Bernoulli Honeywords

Coby Wang
Visa Research

Michael K. Reiter
Duke University

Duke

** Coby participated in this research while at Duke University*

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.*

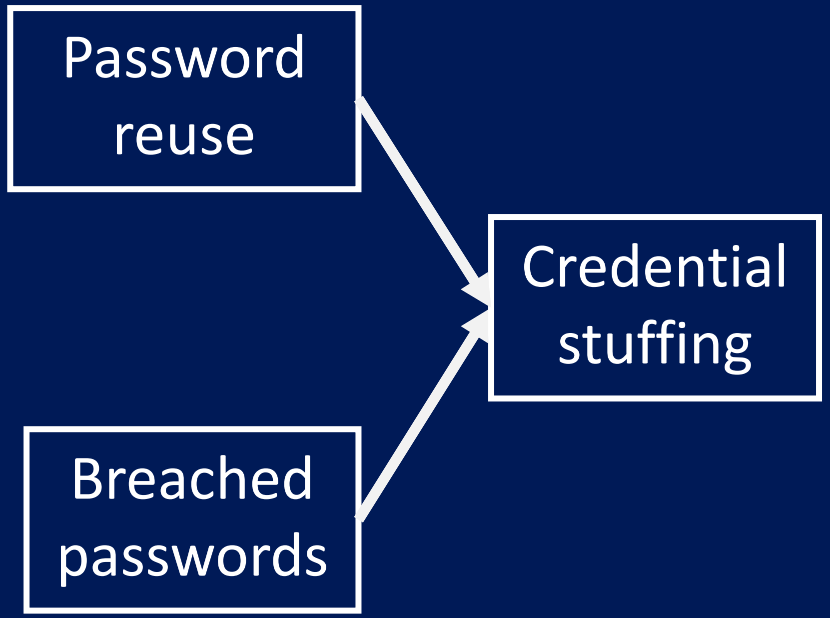
Breached passwords

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

Duke

Credential Abuse across Sites

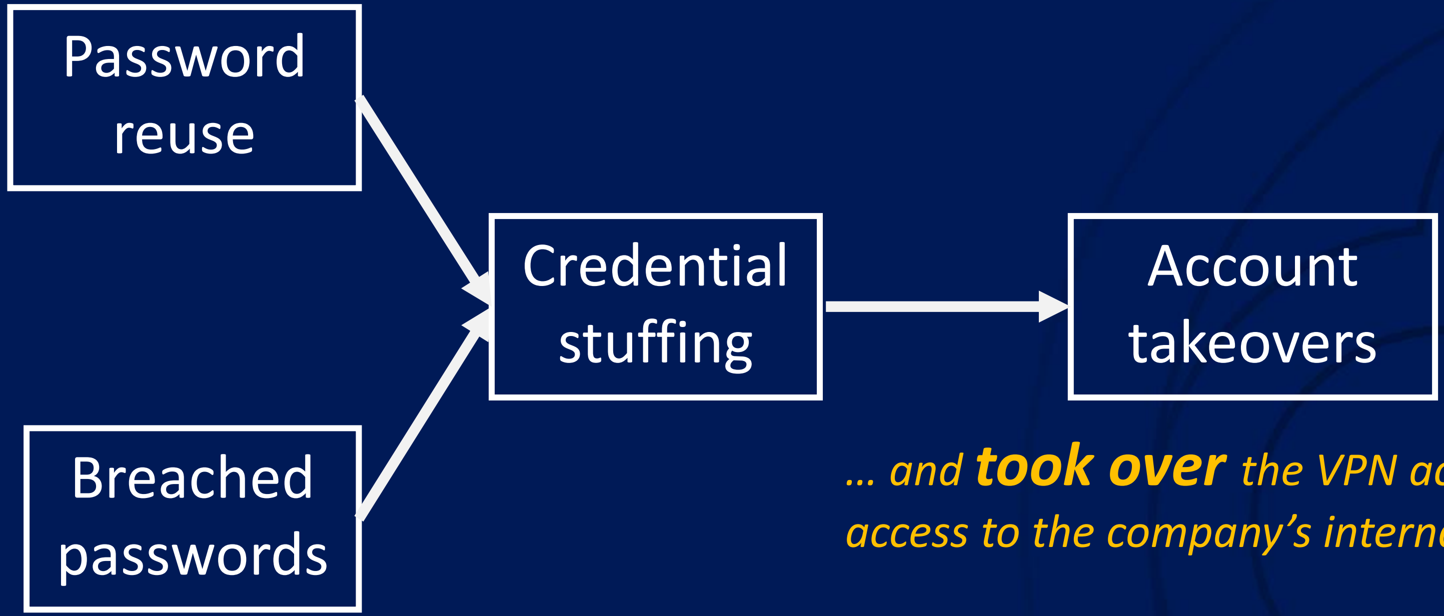
The Colonial Pipeline Attack (May 2021)



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

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

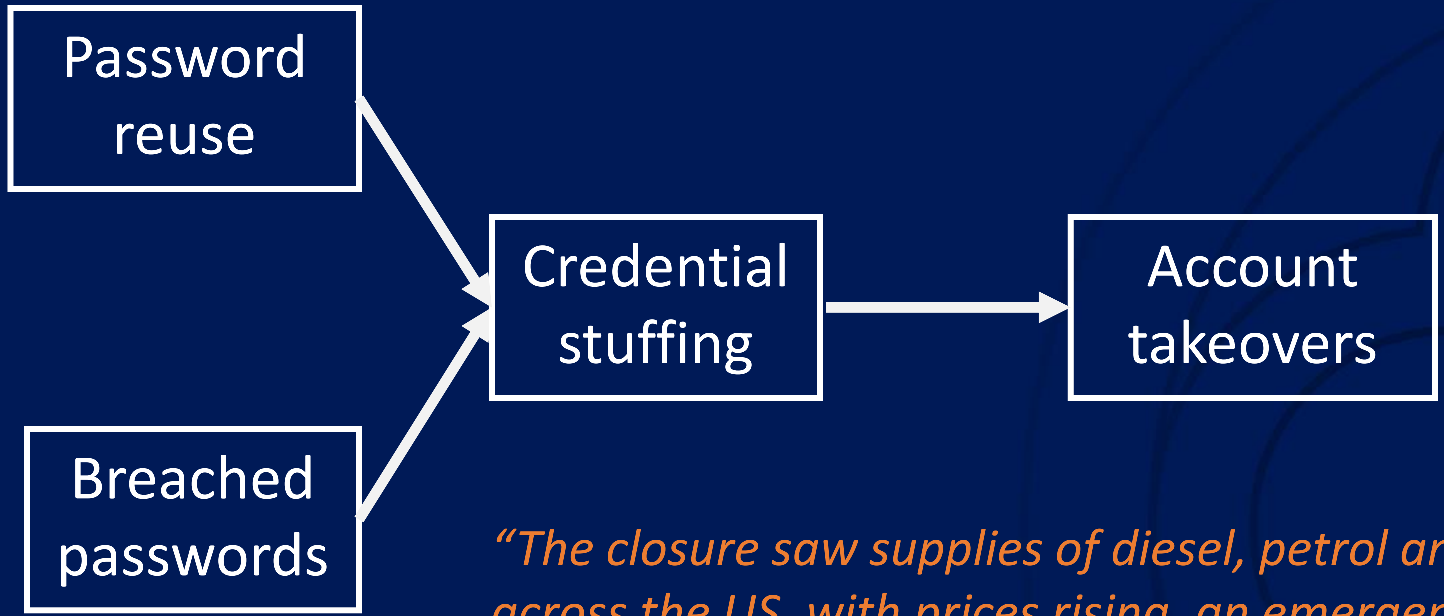


… and ***took over*** *the VPN account, getting access to the company’s internal network.*

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

Credential Abuse across Sites

The Colonial Pipeline Attack (May 2021)

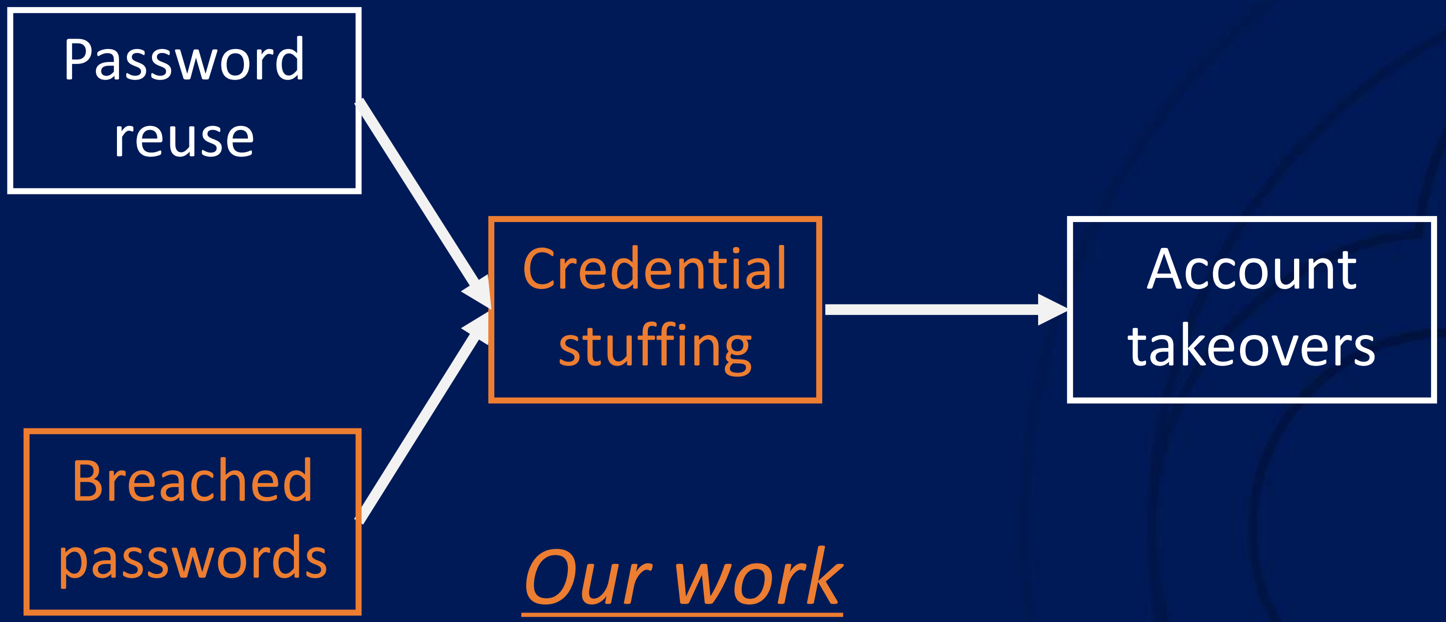


"The closure saw supplies of diesel, petrol and jet fuel tighten across the US, with prices rising, an emergency waiver passed on Monday and a number of states declaring an emergency."

-- BBC

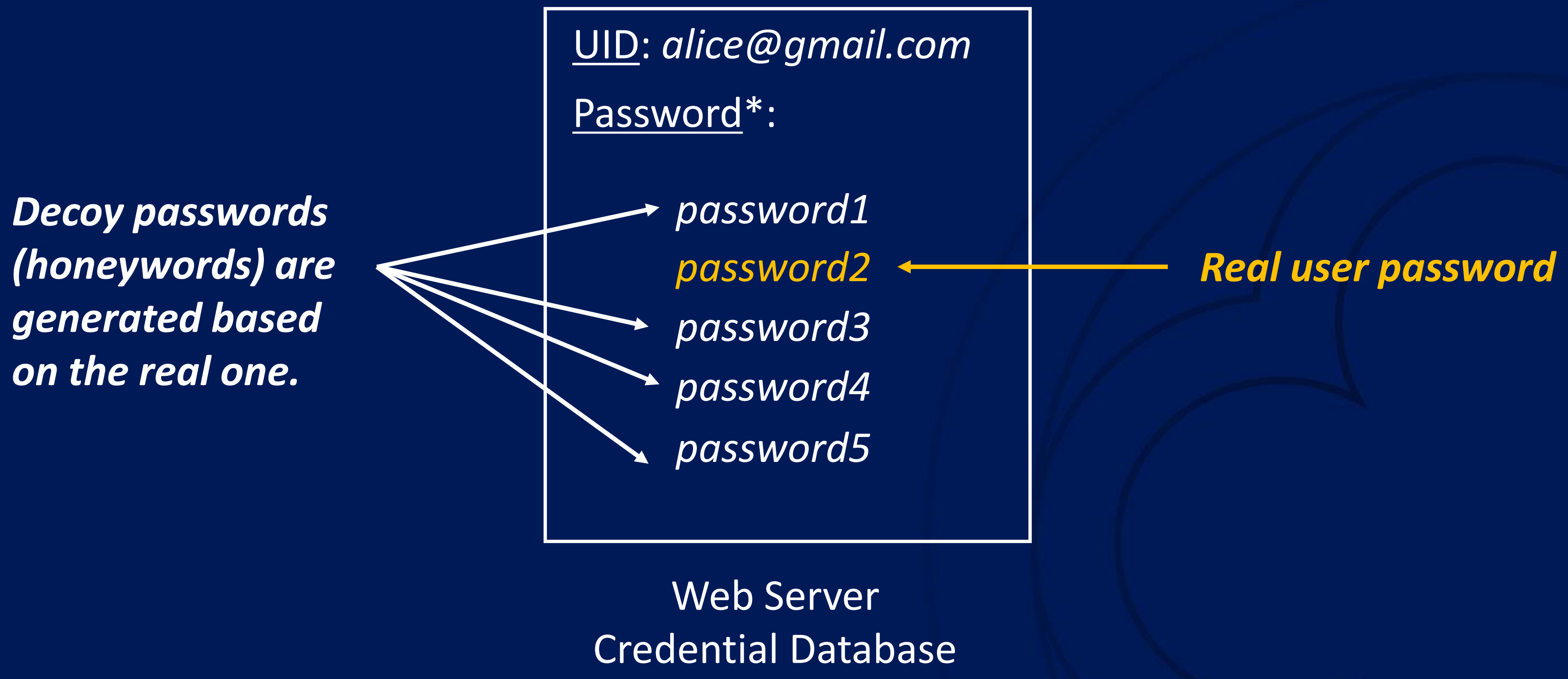
Duke

Where to Tackle this Problem?



Duke

Honeywords
(Juels & Rivest 2013)



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

Duke

Honeywords

(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

???

Web Server
Credential Database

Wait … How can the defender determine whether a given password (in the list) should result in a successful login or a breach alarm?

Duke

Asymmetric Design

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

<

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

2 is real

Honeychecker (Juels & Rivest 2013)

Attacker Knowledge

Defender Knowledge

Duke

Symmetric Design

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

=

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

Attacker Knowledge

Defender Knowledge

Amnesia (Wang & Reiter 2021)

Duke

Asymmetric vs. Symmetric

password1
password2
password3
password4
password5

2 is real

Breach Attacker knowledge

<

Defender knowledge (in the form of persistent storage)

Example: Honeychecker (Juels & Rivest 2013)

password1
password2
password3
password4
password5

Breach Attacker knowledge

=

Defender knowledge (in the form of persistent storage)

Example: Amnesia (Wang & Reiter 2021)

Duke

Asymmetric vs. Symmetric

password1
password2
password3
password4
password5

2 is real

Breach Attacker knowledge
<
Defender knowledge (in the form of persistent storage)

Example: Honeychecker (Juels & Rivest 2013)

password1
password2
password3
password4
password5

Breach Attacker knowledge
=
Defender knowledge (in the form of persistent storage)

Example: Amnesia (Wang & Reiter 2021)

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 information about this user 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 *per account* can still be useful

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

3. Without quantifying false alarms, admins will ignore alarms
   - See the Tripwire study [DeBlasio, Savage, Voelker, and Snoeren 2017]

Duke

Bernoulli Honeywords

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

Web Server
Credential Database

UID: *alice@gmail.com*

Password:

Entire password space

Web Server
Credential Database

Duke

Bernoulli Honeywords

UID: *alice@gmail.com*

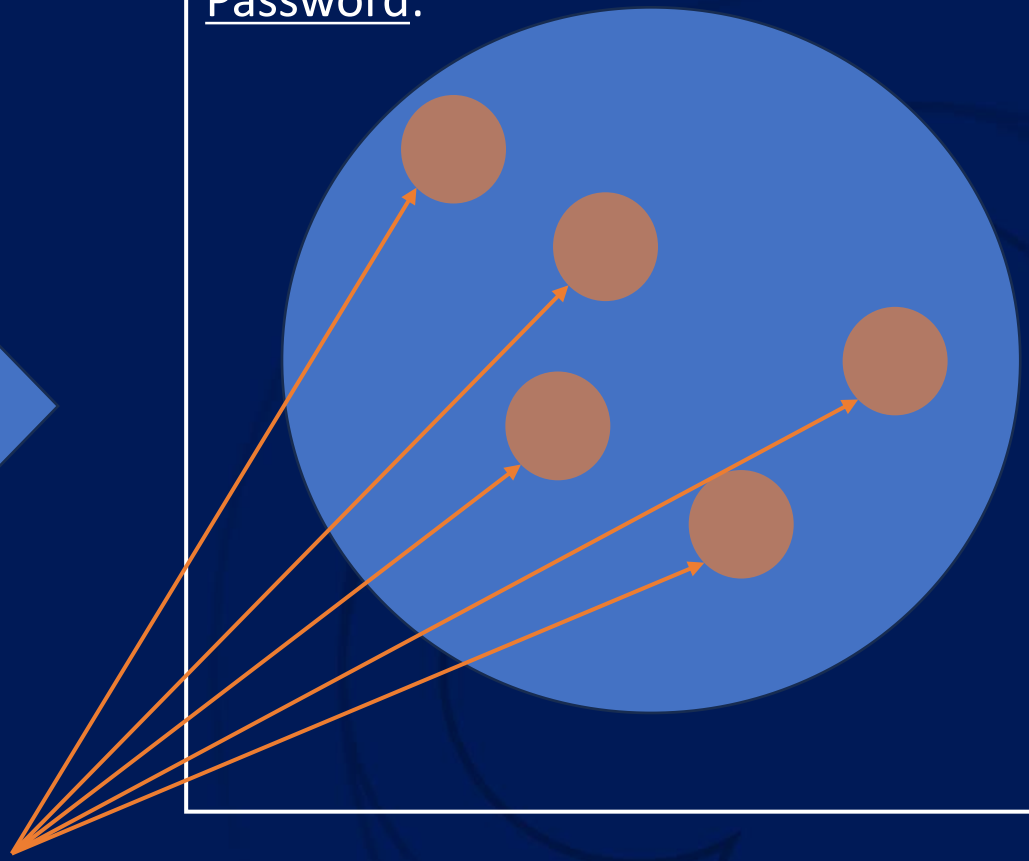
Password:

password1
password2
password3
password4
password5

Web Server
Credential Database

UID: *alice@gmail.com*

Password:



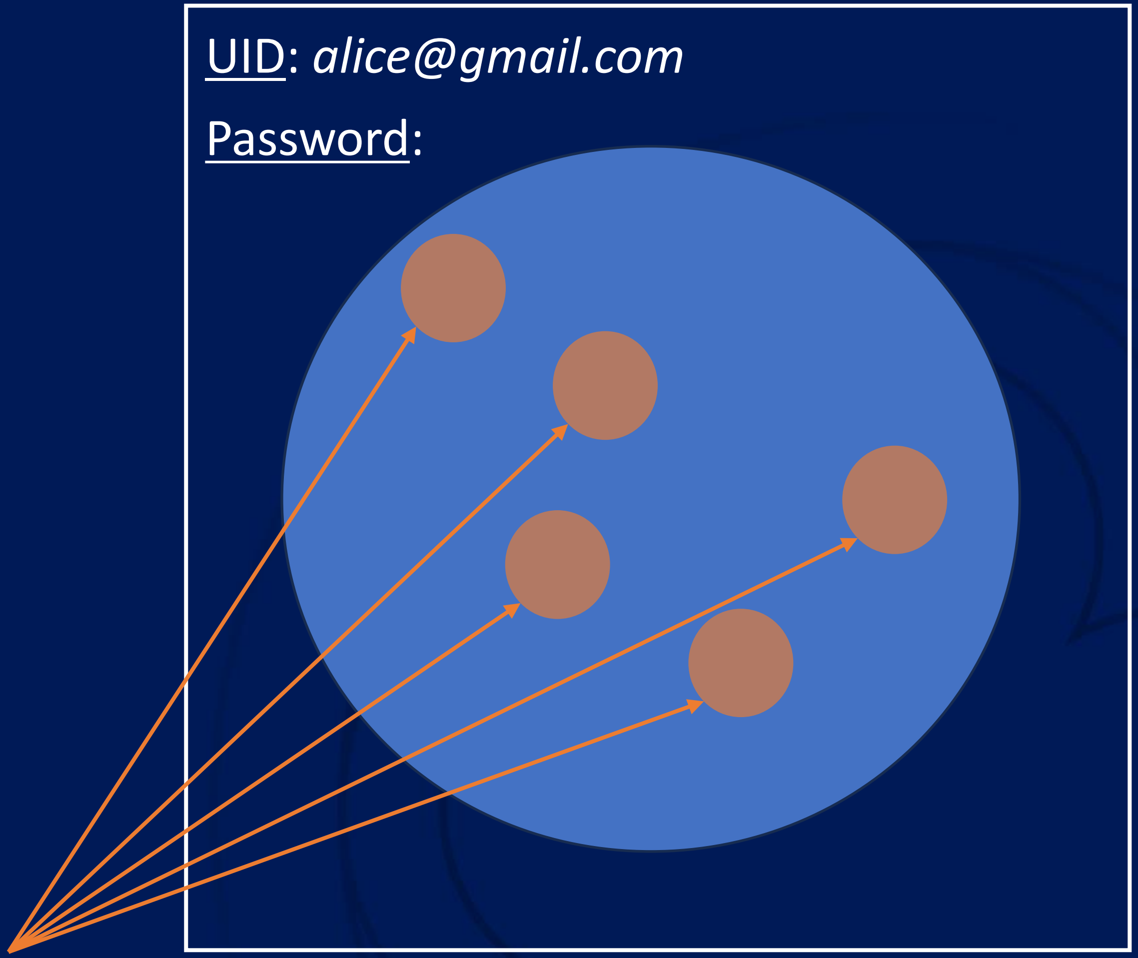
Each incorrect password is chosen as a honeyword according to a **Bernoulli process**

Web Server
Credential Database

Bernoulli Honeywords

Questions:

- *How to efficiently sample and store honeywords from the entire password space?*
- *How to efficiently determine whether a login attempt has a correct, incorrect, or decoy password?*
- *How to allow easy parameterization of Bernoulli honeywords?*



Each incorrect password is chosen as a honeyword according to a **Bernoulli process**

Web Server Credential Database

Duke

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

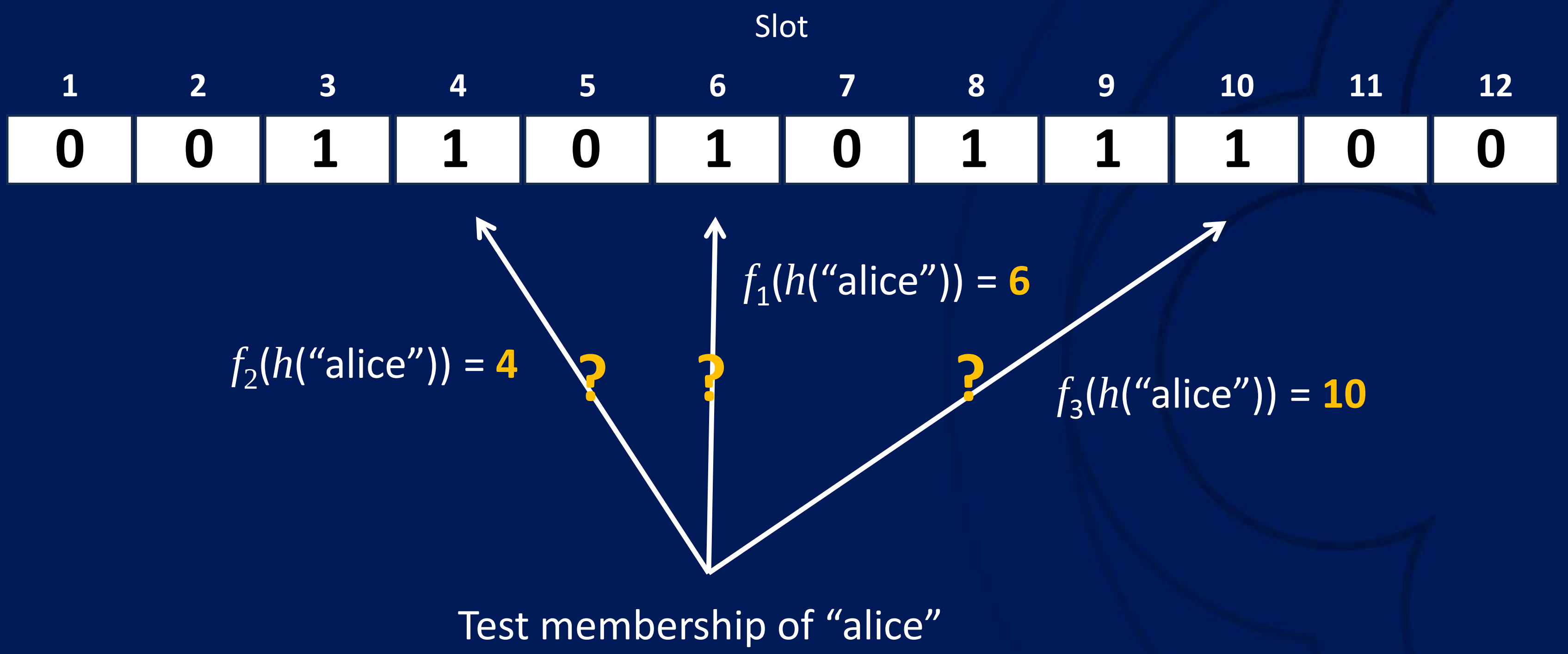
Duke

Bloom Filters

(Bloom 1970)

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

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



Duke

Bloom Filters

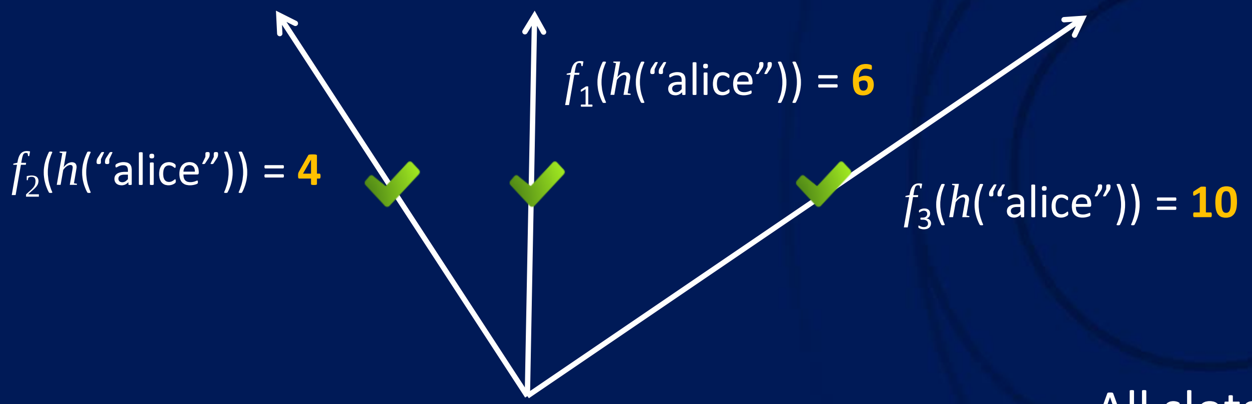
(Bloom 1970)

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



Test membership of “alice”

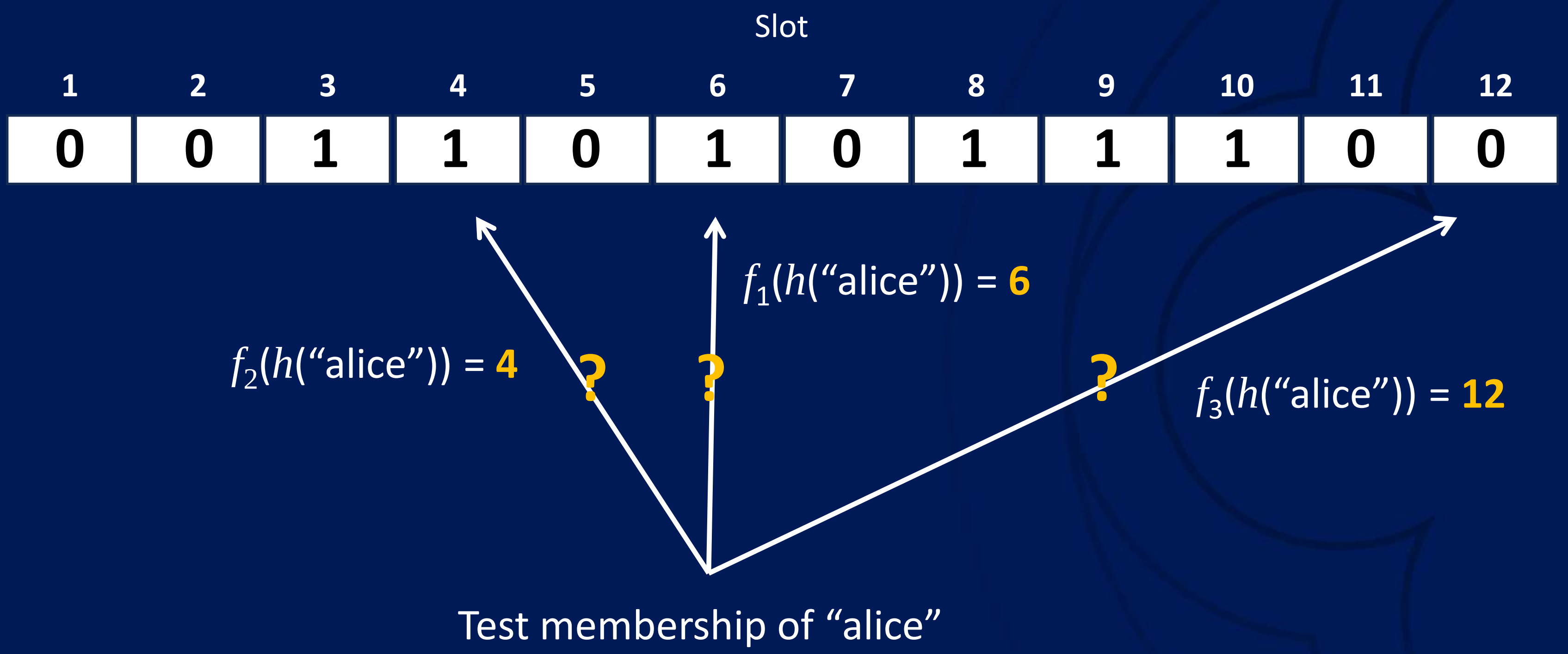
All slots $f_i(h(\text{“alice”})) = 1$, and so membership is *confirmed*

Bloom Filters

(Bloom 1970)

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

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

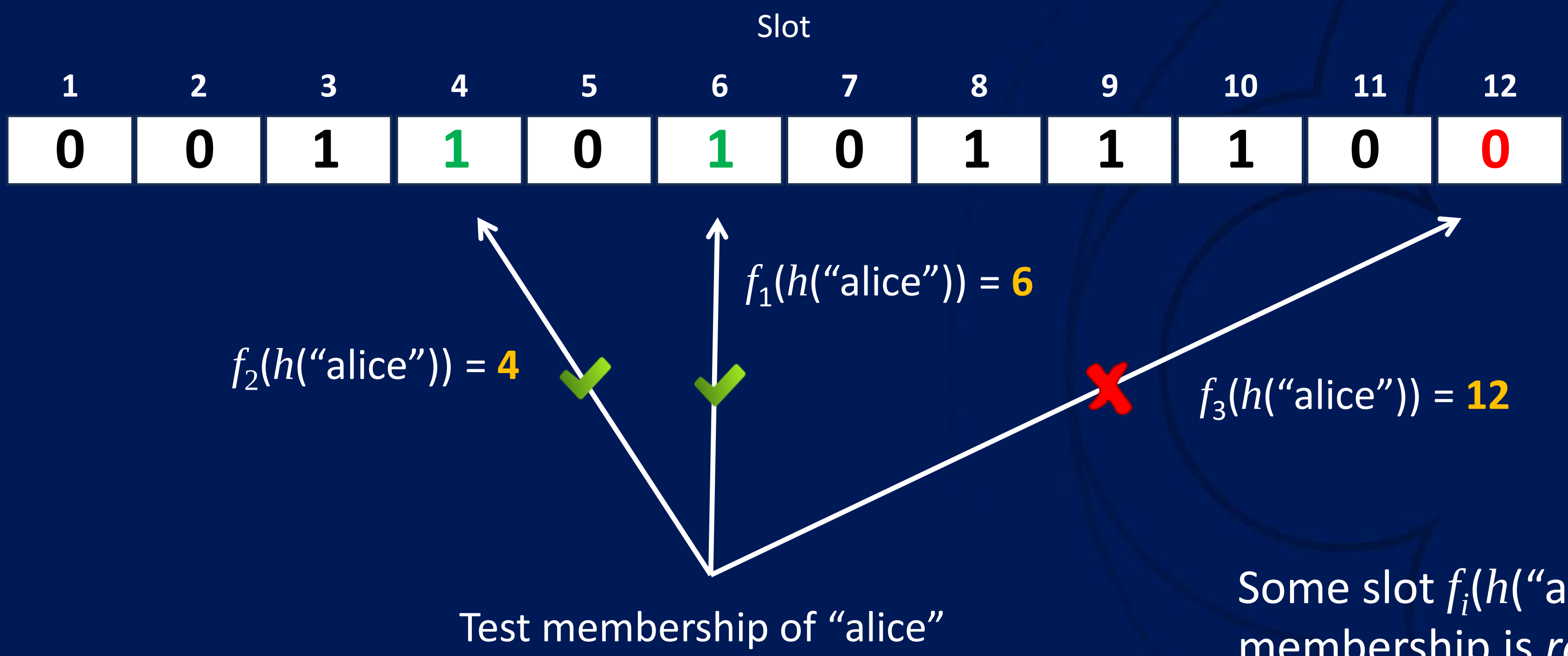


Bloom Filters

(Bloom 1970)

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

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



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

Duke

Bernoulli Honeywords

Bloom Filter

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

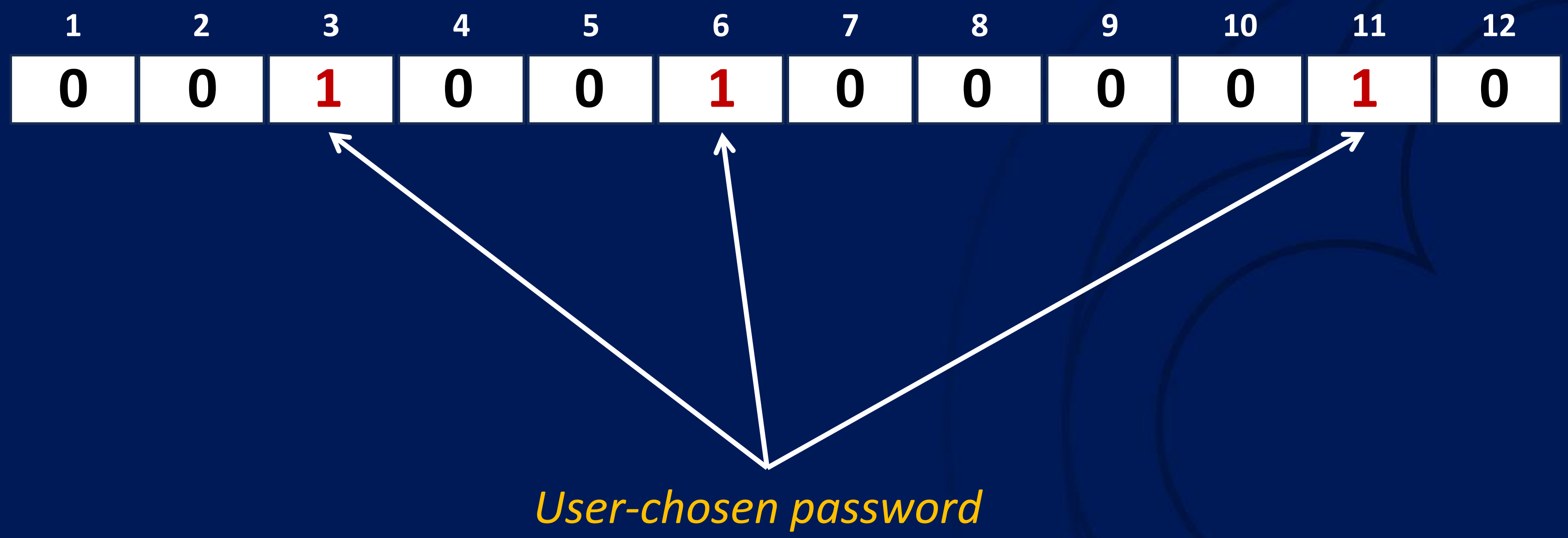
Duke

Bernoulli Honeywords

If integrated with a Honeychecker:

3, 6, 11

Bloom Filter



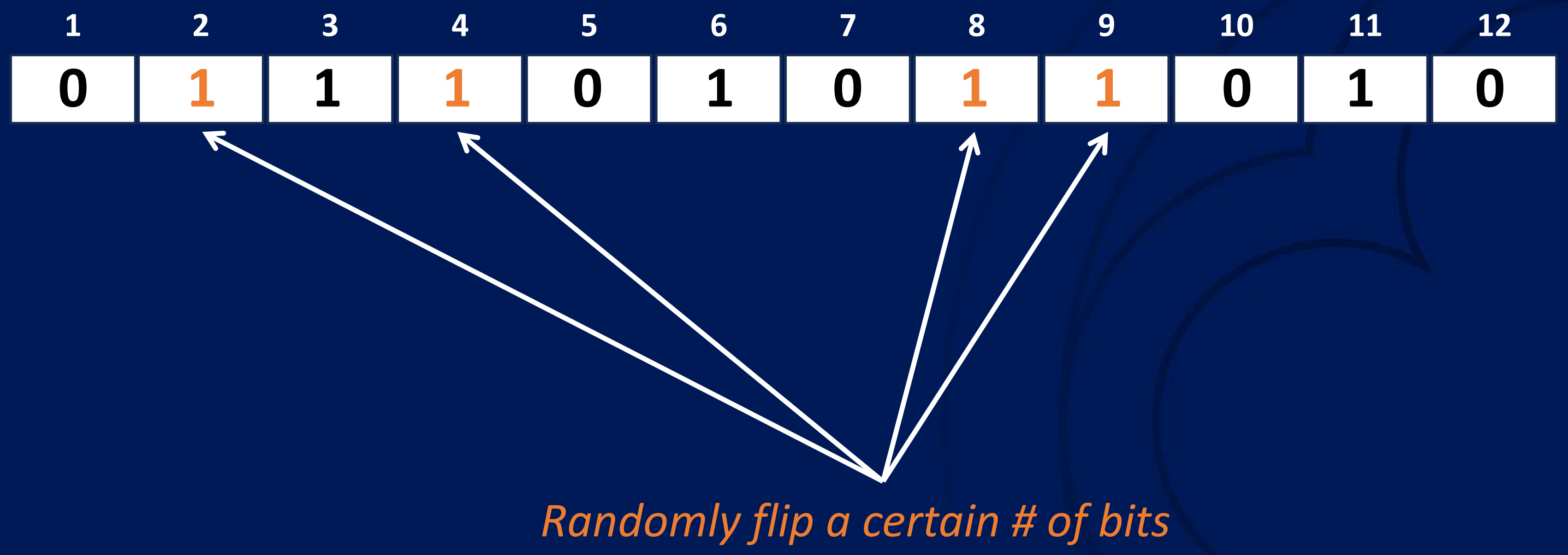
Duke

Bernoulli Honeywords

If integrated with a Honeychecker:

3, 6, 11

Bloom Filter



Bernoulli Honeywords

If integrated with a Honeychecker:

3, 6, 11

Bloom Filter

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

If a submitted password is

- *In the BF & with indices 3, 6, 11 → Successful login*
- *In the BF & with ≥ one index not being 3, 6, or 11 → Breach alarm*
- *Not in the BF → Failed login*

Duke

Bernoulli Honeywords

If integrated with a Honeychecker:

3, 6, 11

Bloom Filter

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

If a submitted password is

- *In the BF & with indices 3, 6, 11 → Successful login*
- *In the BF & with ≥ one index not being 3, 6, or 11 → Breach alarm*
- *Not in the BF --> Failed login*

These passwords are Bernoulli honeywords!

Duke

Can We Analytically Quantify the False Alarm Rate?

Bloom Filter

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

If we generate honeywords heuristically, then we probably cannot.

But for Bernoulli honeywords, we can!

- Recall that each incorrect password in the entire space is randomly chosen as a honeyword according to Bernoulli distribution
- A false alarm attacker can do no better than “blindly” submitting a password hoping it to be a honeyword, which is following the same Bernoulli distribution

Duke

What about True Alarm Rates?

Duke

What about True Alarm Rates?

Breach attacker's view (toy example):

Account #1	Account #2	Account #3
BF: 0101010110 1010100101 …	BF: 1101000110 1000100101 …	BF: 1000010111 1010011011 …

Duke

What about True Alarm Rates?

Breach attacker’s view (toy example):

Account #1	Account #2	Account #3
1. 123456	1. j3dP10	1. mickey
2. qwerty	2. 4mf1k;	2. Simba
3. pwd123	3. As39!2	3. Yoda!!
…	…	…

Passwords in the BF ranked by likelihood of being the user-chosen password from the attacker’s view

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

Account #1

1. 123456
2. qwerty
3. pwd123

…

Account #2

1. j3dP10
2. 4mf1k;
3. As39!2

…

Account #3

1. mickey
2. Simba
3. Yoda!!

…

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

Account #1	Account #3	Account #2
1. 123456	1. mickey	1. j3dP10
2. qwerty	2. Simba	2. 4mf1k;
3. pwd123	3. Yoda!!	3. As39!2
…	…	…

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

The attacker can do no better by trying other passwords than the most likely one (from its view)

Account #1	Account #3	Account #2
1. 123456	1. mickey	1. j3dP10
2. qwerty	2. Simba	2. 4mf1k;
3. pwd123	3. Yoda!!	3. As39!2
…	…	…

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

Account #1	Account #3	Account #2
1. 123456	1. mickey	1. j3dP10
2. qwerty	2. Simba	2. 4mf1k;
3. pwd123	3. Yoda!!	3. As39!2
…	…	…

User-chosen

Honeyword

The attacker starts with the account where it has the most confidence in attacking until it hits an account where the most likely password from the attacker's view is a Bernoulli honeyword, which triggers a breach alarm

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

Compromised	*Breach alarm!*	
Account #1	Account #3	Account #2
1. 123456	1. mickey	1. j3dP10
2. qwerty	2. Simba	2. 4mf1k;
3. pwd123	3. Yoda!!	3. As39!2
…	…	…

Legend: User-chosen (green) — Honeyword (orange)

The attacker starts with the account where it has the most confidence in attacking until it hits an account where the most likely password from the attacker's view is a Bernoulli honeyword, which triggers a breach alarm

Duke

What about True Alarm Rates?

Attack sequence based on the attacker's knowledge and confidence:

Compromised	*Breach alarm!*	
Account #1	Account #3	Account #2
1. 123456	1. mickey	1. j3dP10
2. qwerty	2. Simba	2. 4mf1k;
3. pwd123	3. Yoda!!	3. As39!2
…	…	…

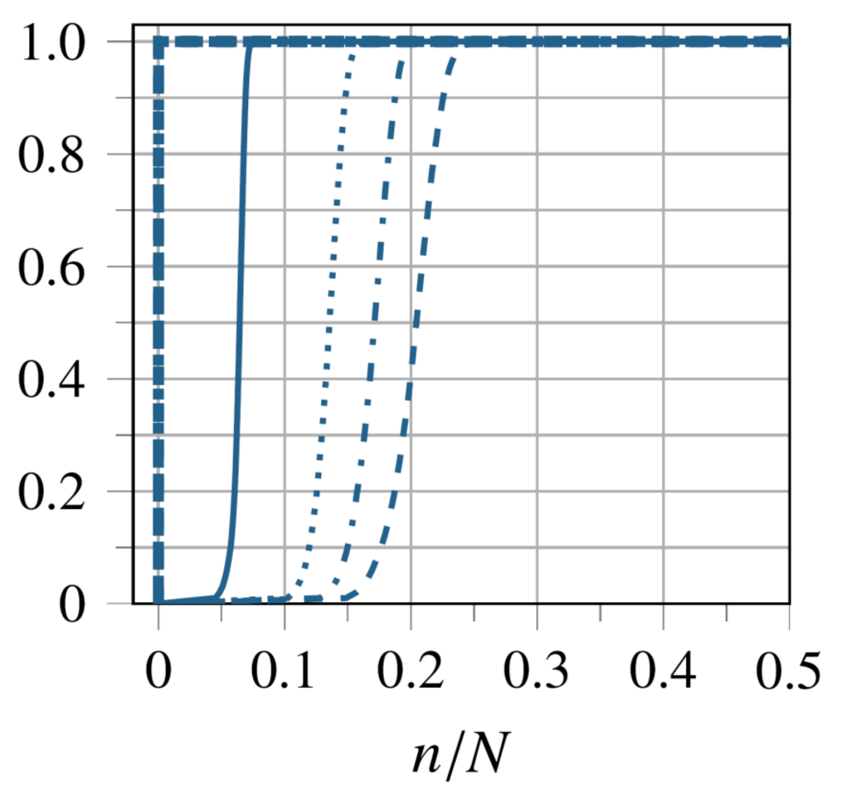
User-chosen

Honeyword

The overall true alarm rate depends on the number of such "vulnerable" accounts where the most likely password in the BF is not a honeyword, which is determined by **1) User password strength and 2) attacker knowledge**.

Duke

Estimates of True Alarm Rate



- Representative true alarm rate 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

Stuffing Honeywords to Avoid Detection

alice@gmail.com:

password1
password2
password3
password4

Site A

alice@gmail.com:

password2

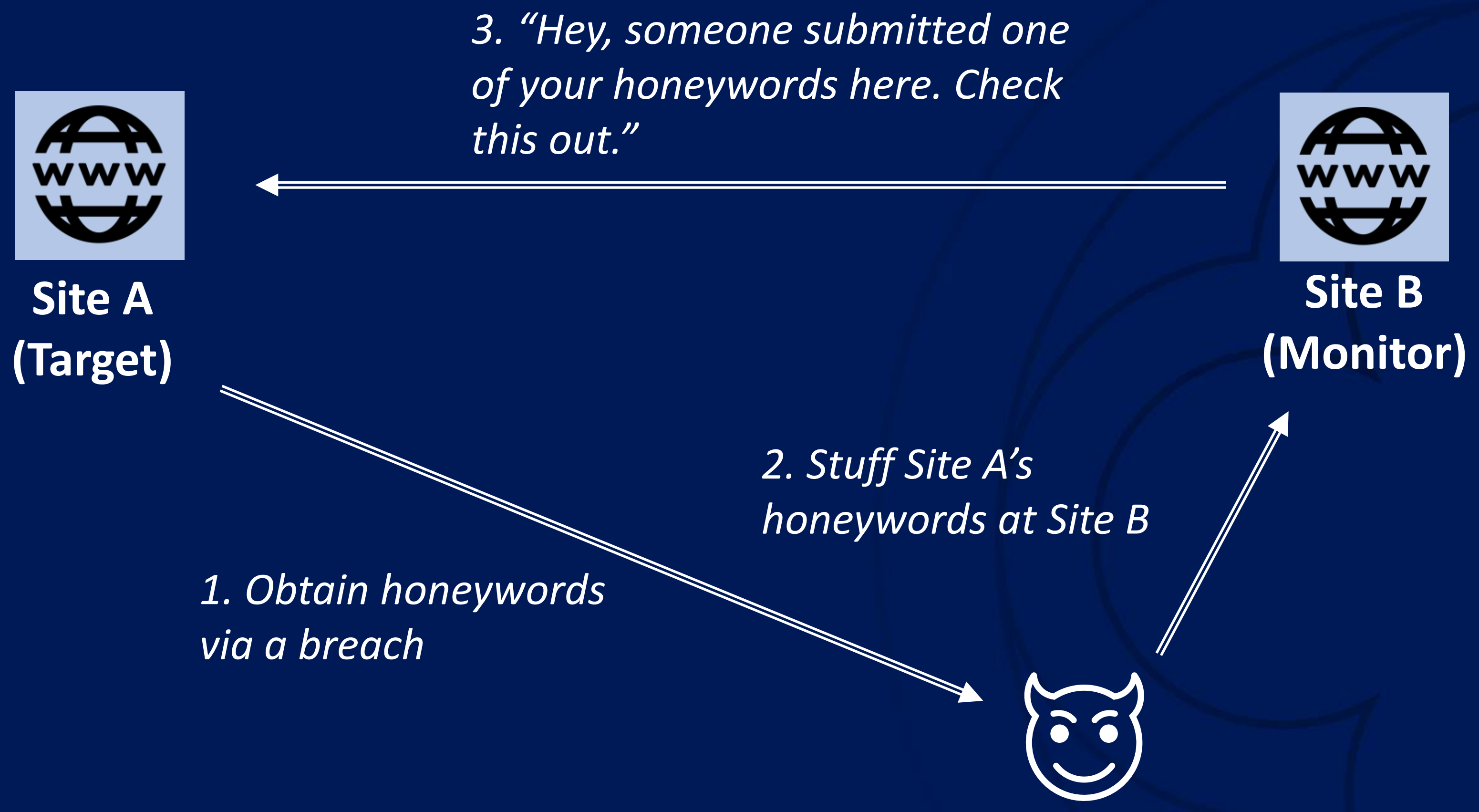
Site B

Duke

Stuffing Honeywords to Avoid Detection

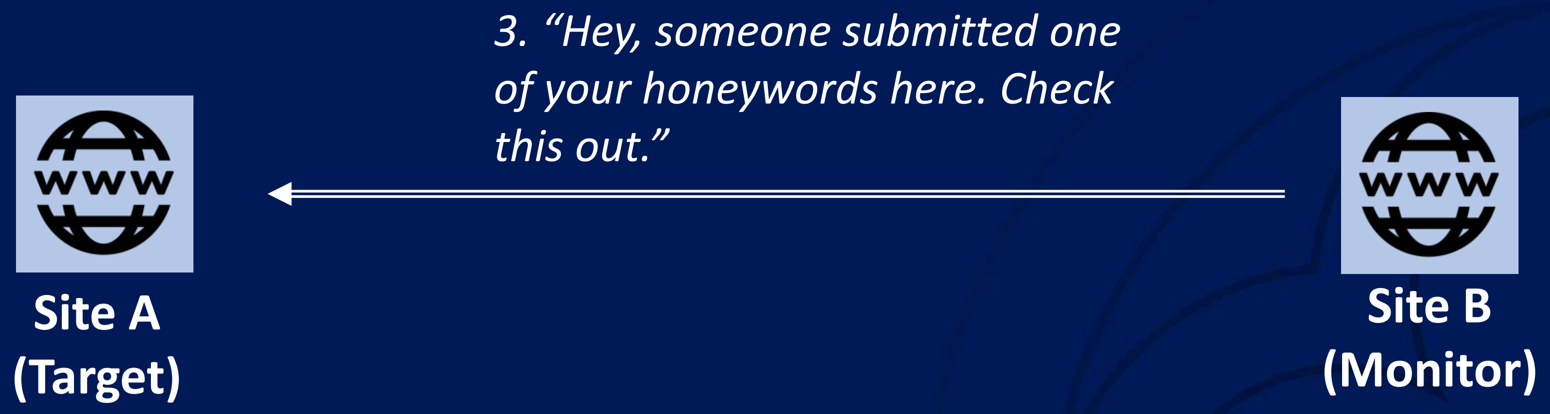
Duke

Detecting Remotely Stuffed Honeywords



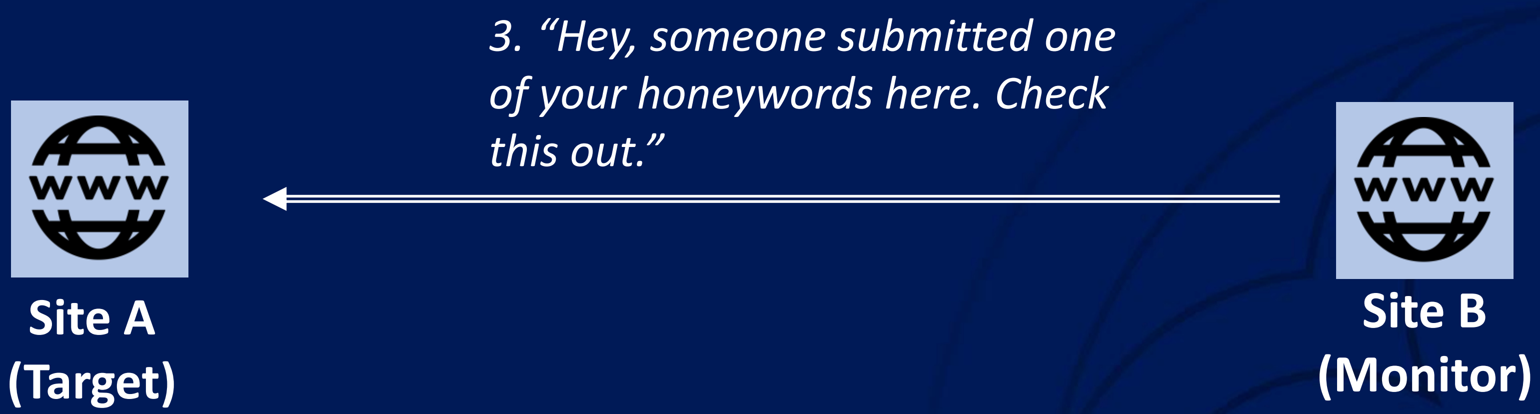
Duke

Detecting Remotely Stuffed Honeywords



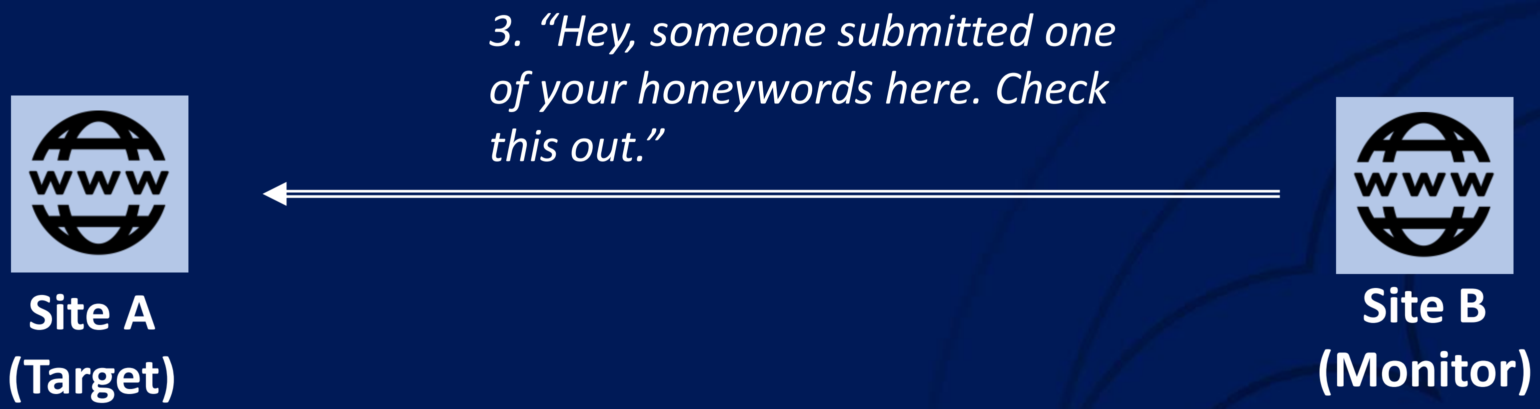
Duke

Detecting Remotely Stuffed Honeywords



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

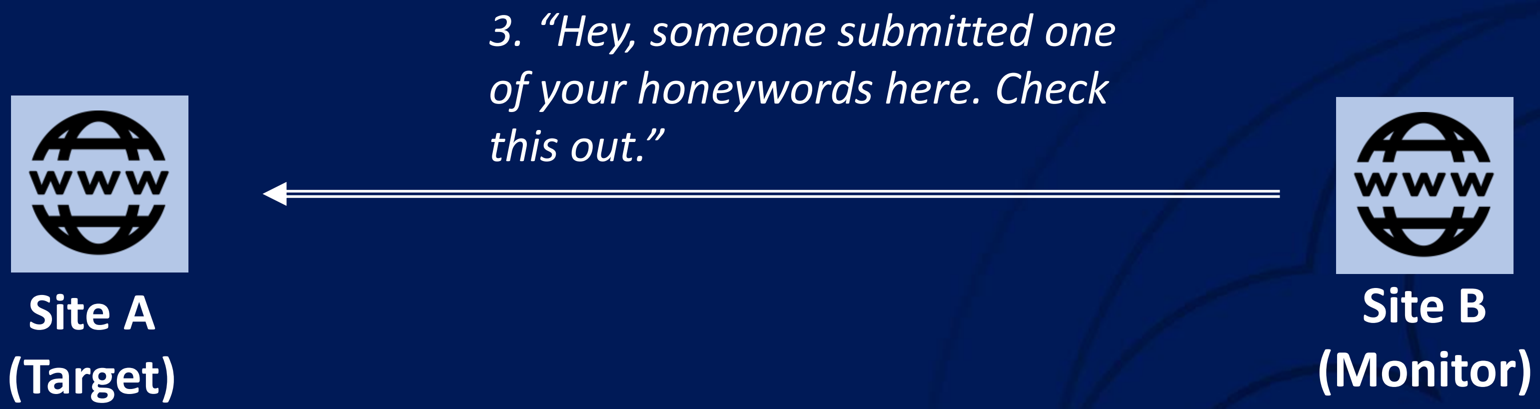
Detecting Remotely Stuffed Honeywords



- 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**

Duke

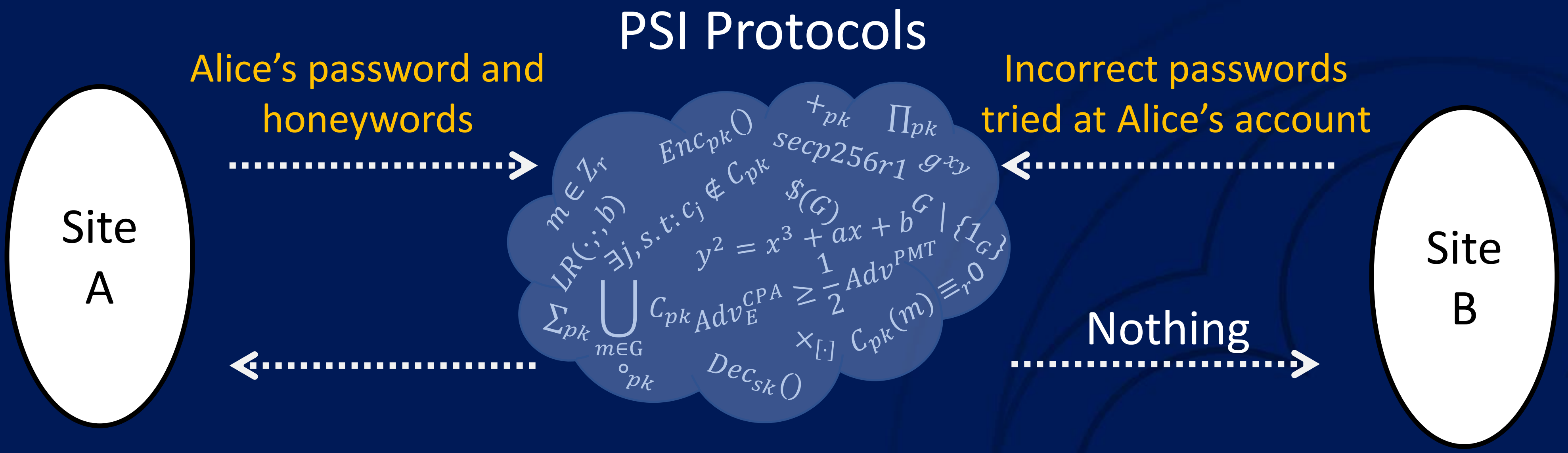
Detecting Remotely Stuffed Honeywords



- 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
- **Should not allow the monitor to trigger a false detection if no breach has happened to Target**

Duke

PSI for Password Database Breach Detection



Needed information:

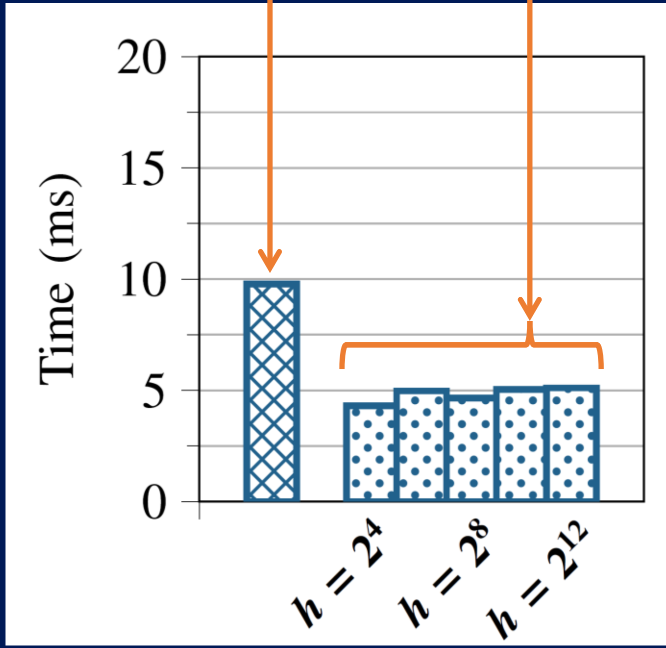
- Set intersection including >= 1 honeyword: password database breach

Response Generation Costs (Frequent)

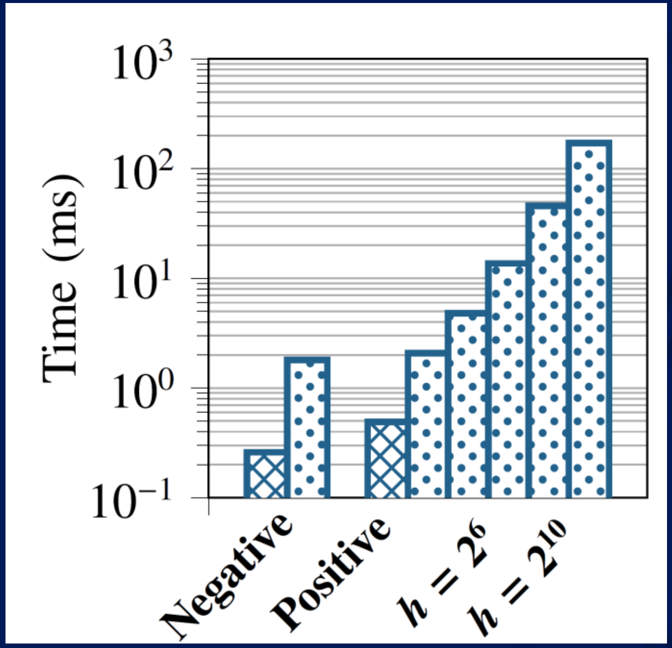
Ours

Cuckoo (WR21)

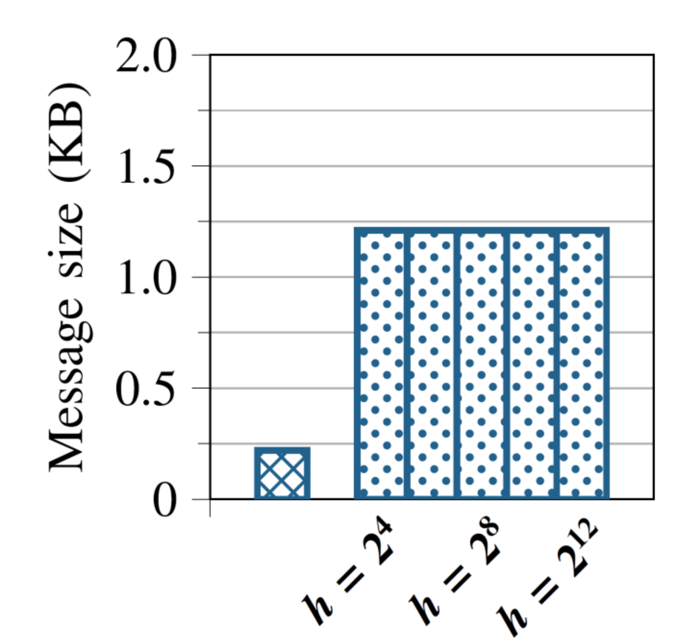
Target and monitor each execute on a single 2.5GHz vCPU



Response generation by monitor



Response processing by target



Response size

Duke

To Summarize

- Bernoulli honeywords allow for a quantifiably low false alarm rate that is independent of the attacker’s knowledge about a user
- Bernoulli honeywords can be integrated with existing honeyword systems and demonstrates compelling detection efficacy
- Our design accommodates a site monitoring for entry of its honeywords at another site, at an expense lower than the latest related work in several important measures

Duke