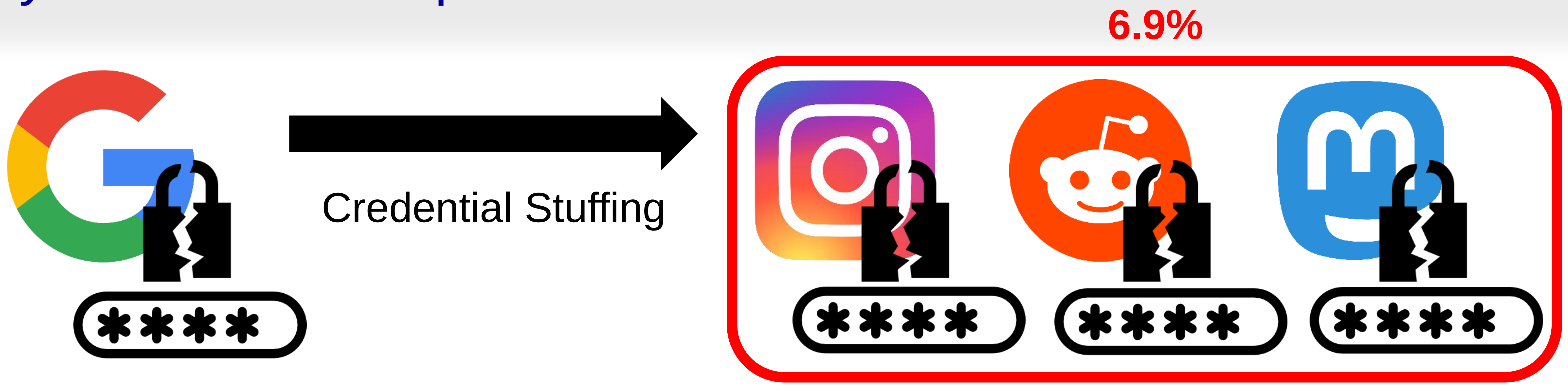
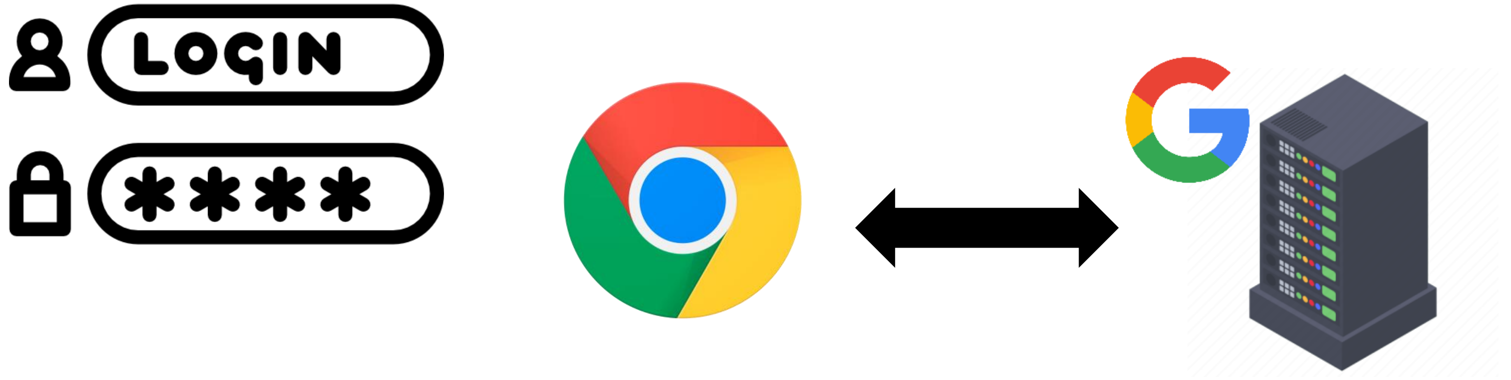
Checking Passwords on Leaky Computers: A Side Channel Analysis of Chrome's Password Leak Detection Protocol

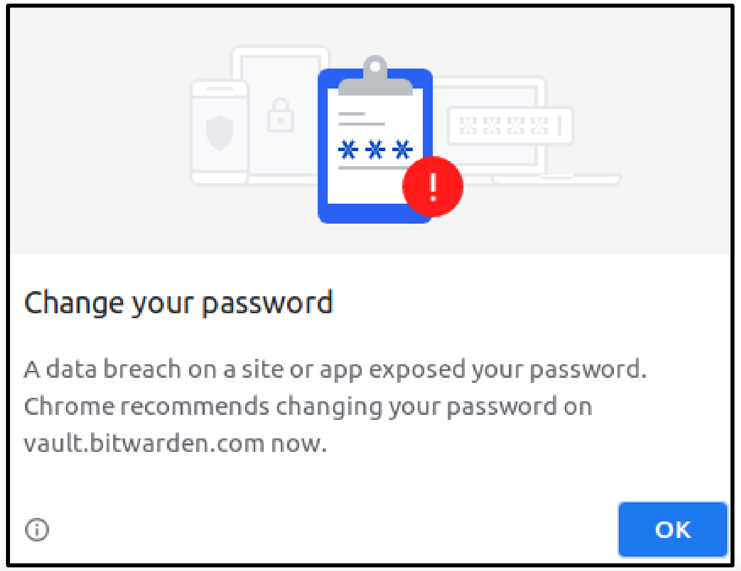
Andrew Kwong, Walter Wang, **Jason Kim**, Jonathan Berger, Daniel Genkin, Eyal Ronen, Hovav Shacham, Riad Wahby, Yuval Yarom

Why Check for Compromised Passwords?



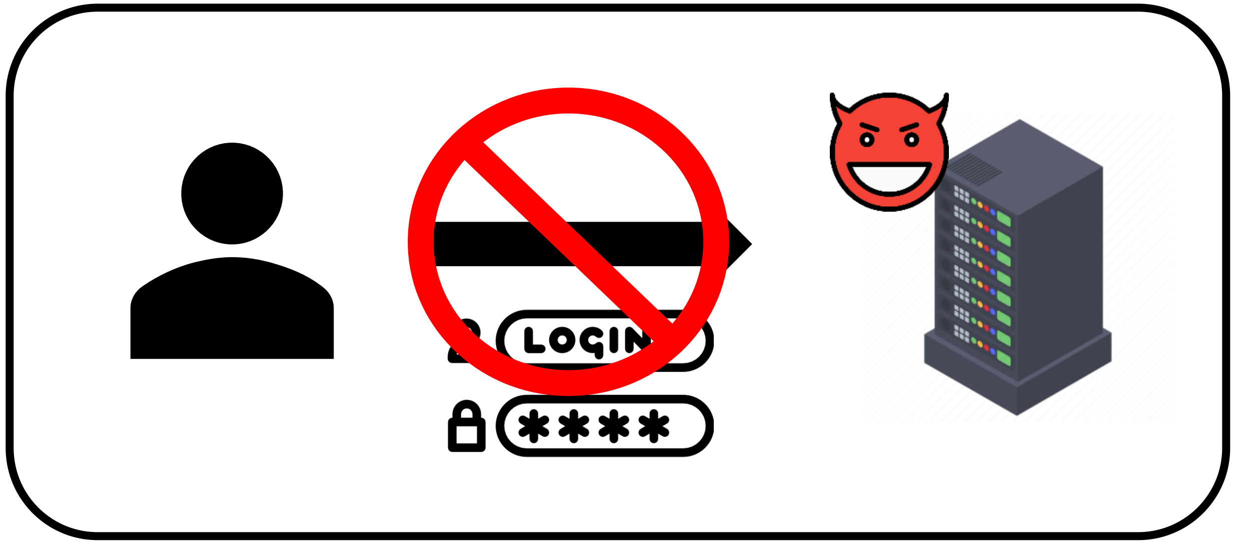
Enabled by Default!

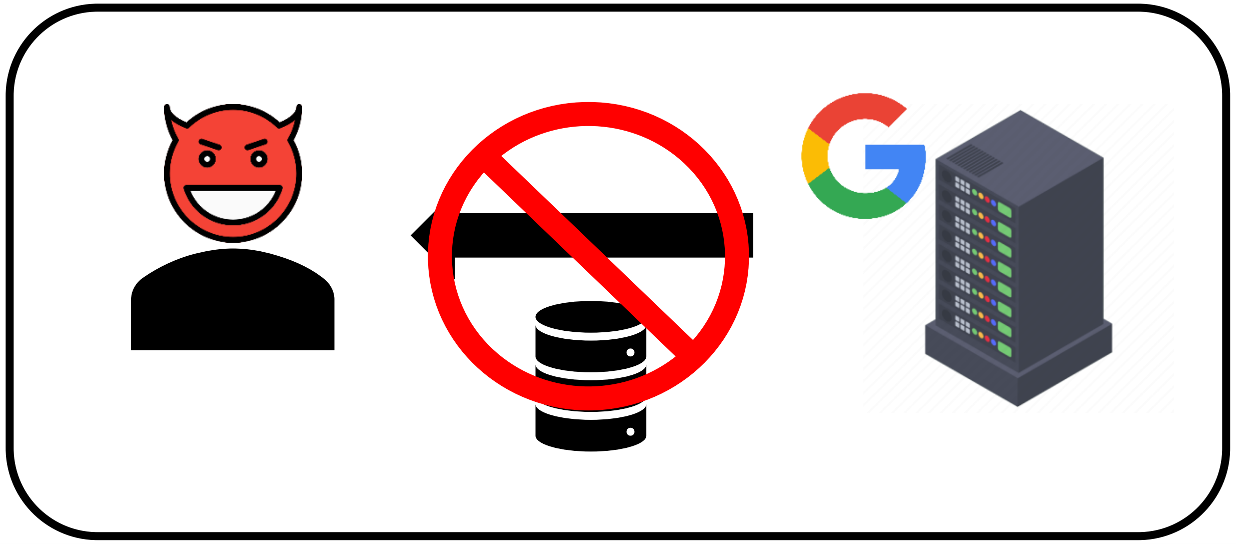



Change your password

A data breach on a site or app exposed your password. Chrome recommends changing your password on vault.bitwarden.com now.

Chrome's Password Leak Detection





Memory-hard Hashing

Hash-to-curve

Private Set Intersection

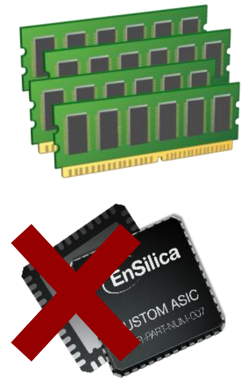
Is Password Leak Detection secure against side-channel attacks?

No!

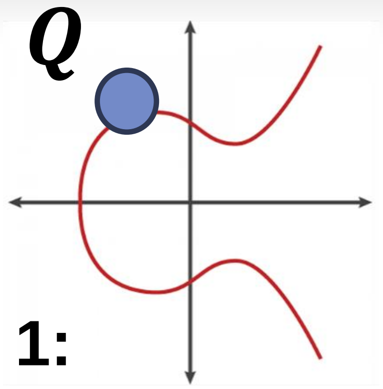
The Protocol in a Nutshell

***PSI:** Private Set Intersection

Memory-hard Hashing:
Scrypt



PSI Step 1:
Hash-to-curve



$$(P^a)^{a^{-1}} = P$$

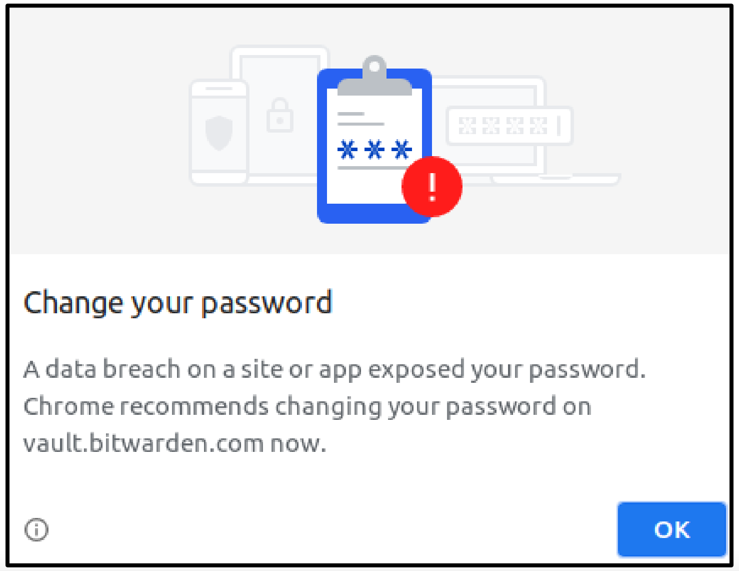
PSI Step 4:
Unblind with
Mod. Inv.

PSI Step 2:
Blind with
Mod. Exp.

Q^a

P^a

PSI Step 3:
Server
Computes

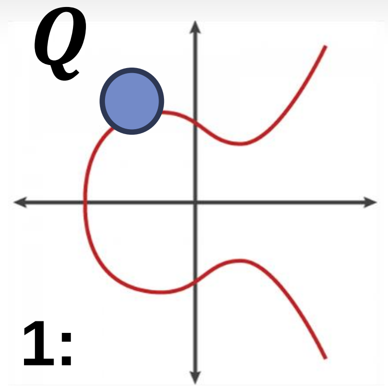

Change your password

A data breach on a site or app exposed your password. Chrome recommends changing your password on vault.bitwarden.com now.

OK

The Protocol in a Nutshell

***PSI:** Private Set Intersection



Memory-hard Hashing:
Scrypt

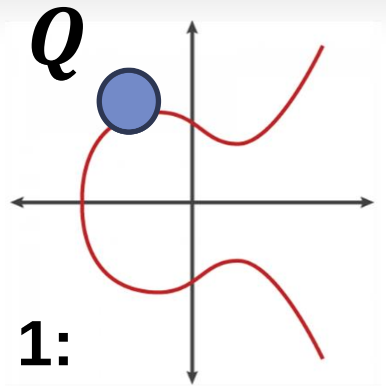
PSI Step 1:
Hash-to-curve

$$(P^a)^{a^{-1}} = P$$

PSI Step 4:
Unblind with Mod. Inv.

Let the Attacks Begin!

***PSI:** Private Set Intersection



$$(P^a)^{a^{-1}} = P$$

Memory-hard Hashing:
Scrypt

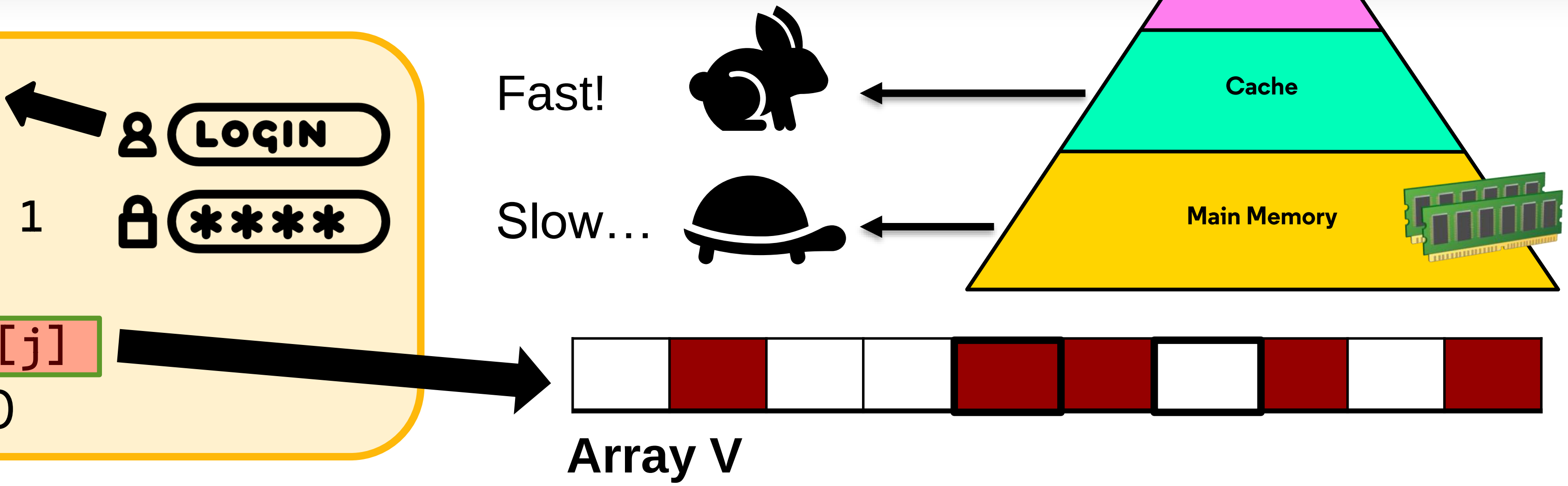
PSI Step 1:
Hash-to-curve

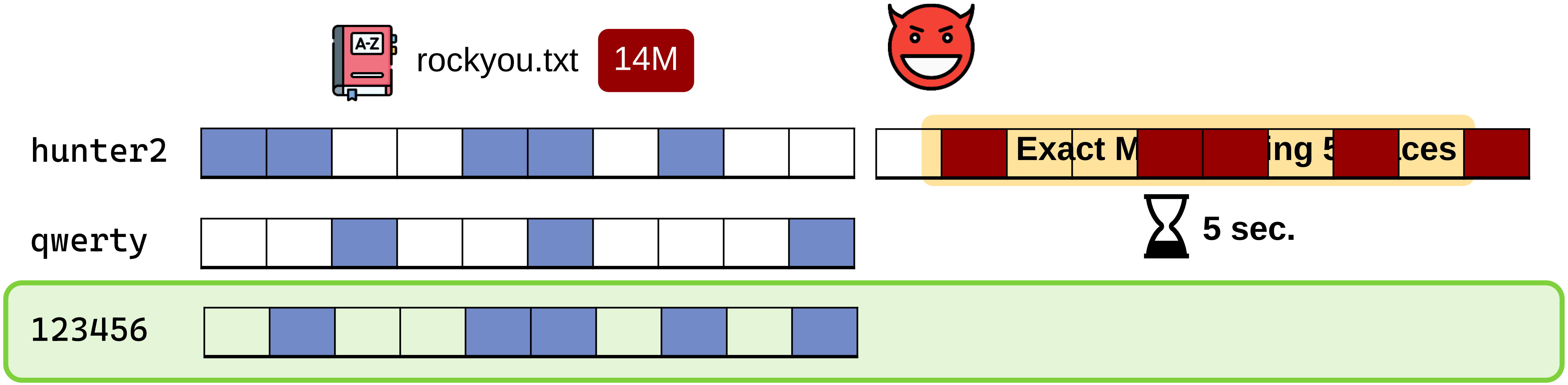
PSI Step 4:
Unblind with Mod. Inv.

Native Adversary

Attacking Input-dependent Memory Accesses

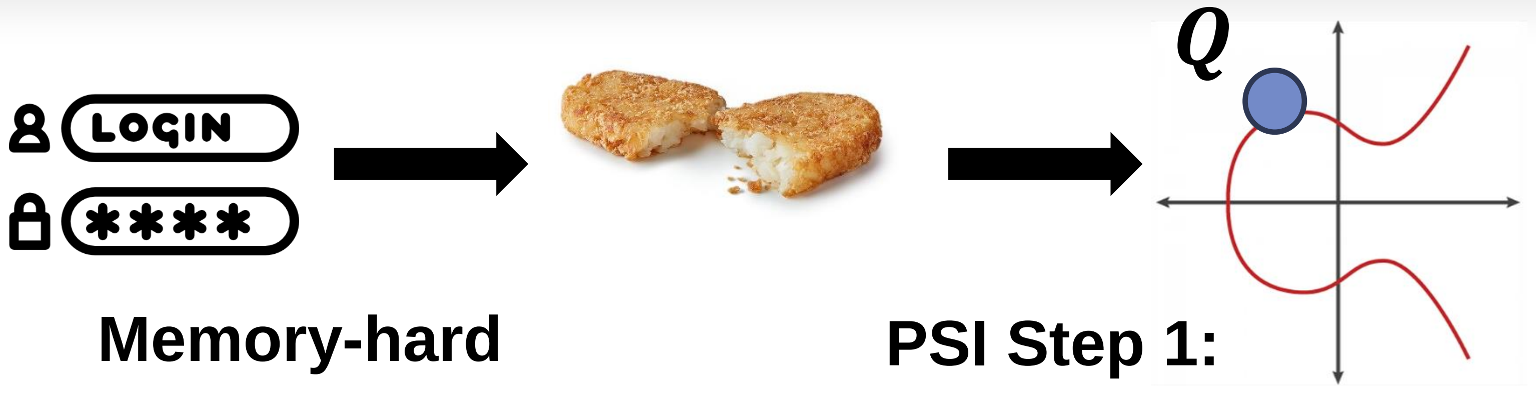
```
function scrypt(inp)
    X = init(inp)
    for i = 0 to N − 1
        j = int(X)
        temp = X ^ V[j]
        X = mix(temp)
```





What About Browser-based Adversaries?

***PSI:** Private Set Intersection



Memory-hard Hashing:

Scrypt

PSI Step 1:

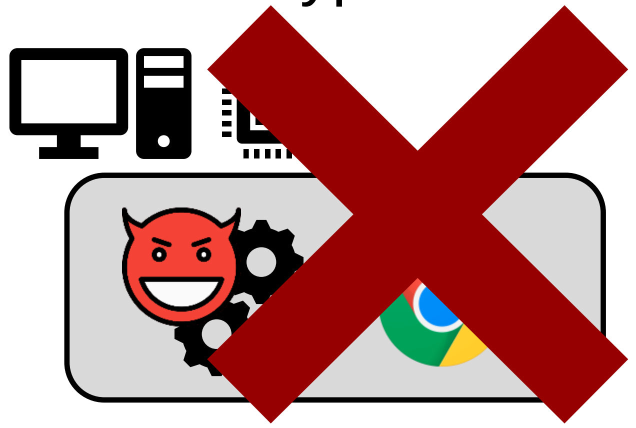
Hash-to-curve

$$(P^a)^{a^{-1}} = P$$

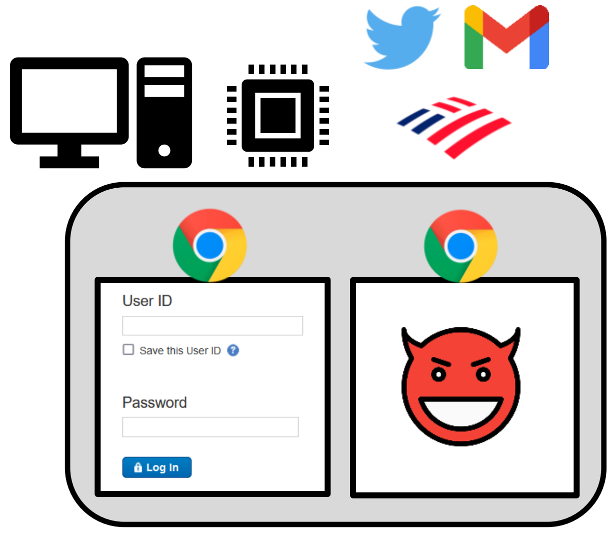
PSI Step 4:

Unblind with

Mod. Inv.



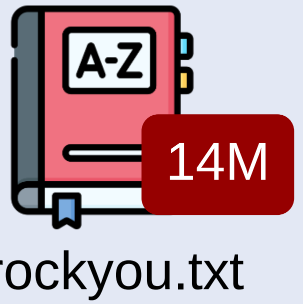
Native Adversary



Browser Adversary

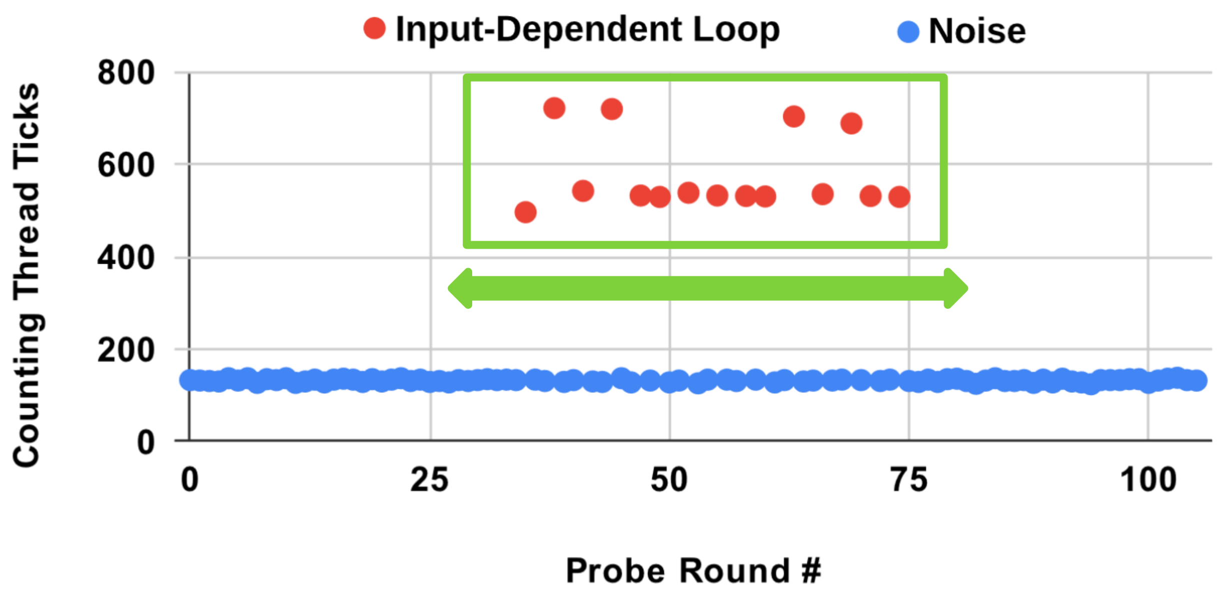
Attacking Input-dependent Loop Iterations

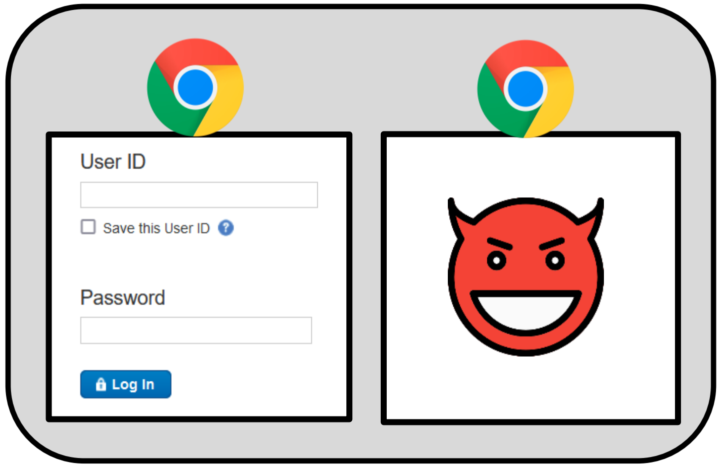
```
function hash2curve(hash)
    point = RandomOracleSHA256(hash)
    while !OnCurve(point) do
        point = RandomOracleSHA256(point)
    …
```



```
(hunter2, 3)
(12345678, 4)
(qwerty, 10)
```

Offline: Make Dictionary





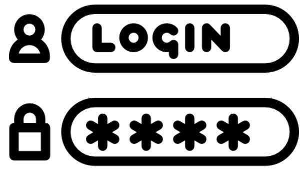
66% Recovery

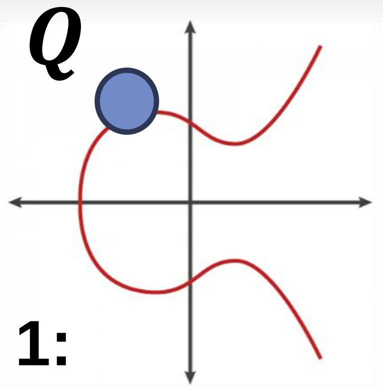
5 Traces

5 sec.

What Can a Malicious Server Do?

***PSI:** Private Set Intersection



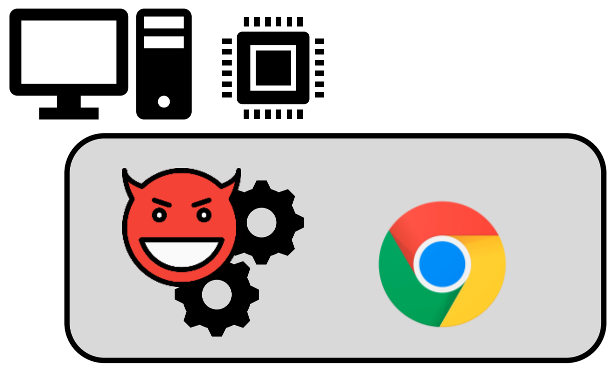


Memory-hard Hashing: Scrypt

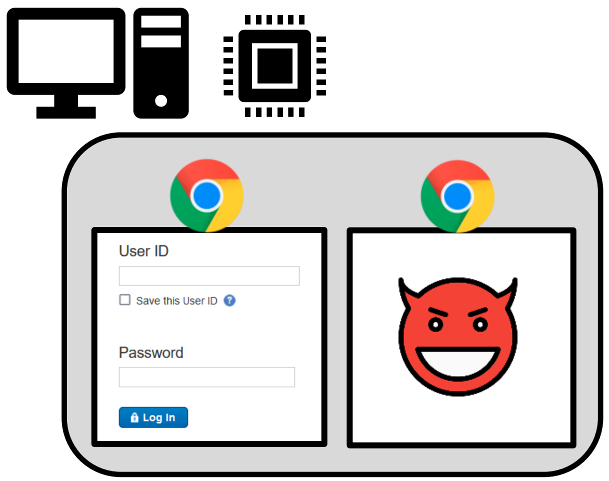
PSI Step 1: Hash-to-curve

$$(P^{a})^{a^{-1}} = P$$

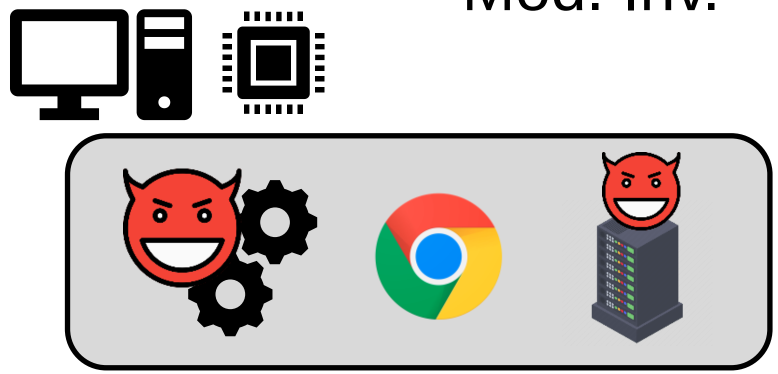
PSI Step 4: Unblind with Mod. Inv.



Native Adversary

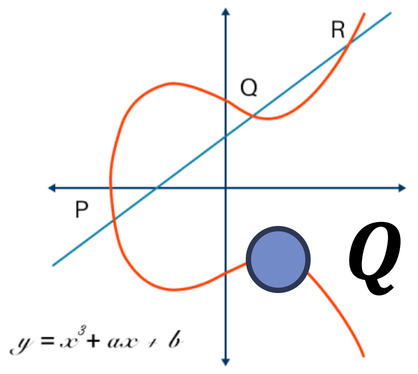


Browser Adversary



Native + Server Adversary

Leaking the Blinded Point



Q^a

$(P^a)^{a^{-1}} = P$

a^{-1}

Q

14M

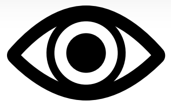
```
hunter2 → (−1, 7)
qwerty  → (5, 0)
abc123  → (−2, −8)
```

BEEA

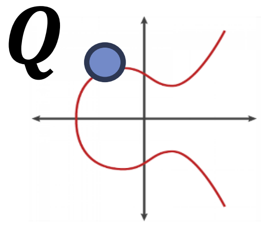
$(Q^a)^{a^{-1}} = Q$

a^{-1} 34 ms

The Final Picture

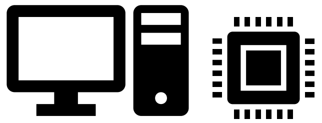


Modular Inversion



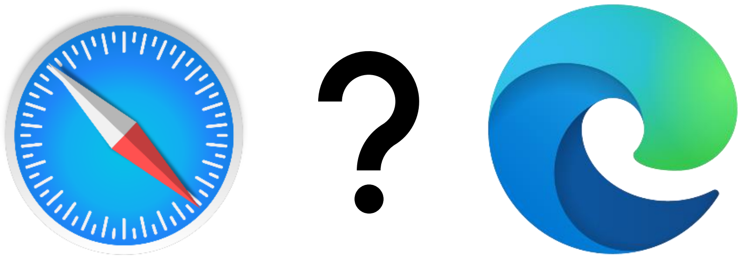
Hash-to-curve

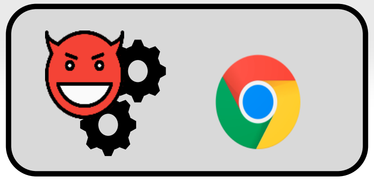
Scrypt



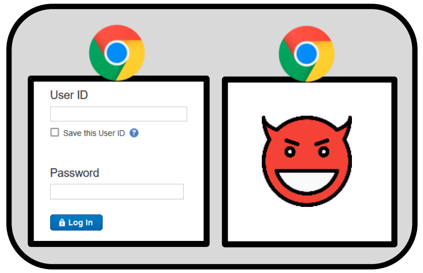
```
foo = arr[secret];
```

Constant-time Implementations!

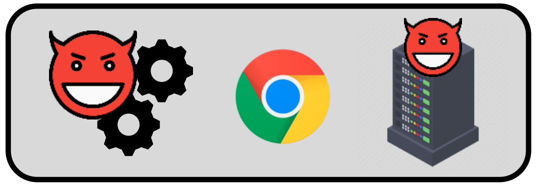




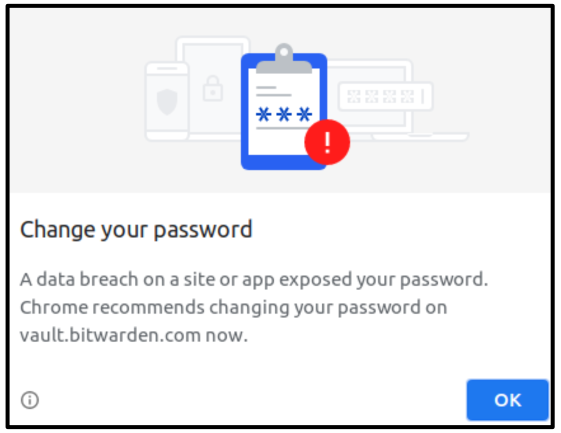
Native



Browser



Server



Change your password

A data breach on a site or app exposed your password. Chrome recommends changing your password on vault.bitwarden.com now.

OK

Thank you for listening!

Jason Kim

nosajmik@gatech.edu