# Protocols for Checking Compromised Credentials

**Lucy Li**
Cornell Tech

Bijeeta Pal
Cornell Tech

Junade Ali
Cloudflare Inc.

Nick Sullivan
Cloudflare Inc.

Rahul Chatterjee
University of Wisconsin—Madison
Cornell Tech

Thomas Ristenpart
Cornell Tech

# Password breaches


…and many more

| Leaked Credentials | |
|---|---|
| Username | Passwords |
| … | … |
| lucy@email.com | myPassword123 |
| alice@yahoo.com | Star246, p4ssw0rd1 |
| … | … |

Have I Been Pwned: **406** breaches

Breach compilation from 2018:
**1.4 billion** user-password pairs

# Credential Stuffing

lucy@email.com
myPassword123

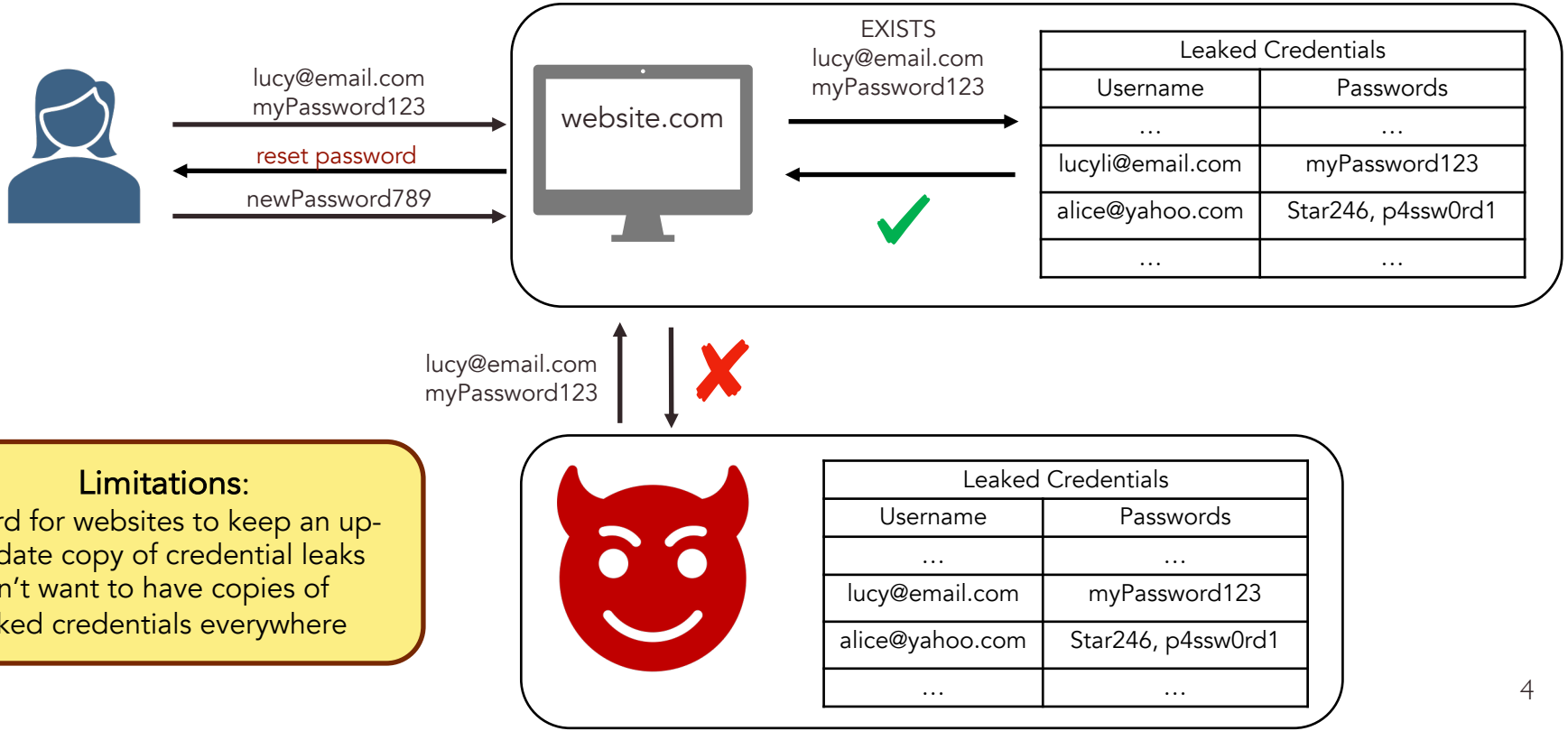
website.com

| Leaked Credentials | |
|---|---|
| Username | Passwords |
| … | … |
| lucy@email.com | myPassword123 |
| alice@yahoo.com | Star246, p4ssw0rd1 |
| … | … |

Around 40% of users reuse passwords across different websites!
[Das et al. 2014, Pearman et al. 2017]

Credential stuffing is the largest source of account takeover and automated fraud
[Shape Security]

# One way to prevent credential stuffing




**Limitations:**

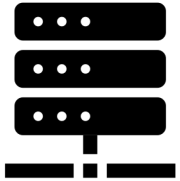
- Hard for websites to keep an up-to-date copy of credential leaks
- Don't want to have copies of leaked credentials everywhere

# Third Party Checker

Don't send credentials in the clear to a third party

lucy@email.com
myPassword123


EXISTS
lucy@email.com
myPassword123



Third party

| Leaked Credentials | |
|---|---|
| Username | Passwords |
| … | … |
| lucy@email.com | myPassword123 |
| alice@yahoo.com | Star246, p4ssw0rd1 |
| … | … |

Can we use a third party checker and still preserve privacy of user credentials?

Two big initial deployments:

';--have i been pwned?

Password Checkup extension

Offered by: google.com

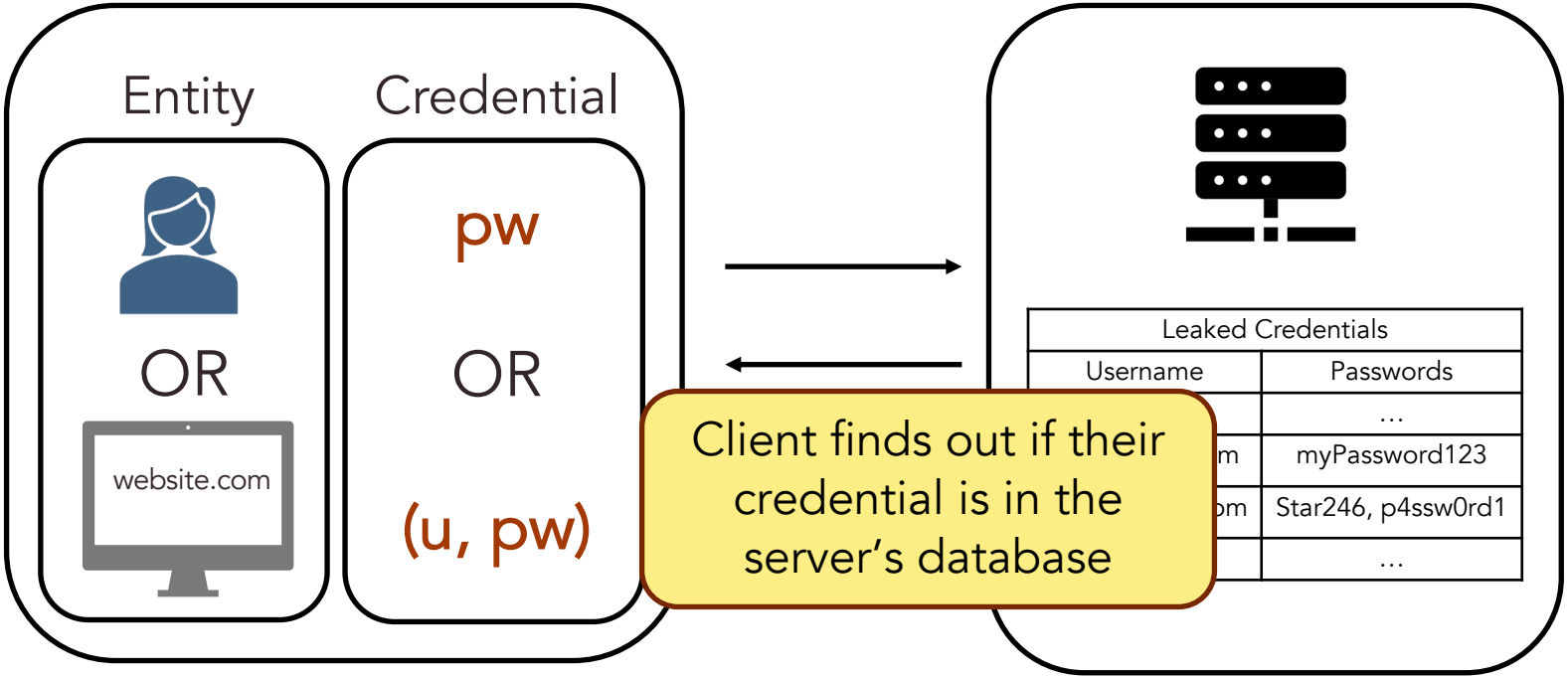
# Our contributions

1. Formalization of compromised credential checking (C3) protocols and threat model

2. HIBP and Google Password Checkup leak information about passwords

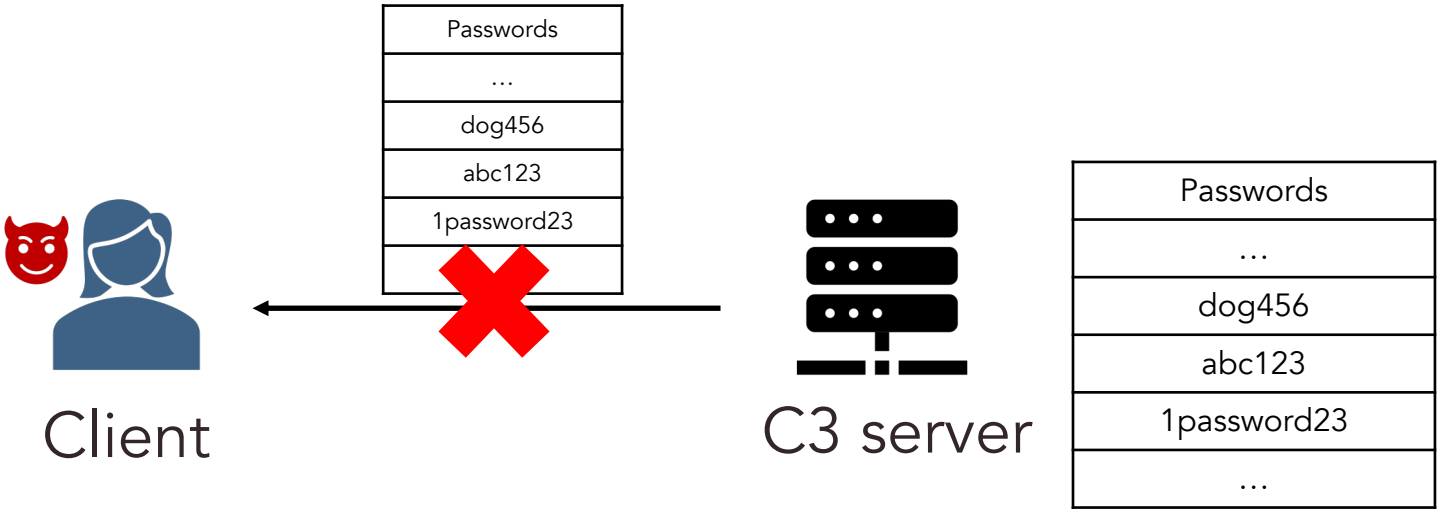
3. New C3 protocols that leak less
   - Frequency-smoothing bucketization
   - ID-based bucketization

# Compromised Credential Checking Services



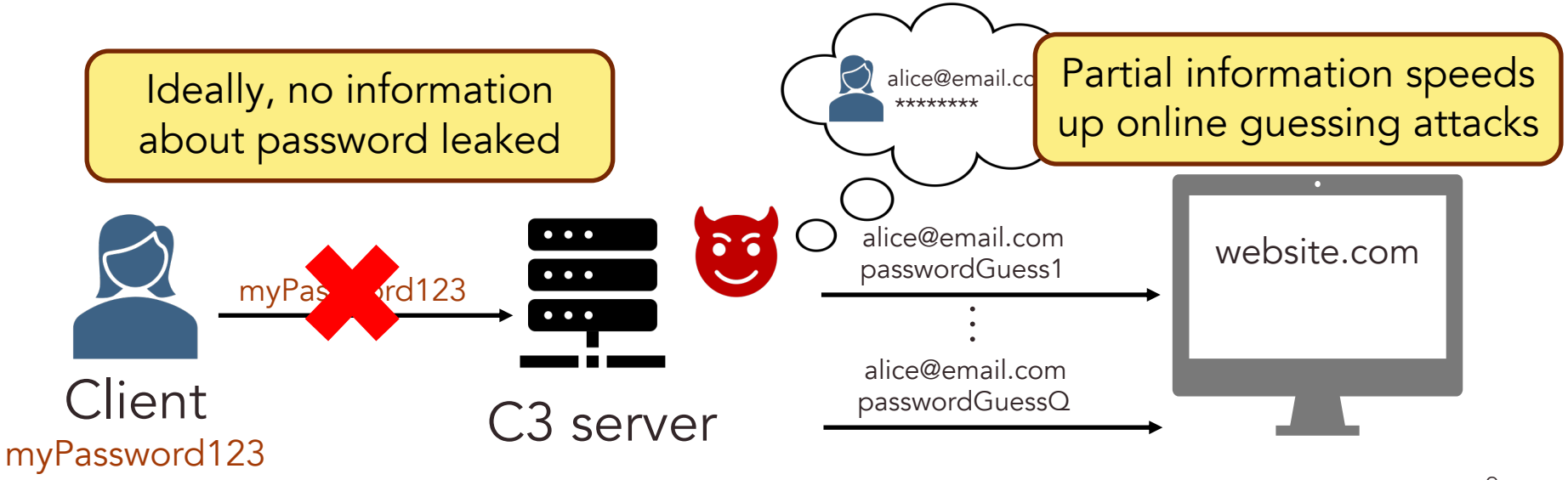
# Threat model

1. Protect leaked credential database from malicious client



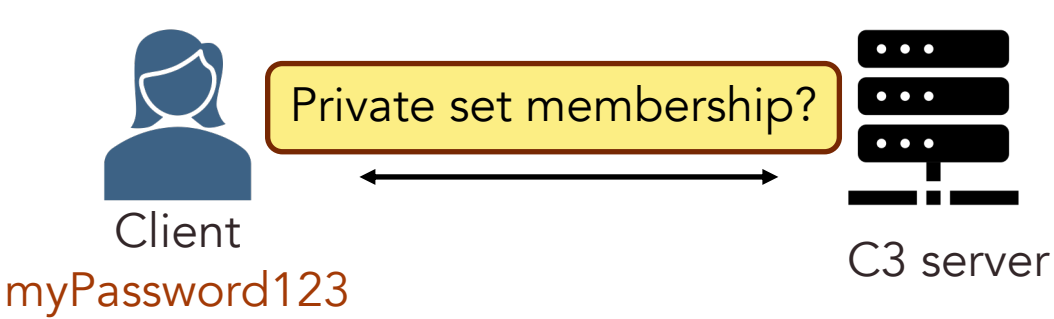
# Threat model

1. Protect leaked credential database from malicious client
2. **Protect client's password against malicious server**



# Private set membership

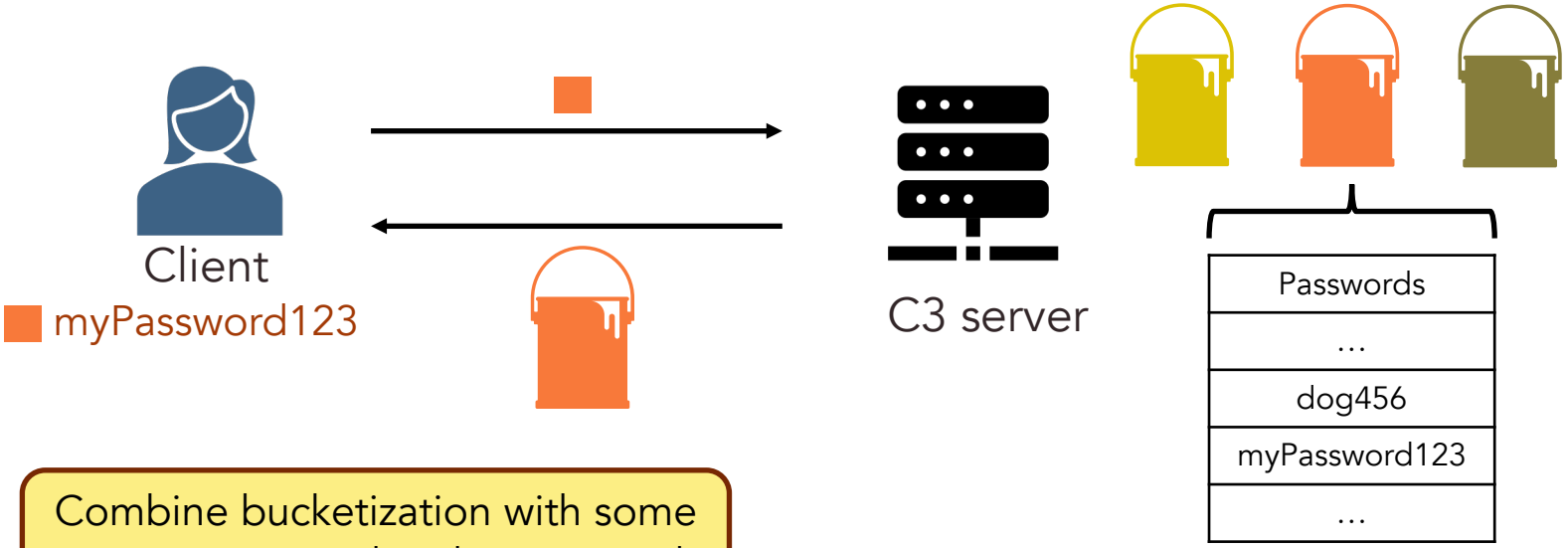
Need to preserve privacy for both client credentials and server database




| Passwords |
| --- |
| … |
| dog456 |
| abc123 |
| 1password23 |
| … |

Doesn't scale to sets containing hundreds of millions of elements

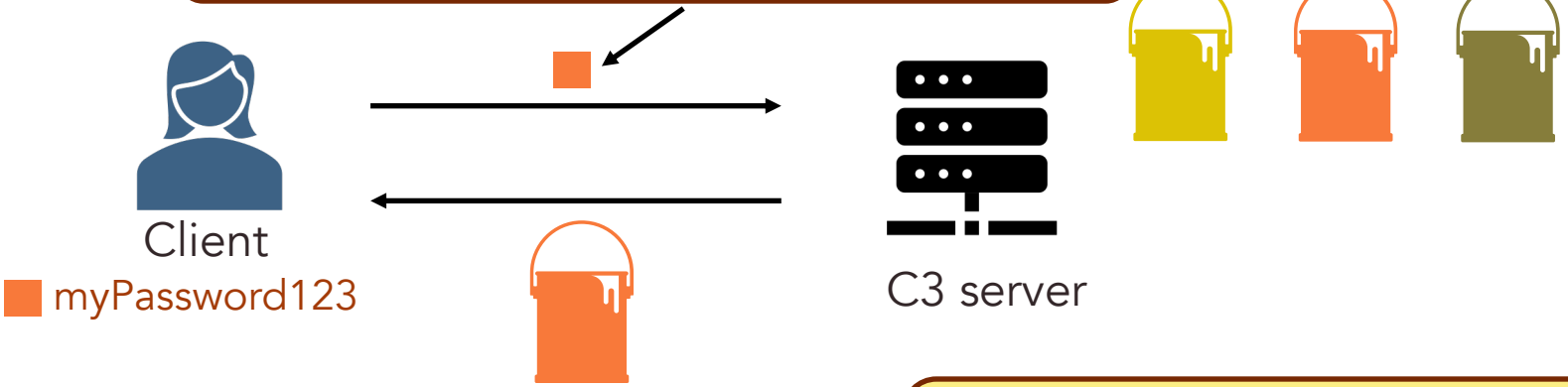
# Efficiency through bucketization




Combine bucketization with some private set membership protocol

# Efficiency through bucketization

Have I Been Pwned and Google Password Checkup use a hash prefix as a bucket identifier

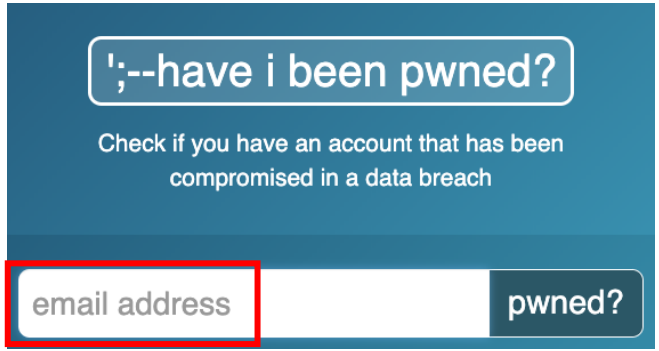


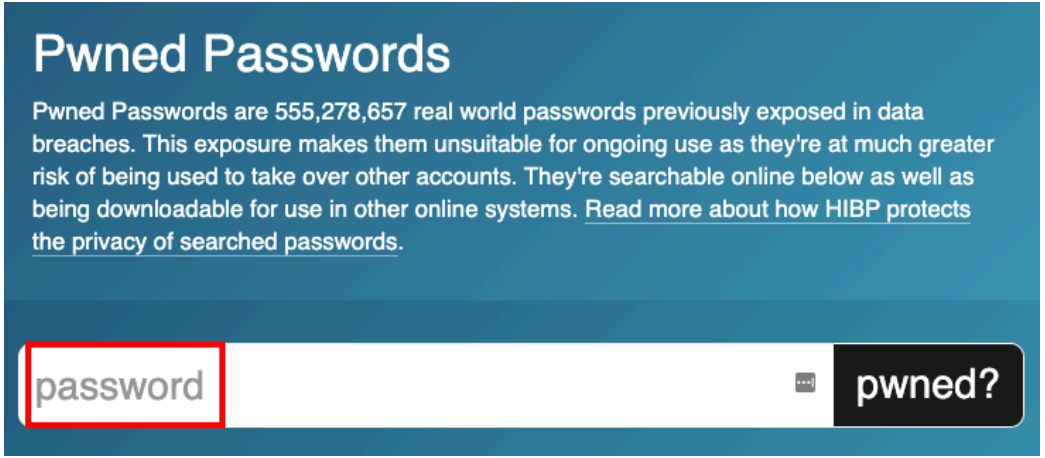

Combine bucketization with some private set membership protocol

**Key security question:** How much does knowing the bucket queried help an adversary guess a client's password?

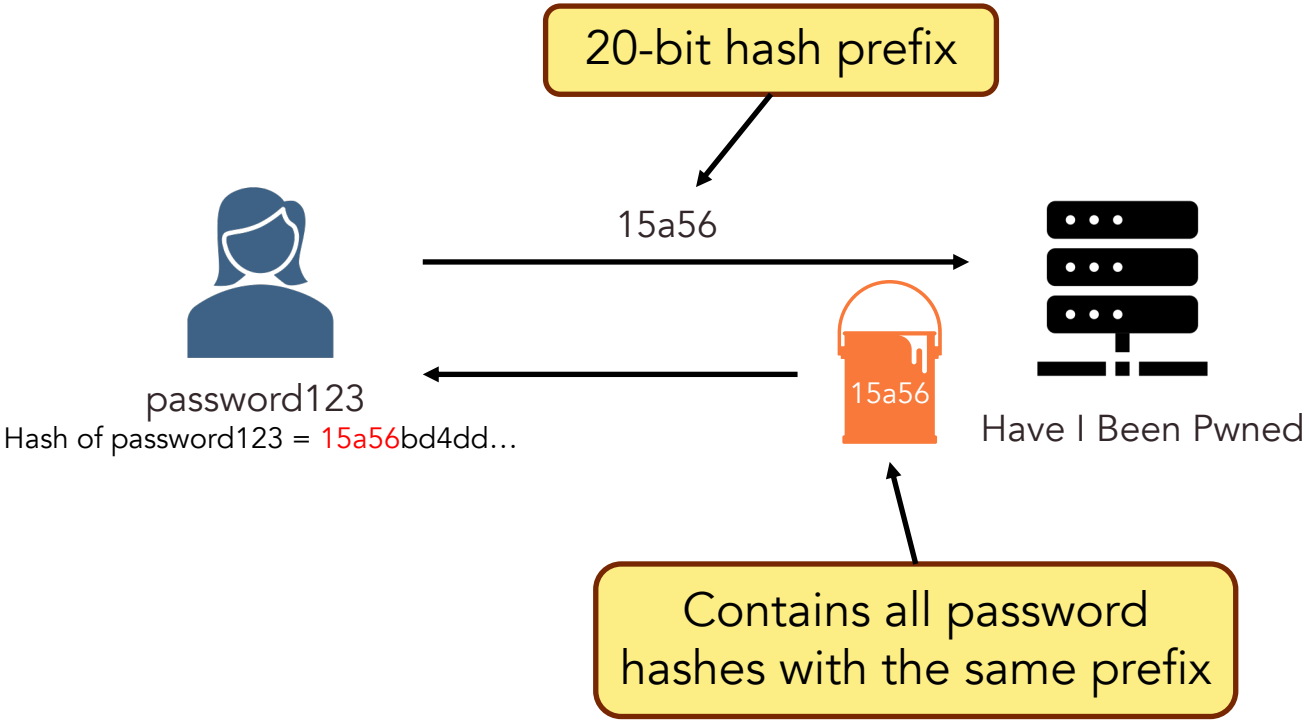
# Have I Been Pwned (HIBP)

- Existing C3 service that checks if your username OR password exists in a known data breach



# Have I Been Pwned (HIBP)



# Security of HIBP


| Bucket 15a56 |
| --- |
| password123 |
| enrique24 |
| lilone55 |
| pourmixl |
| ... |

Attacker's guess order of passwords

More popular

Less popular

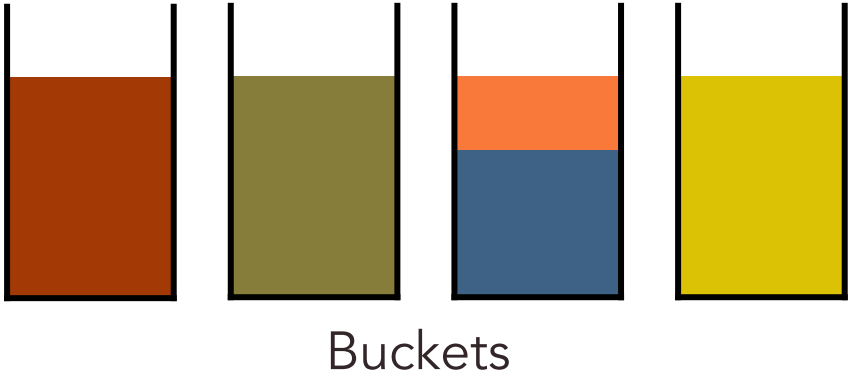
An attacker with access to the bucket and *Q* guesses has about the same success rate as an attacker with *Q*B* guesses!
(*B* is the total number of buckets)

For HIBP, $B = 2^{20} \approx 1$ million

With no info: 431st guess
With hash prefix: 1st guess

Compromise account with Q=10 guesses

# Issue with HIBP

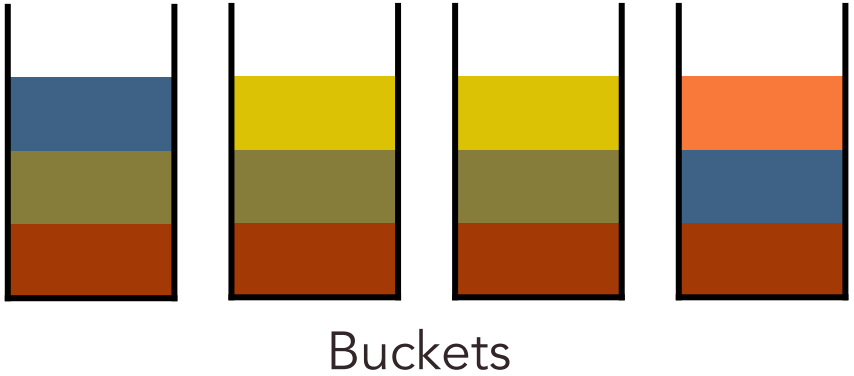




Colors in buckets correspond to probabilities of passwords given the bucket

Easy to guess the password if you know the bucket

# Frequency-smoothing bucketization (FSB)

We propose FSB as a more secure bucketization algorithm

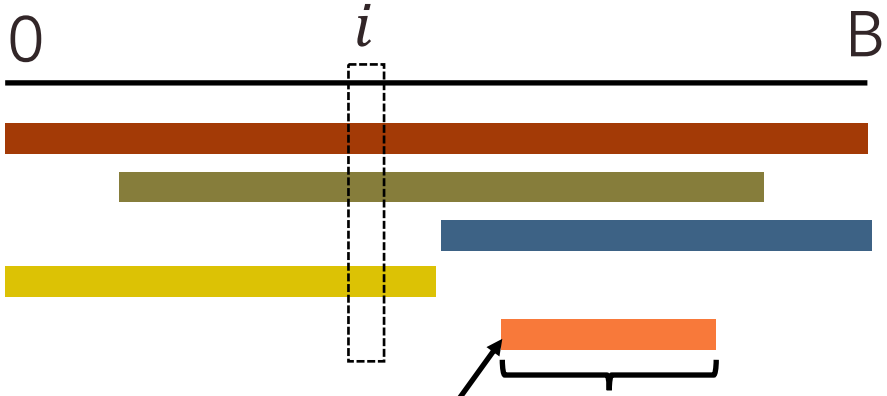




**Goal:** Given a bucket, the probability of each password in the bucket is the same

# FSB implementation details (Q = 1)


B: # buckets

Buckets: 0 $i$ B



Range for password ■
[ H(■) , H(■) + f(Pr(■)) ]

Start bucket: H(■)

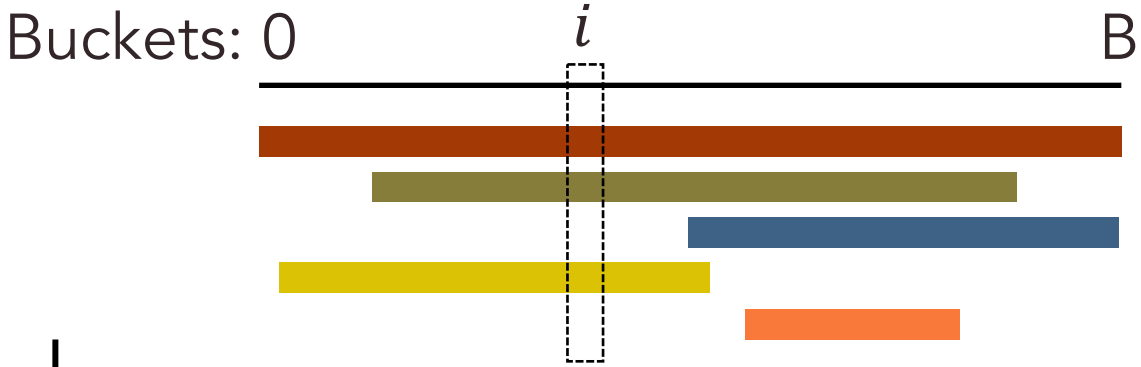
Proportional to probability of password

To check a password with the server:
Client computes range,
picks a bucket randomly

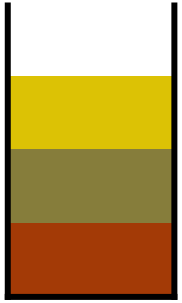
# FSB: what about Q > 1?

- Parameter Q reflects expected online guessing budget
- Include the top Q passwords in every bucket, and distribute the rest proportionally to probability

# FSB: guessing budget parameter

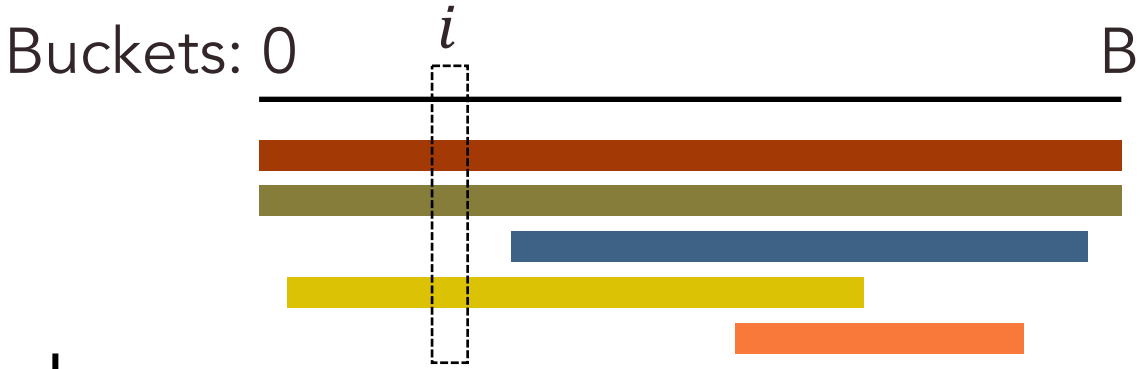




Bucket $i$:

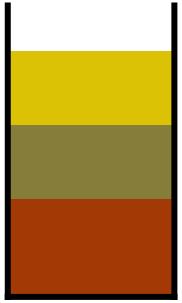


Q = 1

# FSB: guessing budget parameter


Bucket $i$:



Q = 2

# Security of FSB

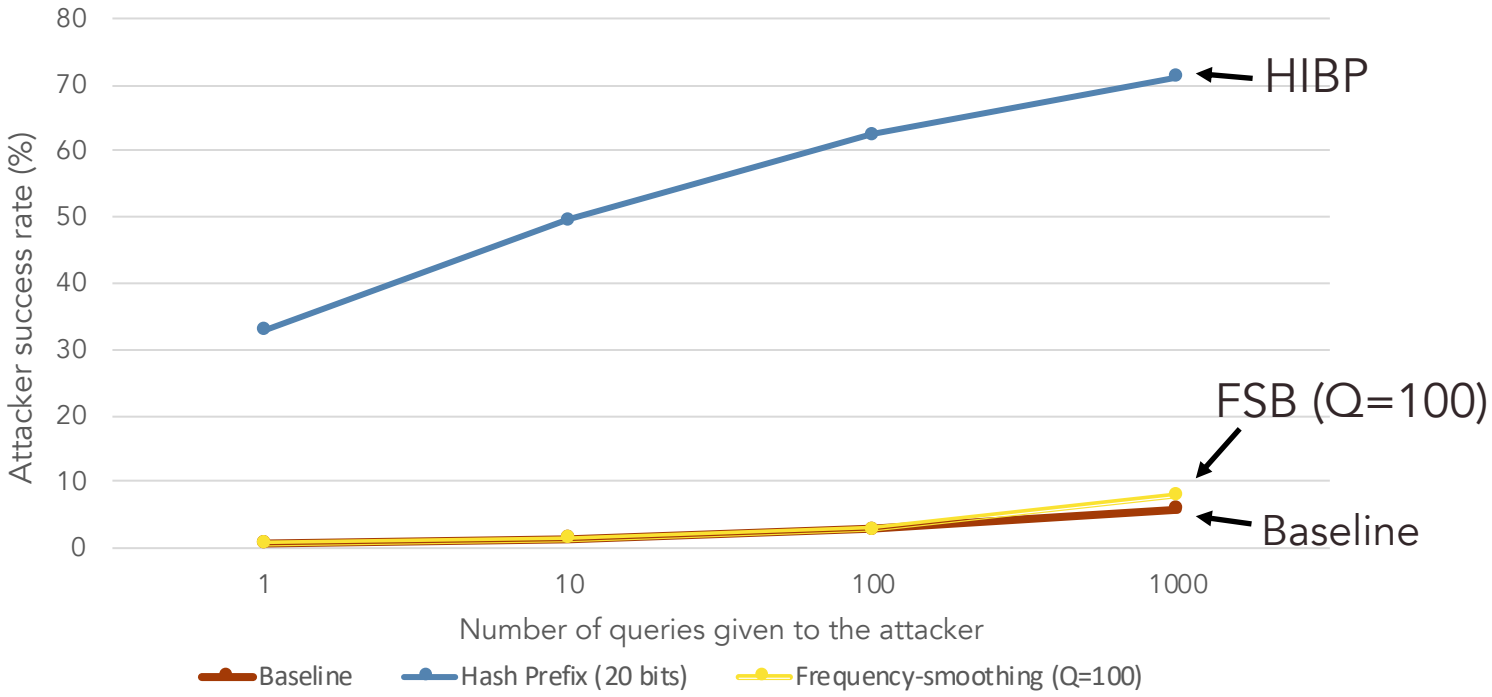
- Theorem: If an attacker has $\leq Q$ guesses, access to the FSB bucket will give no advantage over baseline guessing
- Bounds for $> Q$ guesses shown in our paper
  - Higher $Q \rightarrow$ smaller security loss
    - But also larger bucket sizes

# Empirical security evaluation

- How easily can an attacker guess passwords given the bucket identifiers?
- Breach dataset of 1.4 billion username-password pairs
  - Split into test set and leaked password set
- Sample passwords randomly from test set
- Record the number of guesses needed to recover the correct password

github.com/lucy7li/compromised-credential-checking

# Results


80
70
60
50
40
30
20
10
0
Attacker success rate (%)
1
10
100
1000
Number of queries given to the attacker
HIBP
FSB (Q=100)
Baseline
Baseline
Hash Prefix (20 bits)
Frequency-smoothing (Q=100)

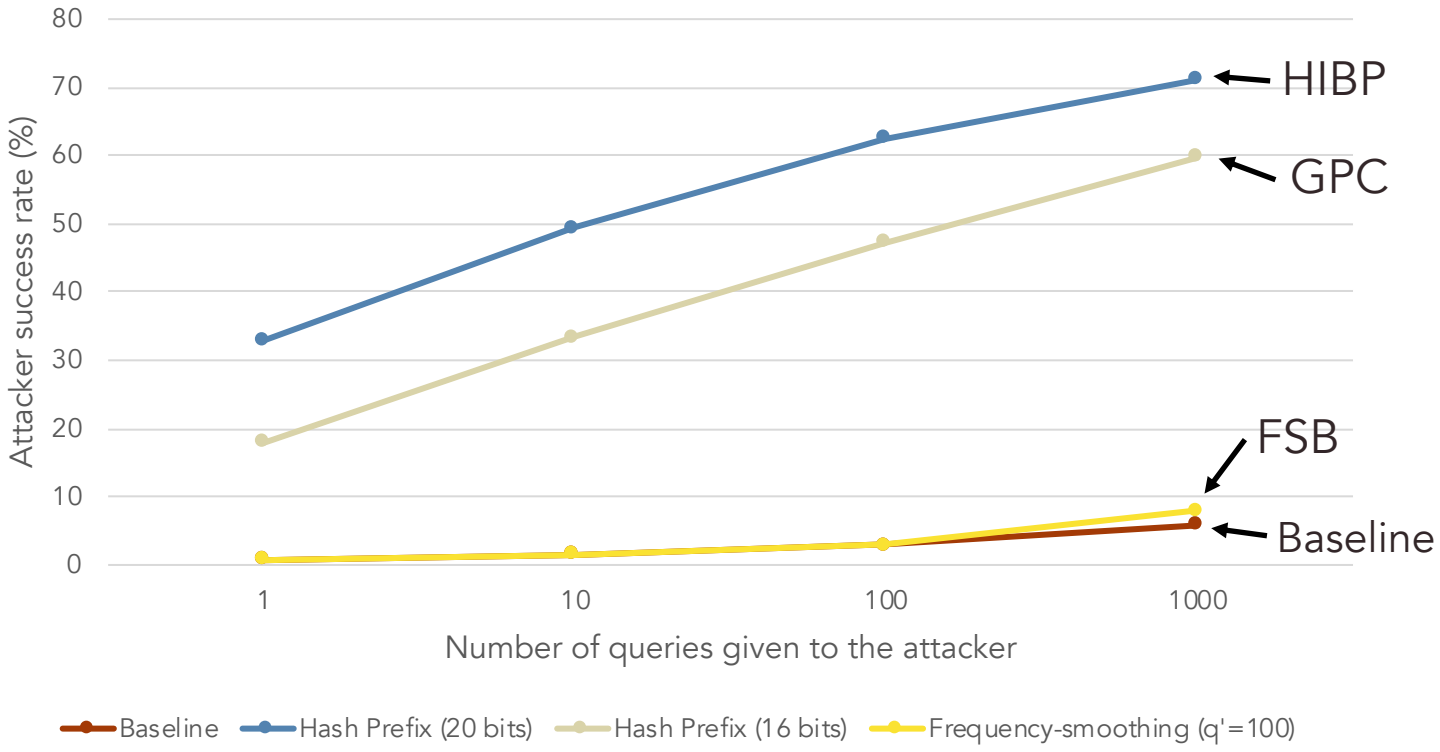
# Recap of password-only C3 setting

- Checks if a password is in breach data
- HIBP leaks information about passwords that speeds up remote guessing attacks
- Frequency-smoothing bucketization leaks less
- However, password-only checks may have false positives

# ID-password C3 protocols

- Check for an exact username-password pair match with a C3 server
- Google Password Checkup (GPC) implemented a protocol that uses the prefix of H(user || pw) as the bucket identifier
  - Bucket contains all H(user || pw) with the same hash prefix
- Runs into same security issues as HIBP, **if username is known**

# GPC empirical security



# ID-based bucketization (IDB)

- We propose a modification of GPC that only uses the hash prefix of the username as a bucket identifier (ID-based bucketization, IDB)
  - Google independently proposed this change in their paper, as a future modification to their initial design [Thomas et al. 2019]
- We show that knowing the IDB bucket identifier gives an attacker no advantage in guessing a user's password (over baseline guessing)!

# Performance

| Setting | Protocol | Bandwidth (KB) | Total time (ms) |
|---|---|---|---|
| Password-only | HIBP | 32 | 220 |
| | FSB | 558 | 527 |
| ID-password | GPC | 1,066 | 489 |
| | IDB | 1,066 | 517 |

**Total time** includes client-server communication and client- and server-side computations

github.com/lucy7li/compromised-credential-checking

# Conclusion

- Existing deployed C3 protocols leak a lot of information about a user's password to the C3 server
- To leak less information, we recommend using:
  - Password-only: Frequency-smoothing bucketization
  - Username-password: ID-based bucketization
- Questions?

github.com/lucy7li/compromised-credential-checking