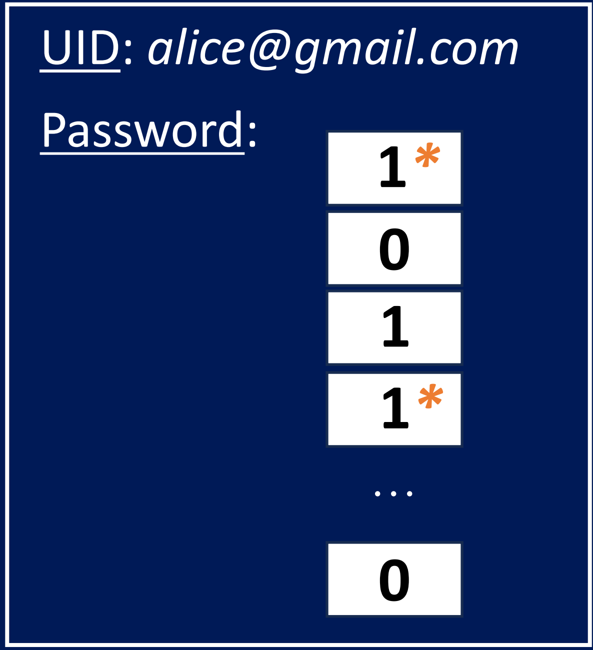
Deploy
monitor
requests

Duke

But Wait … What if Instead, We Did This?



… Whether or Not We Monitor Remotely?

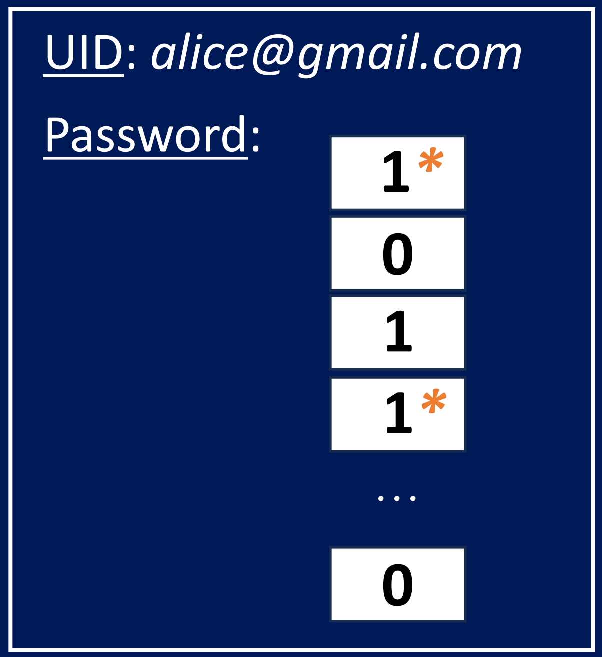


Web Server
Credential Database

New login procedure:

- If password is not in the Bloom filter, then login fails.
- If password is in Bloom filter and all its indices are marked, then login succeeds.
- Otherwise, *breach alarm*!

Bloom-Filter Collisions in Online Attacks



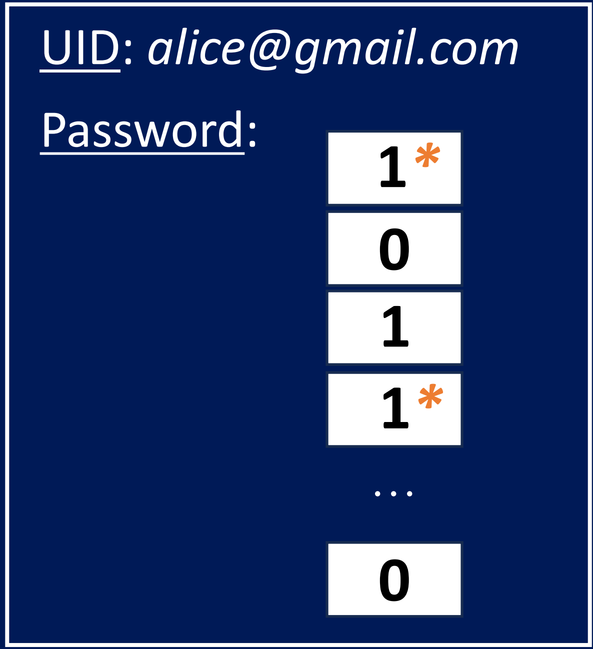
Web Server
Credential Database

The Bloom filter includes the password (hashes) we put there, but also any that collide on the 1 values.

- Some 1 unmarked ⇒ false breach alarm
- All 1's marked ⇒ unauthorized account access

Duke

Bloom-Filter Collisions in Online Attacks



Web Server
Credential Database

The Bloom filter includes the password (hashes) we put there, but also any that collide on the 1 values.

- Some 1 unmarked ⇒ false breach alarm
- All 1's marked ⇒ unauthorized account access

Duke

False Positives (= False Breach Alarms)

- Balancing false positives and false negatives in honeyword selection is notoriously difficult
  - Honeywords too similar to the user-selected password
    $\Rightarrow$ attacker who knows that password can trigger false alarms
  - Honeywords not similar enough to the user-selected password
    $\Rightarrow$ attacker who knows this user’s password elsewhere can avoid true alarm

- Most research has emphasized improving the true alarm rate
  - We believe this has been a mistake

Reasons to Focus on Reducing False Alarms

1. We only need to catch the attacker at one account—and usually the attacker wants to harvest many
   - So, a low true alarm rate can still be useful

2. Breach alarms are expensive!
   - IBM put the average cost of a breach detection and escalation at $1.24 million

Duke

The Tripwire Study

(DeBlasio, Savage, Voelker, and Snoeren 2017)

victim.org

user: notadecoy
em: notadecoy@email.org
pwd: pwd8765!

email.org

user: notadecoy

pwd: pwd8765!

A login here suggests that victim.org was breached

Duke

The Tripwire Study

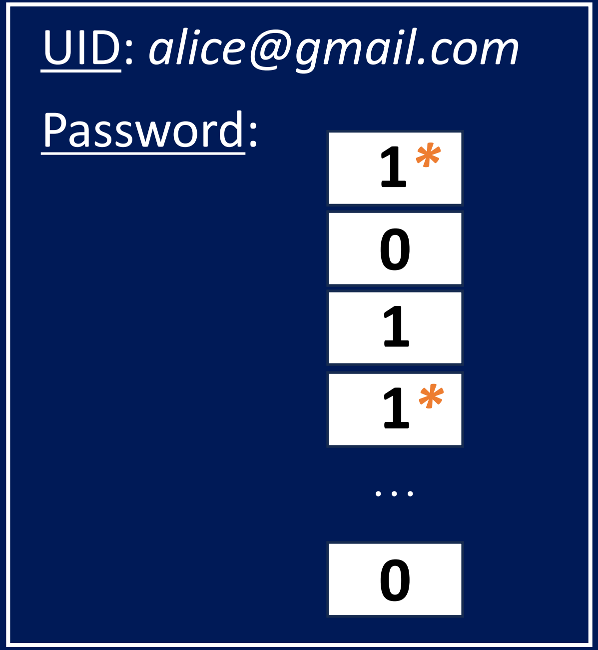
(DeBlasio, Savage, Voelker, and Snoeren 2017)

- Disclosed 18 apparent breaches (and the Tripwire methodology) to site administrators
  - Only 1/3rd responded at all
  - Only 1 indicated it would force a password reset
  - None notified their users

"a major open question ... is how much (probative, but not particularly illustrative) evidence ... is needed to convince operators to act, such as notifying their users and forcing a password reset"

Duke

Can We Analytically Quantify the False Alarm Rate?



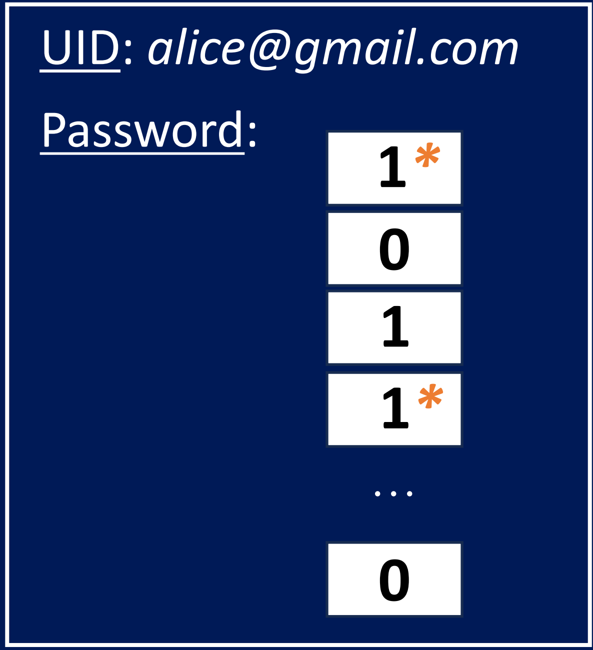
Web Server
Credential Database

If we generate honeywords heuristically, then we probably cannot.

But if we simply generate the Bloom filter *randomly* (while still including the hash of the user-selected password), then we can!

Duke

Bloom-Filter Collisions in Online Attacks



Web Server
Credential Database

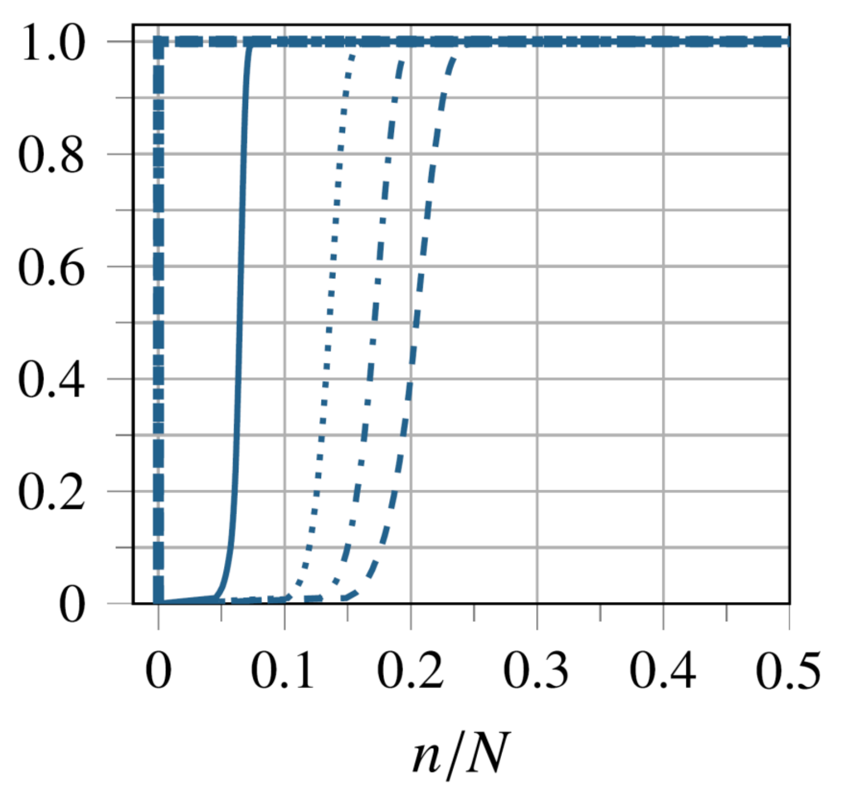
The Bloom filter contains any passwords (hashes) that collide on the 1 values.

- Some 1 unmarked ⇒ false breach alarm
- All 1's marked ⇒ unauthorized account access

Not a problem if the probability of a collision in the allowed number of online guessing attempts is sufficiently small.

Duke

Estimates of True Detection Probability



- Representative TDP plot on left, as a function of the fraction n/N of accounts accessed by the attacker
- Projected from various guessing attacks and datasets in the literature
- Settings ensure a false detection *once every 3 years,* under conservative attack estimates

Duke

To Sum Up

- Configure the Bloom filter so that …

When NO BREACH occurs, the attacker (with few†, ONLINE guesses) has a low probability of guessing passwords in the Bloom filter.

When a BREACH occurs, the attacker (with many‡, OFFLINE guesses) finds numerous passwords in the (marked) Bloom filter.

† $\lessapprox 10^6$ guesses

‡ $\gtrapprox 10^{14}$ guesses

Florêncio, Herley, and van Oorschot 2014

Duke

Coming Full Circle

Collisionful keyed hash functions with selectable collisions

Li Gong [1]

SRI International, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA

Communicated by F.B. Schneider; received 10 March 1994; revised 12 December 1994

"Thus the collision-resistant property can in fact be a liability, especially when the user's secret is a normal password that is typically chosen from a relatively small space ... The existence of easy-to-find collisions ... protects a user's password in that an attacker cannot determine which is the user's real password."

Password Hashing Competition (2014-5)

(https://www.password-hashing.net)

Password hashing is everywhere, from web services' credentials storage to mobile and desktop authentication or disk encryption systems. Yet there wasn't an established standard to fulfill the needs of modern applications and to best protect against attackers. We started the Password Hashing Competition (PHC) to solve this problem.

Submissions will be evaluated according the following criteria:

Security

- Cryptographic security: the function should behave as a random function (random-looking output, one-way, collision resistant, immune to length extension, etc.).

Duke

Has Collision-Resistant Password Hashing for Credential Storage Done More Harm than Good?

- A preimage is almost certainly the password the user chose!
- This certainty …
  - Permits the attacker to confidently end his search
  - Facilitates attacking the user’s accounts at other sites

Duke

Li’s Takeaways

1. Technology transfer from research is a rarity and usually occurs by a researcher playing a central role in that transfer
    - Example: Jerry Saltzer carried the PAKE idea to Kerberos

2. Unless the research is truly transformational, it must be perfectly packaged for someone else to adopt it

Duke

My Takeaways

1. Defenders are self-interested, just like attackers are
    - Until now, collisionful hashing would have served primarily to reduce the confidence that a hash preimage will work at another, unbreached site

2. Practical impact of security research is often as much about timing as it is about the quality of the idea
    - Additional context learned over the last 30 years reveals the potential worth of collisionful hashing

Duke