Using Amnesia to Detect Credential Database Breaches

Ke Coby Wang

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Michael K. Reiter

Duke UNIVERSITY

Credential Database Breaches

- *Verizon[1] confirmed 3950 database breaches globally between Nov. 2018 and Oct. 2019.*

[1] Verizon, "*2020 data breach investigations report*"

Credential Database Breaches

- Verizon[1] confirmed 3950 database breaches globally between Nov. 2018 and Oct. 2019.
  - *60% of these breaches leaked credentials.*

[1] Verizon, "*2020 data breach investigations report*"

Credential Database Breaches

- Verizon[1] confirmed 3950 database breaches globally between Nov. 2018 and Oct. 2019.
  - 60% of these breaches leaked credentials.

- Recent IBM reports[2, 3] with 500+ studied organizations show:

[1] Verizon, "*2020 data breach investigations report*"
[2] IBM Security, "*Cost of a data breach report 2019*"
[3] IBM Security, "*Cost of a data breach report 2020*"

Credential Database Breaches

- Verizon[1] confirmed 3950 database breaches globally between Nov. 2018 and Oct. 2019.
  - 60% of these breaches leaked credentials.

- Recent IBM reports[2, 3] with 500+ studied organizations show:
  - *The likelihood an organization will experience a data breach in the next two years is ~30% [2].*

[1] Verizon, "*2020 data breach investigations report*"
[2] IBM Security, "*Cost of a data breach report 2019*"
[3] IBM Security, "*Cost of a data breach report 2020*"

Credential Database Breaches

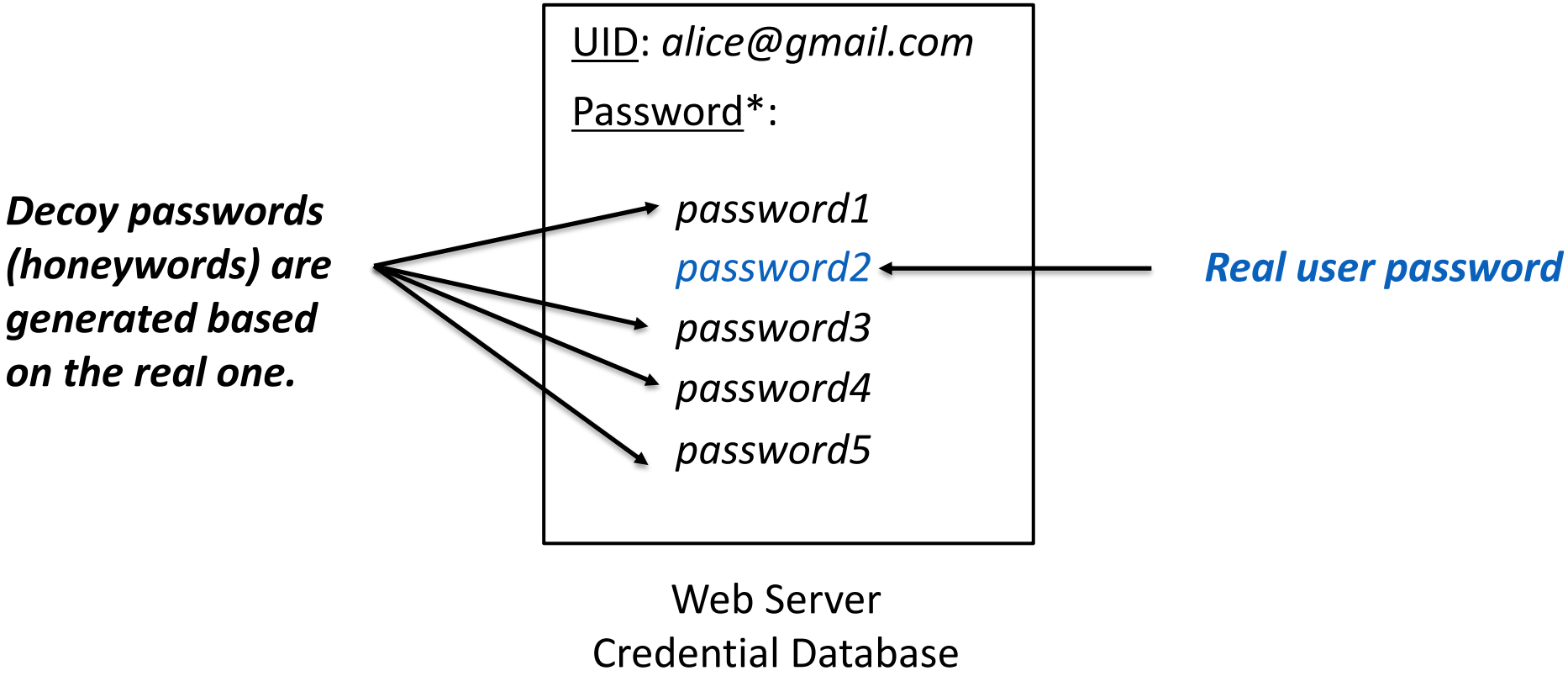
- Verizon[1] confirmed 3950 database breaches globally between Nov. 2018 and Oct. 2019.
  - 60% of these breaches leaked credentials.

- Recent IBM reports[2, 3] with 500+ studied organizations show:
  - The likelihood an organization will experience a data breach in the next two years is ~30% [2].
  - *It takes on average 207 days to identify a breach[3].*

[1] Verizon, "*2020 data breach investigations report*"
[2] IBM Security, "*Cost of a data breach report 2019*"
[3] IBM Security, "*Cost of a data breach report 2020*"

Honeywords

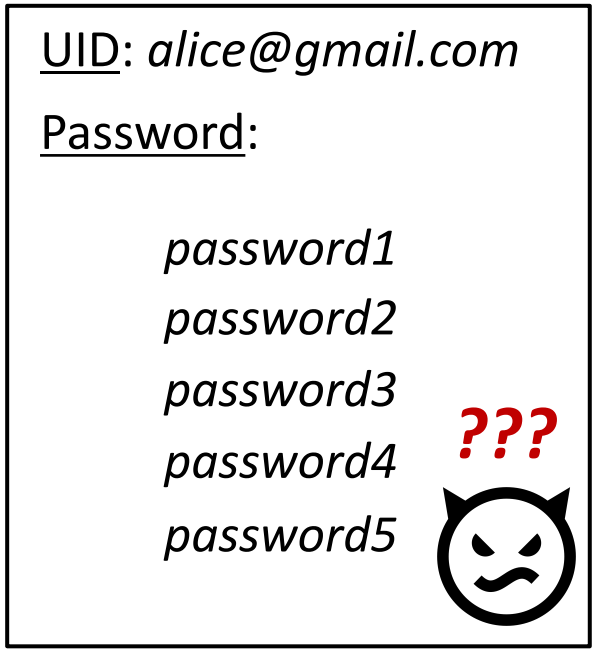
(Juels & Rivest 2013)



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

Honeywords

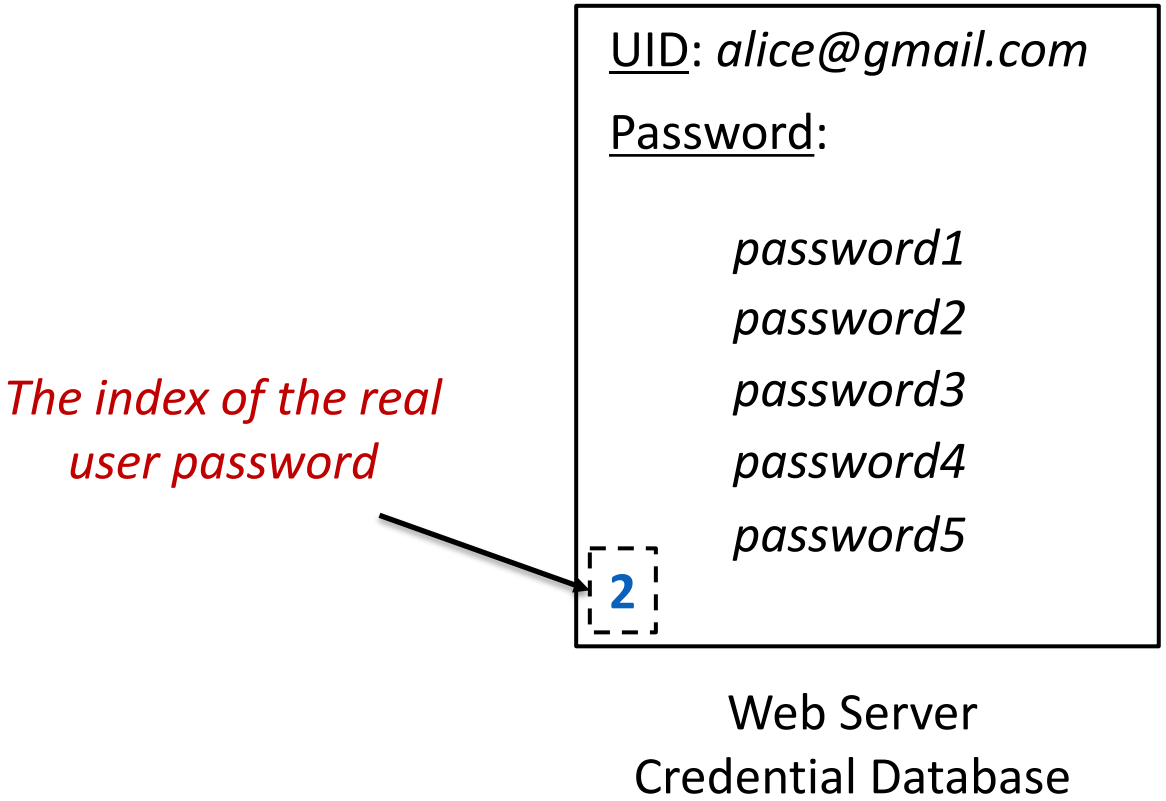
(Juels & Rivest 2013)



Web Server
Credential Database

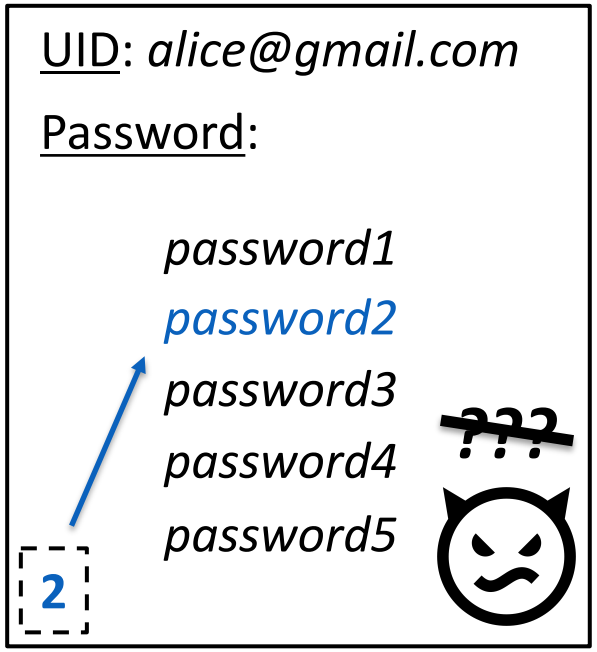
Honeywords

(Juels & Rivest 2013)



Honeywords

(Juels & Rivest 2013)



Web Server
Credential Database

Honeywords

(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

Web Server
Credential Database

Honeychecker

UID: *alice@gmail.com*

Password index:

2

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

Honeywords

(Juels & Rivest 2013)

UID: *alice@gmail.com*

Password:

password1
password2
password3
password4
password5

BREACHED Web Server
Credential Database

Honeychecker

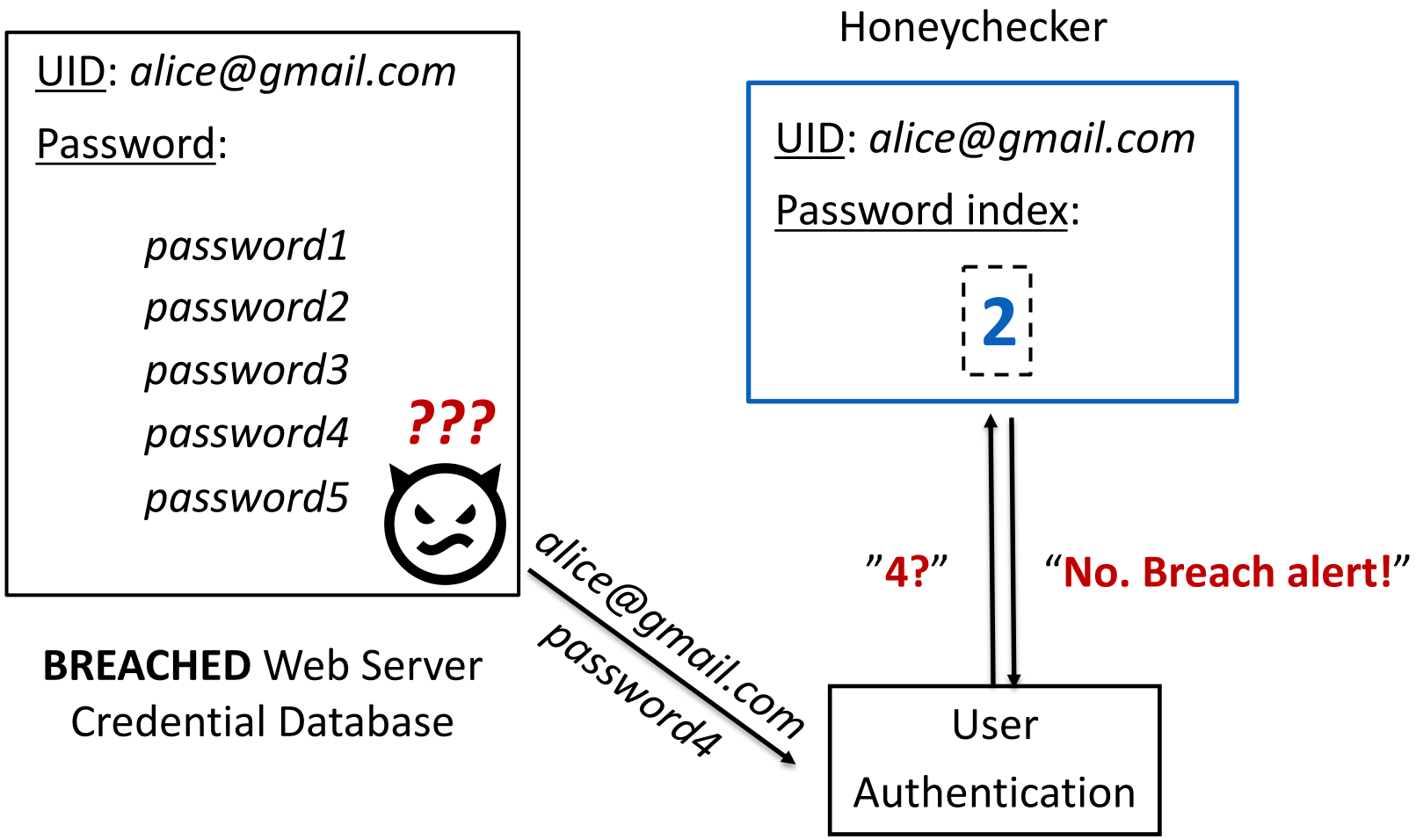
UID: *alice@gmail.com*

Password index:

2

Honeywords

(Juels & Rivest 2013)



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.

Honeywords

(Jules & 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

Honeywords

(Jules & Rivest 2013)

UID: *alice@gmail.com*

Password:

Honeychecker

UID: *alice@gmail.com*

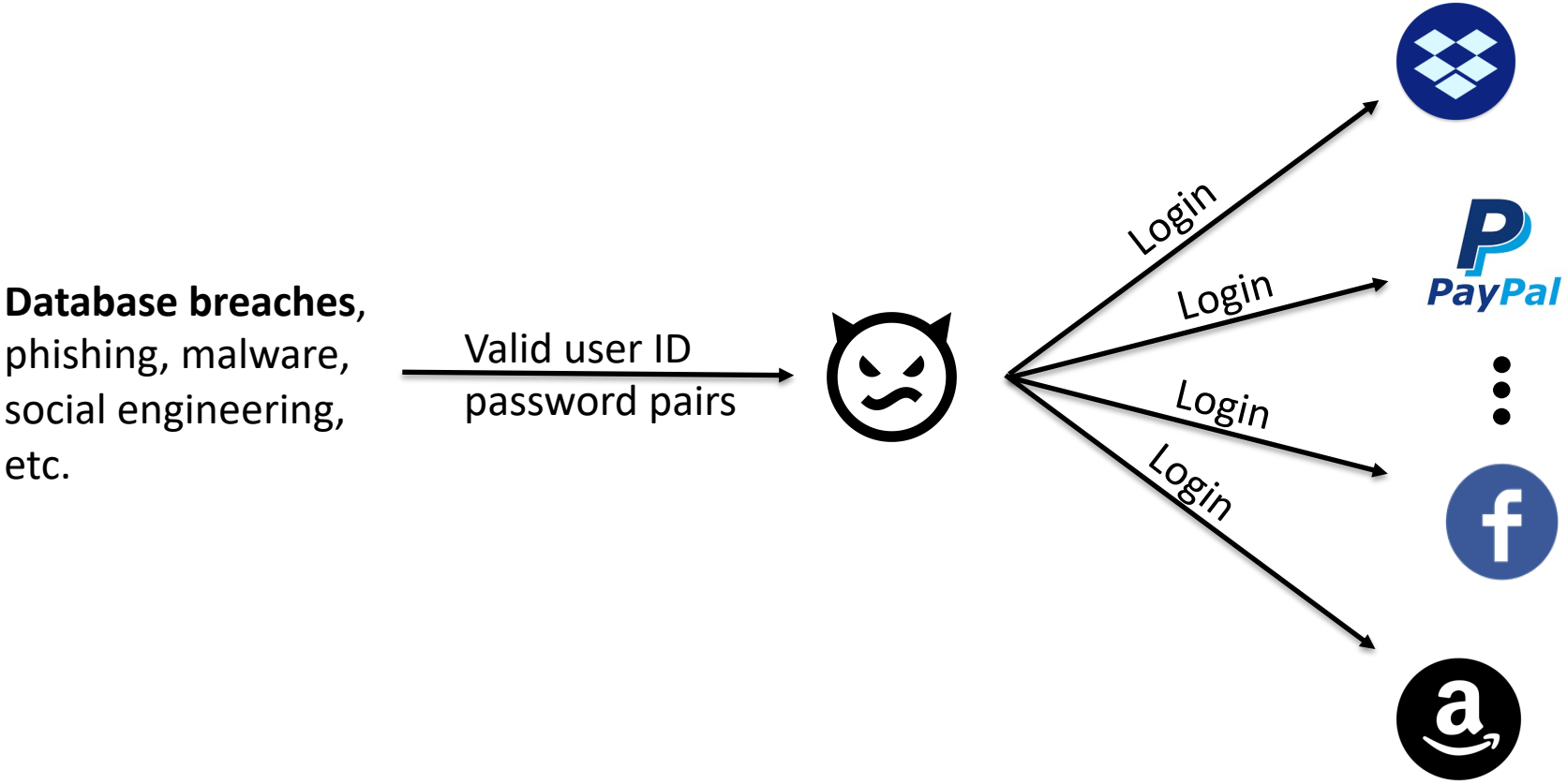
Can we still use honeywords to detect credential database breaches ***without assuming the security of any persistently stored secrets****?*

YES!!

BREACHED Web Server
Credential Database

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

Credential Stuffing



Stuffing Honeywords to Avoid Detection

alice@gmail.com:

password1
password2
password3
password4

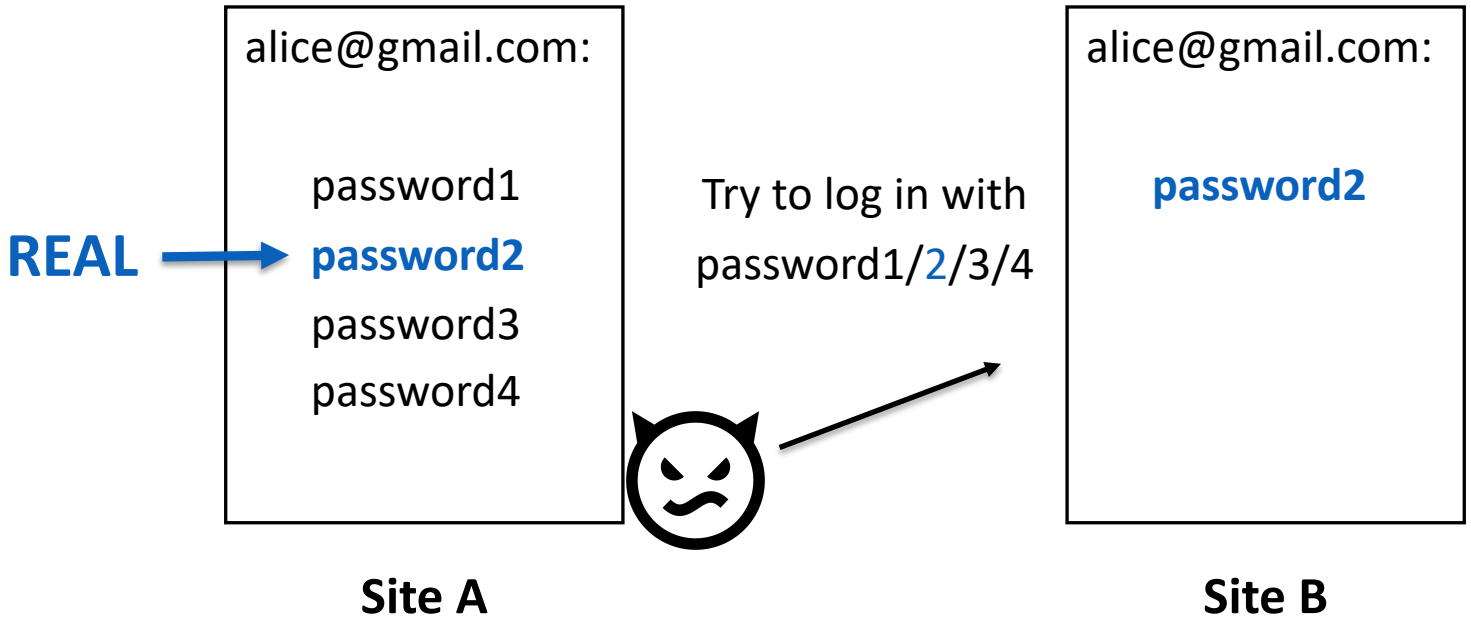
Site A

alice@gmail.com:

password2

Site B

Stuffing Honeywords to Avoid Detection



Two Major Contributions

Two Major Contributions

- *Detecting credential database breaches* ***without relying on any persistent secret state***

Two Major Contributions

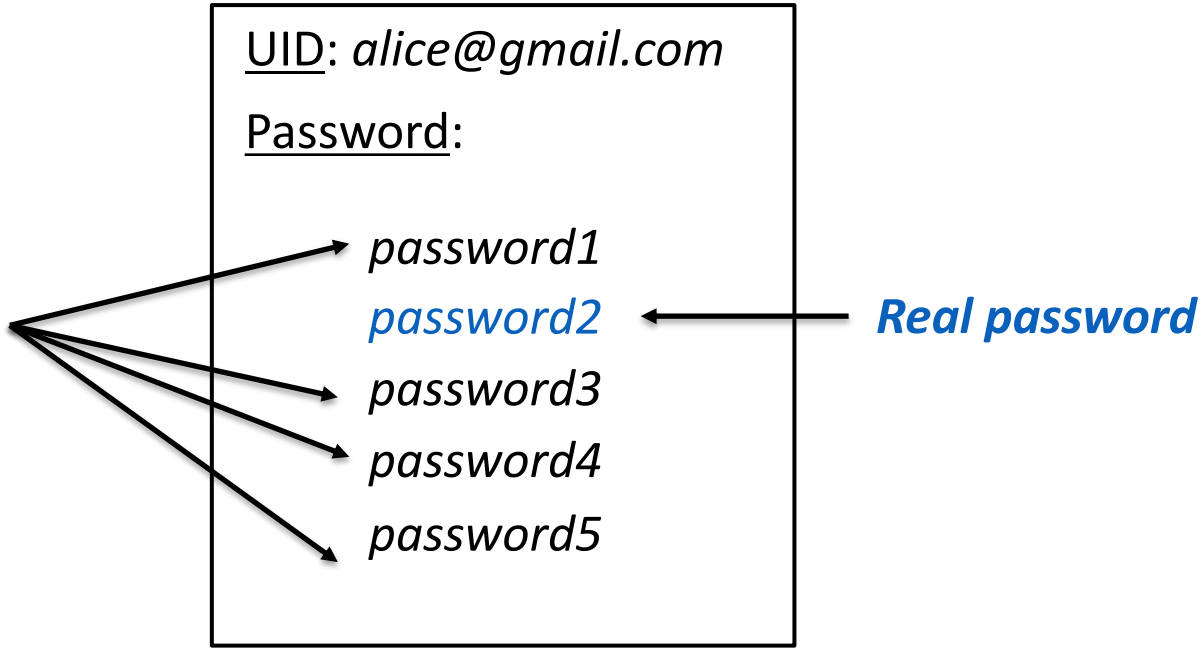
- Detecting credential database breaches without relying on any persistent secret state

- *Detecting credential database breaches* ***against stuffing of honeywords at other websites***

Part I.
Amnesia: Detecting credential database breaches without any persistent secret state

Amnesia :: Decoy Passwords

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



Web Server
Credential Database

Amnesia :: Marking Process

UID: *alice@gmail.com*

Password:

password1
password2*
password3
password4
password5

Web Server
Credential Database

After a successful login:

Amnesia :: Marking Process

UID: *alice@gmail.com*

Password:

password1
password2*
password3
password4
password5

Web Server
Credential Database

After a successful login:

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

Amnesia :: Marking Process

UID: *alice@gmail.com*

Password:

password1*

*password2**

password3

password4*

password5

Web Server
Credential Database

After a successful login:

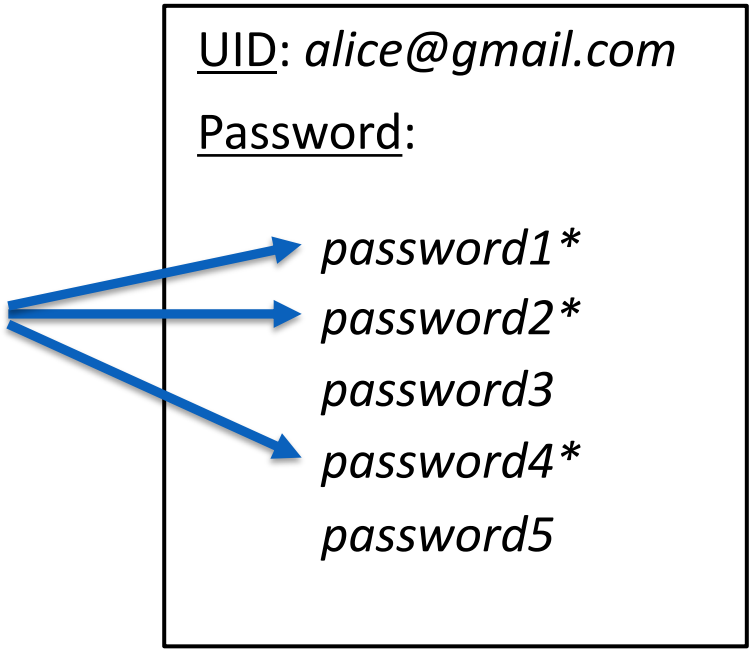
1. Mark the last submitted password
2. ***Mark each of other passwords with a preset probability***

Amnesia :: Detection Process

During a login attempt:

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

Successful login & No breach alert.



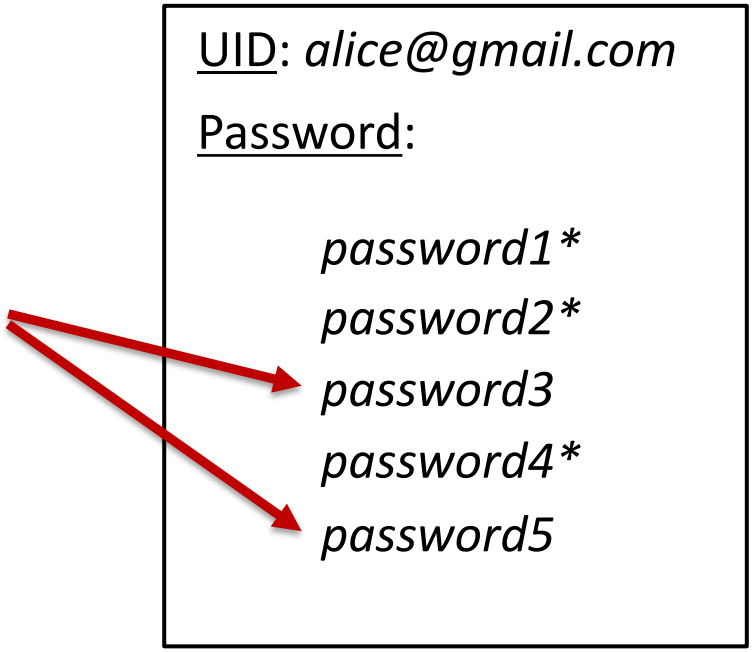
Web Server
Credential Database

Amnesia :: Detection Process

During a login attempt:

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

Breach alert!



Web Server
Credential Database

Amnesia :: User Example

User password: *password2*

Submitted password:
password2

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

The real password remains marked.

Amnesia :: Attacker Example

User password: *password2*

Submitted password:
password4

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

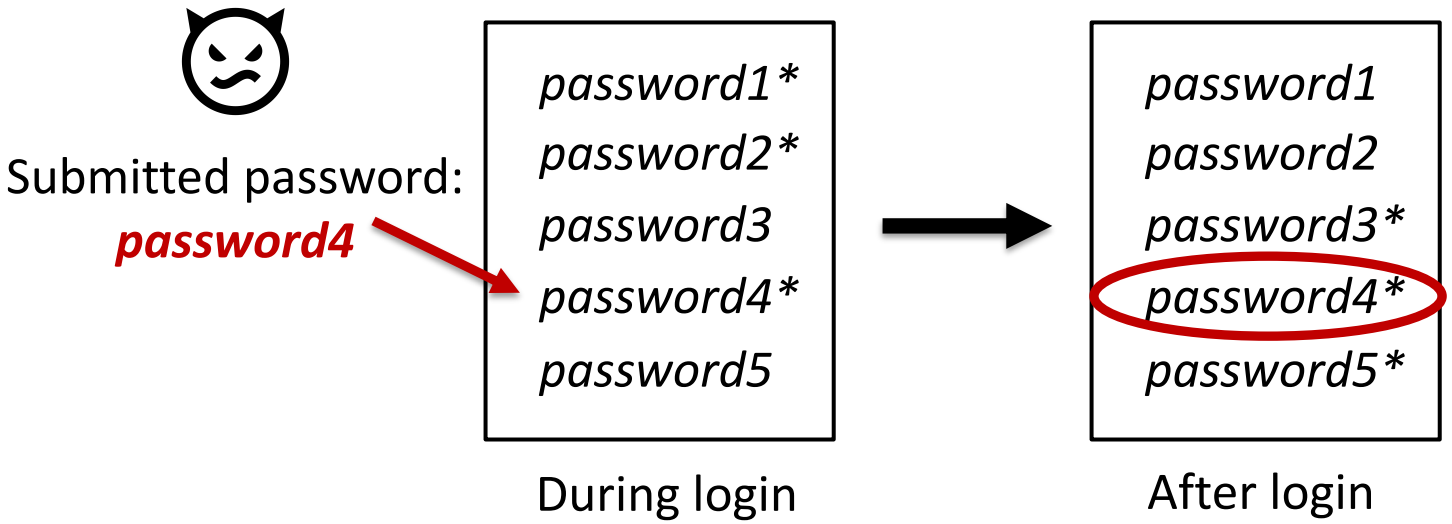
During login

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

After login

Amnesia :: Attacker Example

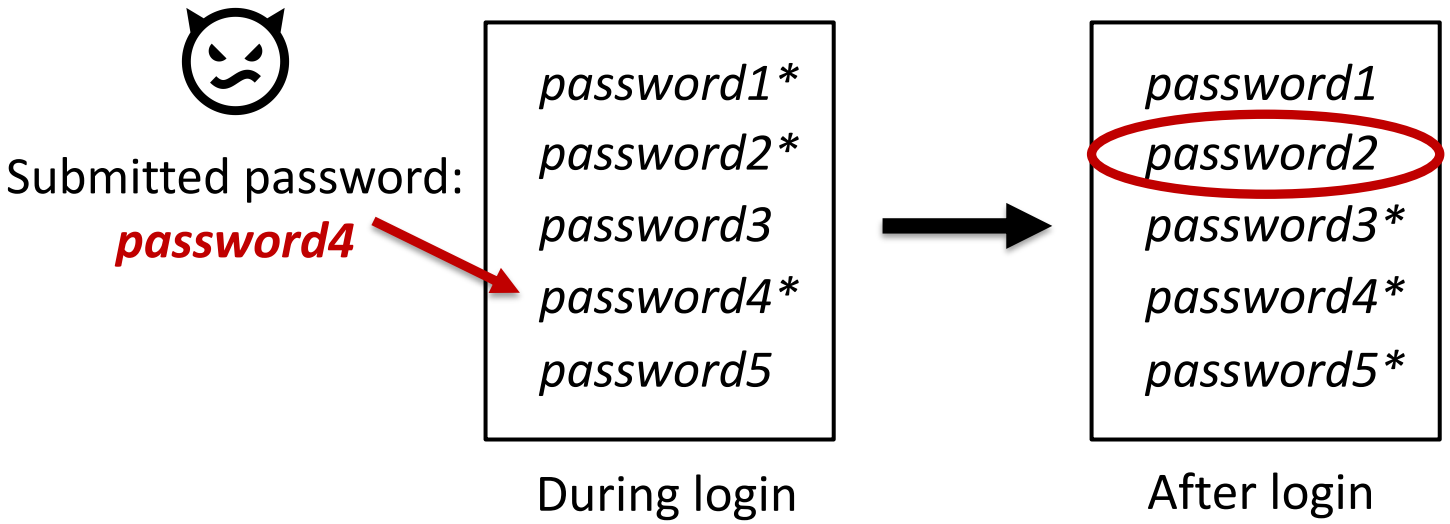
User password: *password2*



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

Amnesia :: Attacker Example

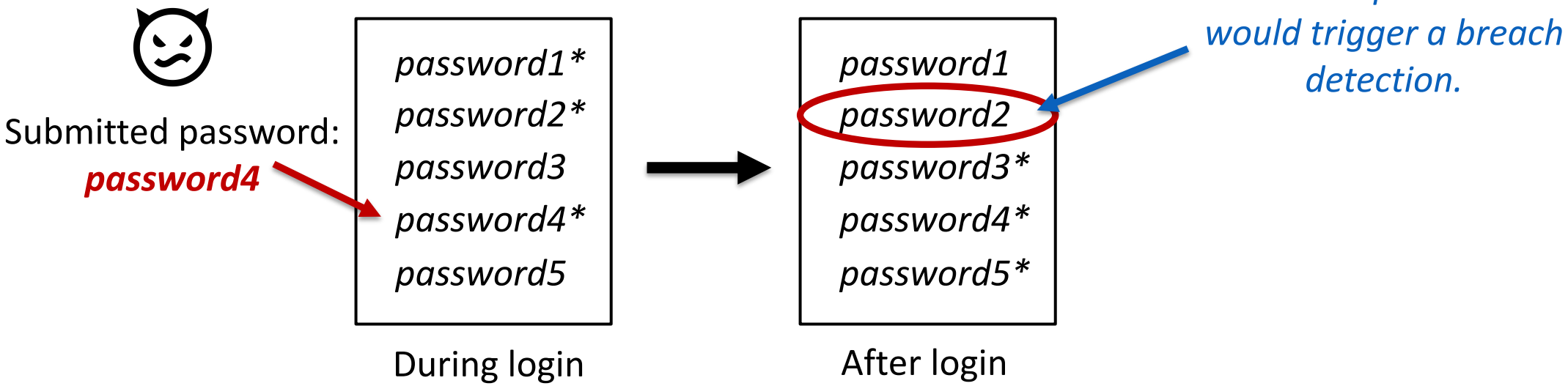
User password: *password2*



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

Amnesia :: Attacker Example

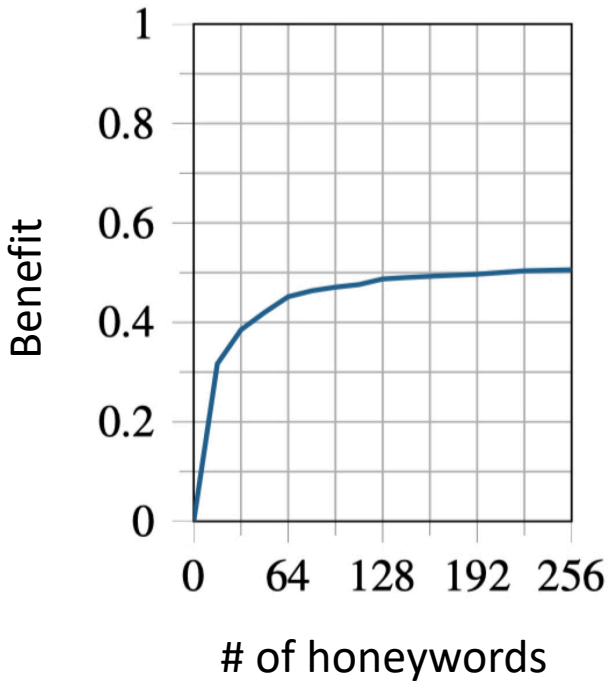
User password: *password2*



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

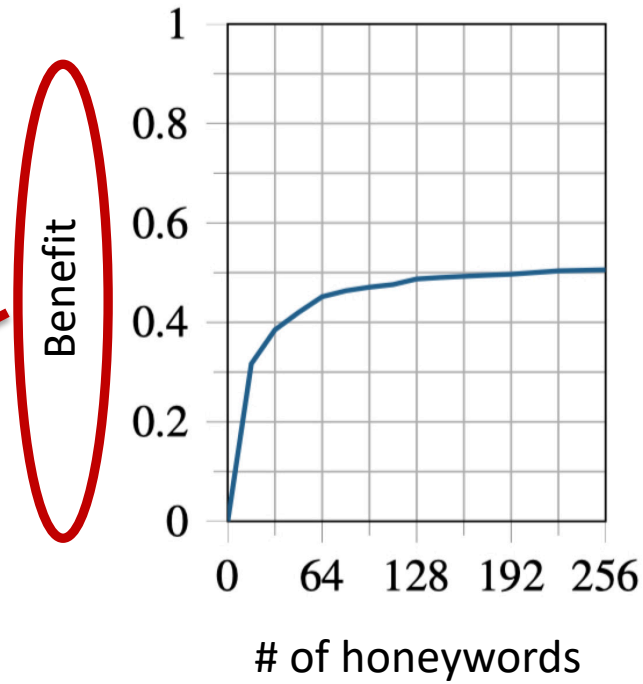
Effectiveness

Effectiveness

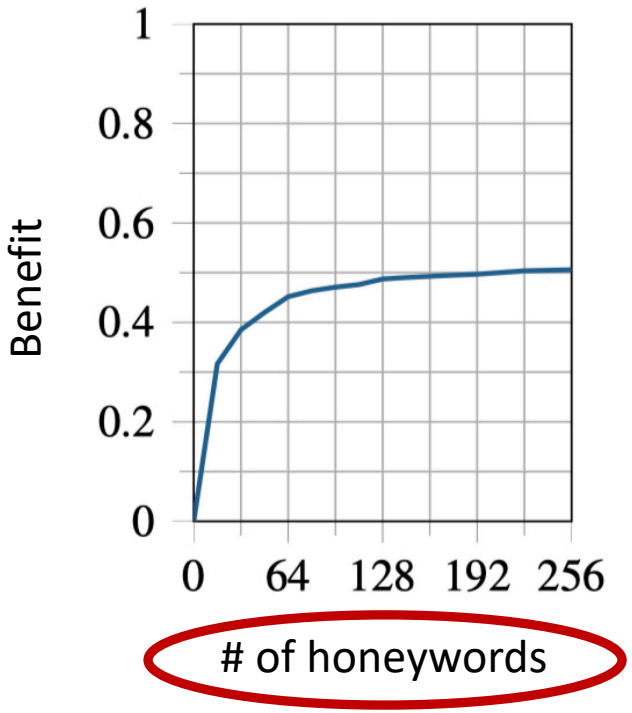


Effectiveness

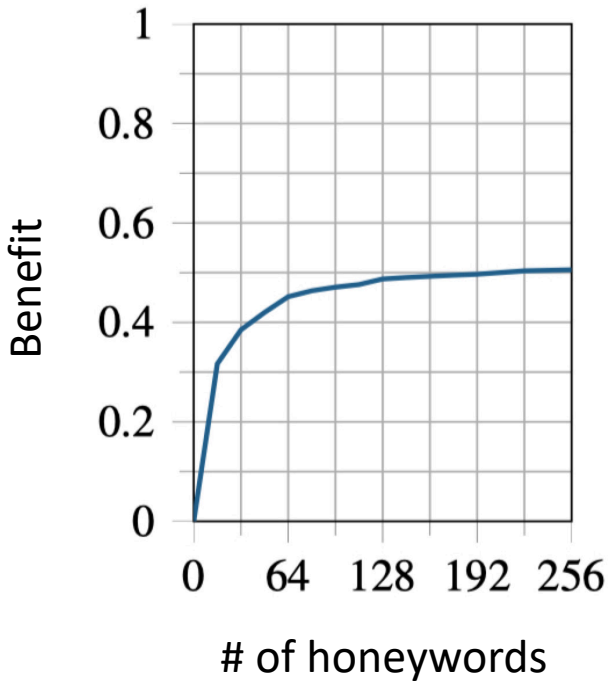
Expected fraction of time that Amnesia deprives the attacker from accessing user accounts before discovery of the breach.



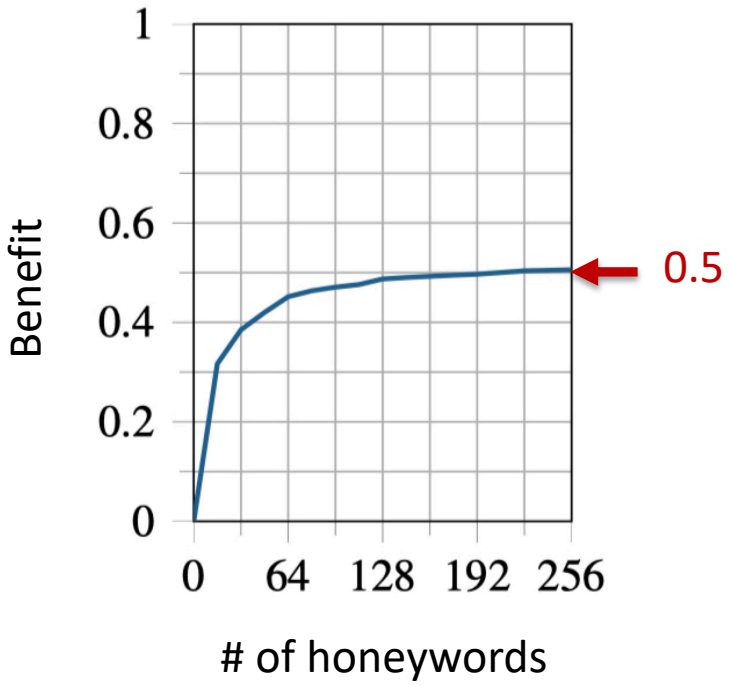
Effectiveness



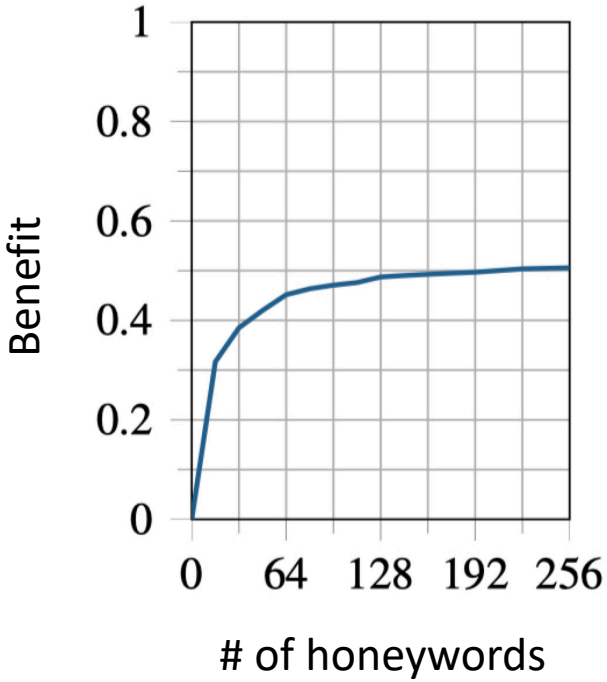
Effectiveness



Effectiveness

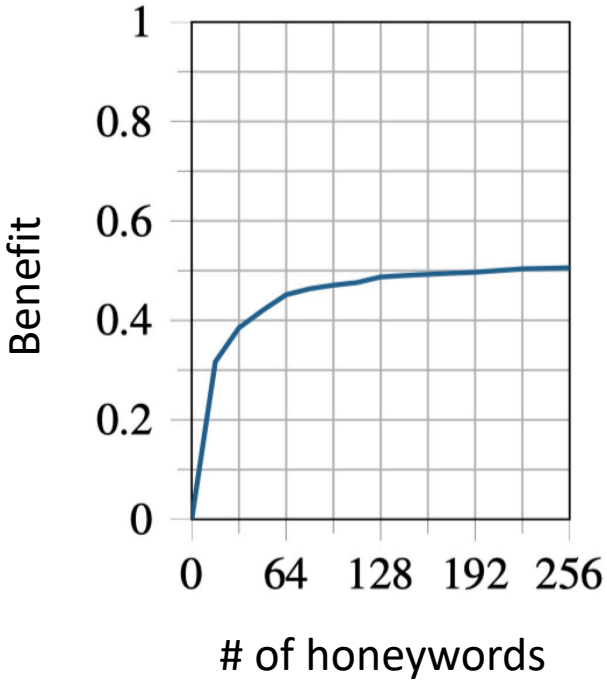


Effectiveness



Conservative evaluation:

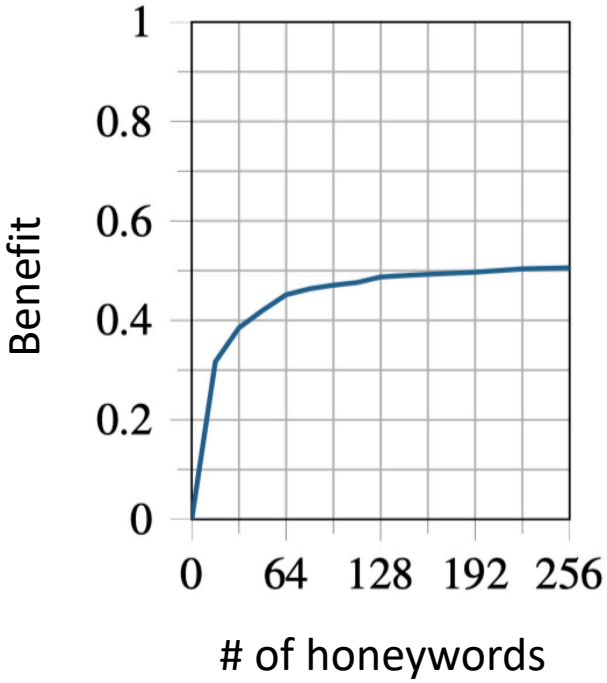
Effectiveness



Conservative evaluation:

- ***The evaluation is for one account only: Attacker wins if not detected after accessing only one account.***

Effectiveness

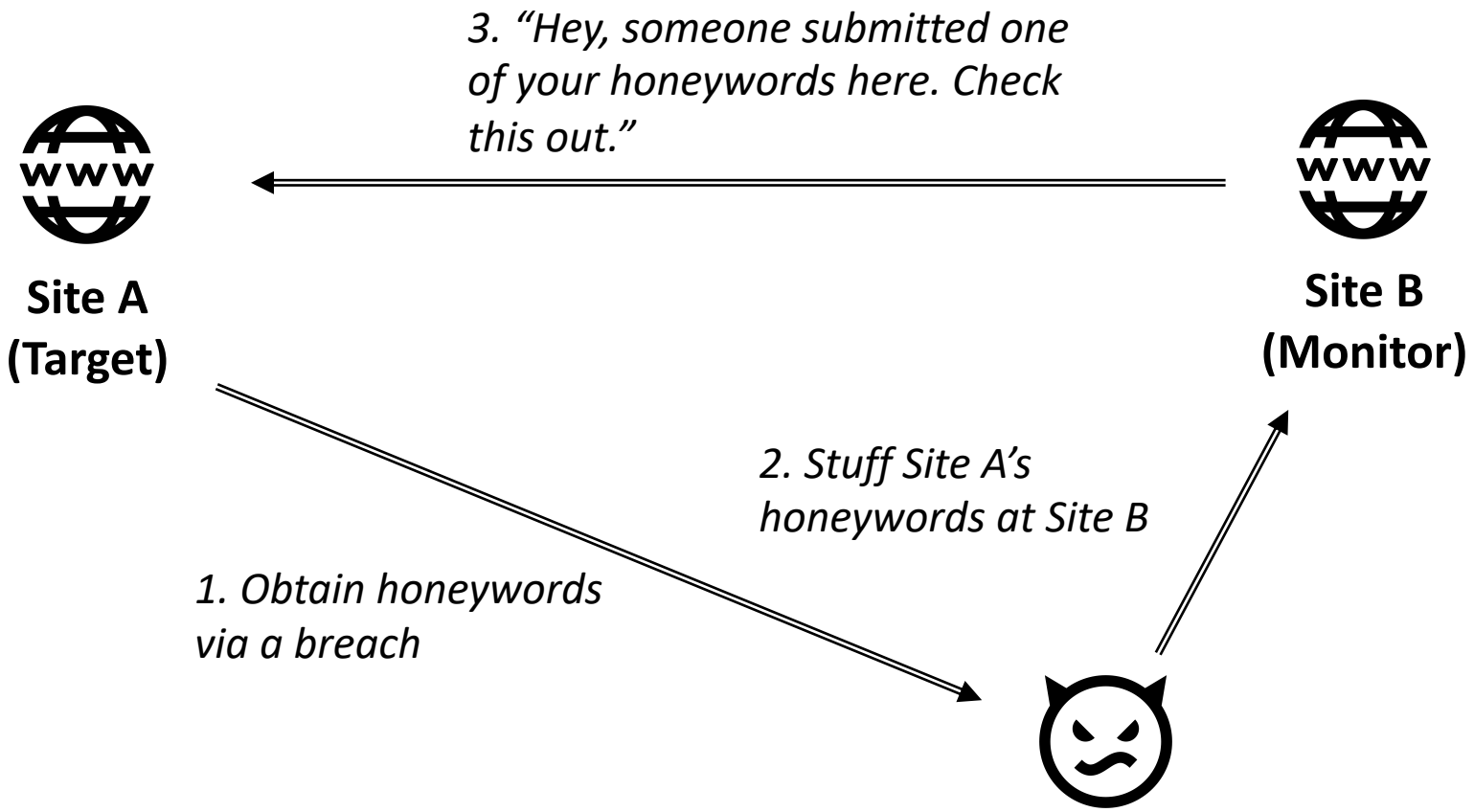


Conservative evaluation:

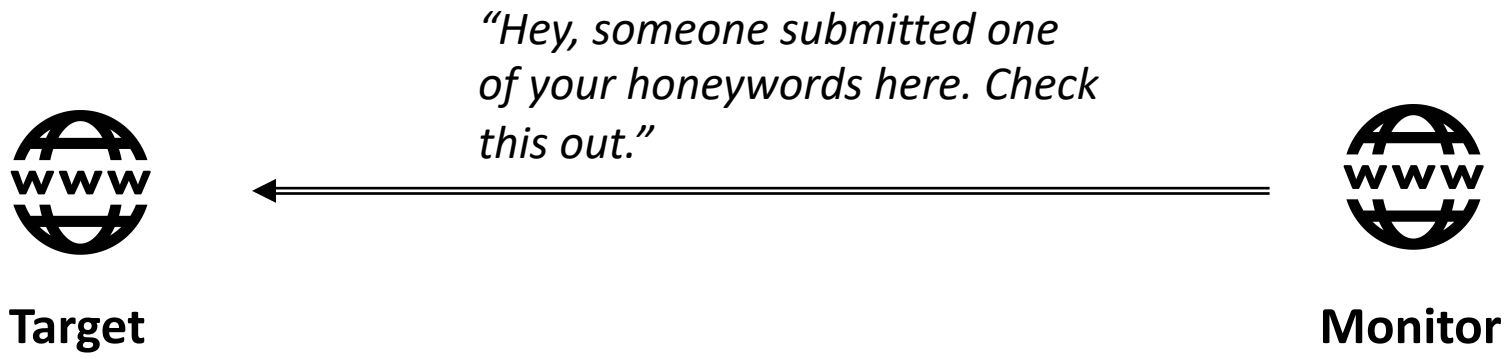
- The evaluation is for one account only: Attacker wins if not detected after accessing only one account.
- ***The model checking computes benefit for the best possible attacker strategy.***

Part II.
Detecting password database breaches against stuffing of honeywords

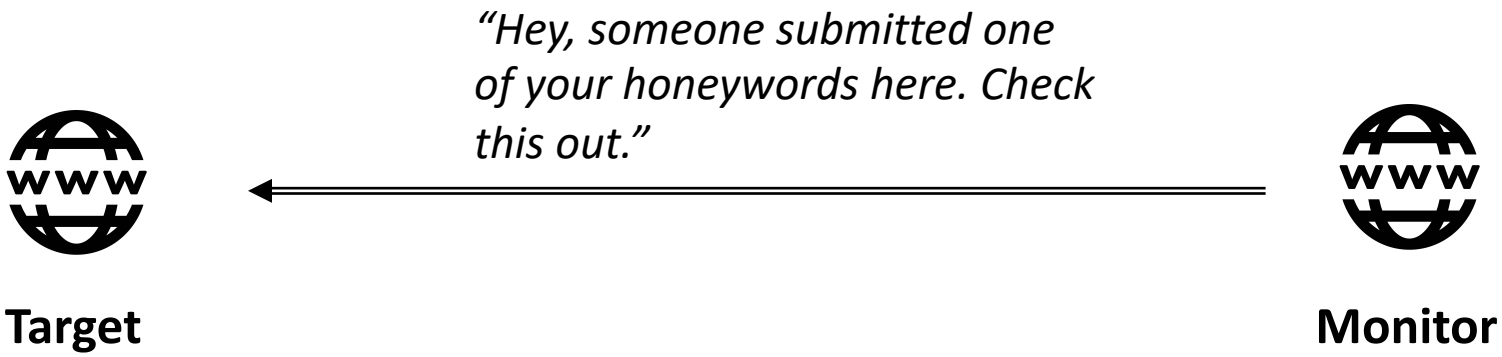
Our Solution :: High-Level Idea



Our Solution :: Security Goals

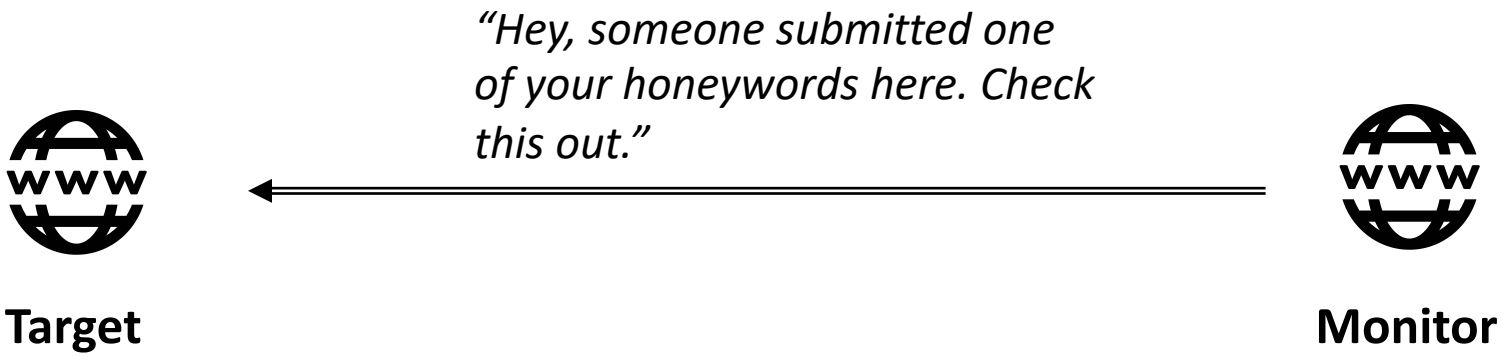


Our Solution :: Security Goals



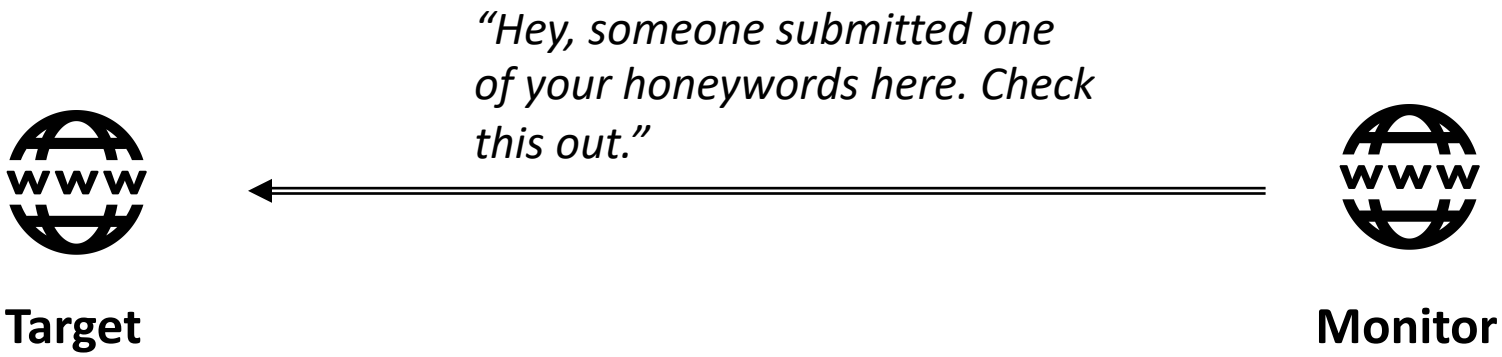
- **Should not leak Target's stored passwords to Monitor**

Our Solution :: Security Goals



- 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**

Our Solution :: Security Goals

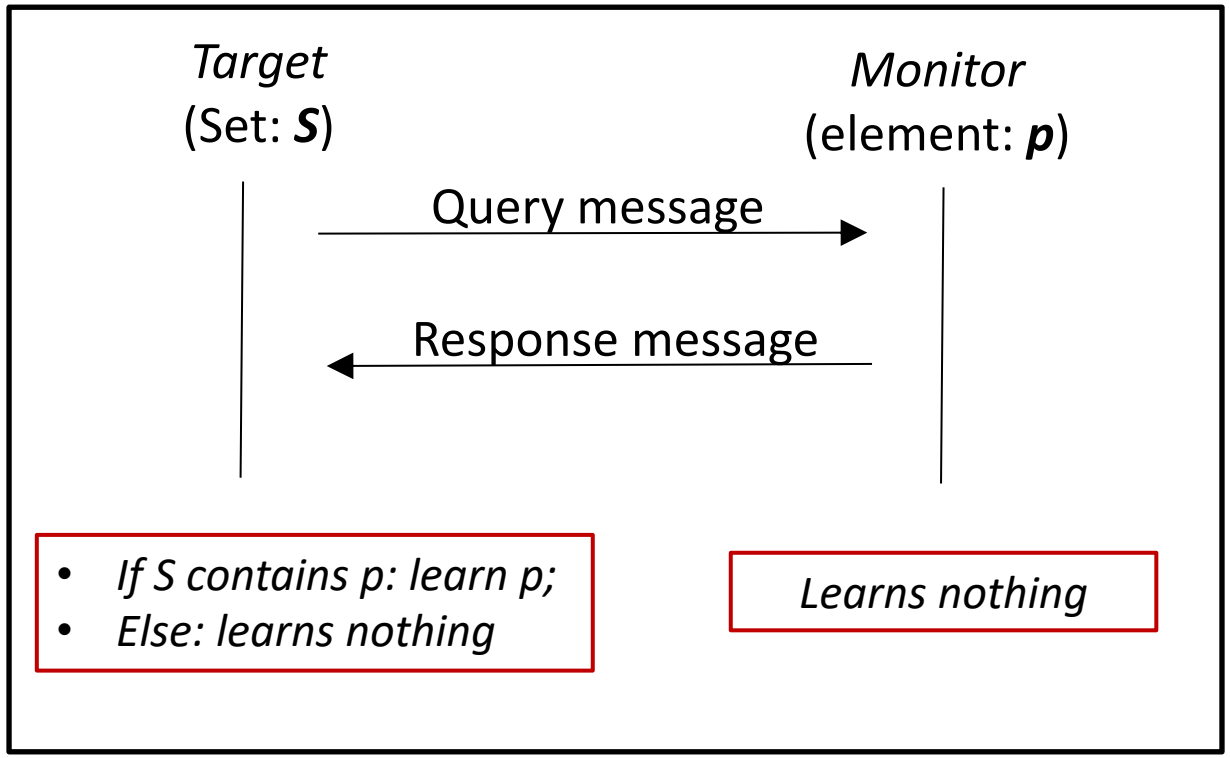


- Should not leak Target's stored passwords to Monitor
- Should not leak the submitted password at Monitor to Target if the password is not one of Target's stored passwords
- **Should not allow the monitor to trigger a false detection if no breach has happened to Target**

Private Containment Retrieval (PCR)

<u>Containment retrieval:</u>

<u>retrieve ***p*** only if ***S*** contains ***p***</u>

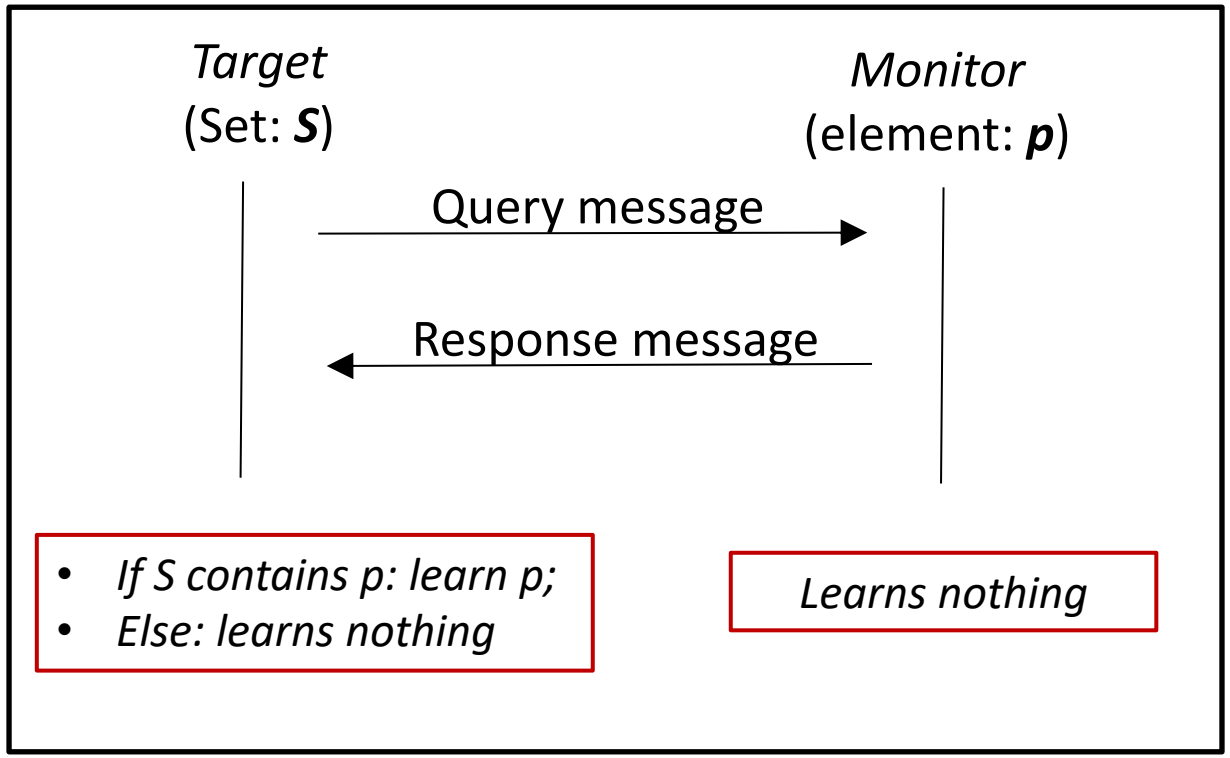


Private Containment Retrieval (PCR)

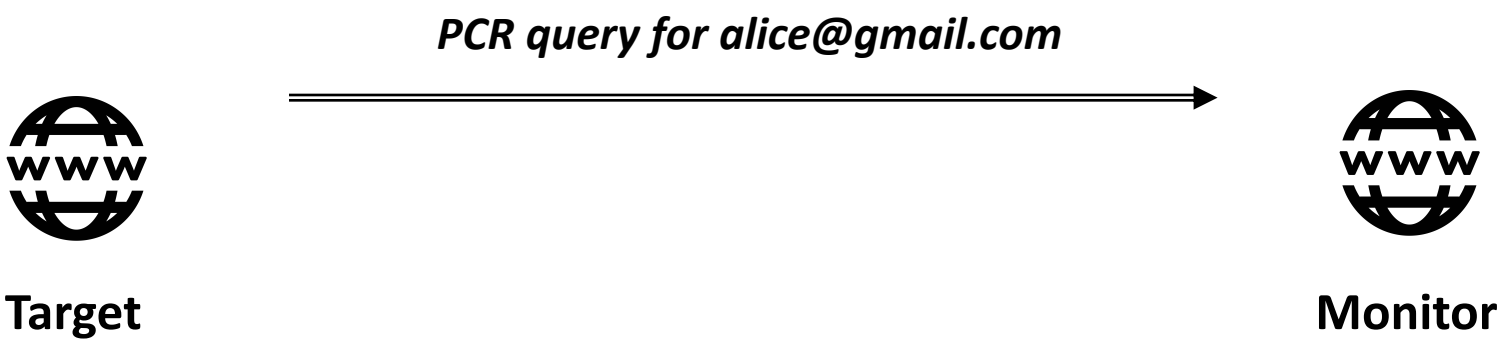
<u>Containment retrieval:</u>

<u>retrieve ***p*** only if ***S*** contains ***p***</u>

Our construction (one round of interaction): Partially homomorphic encryption + Cuckoo filters



Monitoring Stuffing of Honeywords



Target generates a PCR query message on all passwords (real and decoy) and sends it to Monitor.

Monitoring Stuffing of Honeywords

PCR query for alice@gmail.com

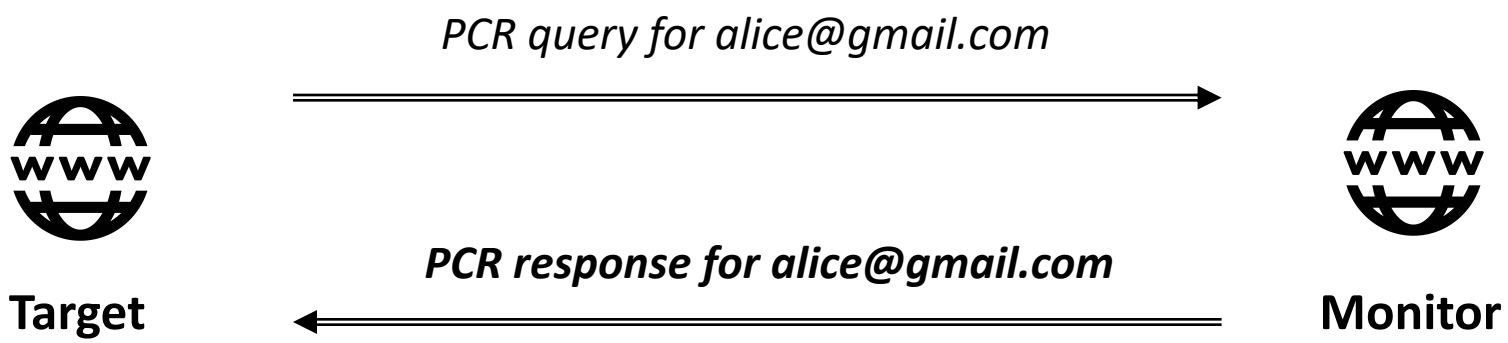
Target

Monitor

PCR response

For each password submitted in a FAILED login attempt at Monitor for alice@gmail.com, Monitor generates a PCR response …

Monitoring Stuffing of Honeywords



… and sends it back to Target

Monitoring Stuffing of Honeywords

Target

Monitor

PCR query for alice@gmail.com →

← *PCR response for alice@gmail.com*

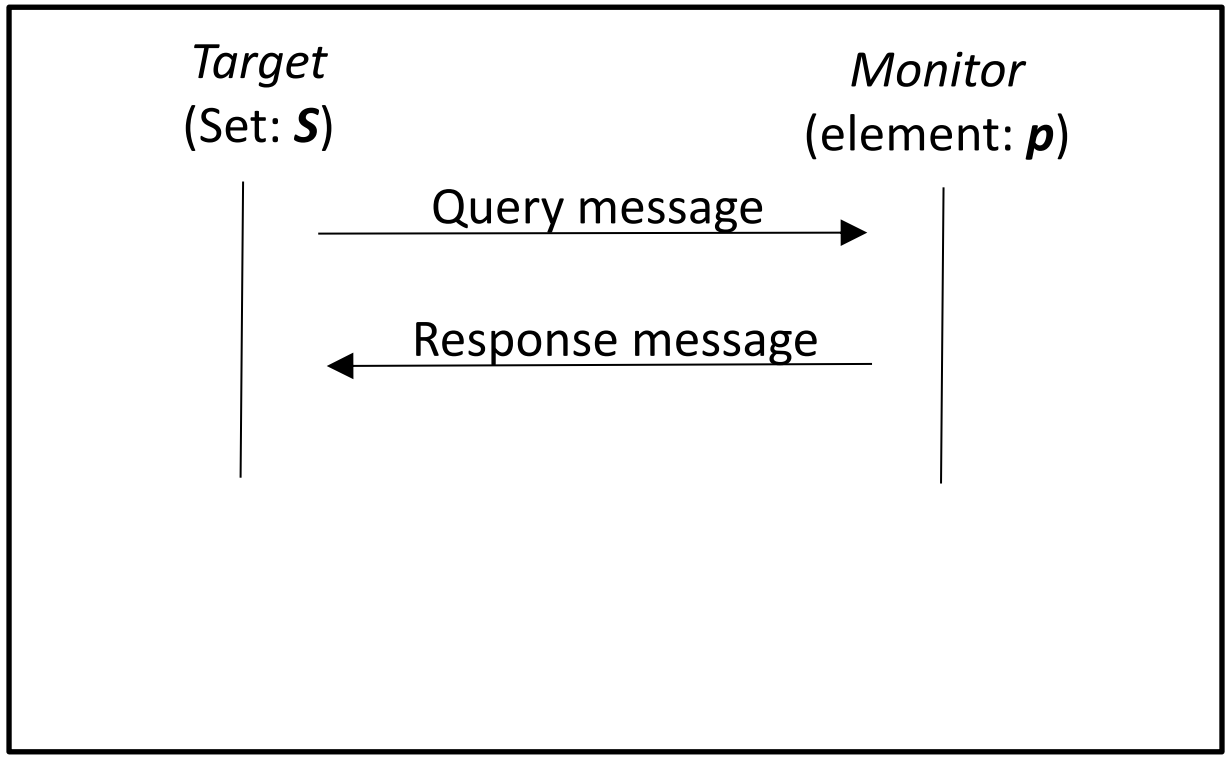
Tries to retrieve the password in the response

For each successfully retrieved password from Monitor's responses, Target treats it as a password tried locally and runs the marking process.

PCR Performance

<u>Containment retrieval:</u>

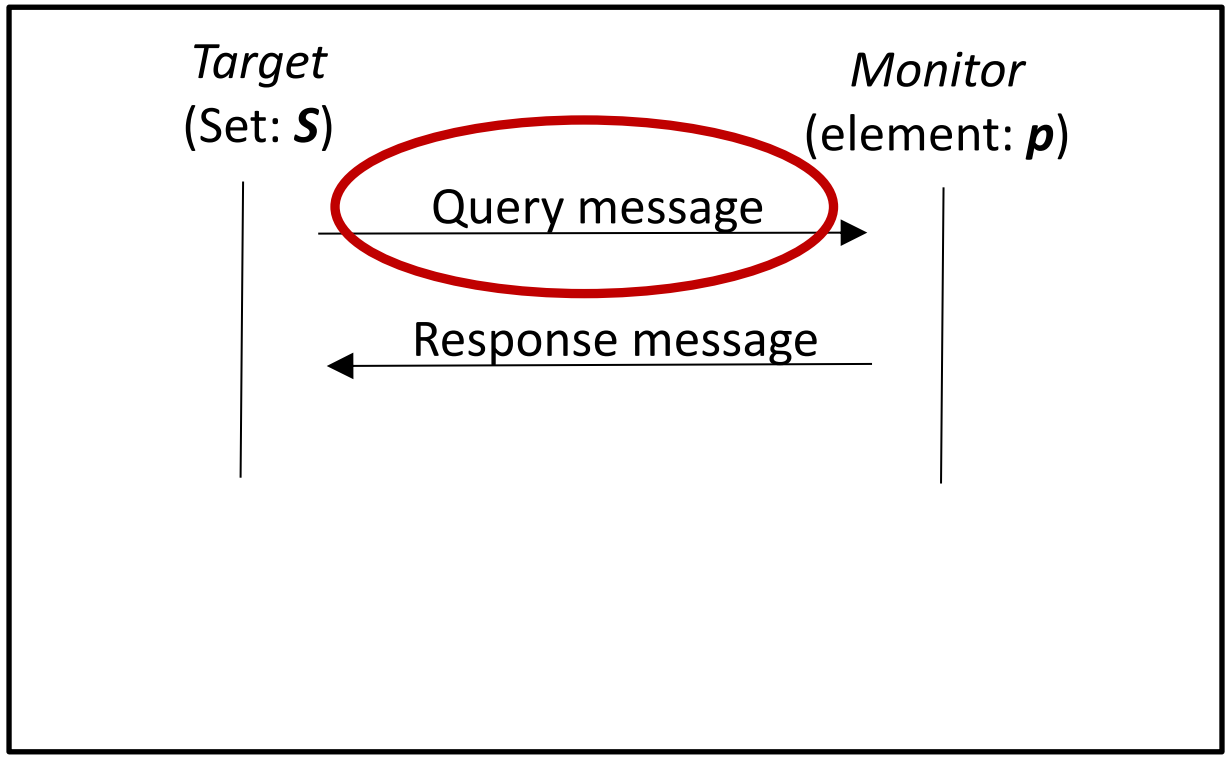
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

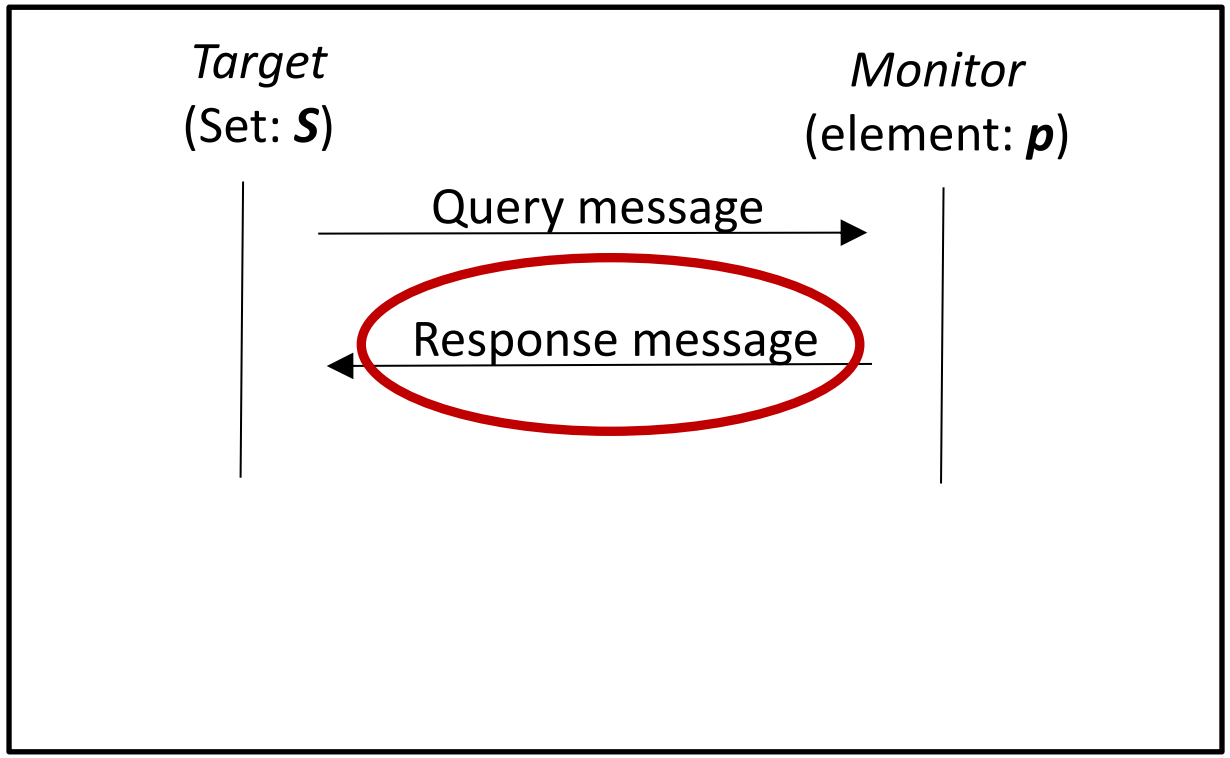
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

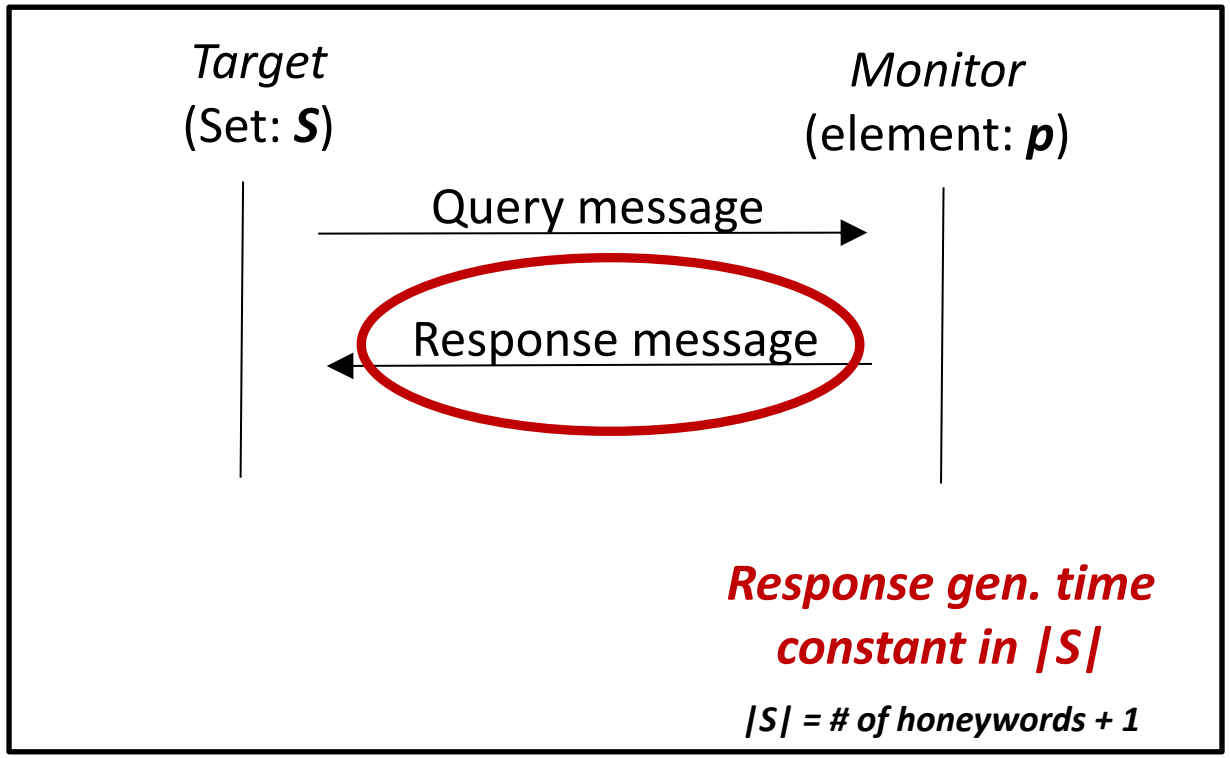
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

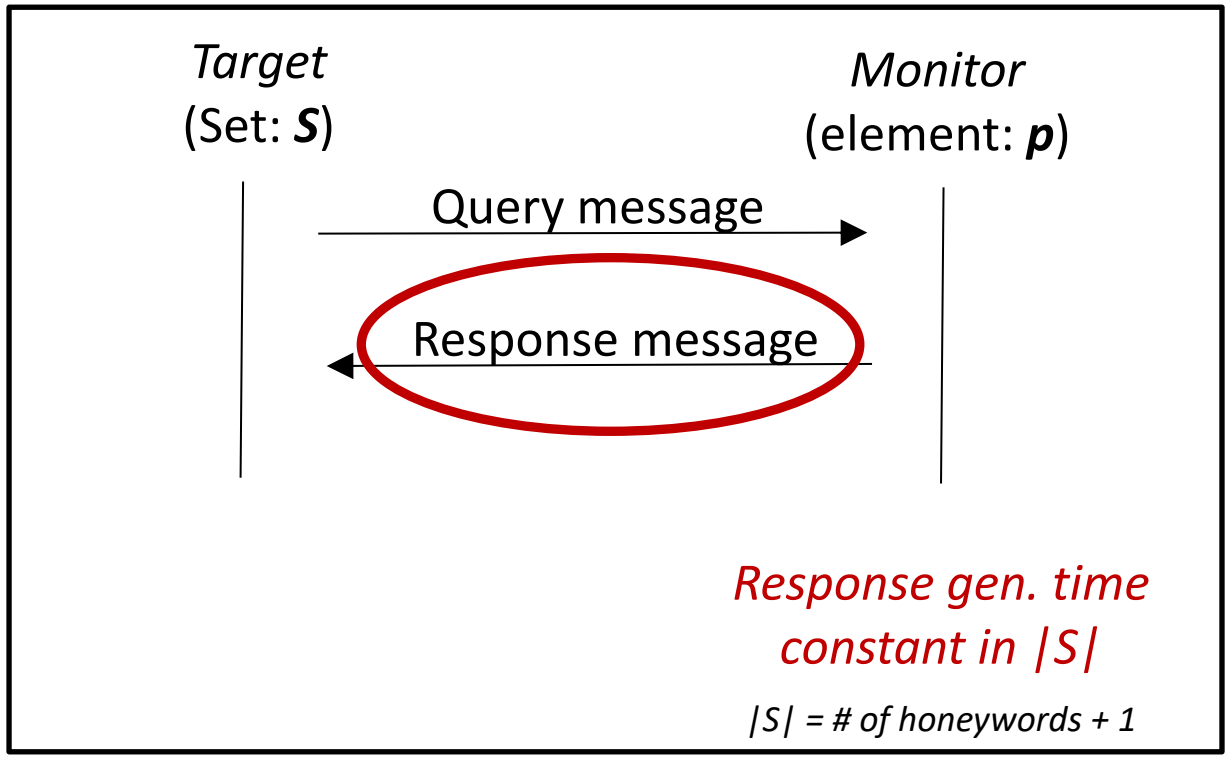
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

<u>retrieve ***p*** only if ***S*** contains ***p***</u>

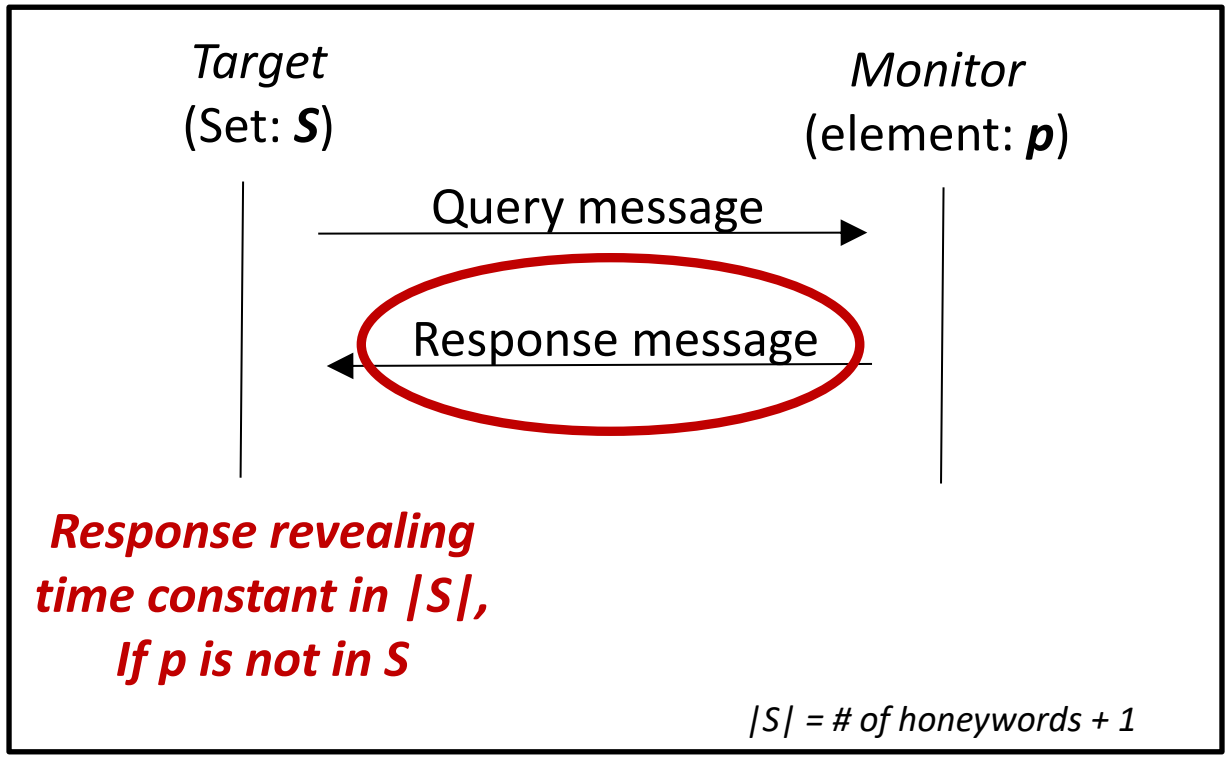


E.g., < 9ms, single core CPU (128-bit symmetric security)

PCR Performance

<u>Containment retrieval:</u>

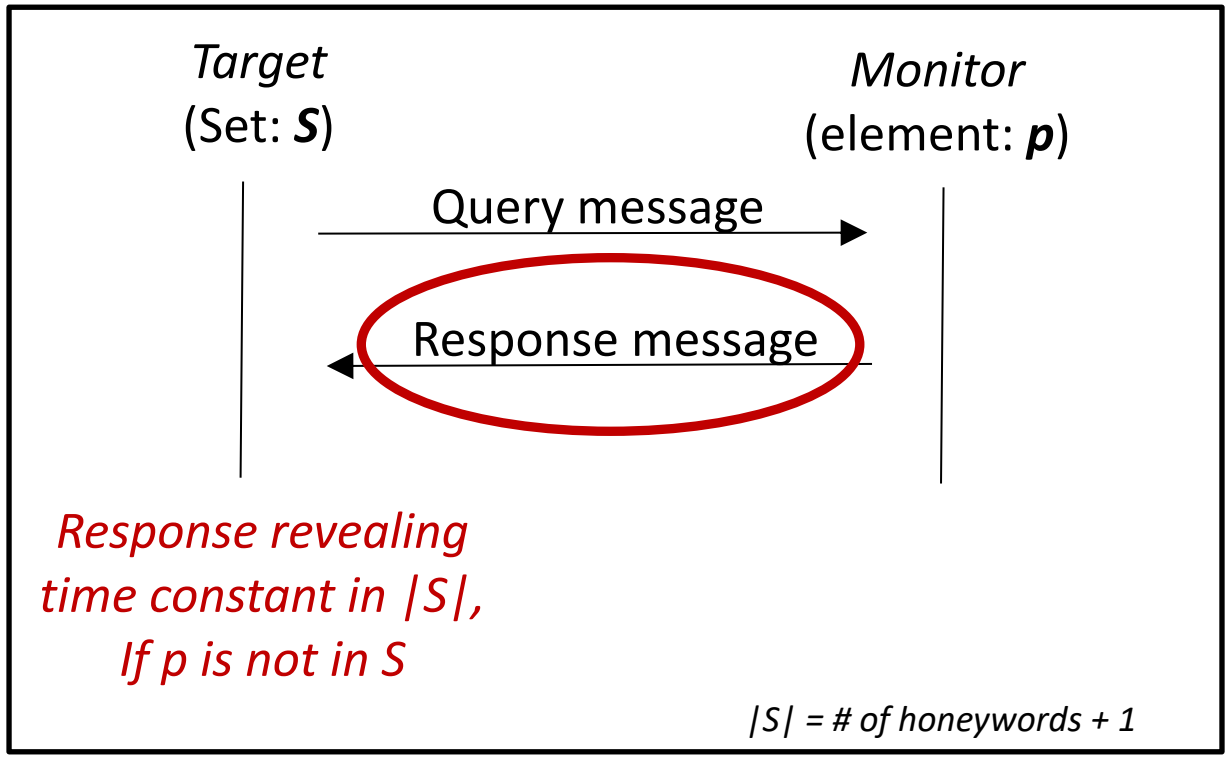
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

<u>retrieve ***p*** only if ***S*** contains ***p***</u>

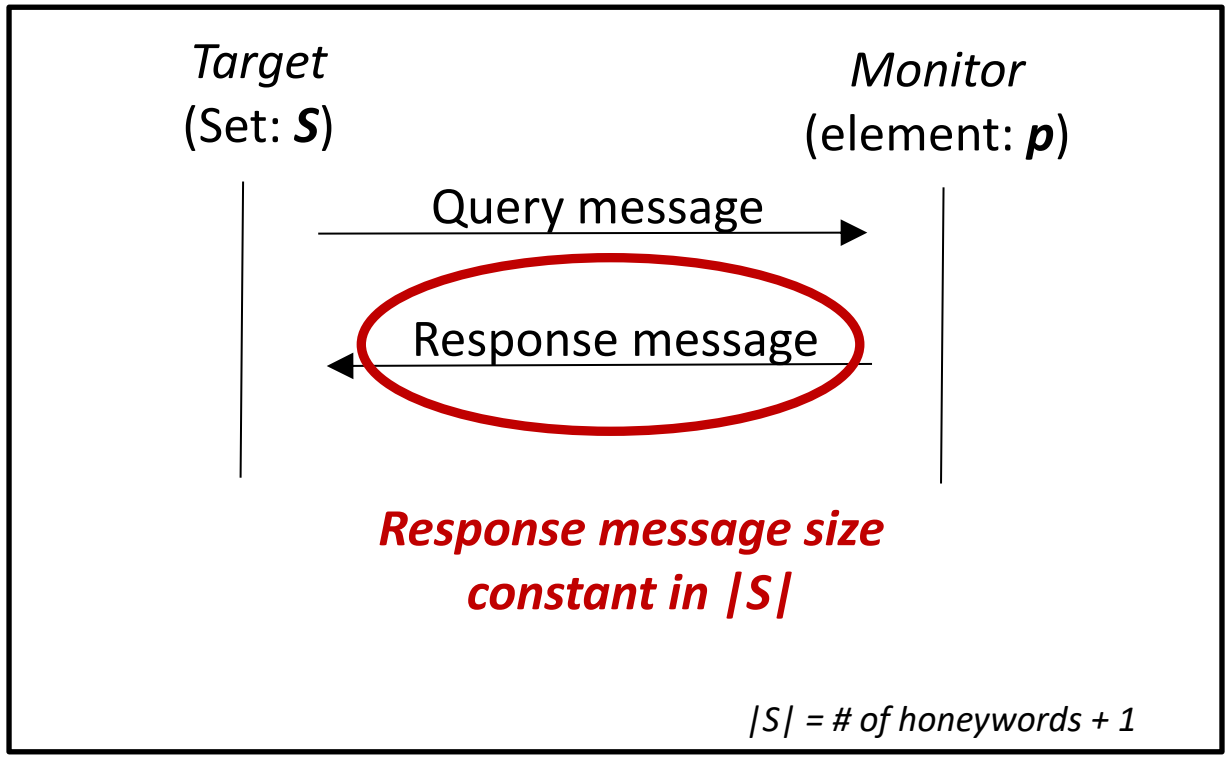


E.g., ~3.5ms, single core CPU

PCR Performance

<u>Containment retrieval:</u>

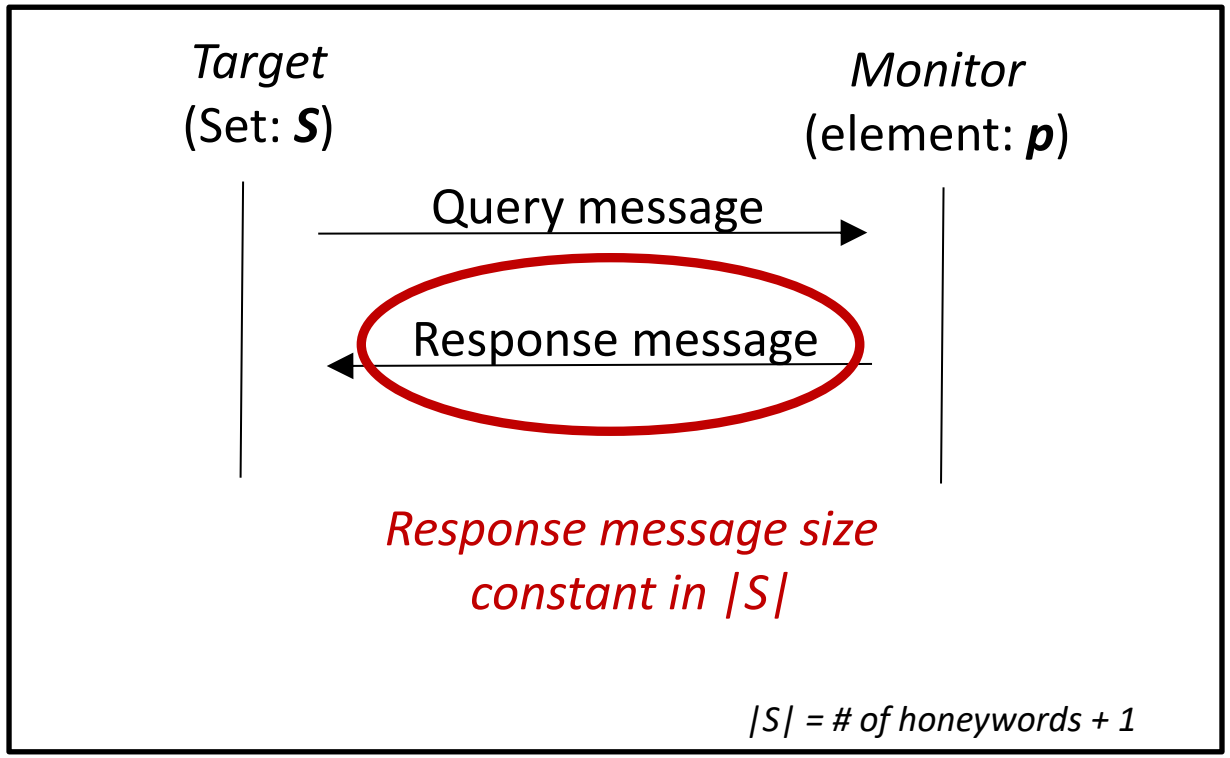
<u>retrieve ***p*** only if ***S*** contains ***p***</u>



PCR Performance

<u>Containment retrieval:</u>

<u>retrieve ***p*** only if ***S*** contains ***p***</u>



E.g., ~1KB per response
(128-bit symmetric security)

Summary

Summary

- ***Amnesia can detect password database breaches without any persistent secret state***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- ***Amnesia is extended to detect password database breaches against stuffing of honeywords***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- Amnesia is extended to detect password database breaches against stuffing of honeywords
  - ***A new two-party private containment retrieval protocol***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- Amnesia is extended to detect password database breaches against stuffing of honeywords
  - A new two-party private containment retrieval protocol
  - ***Converting remote stuffing to local online guessing***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- Amnesia is extended to detect password database breaches against stuffing of honeywords
  - A new two-party private containment retrieval protocol
  - Converting remote stuffing to local online guessing
- Evaluation results:
  - ***Amnesia would substantially shorten the time between credential database breaches and their discovery***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- Amnesia is extended to detect password database breaches against stuffing of honeywords
  - A new two-party private containment retrieval protocol
  - Converting remote stuffing to local online guessing
- Evaluation results:
  - Amnesia would substantially shorten the time between credential database breaches and their discovery
  - ***Our PCR is practical and well suited for Amnesia***

Summary

- Amnesia can detect password database breaches without any persistent secret state
- Amnesia is extended to detect password database breaches against stuffing of honeywords
  - A new two-party private containment retrieval protocol
  - Converting remote stuffing to local online guessing
- Evaluation results:
  - Amnesia would substantially shorten the time between credential database breaches and their discovery
  - Our PCR is practical and well suited for Amnesia

Thank you !