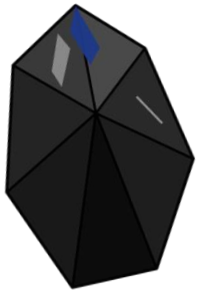
OBSIDIAN

Secure Vickrey Auctions for Online Advertising

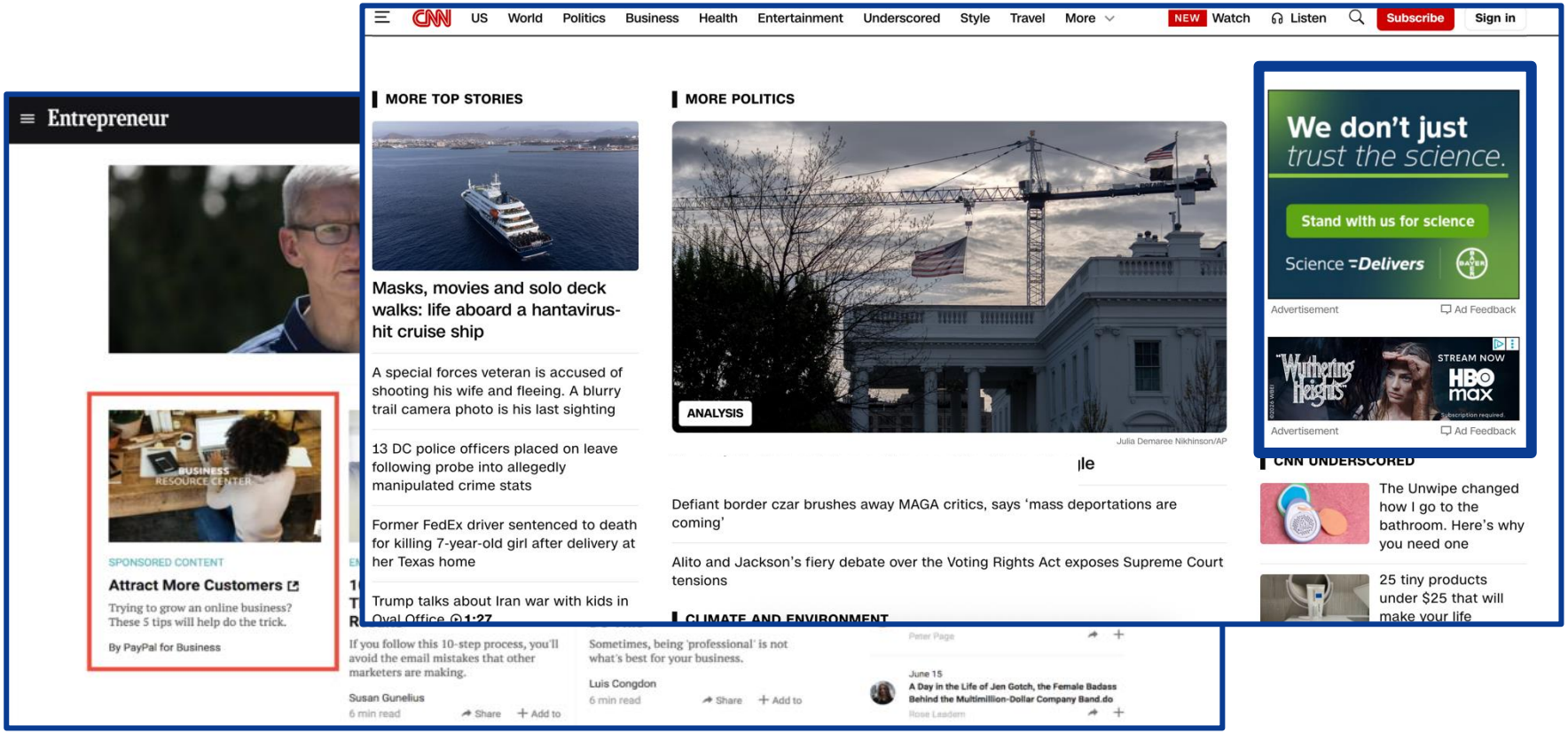


Archit Bhatnagar · Yunming Xiao · Ang Chen · Amrita Roy Chowdhury

NSDI 2026

Renton, WA

We see Advertisements all around on the Web



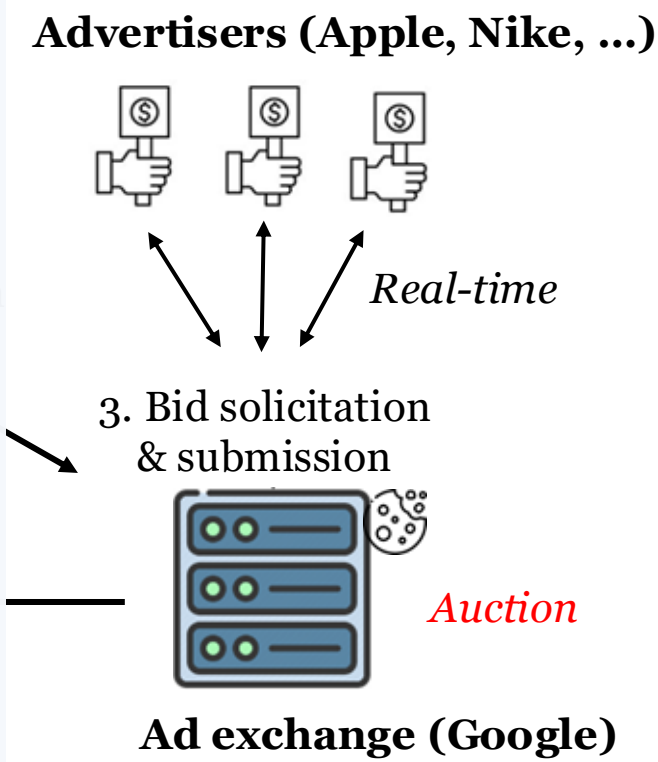
How does a web advertisement get to you?

Web Ad auctions are latency-critical

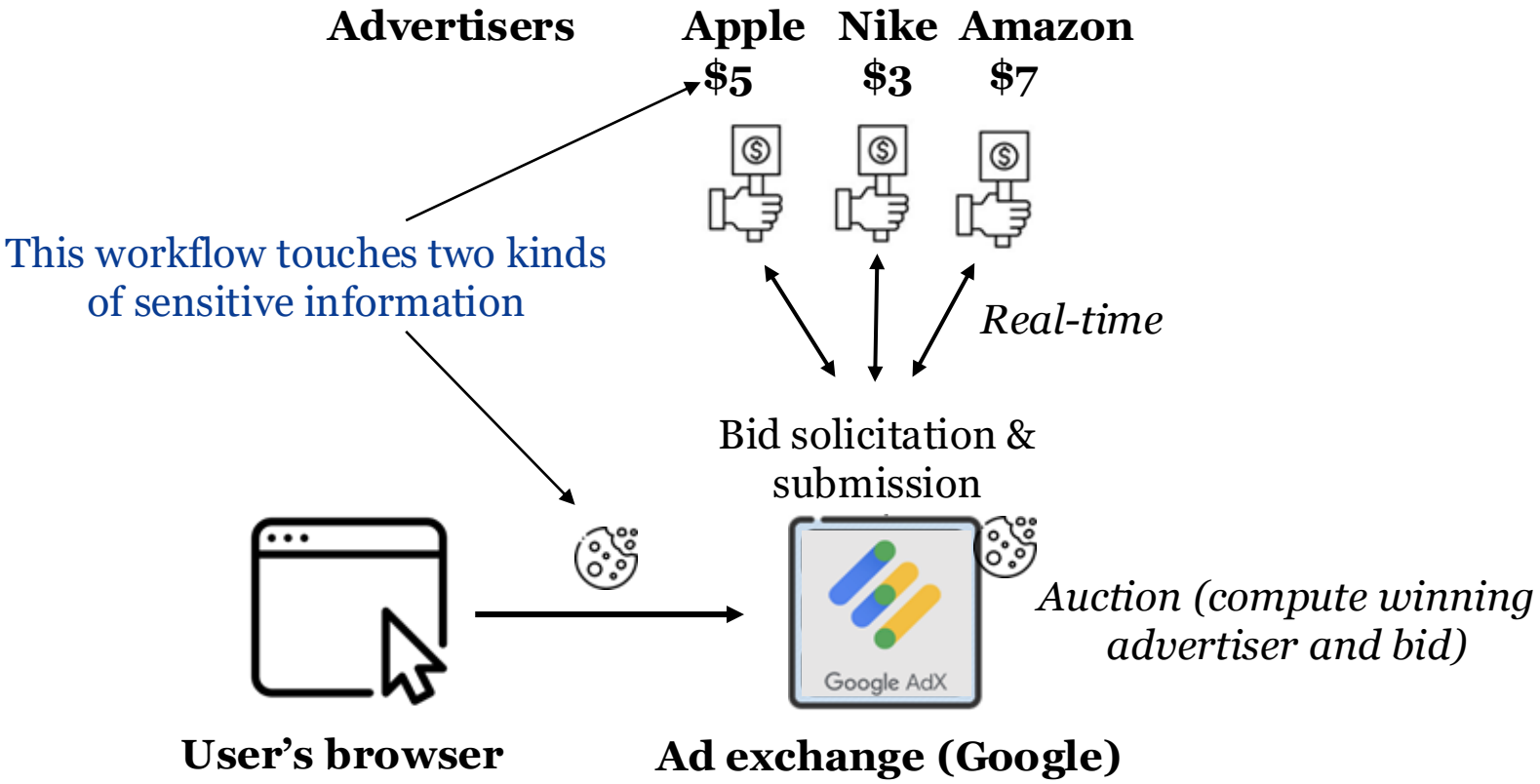
→ Sub-second budget for an entire *distributed protocol*.

Advertisements are critical economic backbone for tech giants

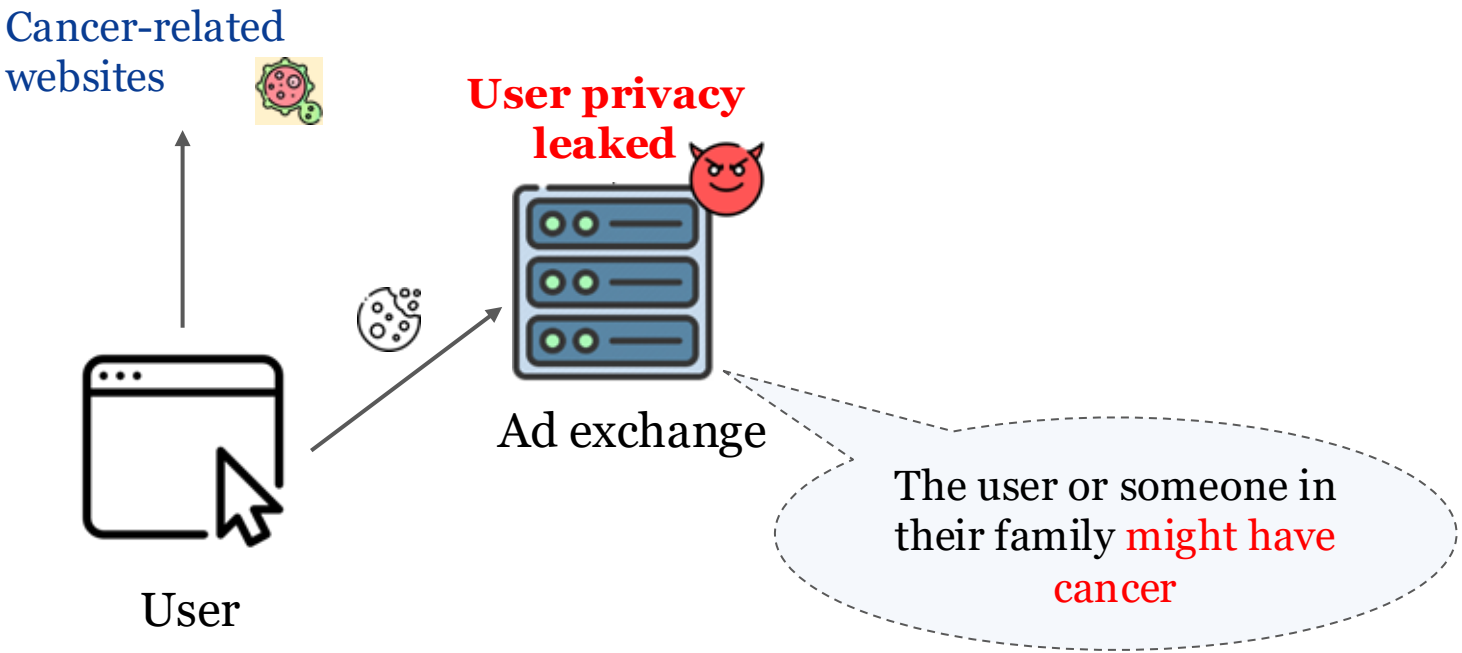
→ Online Ad revenues touched $300B in the US alone (2025)



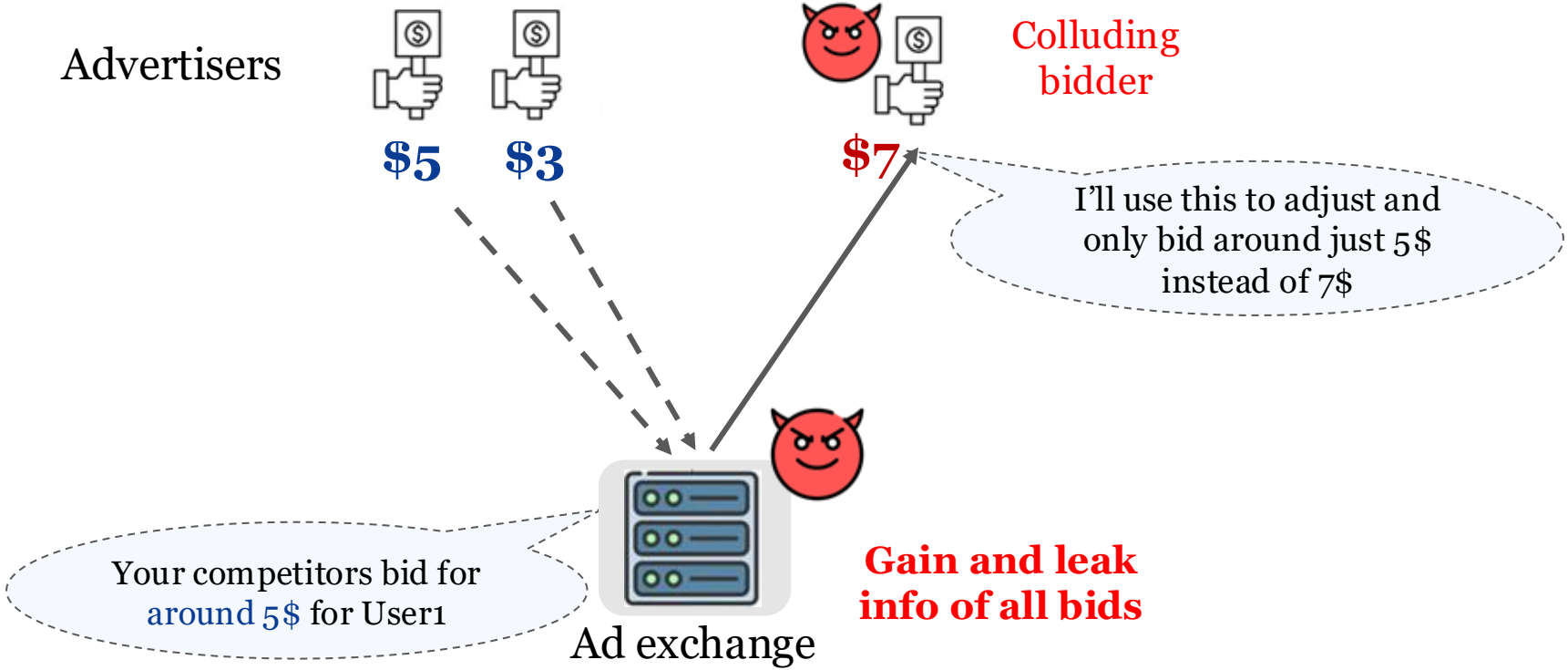
Ad auctions are a critical component



User privacy leaks are possible



And bid confidentiality is at risk too



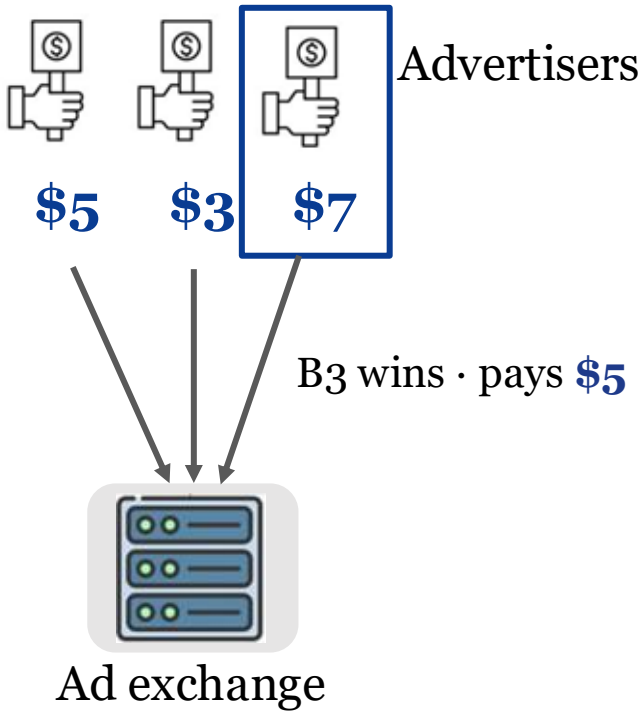
Vickrey auctions for online ads

(a.k.a. second-price)

The winner is the one bidding highest, they pay second-highest bid's price

Economic fair and incentivize higher bidding

Used in ad-exchanges and online auctions (ebay[1])



1. https://community.ebay.com/t5/Bidding/td-p/34065315

How do we conduct secure Vickrey auctions for online advertising?

And what are the guarantees we want?

SOTA leaves gaps

Theoretical Frameworks

Secure Vickrey Auctions with Rational Parties (CCS'24)

- Decentralized → inefficient
- Correctness doesn't hold under malicious setup

Practical Approaches

Addax (NSDI'23)

- Leaks highest bid for Vickrey auctions
- Weaker threat model

Industry Proposals

Google Fledge & GDPR (third-party cookies)

- Proposed to get rid of third party cookies
- Don't address bid confidentiality

Nothing offers **practical, malicious-secure** Vickrey auctions for the modern Web

What do we want from Secure Vickrey auctions?

Bid confidentiality

no bids except the second-highest are revealed

Bidder anonymity

only the winner's identity is revealed

Low latency

hundreds of milliseconds, end-to-end

Compatibility

fits modern in-browser ad infrastructure

Key Insights

Novel combination of Crypto primitives embedded in the existing workflow

1 Bid confidentiality

- Efficient secure auction computation
- encoding, decoupling identities from bids

2 Well-formed bids

- Leveraging correct by design crypto primitives
- validate offline, transform online

3 Bidder anonymity

- lightweight ring signatures, zero-knowledge checks
- public anonymous bulletin boards

Today's talk

1

Bid confidentiality

- Efficient secure auction computation
- encoding decouples identity from bid

2

Well-formed bids

- Leveraging correct by design crypto primitives
- validate offline, transform online

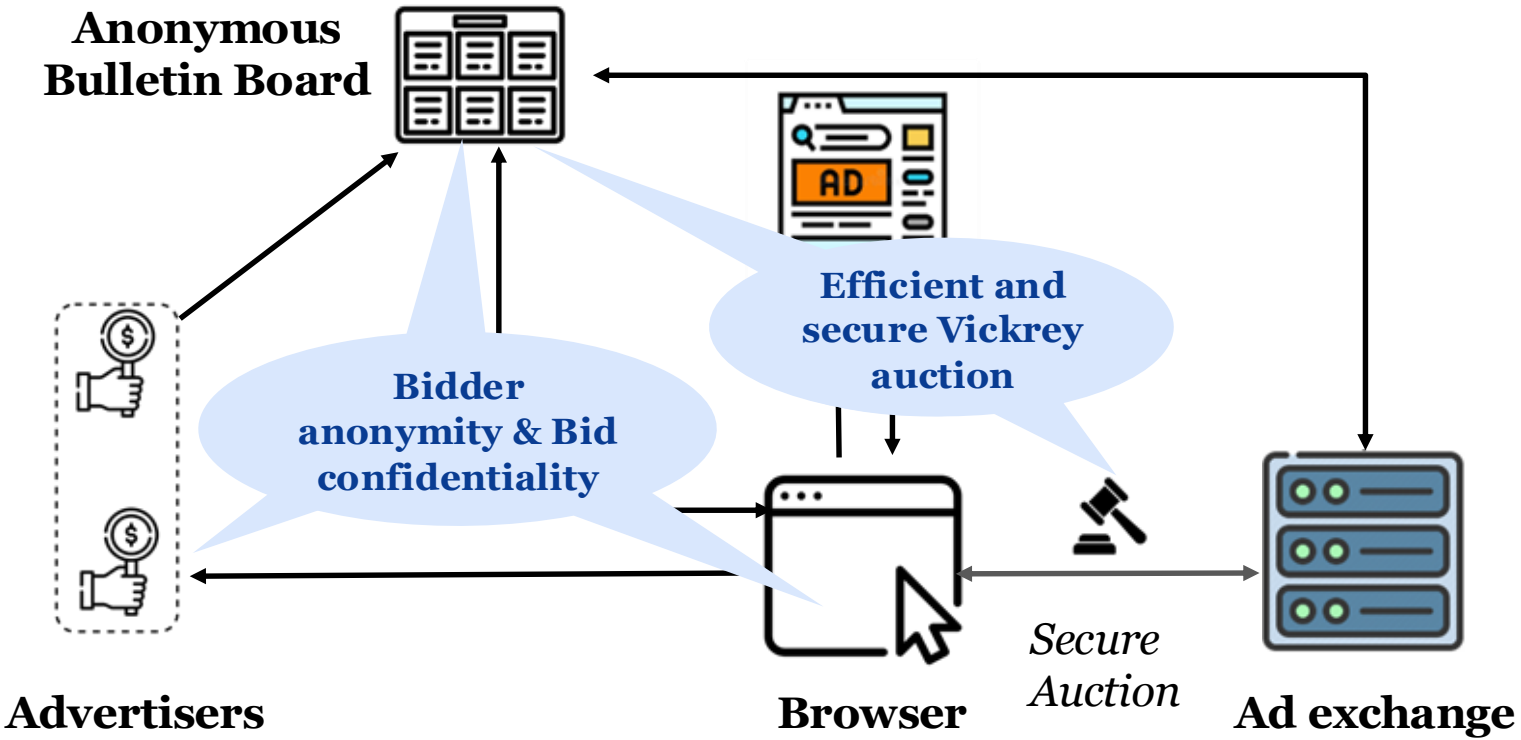
3

Bidder anonymity

→ lightweight ring signatures

→ anonymous bulletin board for legitimacy

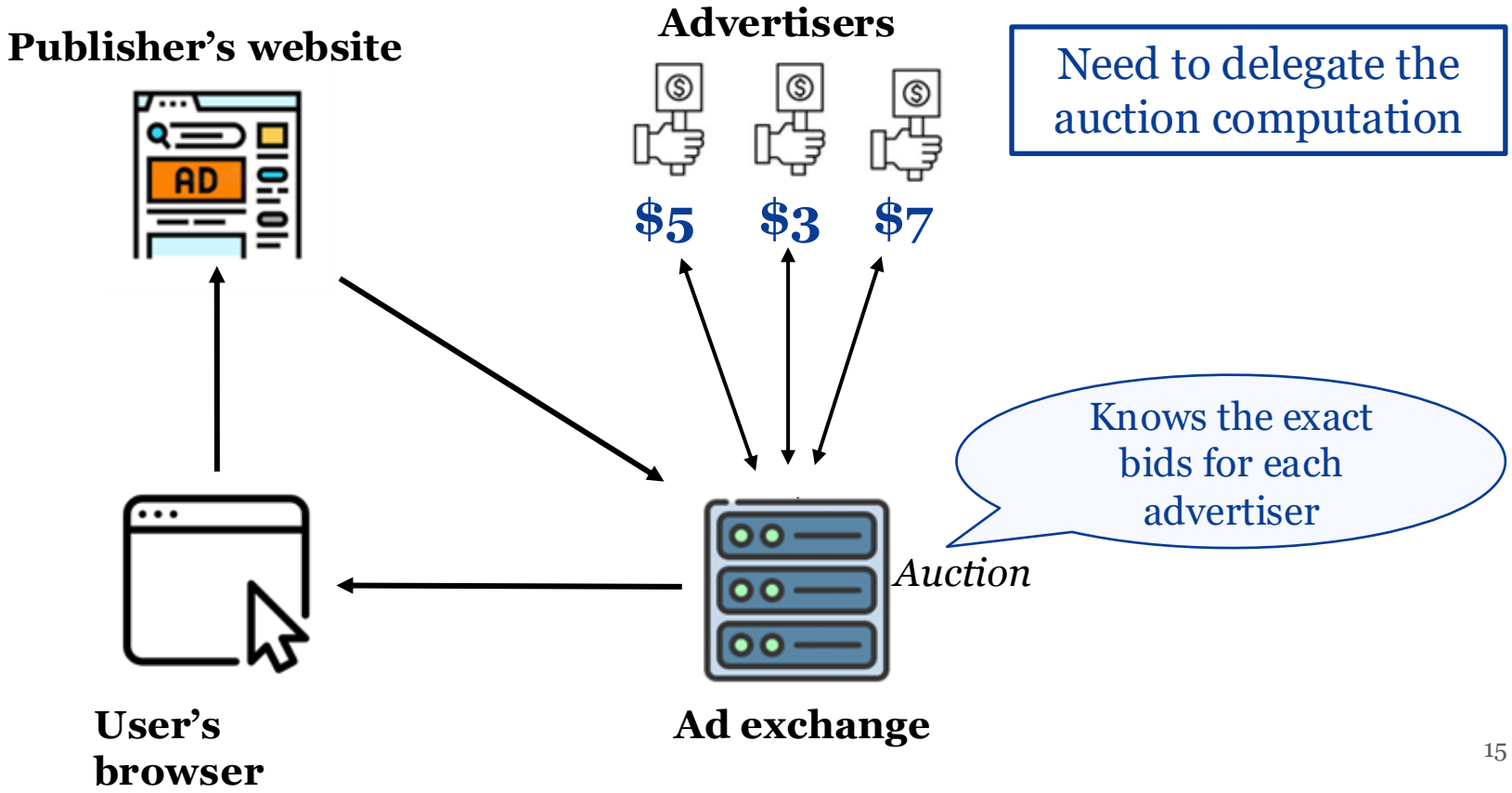
Obsidian at a glance



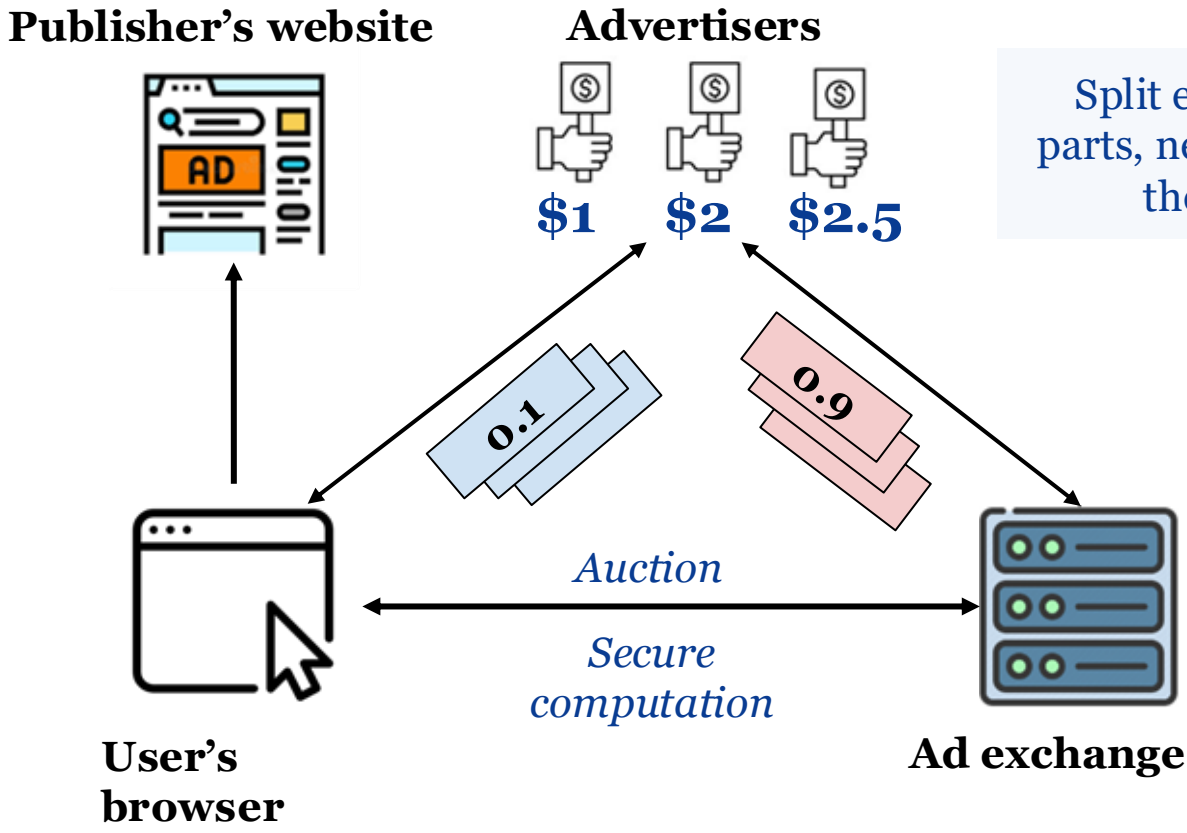
Rest of the talk…

1. Challenges & Solutions (around Bid Confidentiality)
2. Overall Workflow
3. Evaluations

Challenge 1 : Bid confidentiality



Solution: Splitting the computation



Split each bid into 2 parts, neither party sees the exact bid

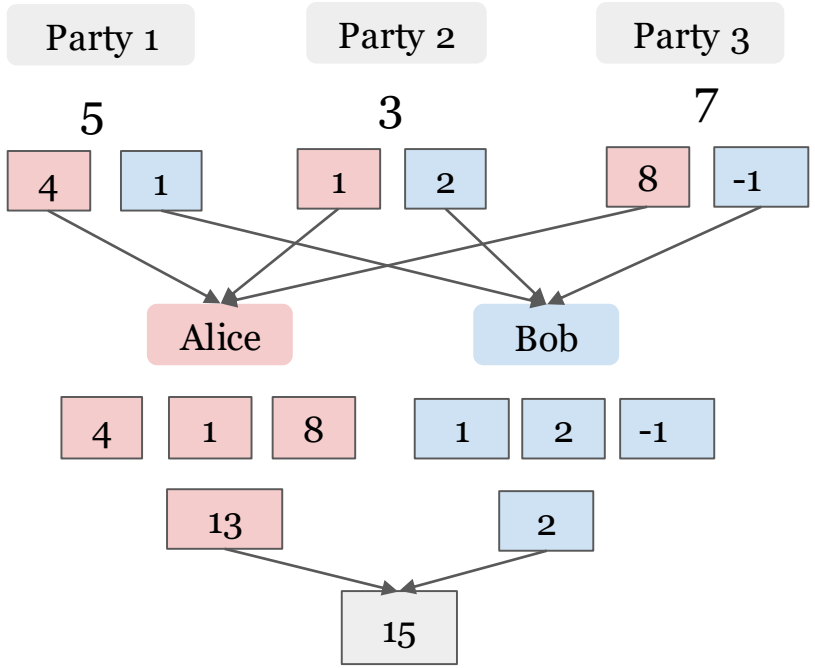
Delegated, Multi-party computation

Goal: jointly compute a function over private inputs, no party learns others' inputs beyond the output.

Function SUM()

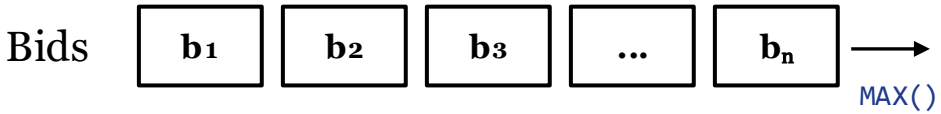
Linear Shares

Delegated MPC



Vickrey under MPC is expensive

Naive Vickrey under MPC (find highest bidder, second highest bid):



→ *find MAX(), remove winner, run MAX() again for the second-highest bid*

Naive computation involves a series of sorting operations on the bids under MPC

Linear ops on shares are **cheap**, comparisons are **expensive** (require rounds of communication between parties)

Cost

Communication rounds scale with the number of bidders, **too slow** for the Web.

Efficient auction algorithm using encoding

Reverse-unary encoding

For bid b over a domain of size d:

`E(b) = 1 1 1 1 0 ... 0 0`
with 1 at index i if i ≥ b, else 0

Compute second-highest bid using column-wise sums

Index from the right with

`col_sum == num_bidders - 1`

→ that index is the **second-highest bid**

Identify the winner

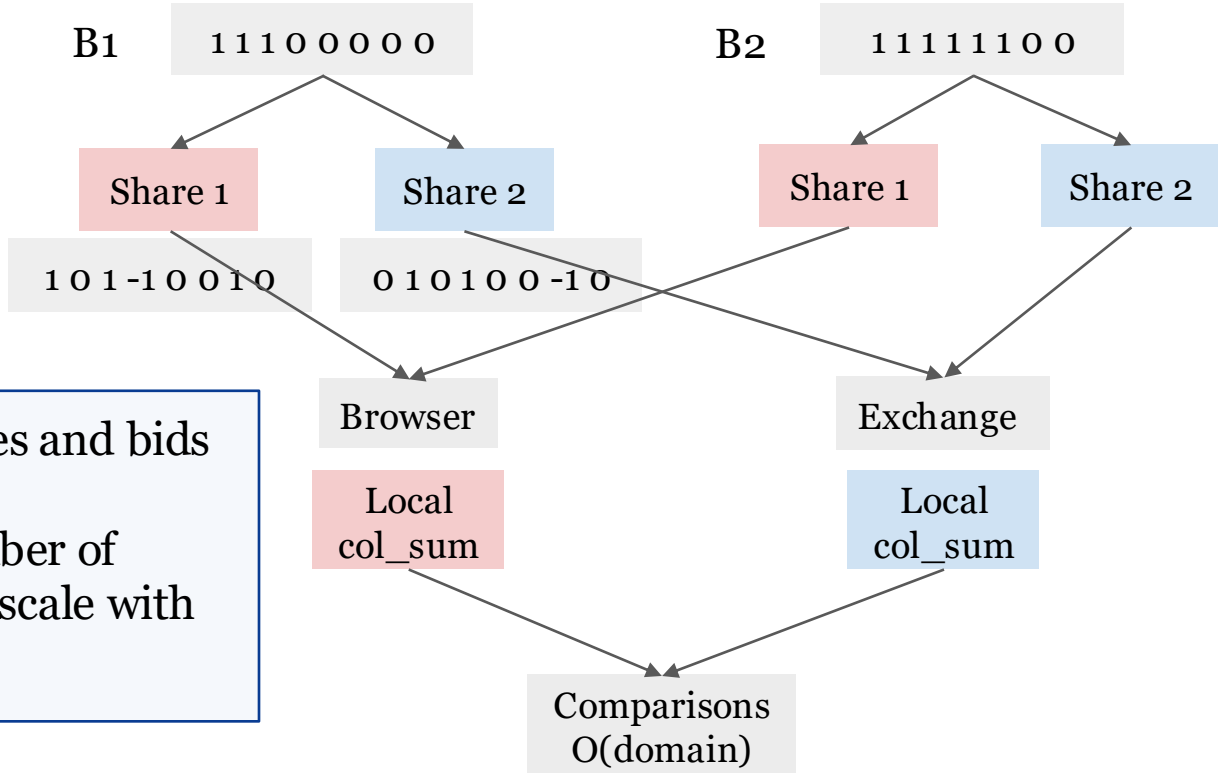
Bidder (row) with a **'0'** at that index

Bidder	8	7	6	5	4	3	2	1
Bidder 1 (bid=5)	1	1	1	1	0	0	0	0
Bidder 2 (bid=3)	1	1	1	1	1	1	0	0
Bidder 3 (bid=7)	1	1	0	0	0	0	0	0
Bidder 4 (bid=2)	1	1	1	1	1	1	1	0
Column Sum	4	4	3	3	2	2	1	0

Column sum at an index is equivalent to number of bidder with utmost that bid

Putting it together

split bids, run encoding, batch comparisins under MPC



Decoupled identities and bids

MPC-friendly Number of comparisons don't scale with number of bidders

Challenge 2: But, what if a bidder lies?

Well-formed encoding (b=5):

1 1 1 1 0 0 0 0

Malformed (malicious):

1 0 1 1 0 0 1 0

Under MPC, parties only see *shares* of the encoding — they can't tell whether it's well-formed.

Consequence

Column sums become wrong, the algorithm picks the wrong winner.

Naive fix

Validate the encoding under MPC online — but that's **expensive** and blows the latency budget.

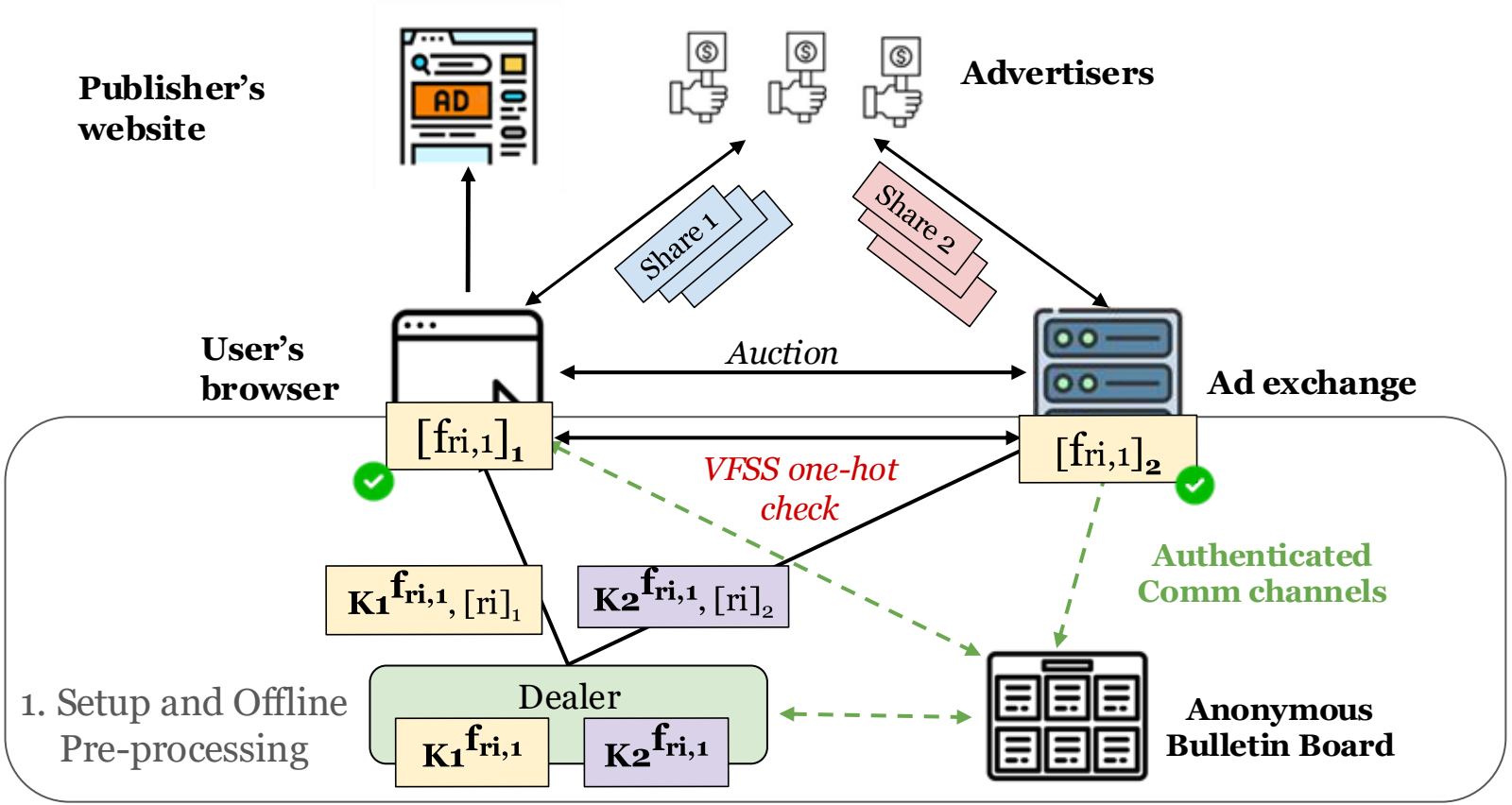
Solution: Moving majority of the validation cost offline

Verifiable Function Secret Sharing (VFSS) *(Boyle et al.)*

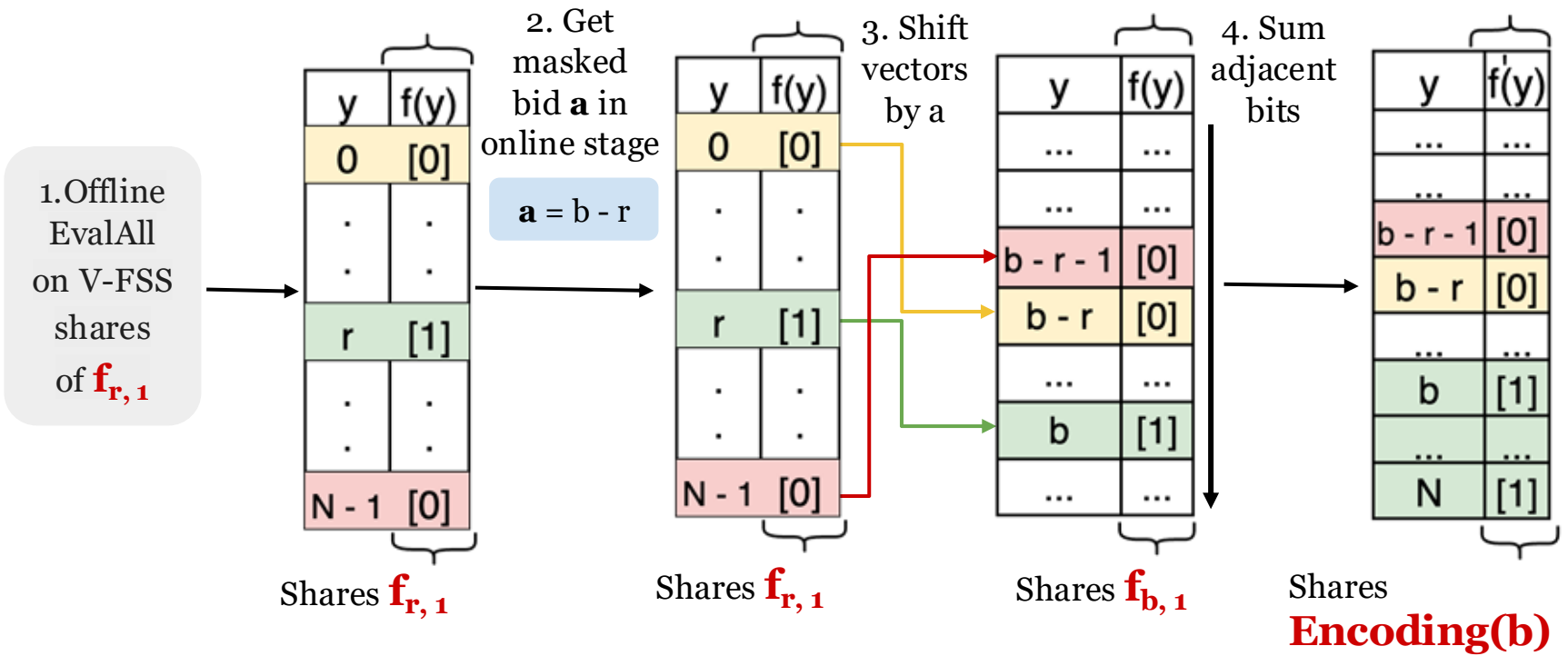
Bidders send raw bid shares instead of encoded bids and we convert it to reverse unary encoding shares

- Function Secret Sharing allows efficiently secret sharing a function (think of it like a lookup table) across parties, valid structure by construction
- Validation & verifiable shifting/lookup is optimal for one-hot functions
- Parties setup random function shares offline, shift and convert one-hot shares to a reverse-unary bids online

Offline Workflow for setting up validated FSS

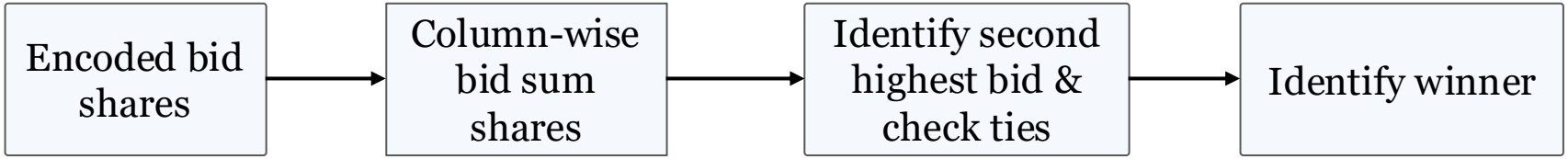


Converting FSS shares to encoding



Checkout the paper for more details!

Leveraging FSS to optimize further



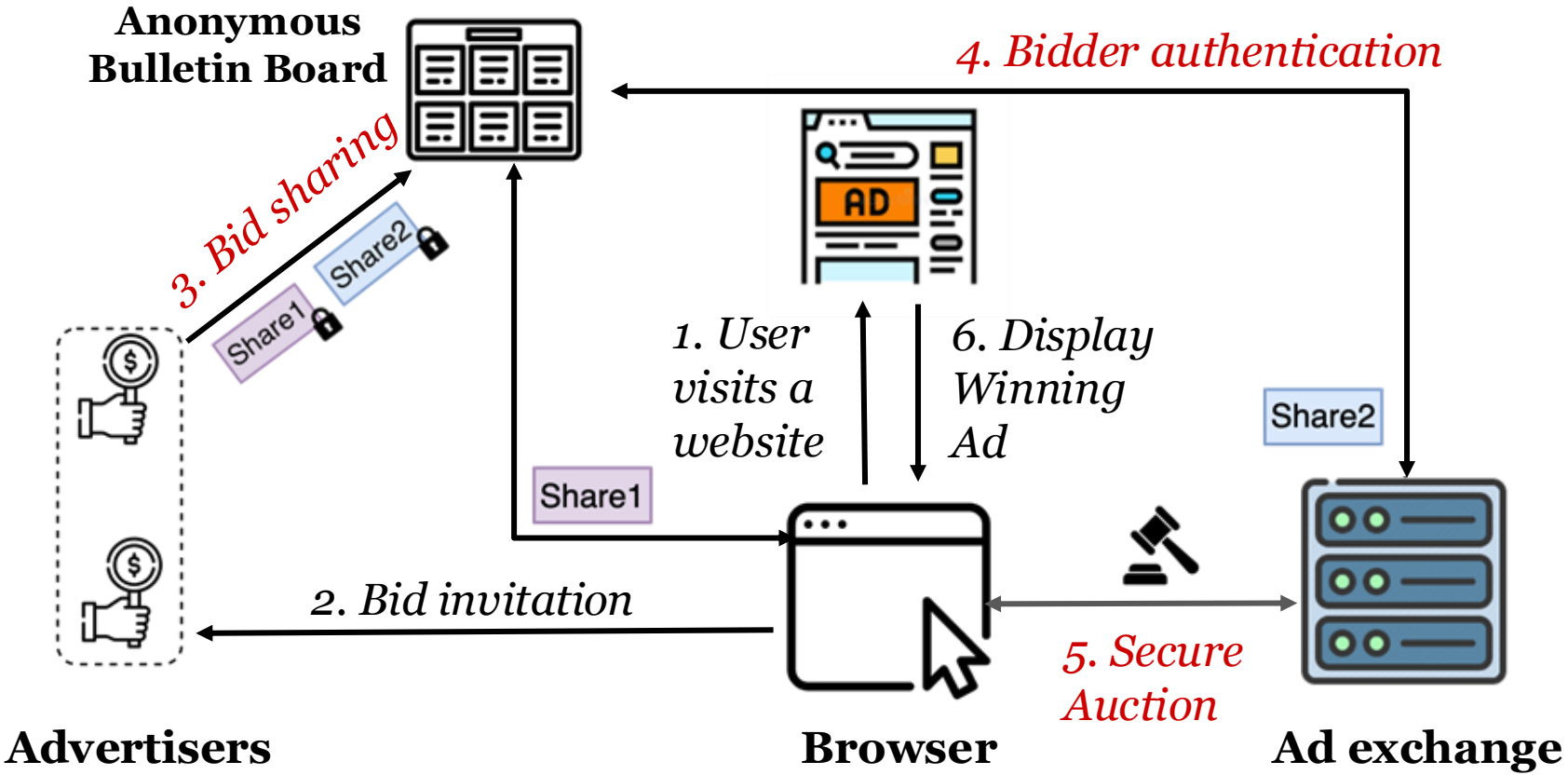
Naive search (for smallest index) does sequential, individual comparisons for second-highest bid and then the index with zero, scaling with bidders and domain

FSS-based evals and shifts help in replacing these individual comparisons with efficient lookups and help us achieve:

Constant round communication

independent of # bidders and bid domain size

OBSIDIAN Workflow



Evaluation: Baseline Comparison

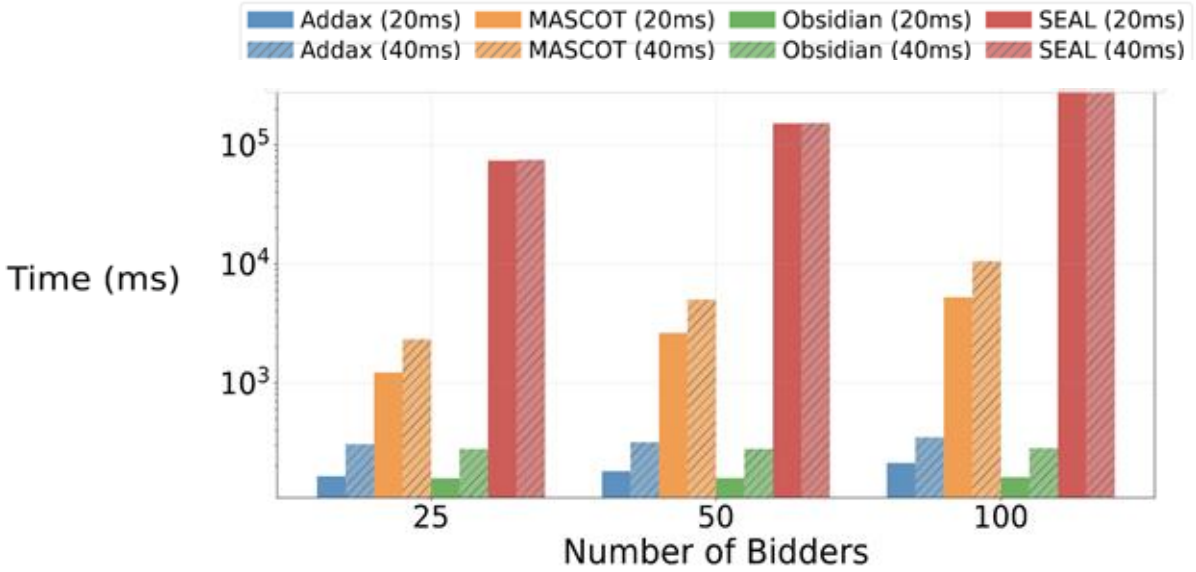
On a WAN network (with diff RTTs), compare auction computation over 2 Intel Xeon server, for 3 baselines enabling a secure computation:

1. **Addax** (leaks highest bid, weaker model) uses Prio-based encoding for finding Max efficiently, works well for first price

2. **MPC** framework - MPSPDZ, running an efficient sorting algorithm using MASCOT, typical for secure computation

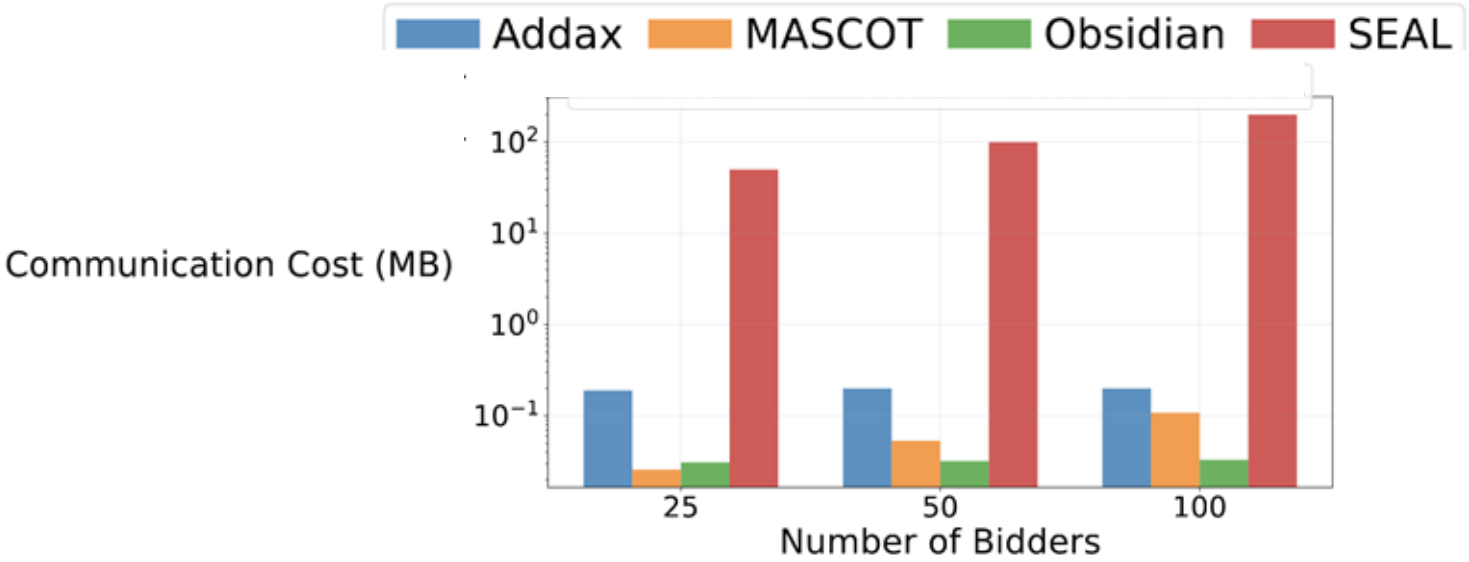
3. **Homomorphic** encryption– SEAL, running the topk maxID algorithm, computing on encrypted data, secure but slow

Comparison: Auction Latency



Takeaway: Constant communication rounds helps Obsidian outperforms all other baselines while scaling up on bidders and RTT

Comparison: Communication Cost



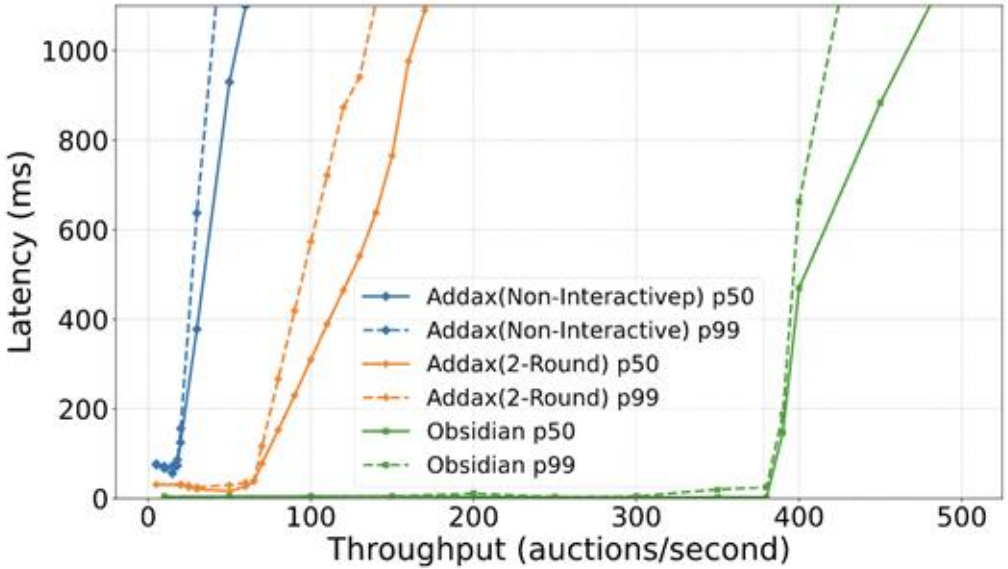
Takeaway: Even for the high number of bidders, cost is < 1MB

Communication cost is minimal given the modern fast internet links

Evaluation: Auction Throughput

Ad-exchanges handle **many auctions concurrently**

We stress test with varying auction requests/sec and measure end-to-end tail latencies



Takeaway: Obsidian out performs Addax by 4x - 10x, amenable to practical real-time auctions

Conclusion

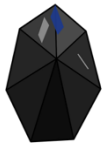
→ A **secure Vickrey auction system** for the present-day web

→ Novel use of **encoded bids + VFSS** → constant-round, malicious-security

→ **15.4% faster** than the closest baseline; **< 1 MB** communication cost

✓ **Bid confidentiality**

✓ **Low latency**



✓ **Bidder anonymity**

✓ **Compatibility**

GitHub Link

Thank you!

Questions?