

# Designing Private Ads Measurement for the Web

Guest Lecture for Ni Trieu's Cryptography Class  
11/1/2023

# Team Effort



Benjamin Case  
(Meta)



Ben Savage  
(Meta)



Erik Taubeneck  
(Meta)



Martin Thomson  
(Mozilla)



Richa Jain  
(Meta)



Taiki Yamaguchi  
(Meta, fmr)



Alex Koshelev  
(Meta)



Andy Leiserson  
(Mozilla)



Victor Miller  
(Meta, fmr)

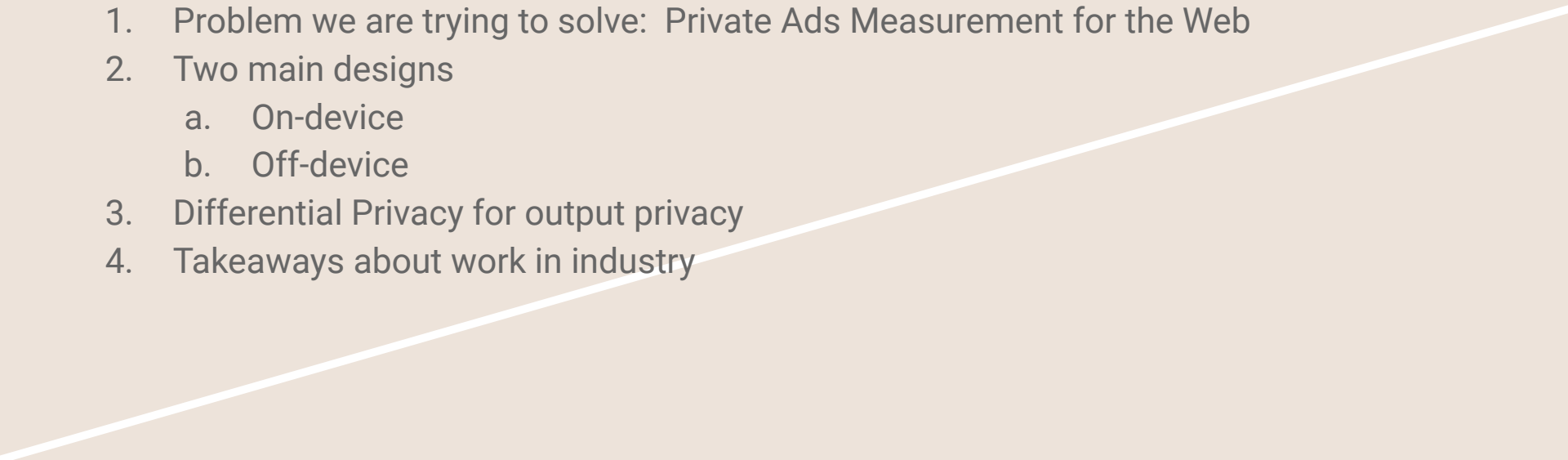


Daniel Masny  
(Meta)

# About Me

- Educational/Career journey:
  - 2016: B.S. in Mathematics, Bob Jones University
  - 2016 – 2020: Ph.D. and M.S. in Mathematics, Clemson University
  - 2020 – current: Research Scientist / SWE at Meta
- Journey through Privacy Enhancing Technologies (PETs):
  - Fully Homomorphic Encryption (FHE)
  - Private Set Intersection (PSI)
  - Multi-Party Computation (MPC)
  - Differential Privacy (DP)

# Outline for today


1. Problem we are trying to solve: Private Ads Measurement for the Web
  2. Two main designs
    - a. On-device
    - b. Off-device
  3. Differential Privacy for output privacy
  4. Takeaways about work in industry
- 

# Advertising Measurement




**Headline News**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin et sem sed metus feugiat porta. Donec pulvinar sem nec leo aliquam tincidunt. Curabitur ut auctor orci. Maecenas augue nisi, tincidunt non erat id, convallis tincidunt ante.

Sunglasses!



Ut id turpis vel sapien fringilla viverra. Donec sed ligula nisl. Nunc sit amet pharetra dolor, quis interdum orci.

<https://sunglasses.example>

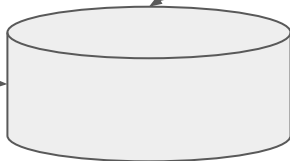


**\$39.99**

Add to Cart

Free shipping on orders over \$50.

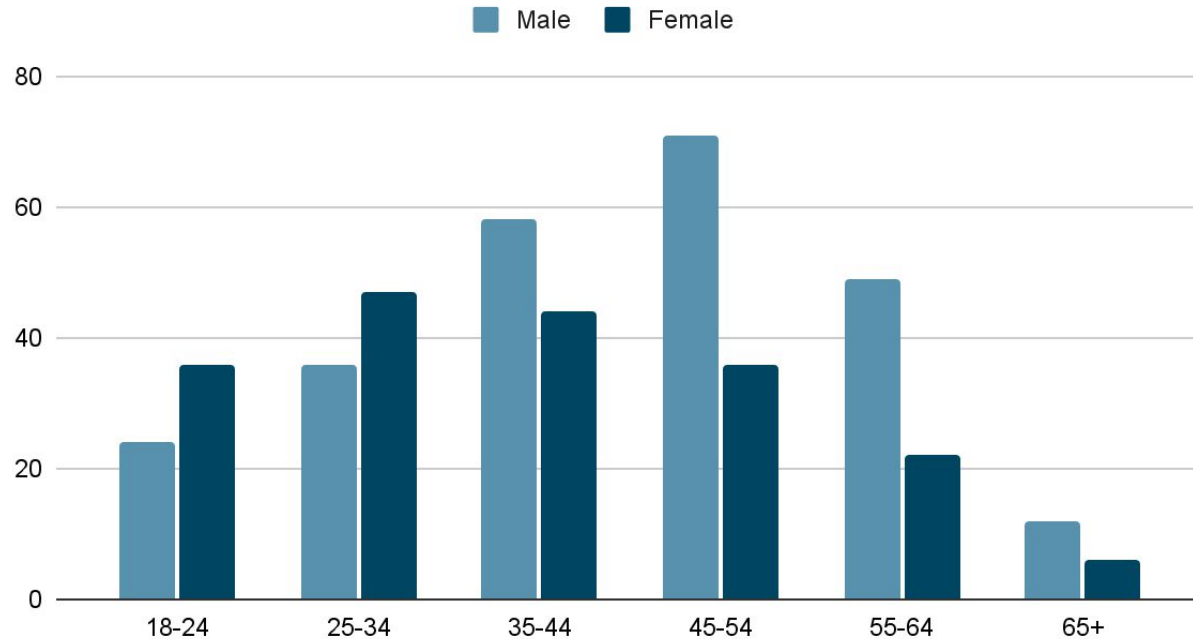
**Product Description**  
Suspendisse varius, metus eget vehicula pulvinar, urna nibh gravida orci, in volutpat ligula purus vitae risus. Morbi feugiat ultricies nisl, blandit pharetra felis ornare quis.



- Conversion Rate
- Cost per Conversion
- Return on Ad Spend

# Outputs we want to see – Measurement

Num Attributed Purchases



# History of Proposals

- *“Privacy Preserving Ad Click Attribution For the Web”*
  - May 22, 2019 by John Wilander (Apple)
  - Later renamed “Private Click Measurement” (PCM)
- *“Building a more private web”*
  - Aug 22, 2019 by Justin Schuh (Google)
  - Later renamed “Attribution Reporting API” (ARA)
- *“Privacy Preserving Attribution for Advertising”*
  - Feb 8, 2022 by Martin Thomson (Mozilla)
  - Always known as “Interoperable Private Attribution” (IPA)
- *“Private Ad Measurement explainer”*
  - Aug 28, 2023 by Luke Winstrom (Apple)

# Threat Model

- Designed not to reveal people's cross context site activity; *even in the face of malicious adversaries.*
- Assume both Publisher and Advertiser sites are colluding
- Any MPC involved should have no single trusted parties.



# Two major design paradigms

## On-device attribution

- Device collects ad impressions and then checks to see if a new conversion is attributed to one
- Sends back secret shares of the result encrypted under the public keys of MPC helper parties.
- Publishers and advertisers can submit these reports from many users grouped by different user properties to the MPC to have them aggregated

## Off-device attribution

- Device generates an consistent identifier which is encrypted and sent to different sites.
- Sites can append data to this identifier (purchase value, breakdowns)
- Sites submit to an MPC which does the full computation of attribution and aggregation

# Some background ... Additive Secret Sharing

```
let secret = 8;
```

```
let p = 31;
```

```
let share_a = RandBetween(0, p);
```

```
let share_b = Mod(secret - share_a, p);
```

`share_a` and `share_b` are randomly distributed numbers in the range  $[0, 31)$ . If you possess just one, or even two of them, you know absolutely nothing about the secret value.

```
share_a + share_b = secret
```

# Additive Secret Sharing – Addition

You have secrets:  $secret_1, secret_2, \dots, secret_n$

Create shares of them  $secret_{i_a}$  and  $secret_{i_b}$  give to Parties A and B.

Then if Party A sums all of their shares

$sum_a = \sum \{ secret_{i_a} \text{ for all } i \}$  (computed modulo  $p$ )

And Party B sums all of their shares

$sum_b = \sum \{ secret_{i_b} \text{ for all } i \}$  (computed modulo  $p$ )

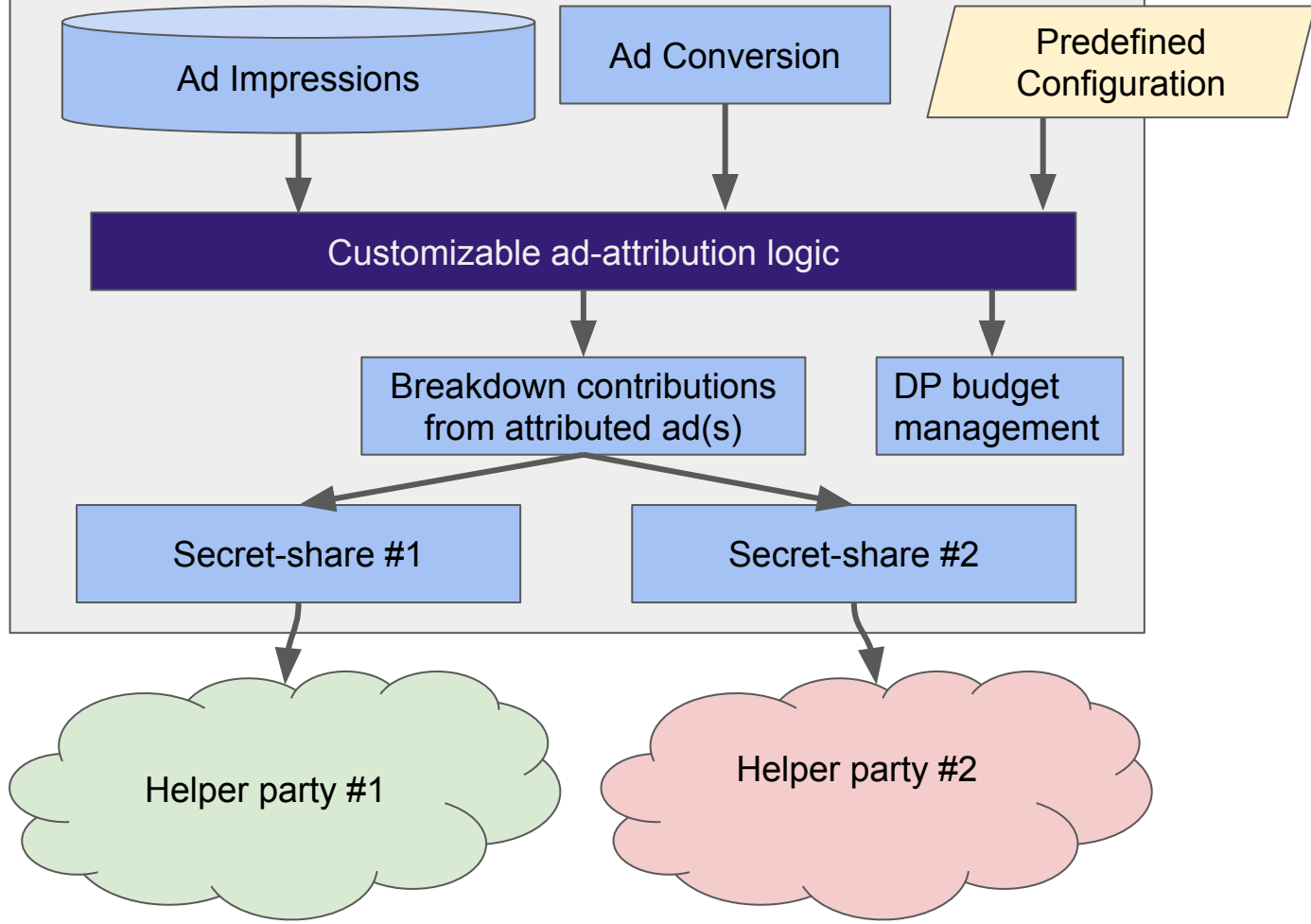
Then

$sum_a + sum_b = \sum \{ secret_i \text{ for all } i \}$

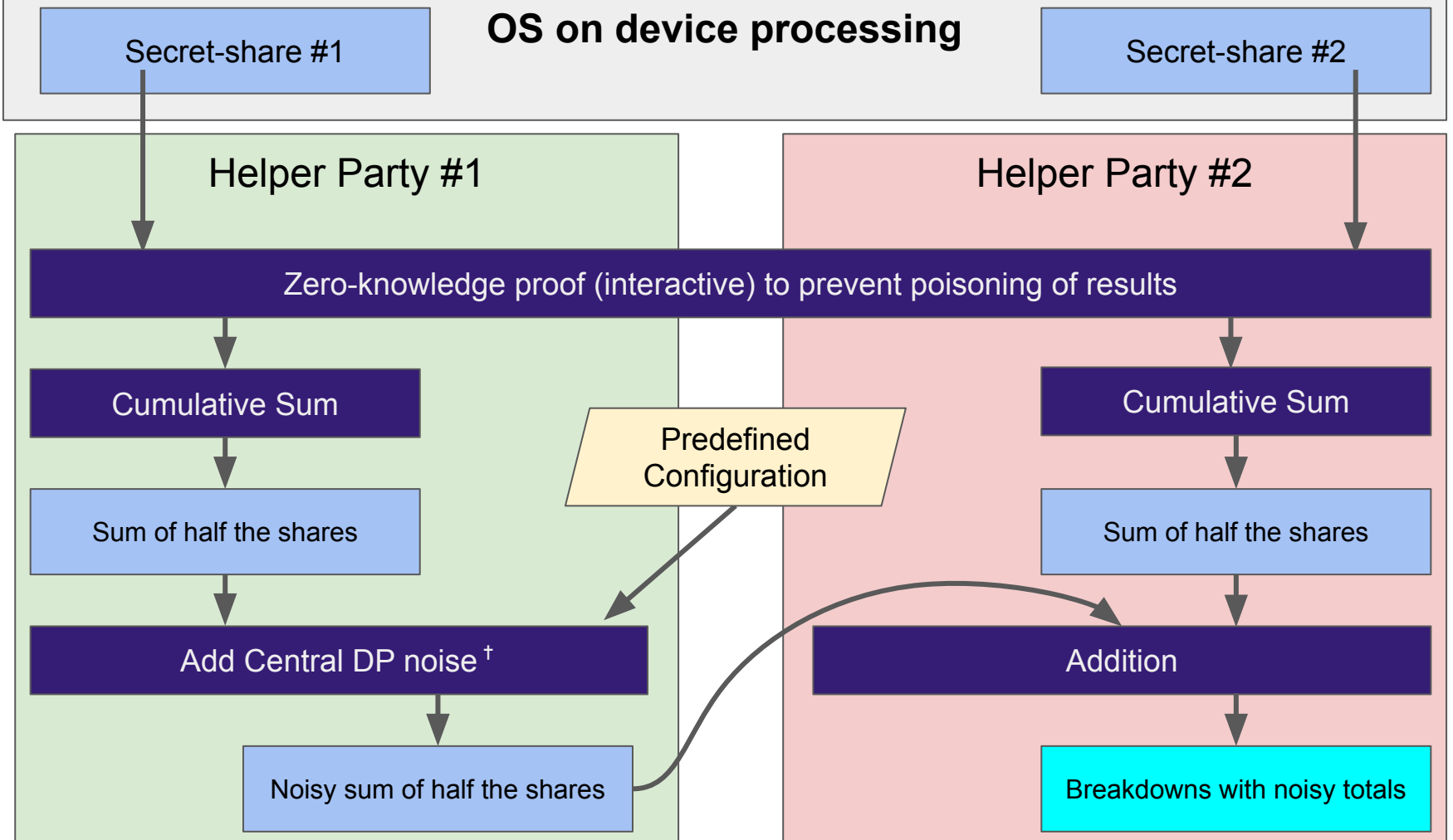
# On-device: Apple's Private Ads Measurement



# OS on device processing



# OS on device processing



<sup>†</sup> Underspecified. Unclear if it's just one party adding noise or an interactive protocol

# Advertiser-side

Ad Impression #1  
occurs

Ad Impression #2  
occurs

Ad Conversion  
occurs

Preconfigured config for  
this "conversion ID"

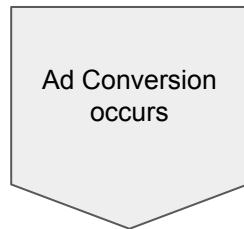
- Cross-publisher  
and multi-touch  
logic
- Campaign ID =>  
breakdown key  
config



Secret-shares  
sent out

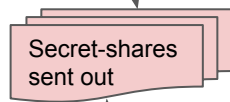
Shares of non-zero  
breakdown contribution

# Advertiser-side



Preconfigured config for this "conversion ID"

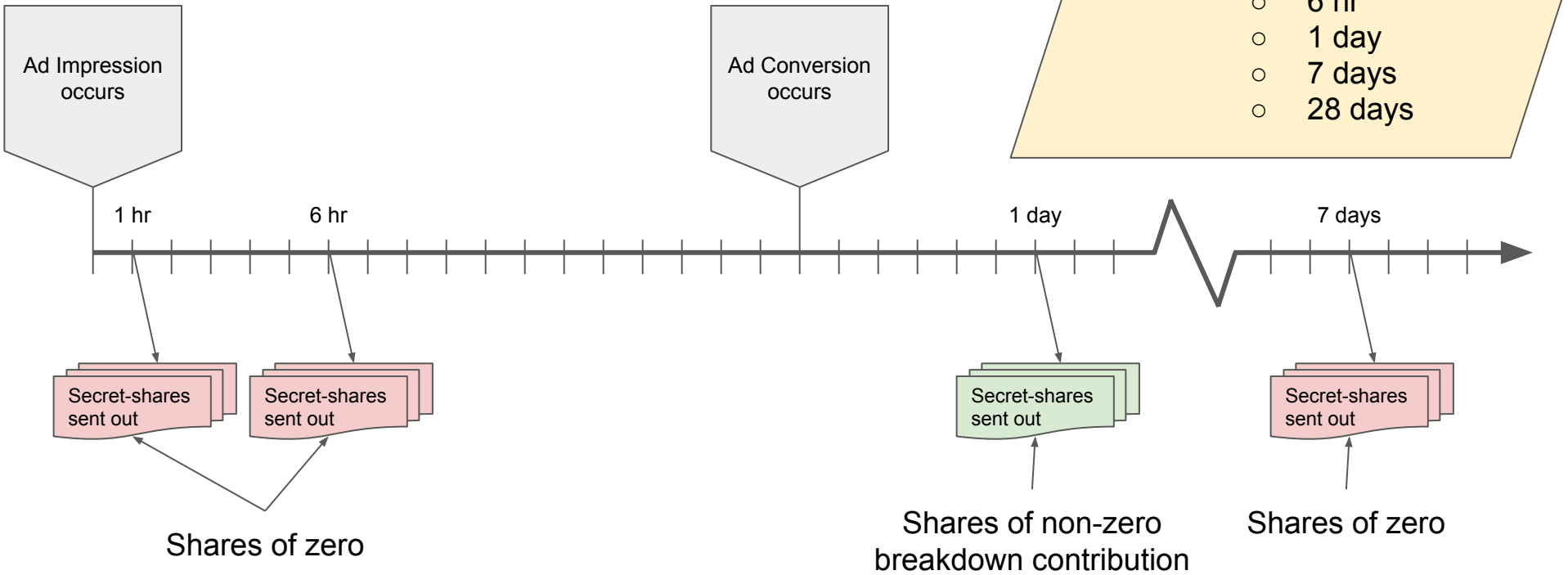
- Cross-publisher and multi-touch logic
- Campaign ID => breakdown key config



Shares of zero  
(or non-zero to "unattributed" bucket)



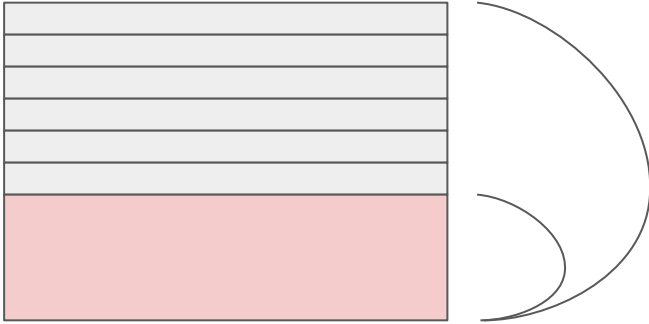
# Publisher-side



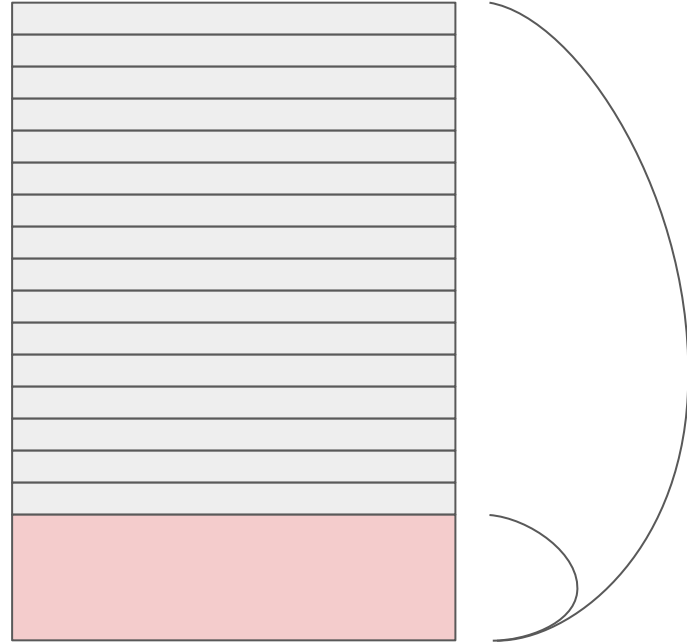
# Collect and Aggregate

- Publishers and Advertisers collect these reports and then when they have enough they submit them to an MPC to have them aggregated.
- Differentially Private noise will be added to each sum, so the more reports they wait for the better signal-to-noise ratio they can get.

Wait less, worse  
signal-to-noise ratio



Wait longer, better  
signal-to-noise ratio



# Off-device: Interoperable Private Attribution



## Recall our additive secret sharing...now with 3 parties

```
let secret = 8;
```

```
let p = 31;
```

```
let share_1 = RandBetween(0, p);
```

```
let share_2 = RandBetween(0, p);
```

```
let share_3 = Mod(secret - share_1 - share_2, p);
```

All of `share_1`, `share_2` and `share_3` are randomly distributed numbers in the range  $[0, 31)$ . If you possess just one, or even two of them, you know absolutely nothing about the secret value.

# Replicated Secret Sharing

Helper 1: (share\_1, share\_2)

Helper 2: (share\_2, share\_3)

Helper 3: (share\_3, share\_1)

Consider the “additive secret sharing” from the last slide, and give each helper two of the three shares

That’s not enough information for any one helper to reconstruct the secret

This can make **multiplication** much simpler

Our current prototypes use this type of secret sharing

# Multiplication with replicated secret shares

Suppose 3 parties hold replicated secret sharings of secret values  $x$  and  $y$

- Helper 1:  $(\text{share\_x\_1}, \text{share\_x\_2}); (\text{share\_y\_1}, \text{share\_y\_2})$
- Helper 2:  $(\text{share\_x\_2}, \text{share\_x\_3}); (\text{share\_y\_2}, \text{share\_y\_3})$
- Helper 3:  $(\text{share\_x\_3}, \text{share\_x\_1}); (\text{share\_y\_3}, \text{share\_y\_1})$
-

# Performance

- The bottleneck in such systems tends to be communication between the MPC helper nodes
- CPU tends not to be a bottleneck
- **Addition** of secret shared numbers requires no communication between the nodes
- **Multiplication** of secret shared numbers **does** require communication between the nodes
- So it's important to minimize the number of multiplications you need to do



# The cost of a multiplication

In the honest majority with 3 parties setting, the communication cost of performing a multiplication is:

**Semi-honest setting:** 1 number exchanged per helper

**Malicious setting:** 2 numbers exchanged per helper

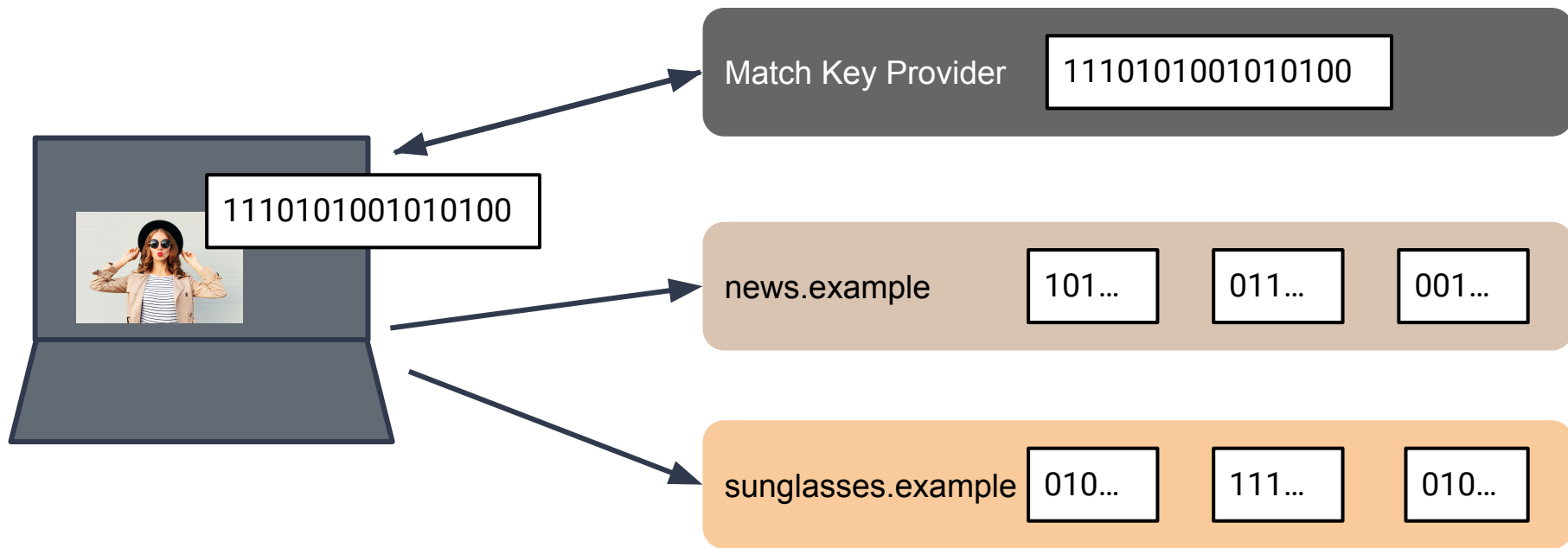
- Upgrades to malicious in dishonest majority are  $\sim 10x$  the semi-honest cost

# getEncryptedMatchKey(provider)

```
fn get_encrypted_match_key(provider) -> [CipherText; 3] {  
    (  
        Encrypt(pk_1, mk_share_1, mk_share_2),  
        Encrypt(pk_2, mk_share_2, mk_share_3),  
        Encrypt(pk_3, mk_share_3, mk_share_1)  
    )  
}
```

where  $pk_i$  is the  $i$ th Helper Party's public key.

# Publisher and Advertiser Sites



# Report Collector

## Publisher Site:

Campaign: 12	ts: 06:32	101...	011...	001...
Campaign: 92	ts: 09:15	010...	101...	110...
...	...	...	...	...

## Advertiser Site:

Value: \$25	ts: 07:41	010...	111...	010...
Value: \$12	ts: 11:04	100...	011...	000...
...	...	...	...	...

## Report Collector:

Campaign: 12	ts: 06:32	1
Value: \$25	ts: 07:41	0
Campaign: 92	ts: 09:15	0
Value: \$12	ts: 11:04	1
...	...	...

# Helper Party Networks

Report Collector

Helper Party #1:

579...	793...	1	101...
811...	544...	1	010...
880...	419...	0	010...
292...	437...	0	100...

...

Helper Party #2:

173...	013...	0	011...
960...	635...	1	111...
621...	835...	0	101...
379...	197...	0	011...

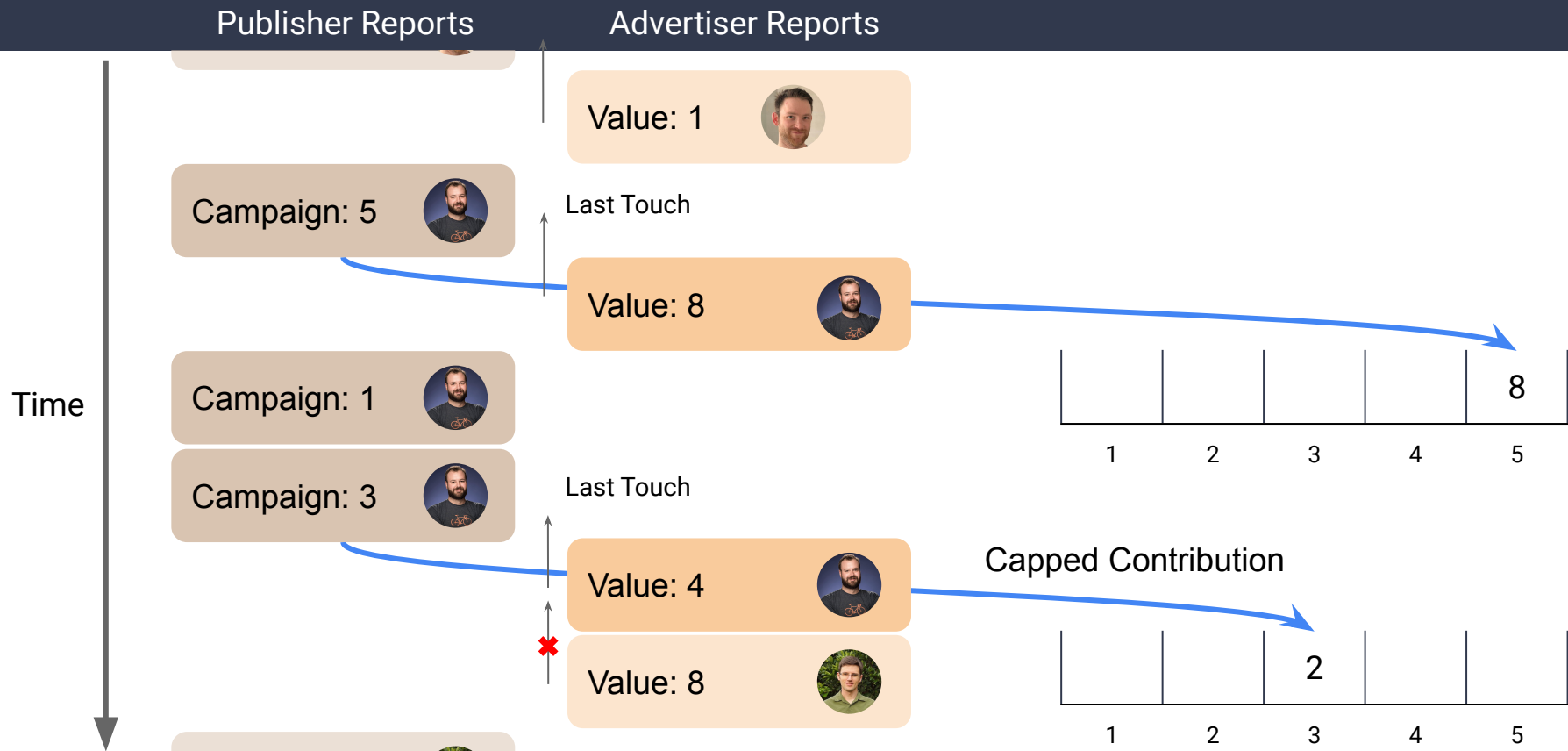
...

Helper Party #3:

282...	365...	1	001...
373...	712...	1	010...
926...	410...	0	110...
162...	835...	1	000...

...

# Ideal Functionality



# Ideal Functionality

				8
		2		
				9
1				
-3	4	-1	-2	3
$\Sigma$	$\Sigma$	$\Sigma$	$\Sigma$	$\Sigma$
-2	4	1	-2	20
1	2	3	4	5

All Capped Contributions

Generate Differentially Private Noise

Aggregate

Release Output Histogram

# How the updated MPC protocol works

1. First, helpers receive a big list of secret shared events
2. Next, they sort these events by match\_key, ~~then timestamp~~
3. Then, they run an “oblivious attribution algorithm” over these events
4. At this point, we need to “cap” the maximum contribution each individual person can contribute to the output
5. Then, they sum up the secret shared “conversion values” (per breakdown key)
6. After this, they generate some random noise and add it to these totals
7. Finally, they “reveal” these totals to the API caller (i.e., give them all 3 shares)



# Oblivious Attribution Algorithms

- In principle, almost any type of attribution heuristic is possible
  - Only limited by the data that is provided in events
  - Simple multi-touch (e.g. equal credit last 3 touches) is definitely possible
  - Addition is cheap
  - Multiplication is a *little* expensive
  - Division / exponentiation / etc. is going to cost you =)
- We have shared a specific algorithm which performs “last-touch attribution”
  - It only requires addition and multiplication
  - It’s very inexpensive; only requiring a few multiplications per event

# Explaining Differential Privacy

# Privacy Example

Suppose you are a student needing to do teacher evaluations.

- “Would you want to take a class with this professor again?” Yes/No

Suppose there are 10 students in the class

You don't like your professor. Plan to respond “No” ... but you do NOT want your professor to know how you answered this question.

The department adds up all the responses of 10 students and discloses the total.

Does that preserve your privacy?

# K-anonymity

This is called K-anonymity; in this example  $K=10$ .

Think of it as “hiding in the crowd” – most of the time this is **just fine**.

Imagine the total is 4.

If that is the **ONLY** information your professor gets, that there were 4 people who would not want to take a class with her again, then your privacy is preserved. She can't know how you voted.

# Differential Privacy is a pessimist

Differential Privacy is focused on **worst case scenario** if everything went wrong

Worst case scenario:

- What if all 10 people voted “No”. Then the total would be 0...and your professor would know exactly how ALL students voted
- What if the department also disclosed the “Total from people who’ve been enrolled more than a year”, and that was the other 9 people...
- ...and that total was also 4...
- Even though both statistics were from a group of at least 9 people, your professor can learn exactly how you voted by subtracting the two totals

**Differential Privacy is all about protecting your privacy even in these kinds of worst case scenarios!**

- The way to do that, is to add some randomness to the total
- The randomness can sort of “mask” any one person’s contributions.
- You can hide in the random noise!

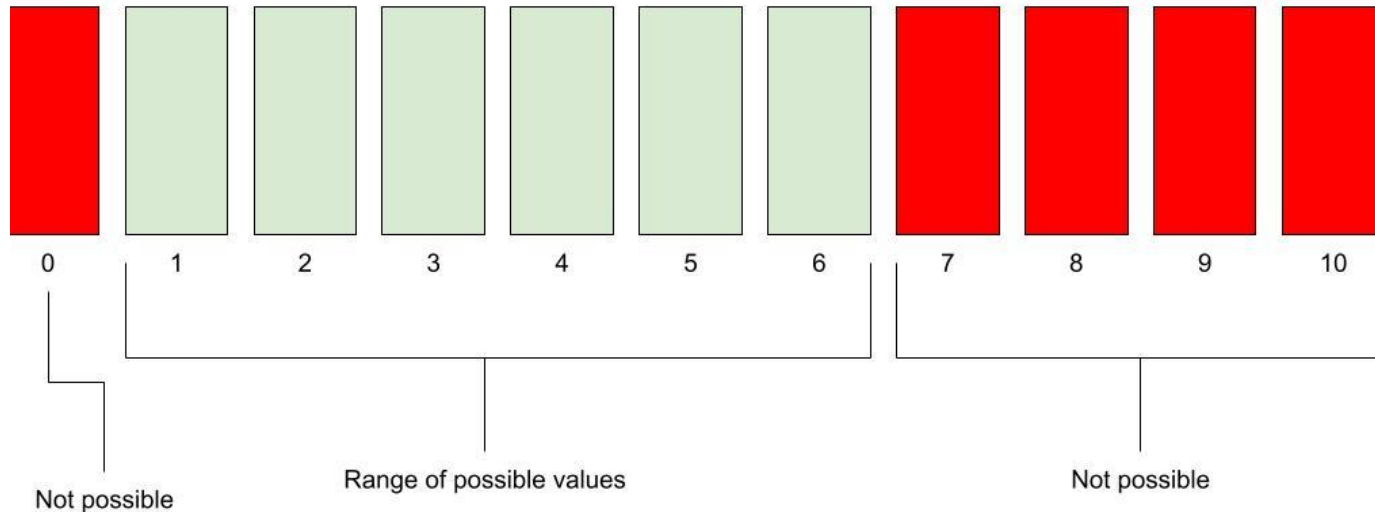
# Updated example

- The department takes the accurate total, and then they flip **5 coins**.
- They add on the number of “heads”.
- They report this total

Let's say the total is now "6"...how should we think about what that means?

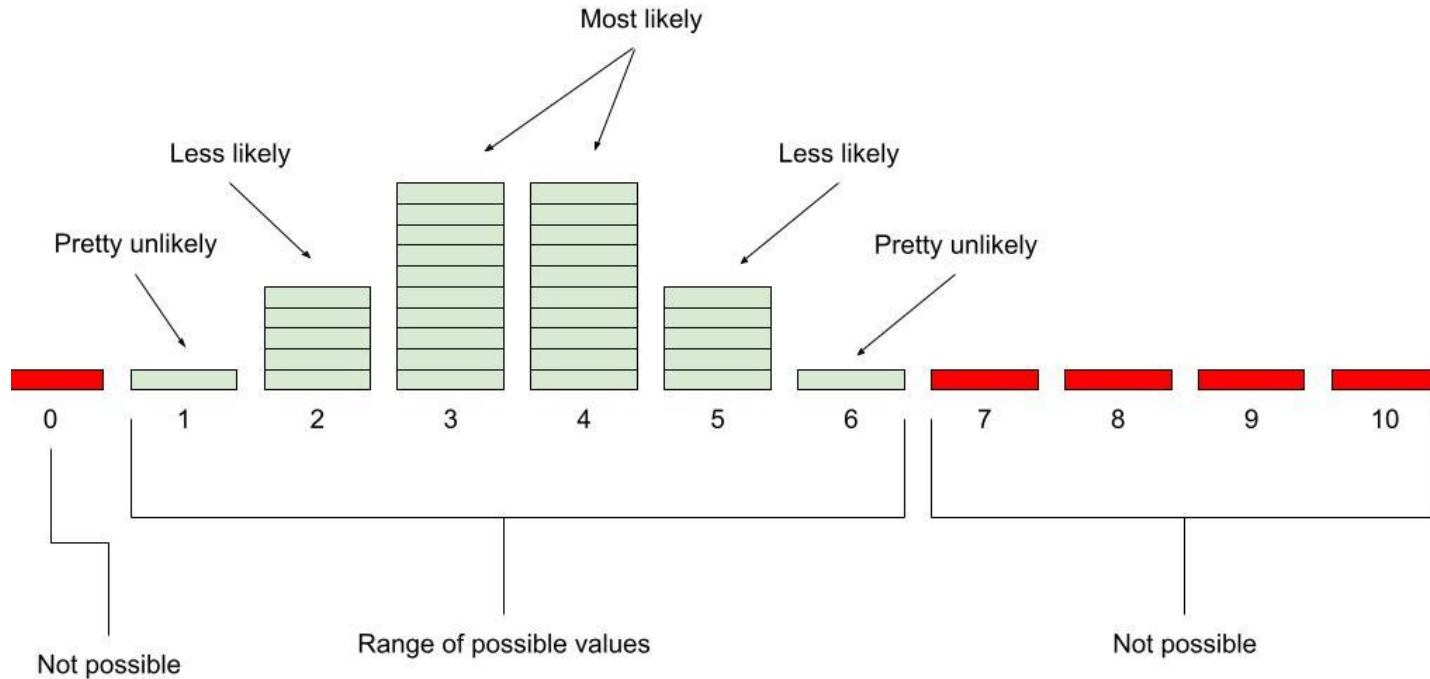
We know that the actual number of people who voted "Yes" is somewhere between 1 and 6.

- The minimum is 1, because even if all 5 coins here "heads" we are max adding 5
- The maximum is 6. If 6 people all voted "yes", and we got 5 "tails", the answer would be 6.





# Not all outcomes equally likely



# Worst case scenario, revisited

- As mentioned earlier, K-anon fails if everyone votes “No” and the total is 0.
- What would happen now?
- Instead of reporting 0, our new “differentially private” system would flip 5 coins and add the number of heads to that total. So we would wind up reporting something between 0 and 5.
- What’s the worst case scenario? We get “tails” 5 times in a row, and the total is still 0.
- OK, in this case the system still failed, but now it’s a LOT less likely. There was only a 1 in 32 chance that you’ll get 5 “tails” in a row.

# What's the worst case scenario where it didn't fail completely?

Suppose the professor knows all the other 9 students voted "Yes". The result is 14. You voted "No". The professor can't learn for sure what you voted but can see how likely it was you voted "No" or "Yes"

- Case 1: You voted "Yes" and 4 coins heads
- Case 2: You voted "No" and 5 coins heads

$$\text{Prob}(\text{Case 1}) / \text{Prob}(\text{Case 2}) = 5$$

What is more likely? It is more likely that you voted "No" – 5 times more likely. Your professor knows it is 5 times more likely that you said "No" than "Yes"

This “5x ratio” is important here. This tells us *how much differential privacy* you have. How much plausible deniability you have.

# Pure Differential Privacy

More formally, suppose we have an algorithm  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ . Consider any two datasets  $X, X' \in \mathcal{X}^n$  which differ in exactly one entry. We call these *neighbouring datasets*, and sometimes denote this by  $X \sim X'$ . We say that  $M$  is  $\varepsilon$ -*(pure) differentially private* ( $\varepsilon$ -(pure) DP) if, for all neighbouring  $X, X'$ , and all  $T \subseteq \mathcal{Y}$ , we have

$$\Pr[M(X) \in T] \leq e^\varepsilon \Pr[M(X') \in T],$$

where the randomness is over the choices made by  $M$ .

# Approximate Differential Privacy

**Definition 1** (Approximate Differential Privacy). *An algorithm  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$  is  $(\epsilon, \delta)$ -differentially private (i.e., it satisfies approximate differential privacy) if, for all neighbouring databases  $X, X' \in \mathcal{X}^n$ , and all  $T \subseteq \mathcal{Y}$ ,*

$$\Pr[M(X) \in T] \leq e^\epsilon \Pr[M(X') \in T] + \delta.$$

In our example, you had privacy preserved with  $e^\epsilon = 5$  or  $\epsilon = 1.609$ .  
Also  $\delta$  was  $1/32$ .

# Takeaways

1. System design – need to understand the actual problem deeply to design an impactful solution
2. Cryptography and PETs are valuable tools – learn all you can!
3. Working in industry
  - a. Focused on having real world impact
  - b. Shifting focus to follow the most important things – enjoy learning new things!

# References

1. Interoperable Private Attribution:  
<https://github.com/patcg-individual-drafts/ipa>
2. Apple's Private Ad Attribution:  
<https://github.com/patcg-individual-drafts/private-ad-measurement>
3. Differential Privacy: <http://www.gautamkamath.com/CS860-fa2020.html>