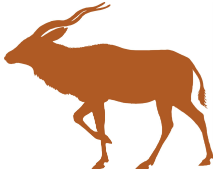
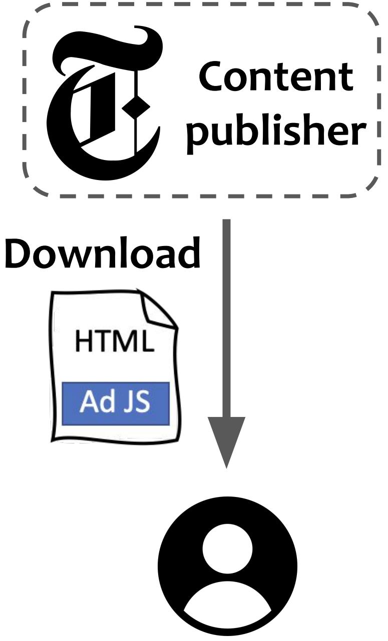


Addax: A fast, private, and accountable ad exchange infrastructure

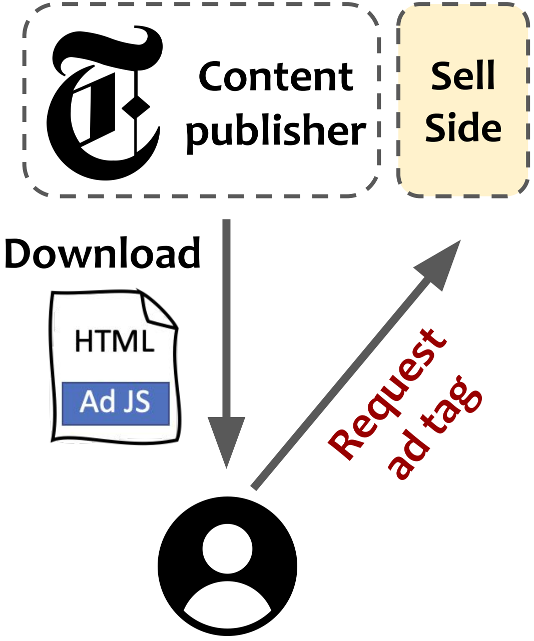
Ke Zhong[1], Yiping Ma[1], Yifeng Mao[1], Sebastian Angel[1,2]

[1]University of Pennsylvania [2]Microsoft Research

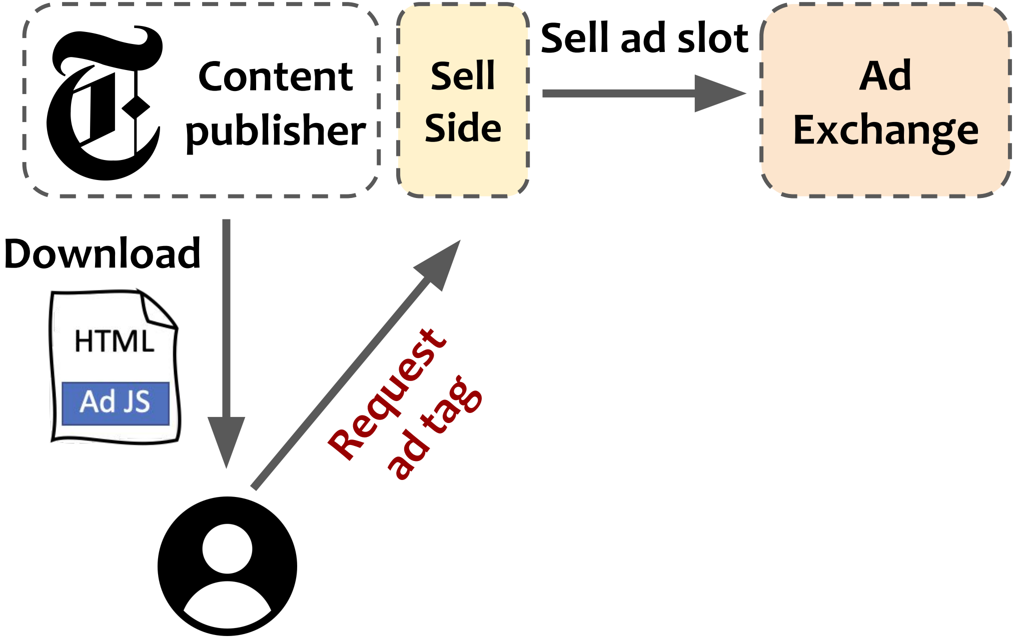
Current ads architecture



Current ads architecture



Current ads architecture



Content publisher

Sell Side

Sell ad slot

Ad Exchange

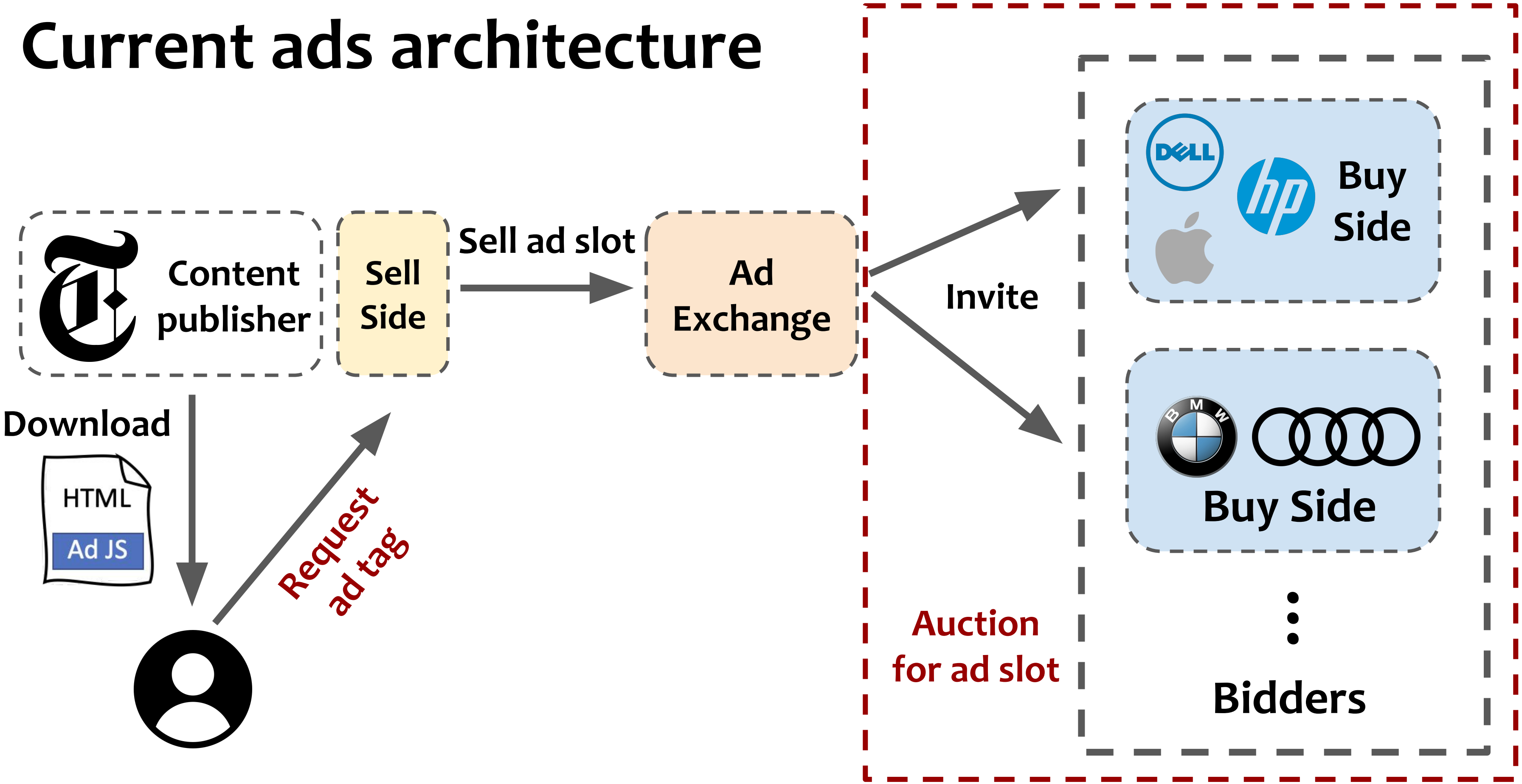
Download

HTML

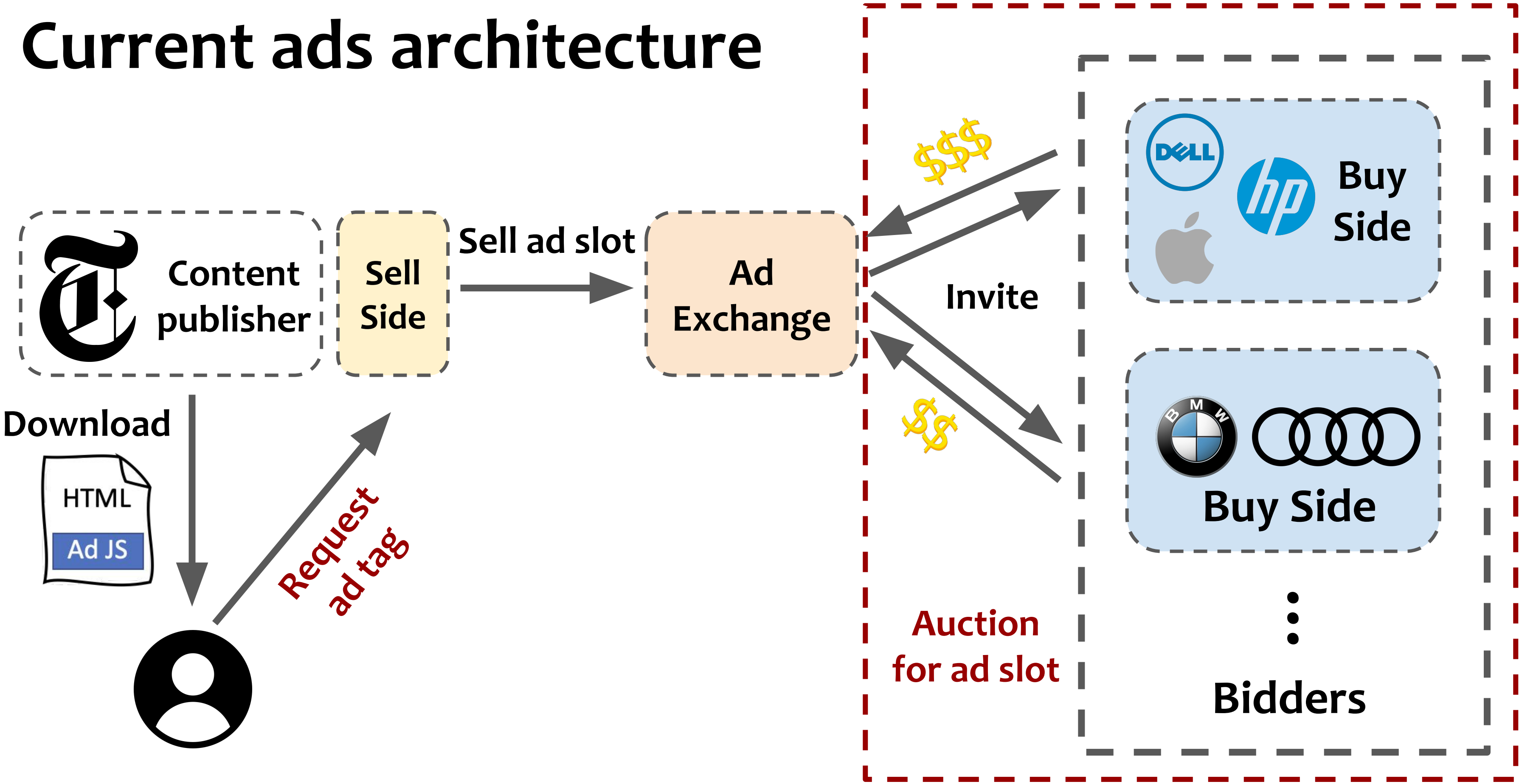
Ad JS

Request ad tag

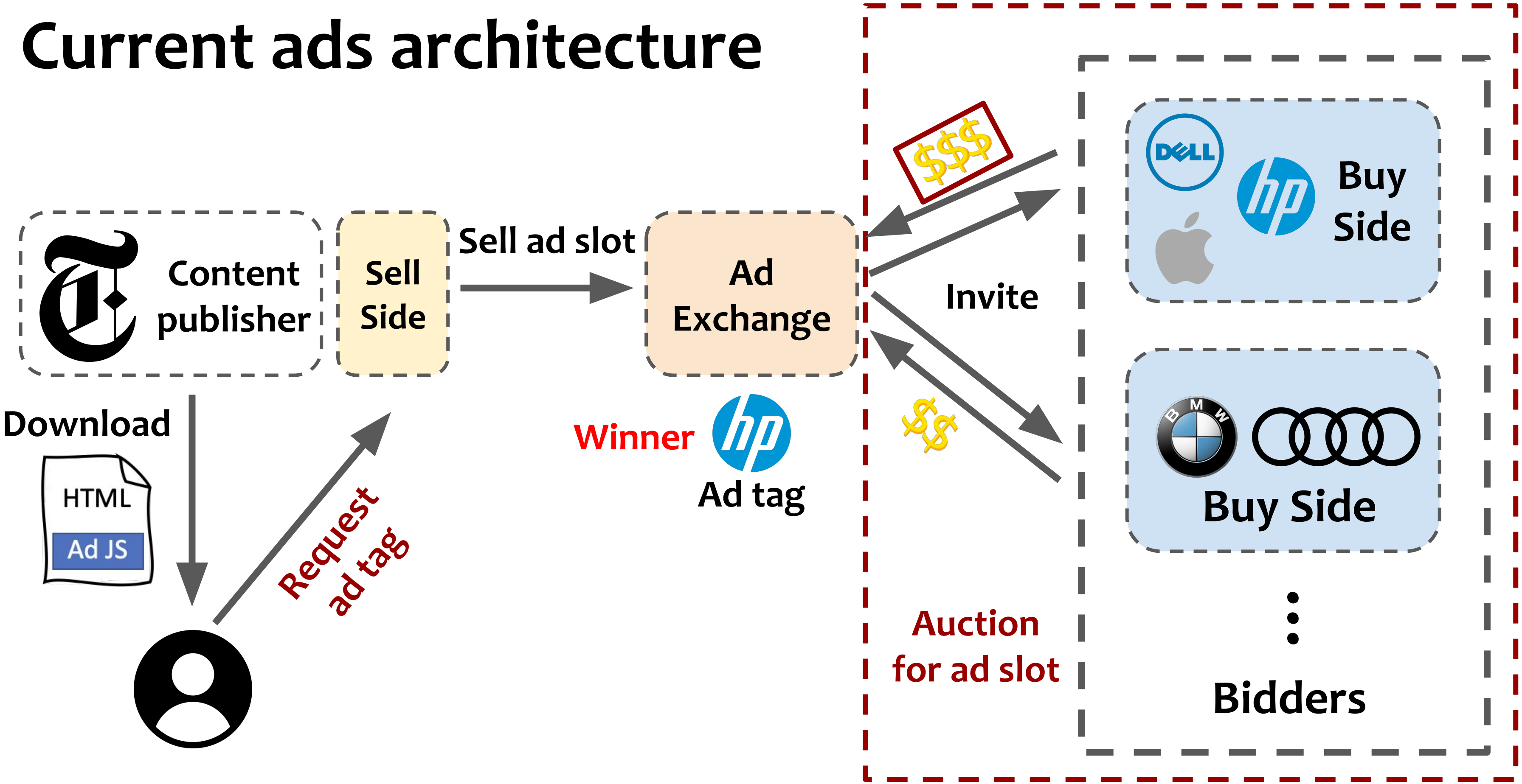
Current ads architecture



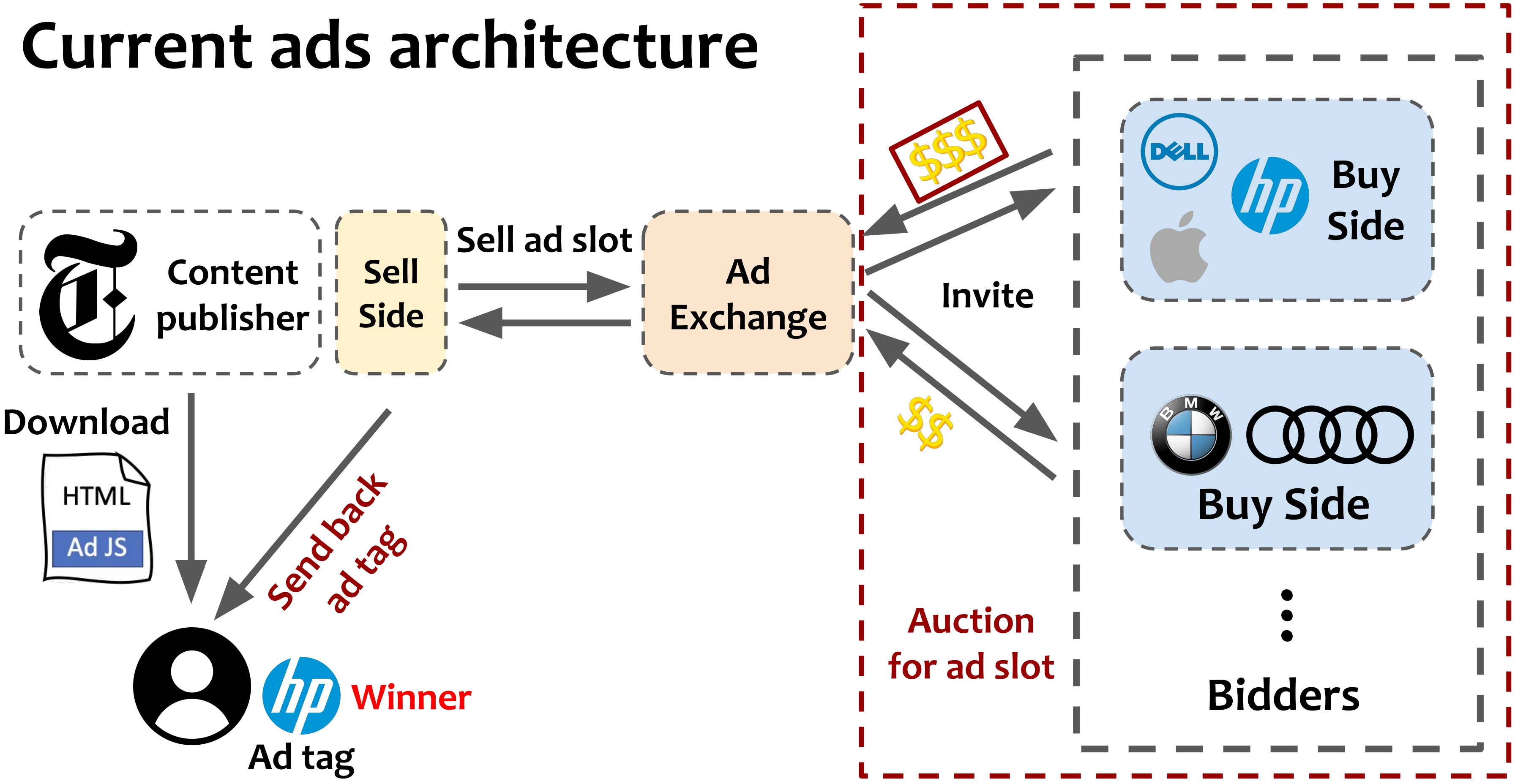
Current ads architecture



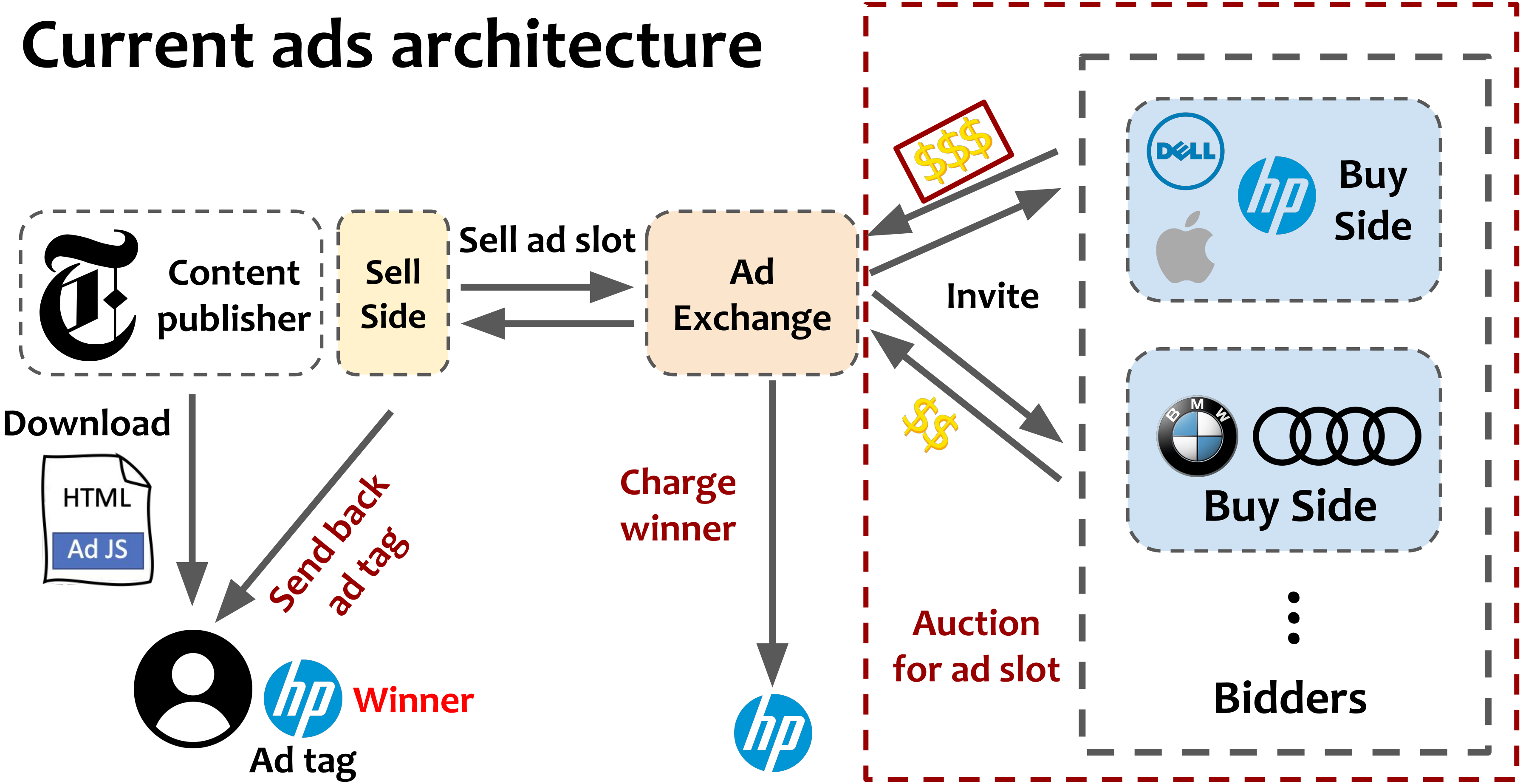
Current ads architecture



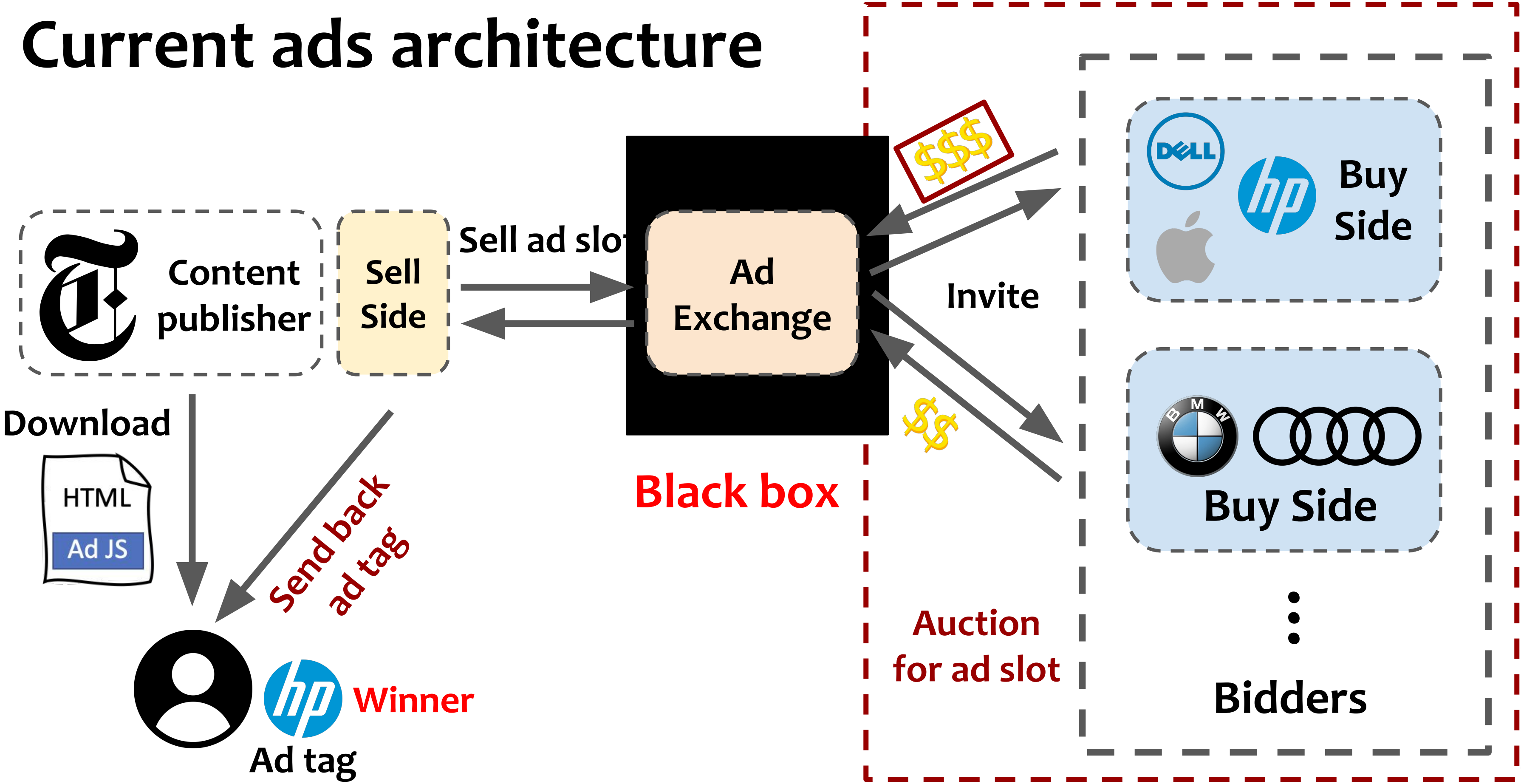
Current ads architecture



Current ads architecture



Current ads architecture



Justice Department Sues Google for Monopolizing Digital Advertising Technologies

“Manipulating auction mechanics across several of its products to insulate Google from competition, deprive rivals of scale, and halt the rise of rival technologies.”, 2023

Department of Justice

Justice Department Sues Google for Monopolizing Digital Advertising Technologies

"Manipulating auction mechanics across several of its products to insulate Google from competition, deprive rivals of scale, and halt the rise of rival technologies.", 2023

Department of Justice

"Google used insider knowledge of past bids submitted by advertisers to gain unfair advantages whenever its subsidiaries participated in auctions" 2021

THE WALL STREET JOURNAL.

Justice Department Sues Google for Monopol
Advertising Technologies
“Manipulating auction mechanics ac
products to insulate Google from co
of scale, and halt t

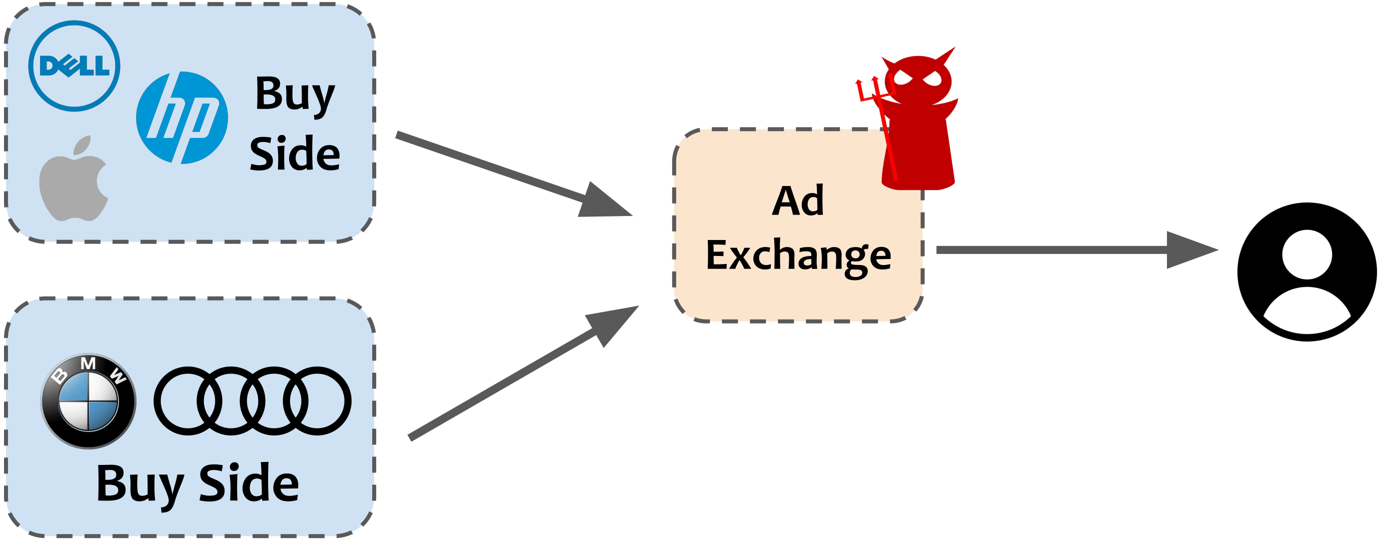
De

Not sure whether these claims are true or not, but they make the ad exchanges look untrustworthy.

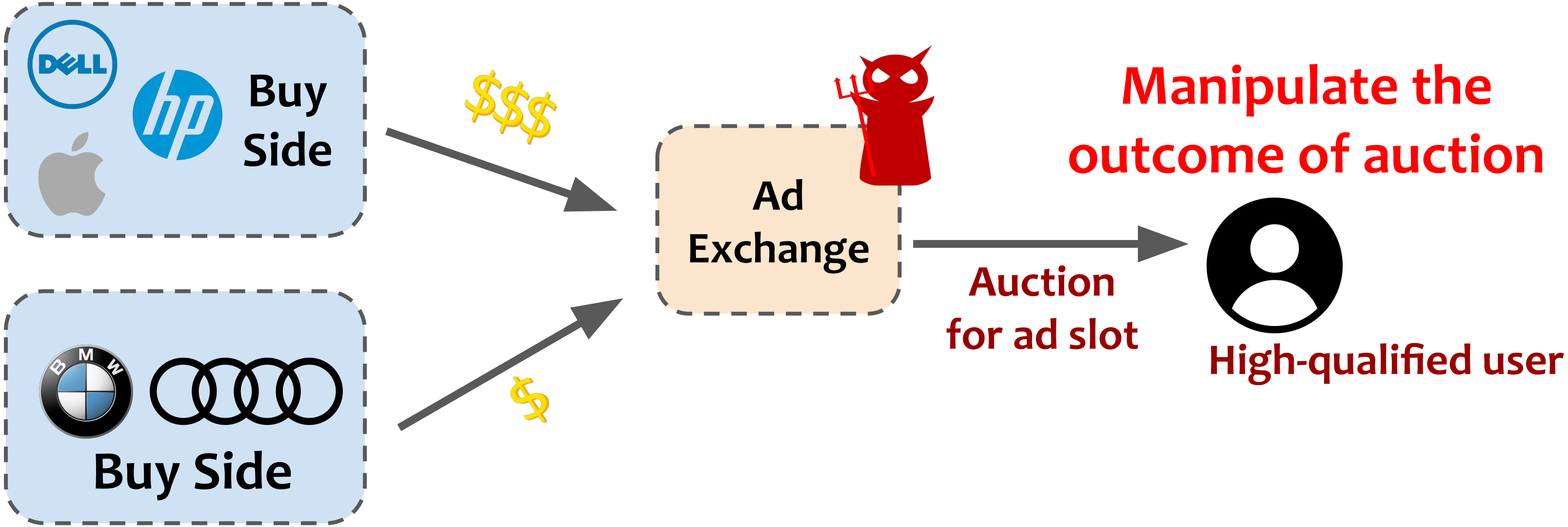
nsider knowledge of past bids submitted
to gain unfair advantages whenever its
rticipated in auctions” 2021

THE WALL STREET JOURNAL.

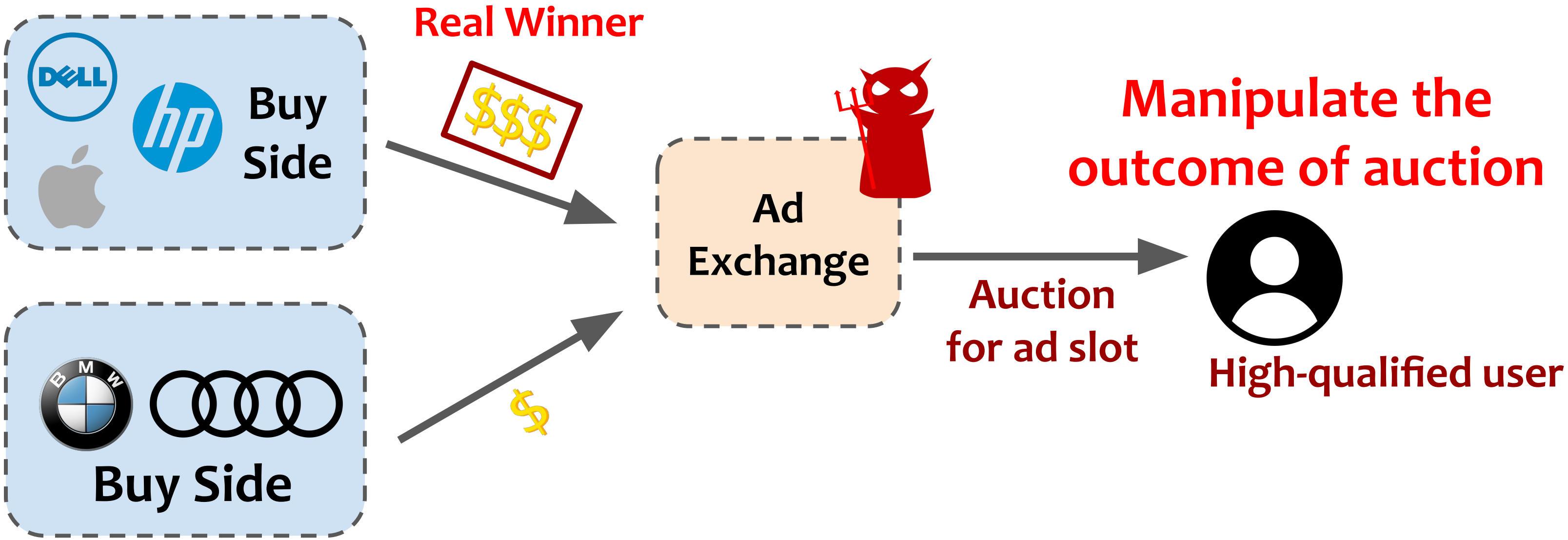
Distrust of current ad exchanges



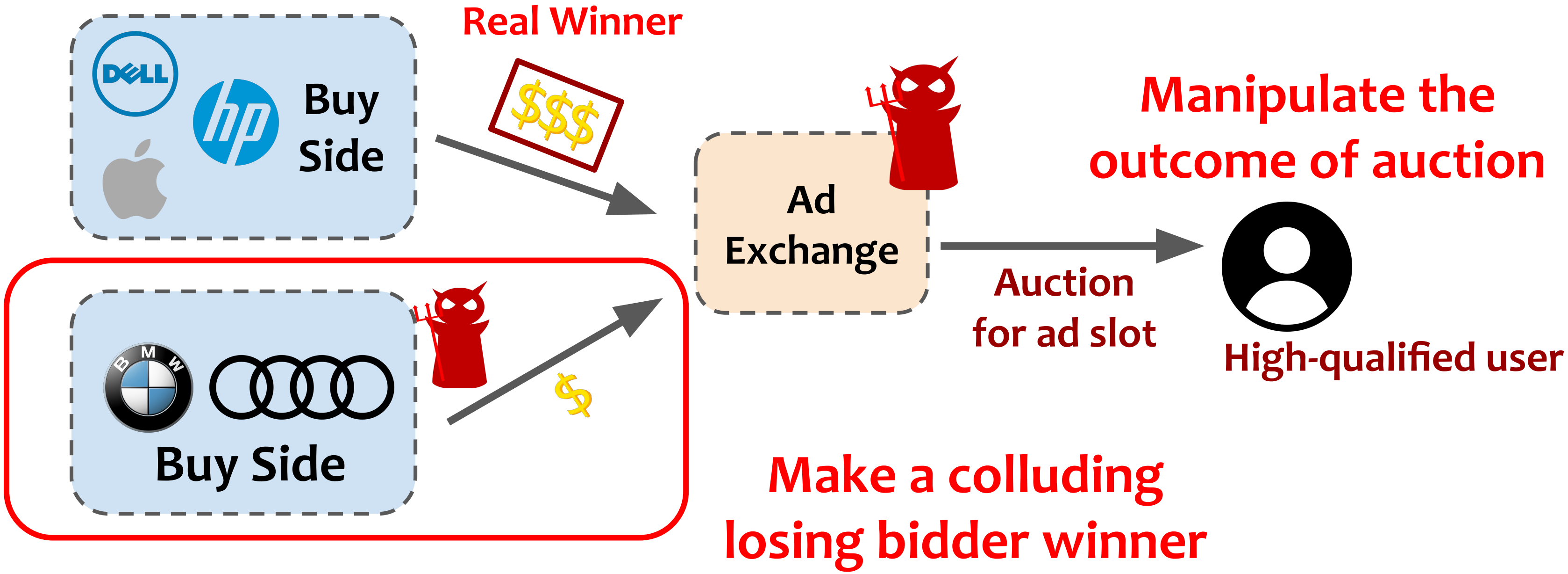
Distrust of current ad exchanges



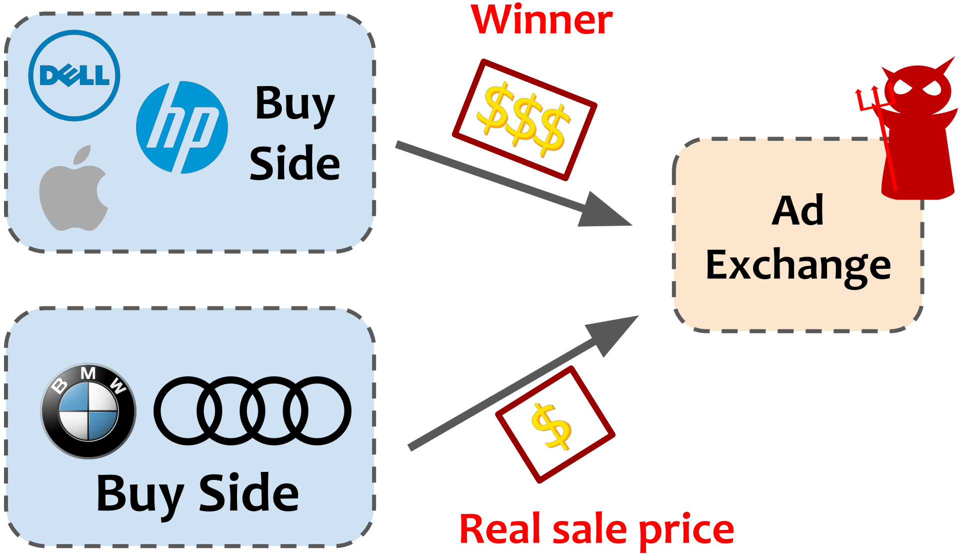
Distrust of current ad exchanges



Distrust of current ad exchanges



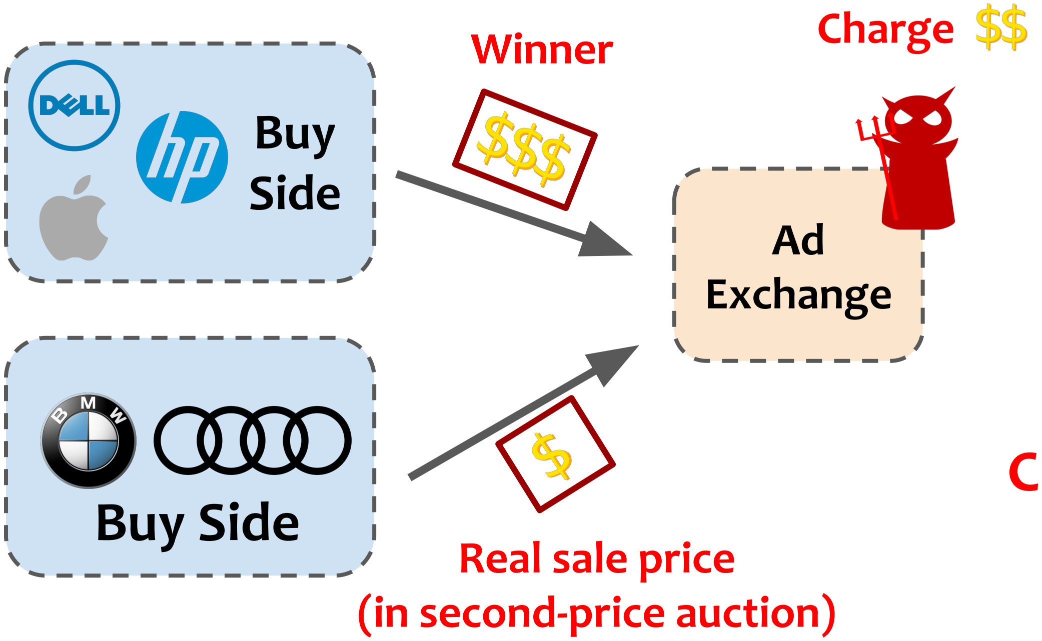
Distrust of current ad exchanges



Real sale price
(in second-price auction)

Manipulate the outcome of auction

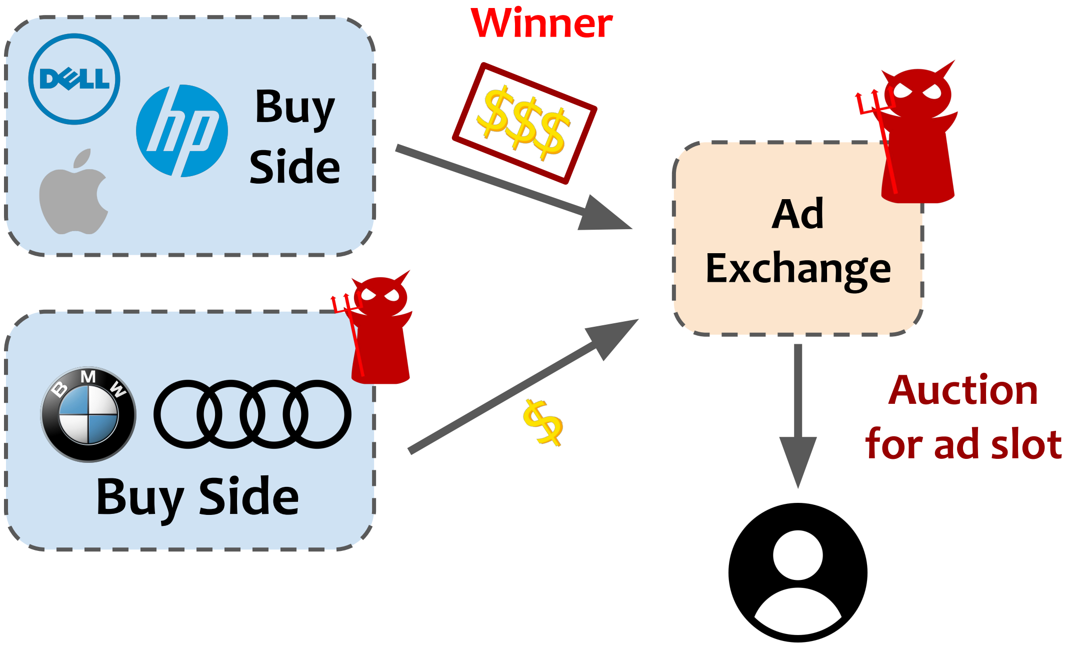
Distrust of current ad exchanges



Manipulate the outcome of auction

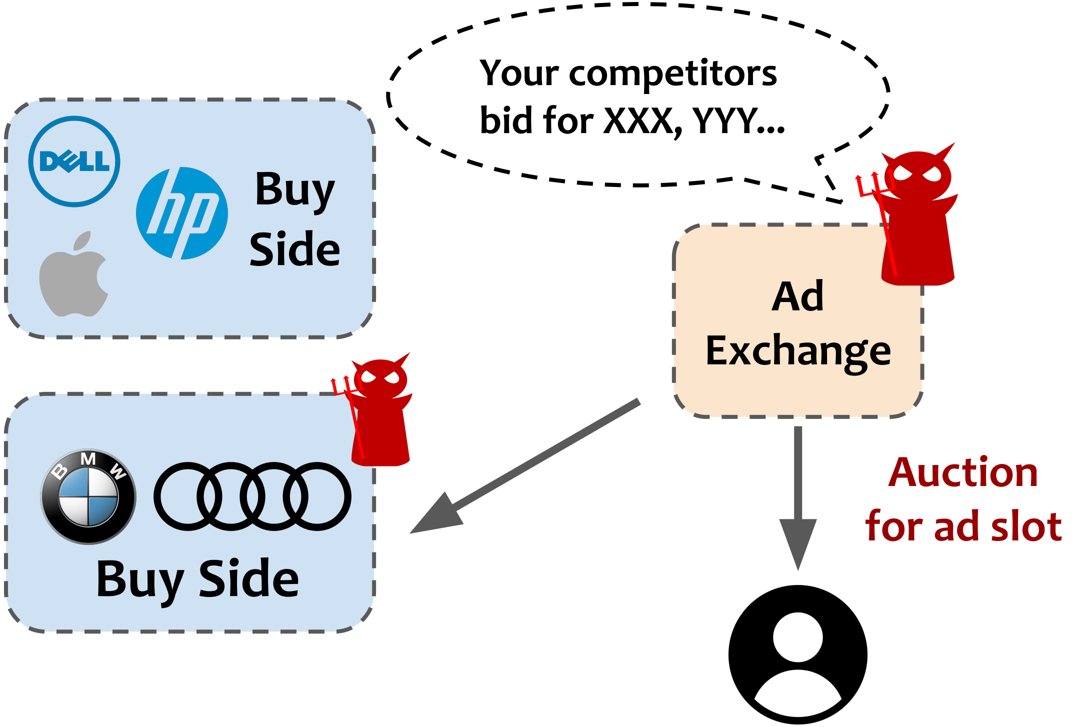
Charge the winner more

Distrust of current ad exchanges



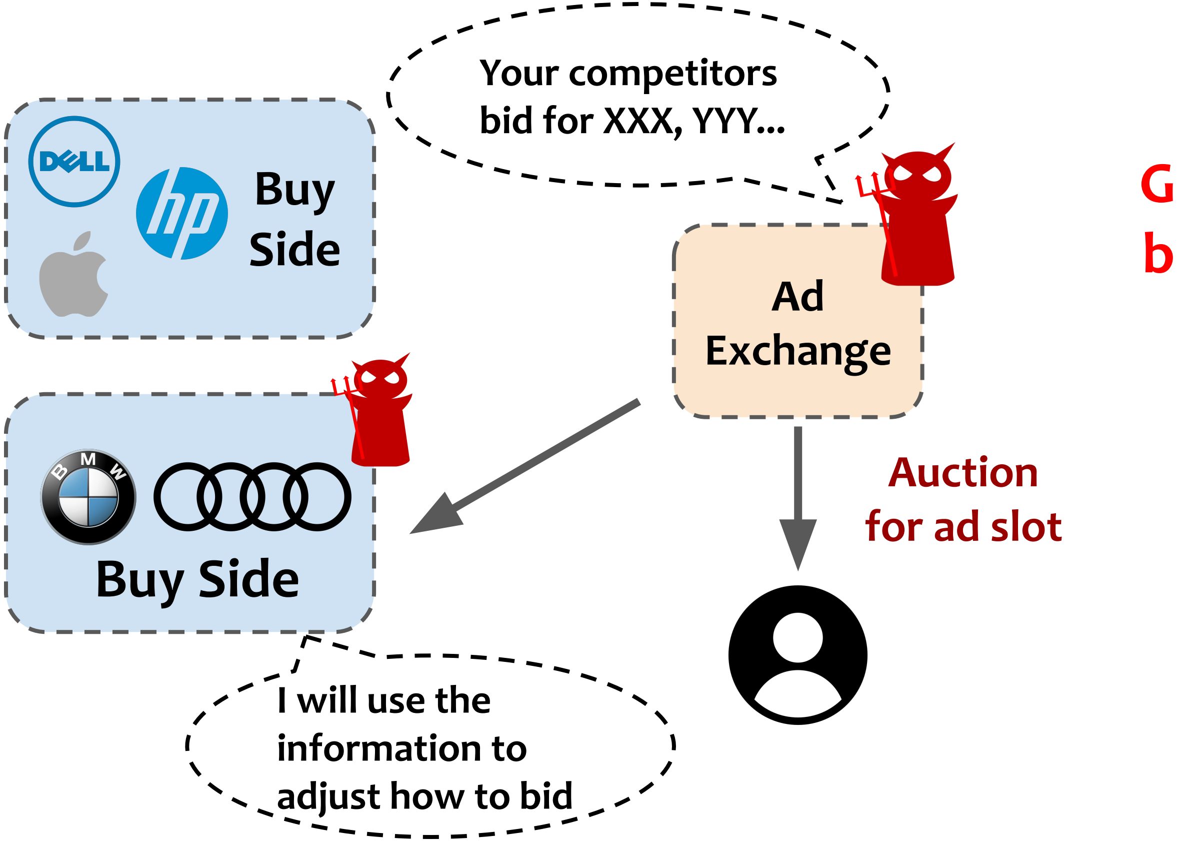
Gain all bidders' bid information

Distrust of current ad exchanges



Gain all bidders' bid information

Distrust of current ad exchanges


Your competitors bid for XXX, YYY...
Buy Side
Ad Exchange
Buy Side
Auction for ad slot
I will use the information to adjust how to bid

Gain all bidders' bid information

Crux of the distrust

- **Lack ways to prove that ad exchanges conduct auctions correctly.**

- **Lack ways to prove that ad exchanges are not misusing additional bid information.**

Crux of the distrust

- La ons
  co

- La ng
  ad

We propose Addax to provide mechanisms to help ad exchange companies to *build up trust* again!

Goals

- **Public verifiability for auction**
    - **Ad exchanges can prove that they conduct auctions correctly.**

Goals

- **Public verifiability for auction**
  - **Ad exchanges can prove that they conduct auctions correctly.**
- **Bids privacy for losing bidders**
  - **Ad exchanges cannot learn values of losing bidders' bids.**

Goals

- **Public verifiability for auction**
  - **Ad exchanges can prove that they conduct auctions correctly.**
- **Bids privacy for losing bidders**
  - **Ad exchanges cannot learn values of losing bidders' bids.**
- **Practicability for real-time bidding**
  - **Low latency (hundreds of ms) and high throughput.**

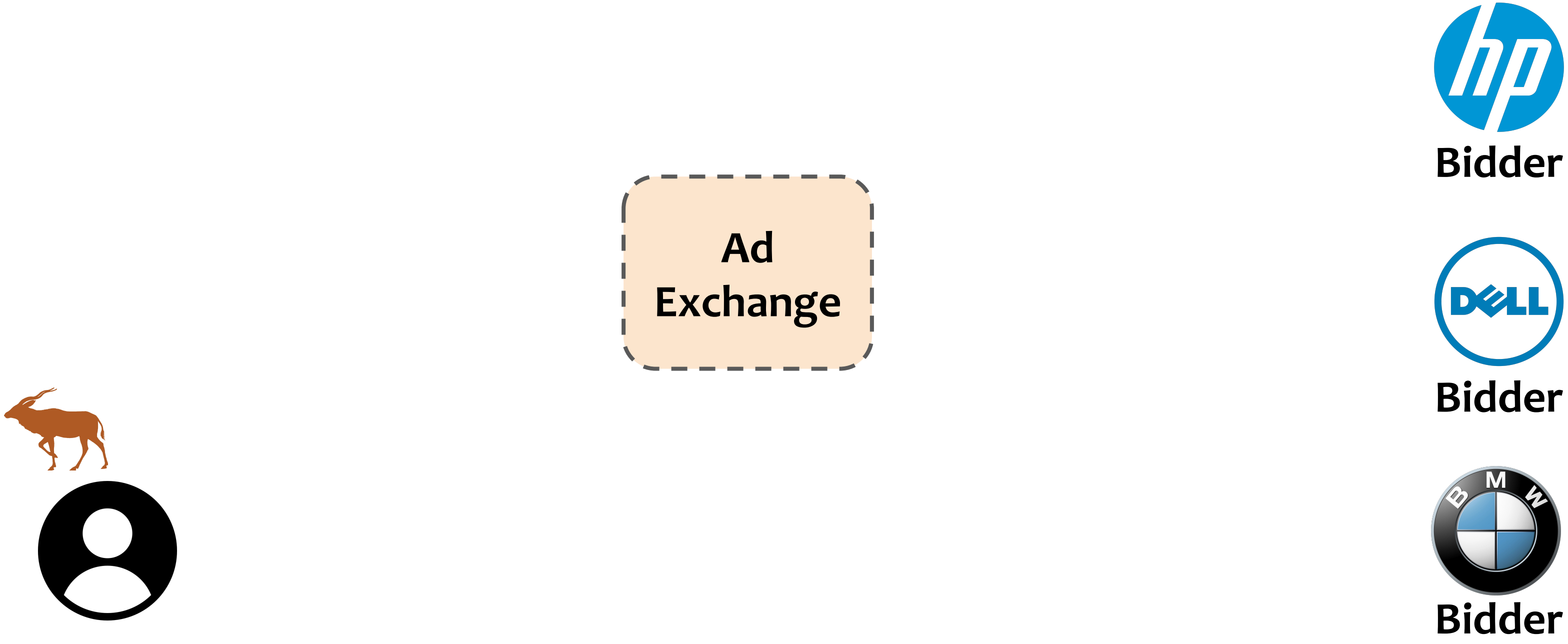
Rest of this talk

- Overview of Addax
- Private auction protocol
- Make auction verifiable
- Experimental evaluation

Rest of this talk

- **Overview of Addax**
- Private auction protocol
- Make auction verifiable
- Experimental evaluation

Architecture of Addax



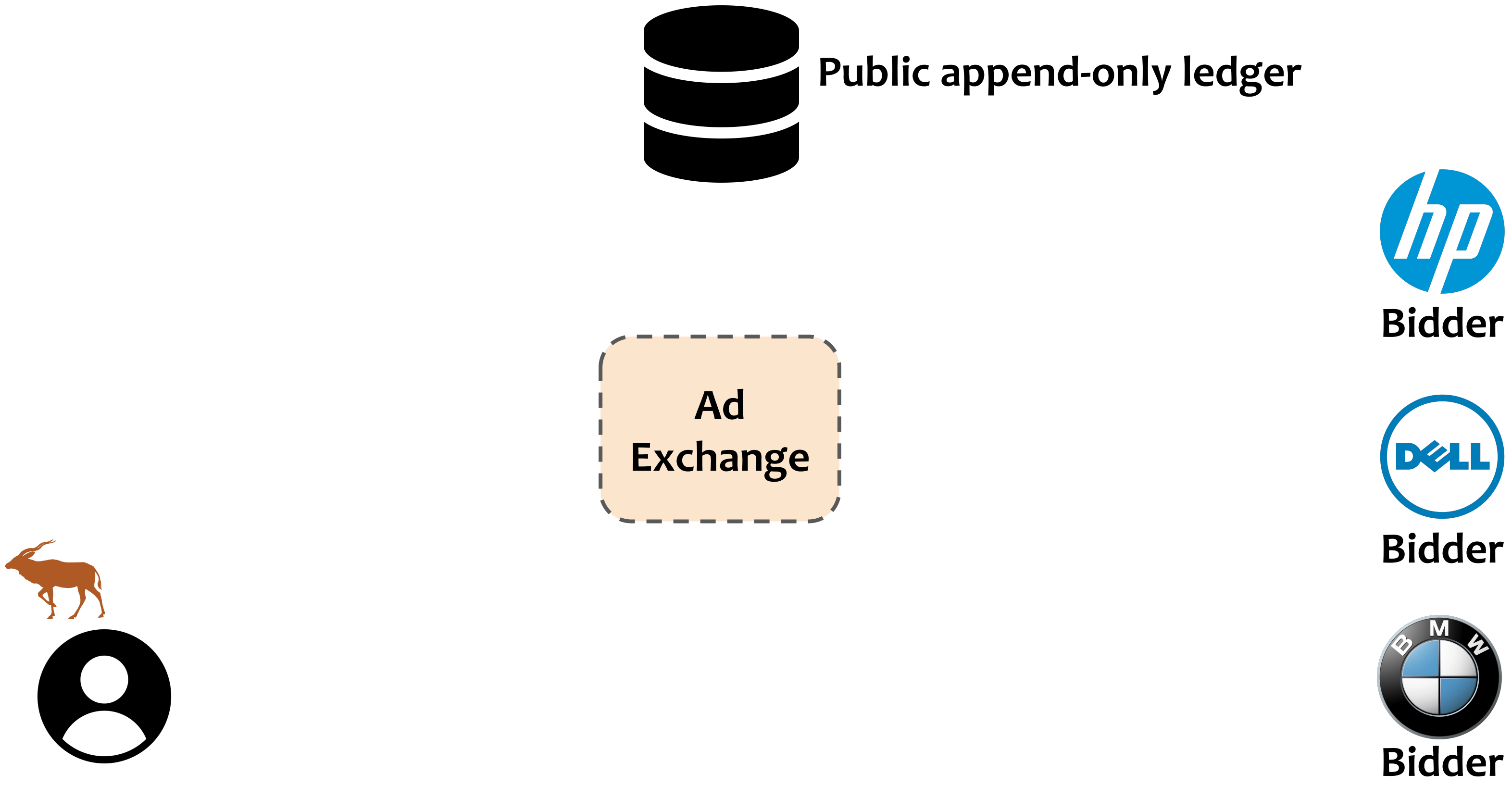
Ad Exchange

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

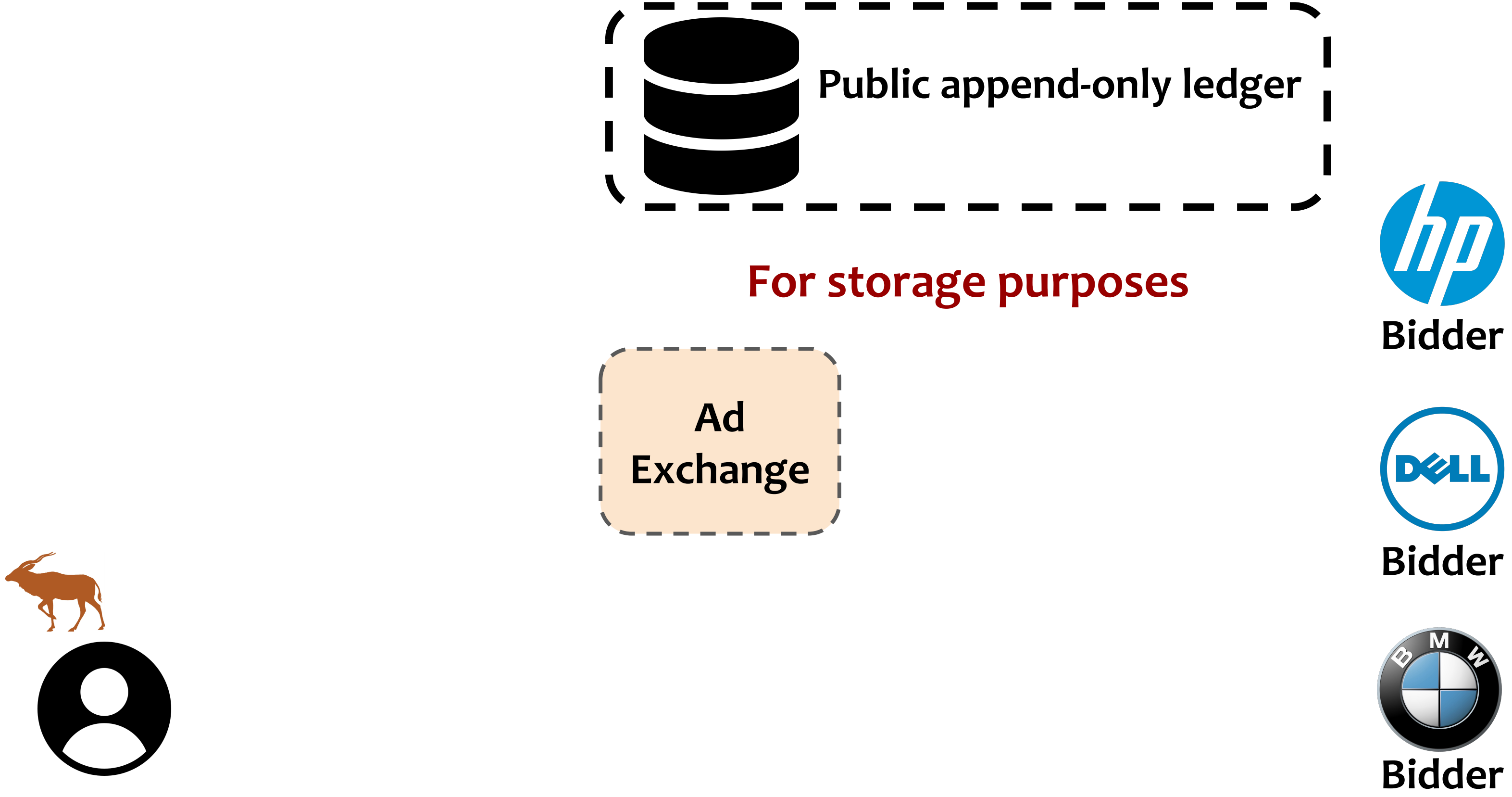
Ad Exchange

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

For storage purposes

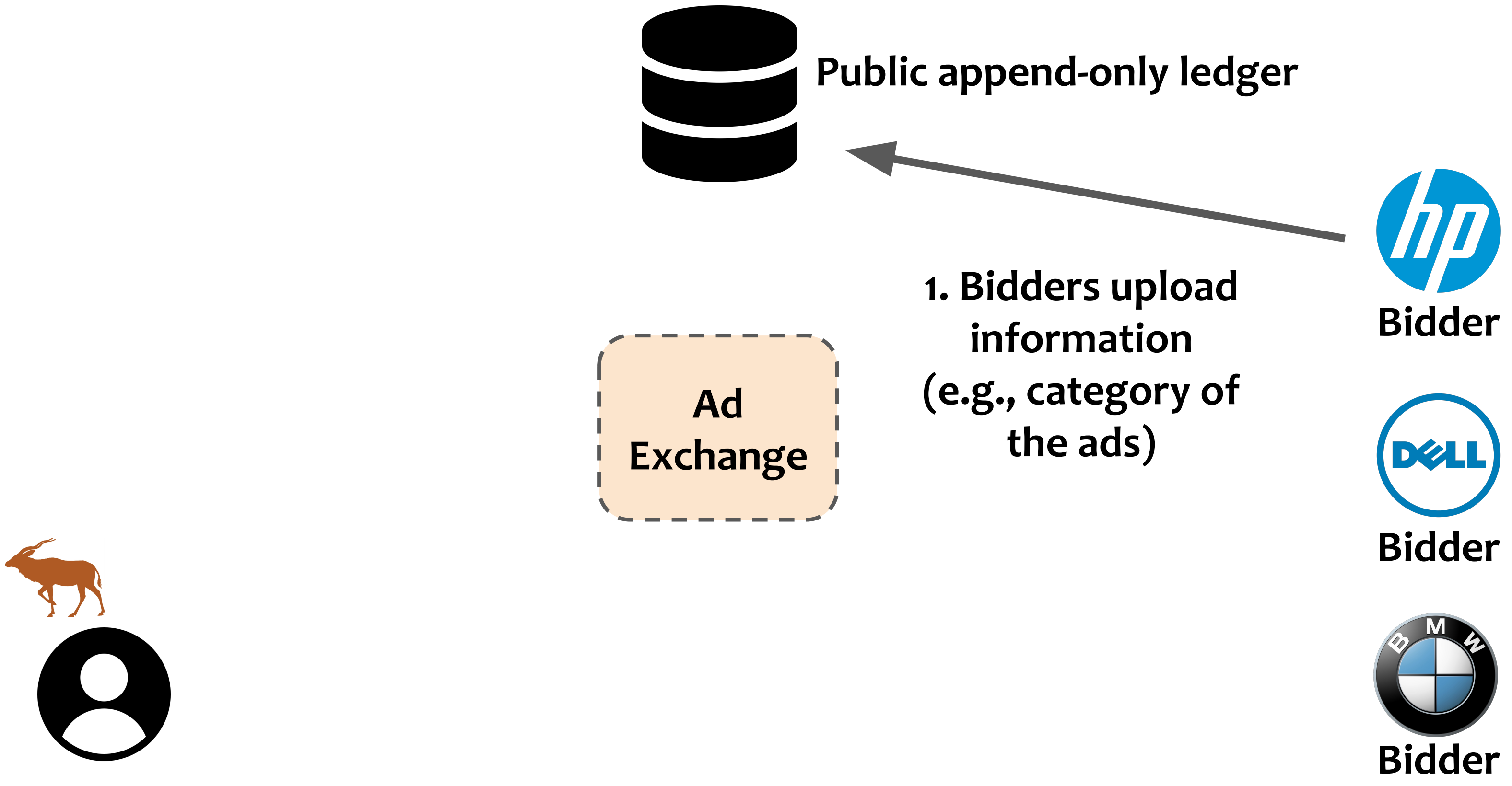
Ad Exchange

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

1. Bidders upload information (e.g., category of the ads)

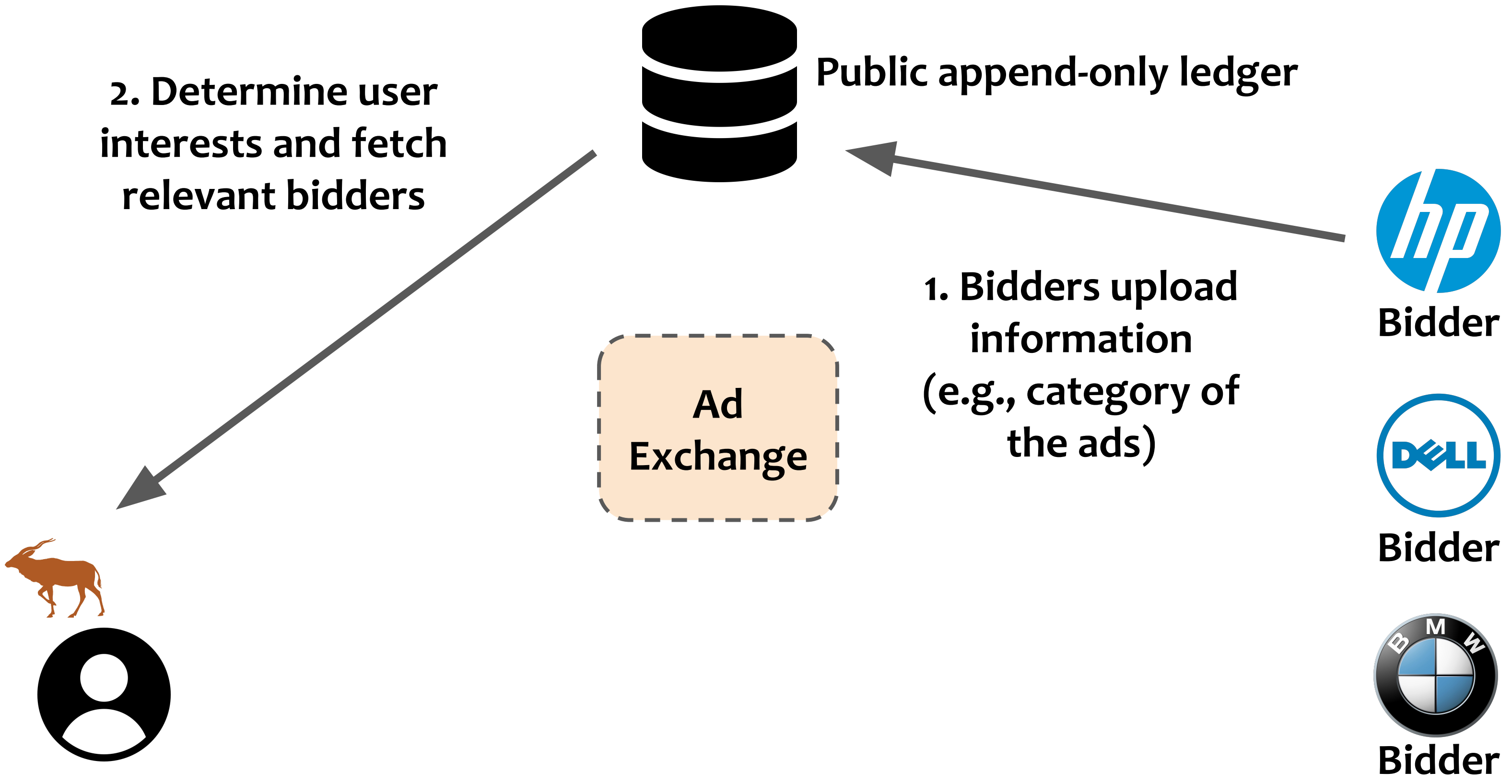
Ad Exchange

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

2. Determine user interests and fetch relevant bidders

1. Bidders upload information (e.g., category of the ads)

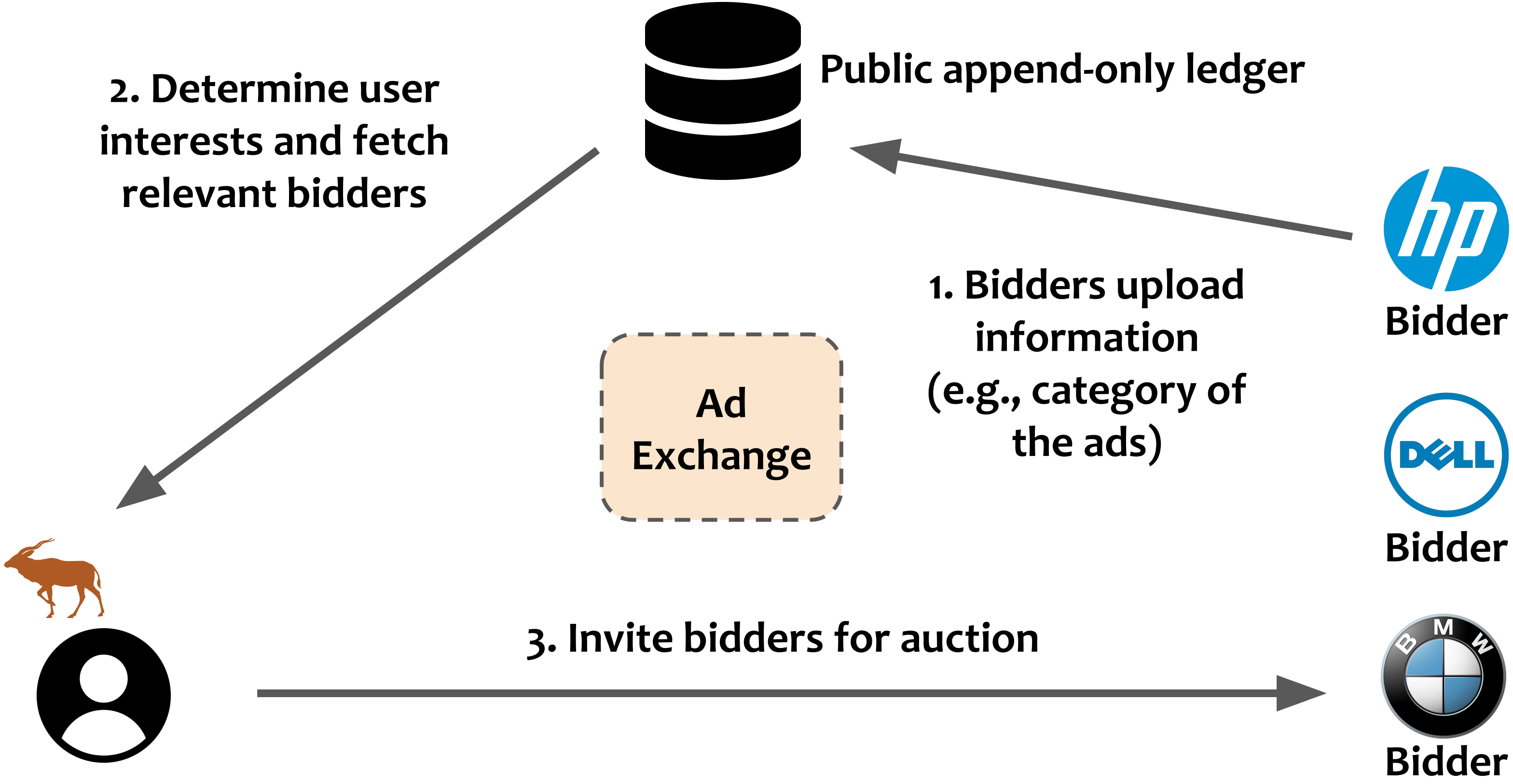
Ad Exchange

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

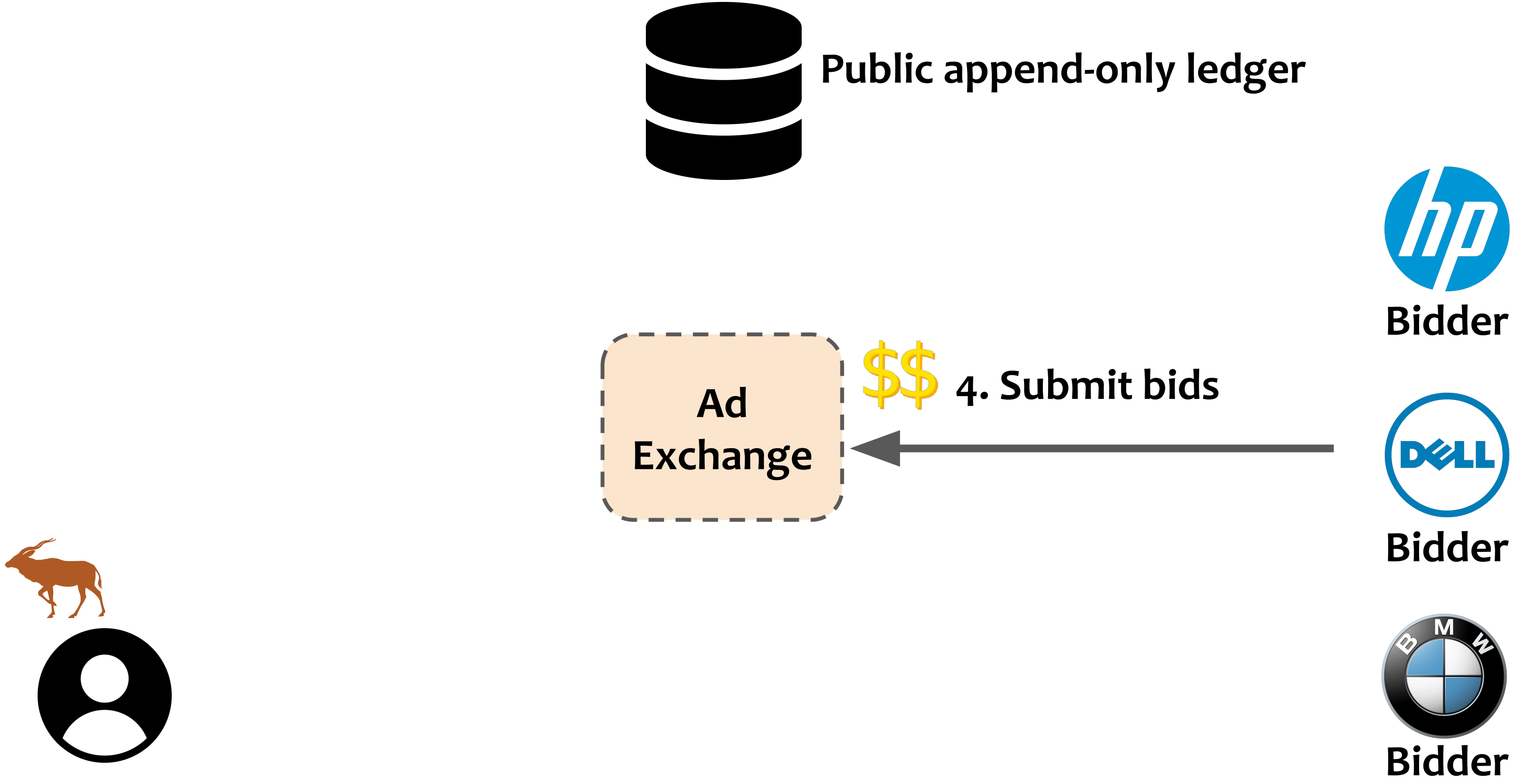
2. Determine user interests and fetch relevant bidders

1. Bidders upload information (e.g., category of the ads)

Ad Exchange

3. Invite bidders for auction

Architecture of Addax



Public append-only ledger

Ad Exchange

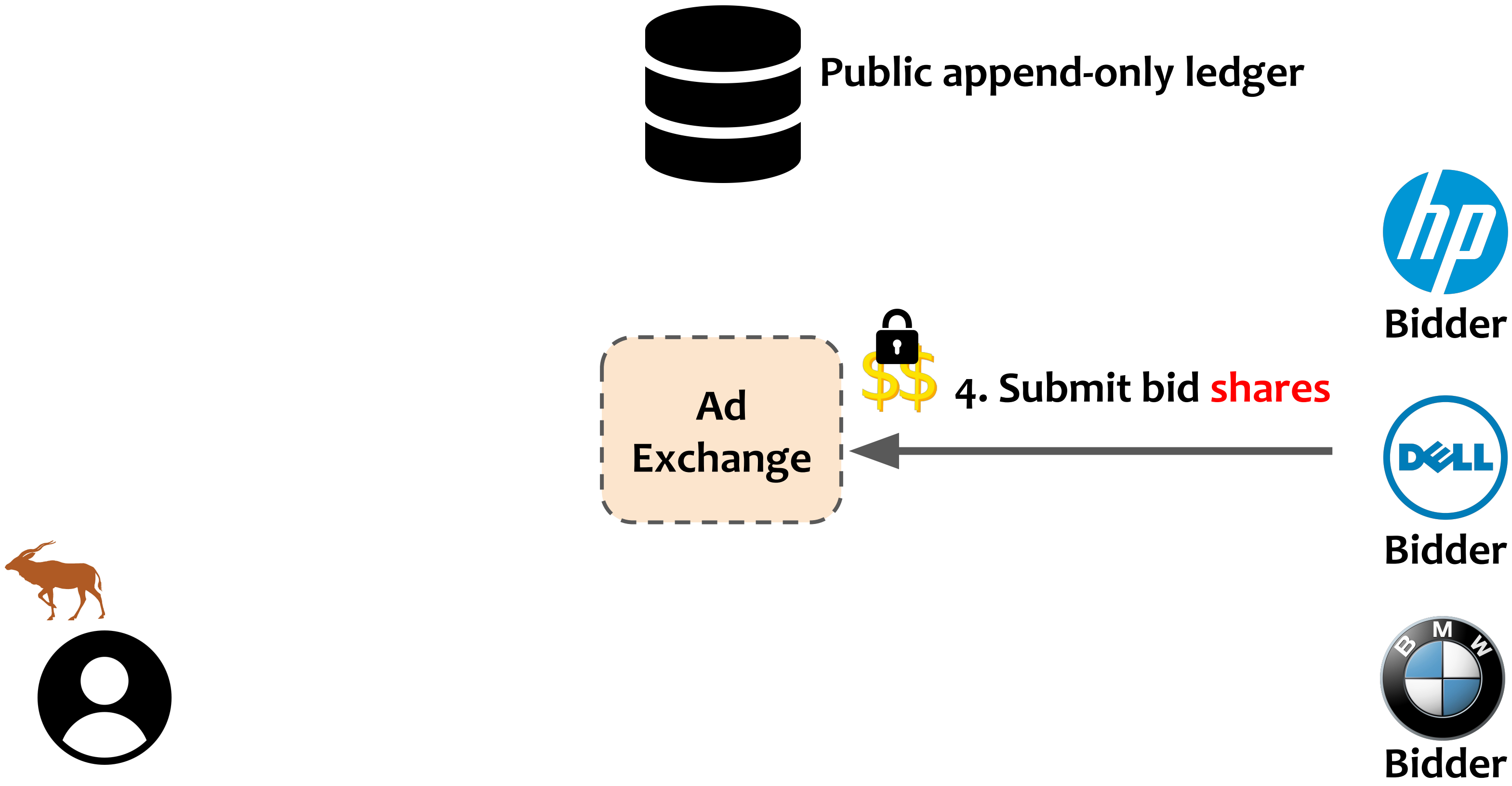
$$ 4. Submit bids

Bidder

Bidder

Bidder

Architecture of Addax



Public append-only ledger

Ad Exchange

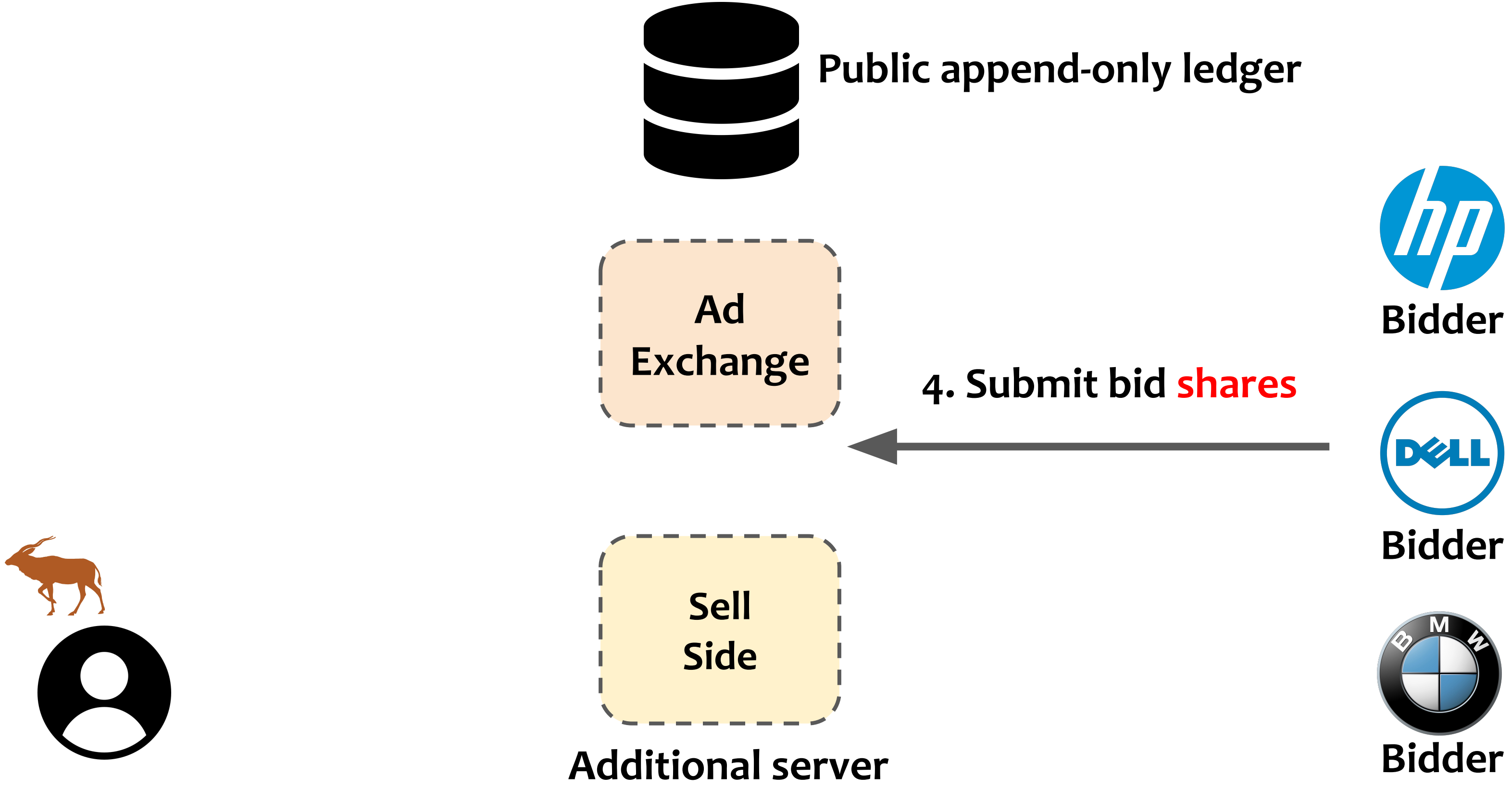
4. Submit bid shares

Bidder

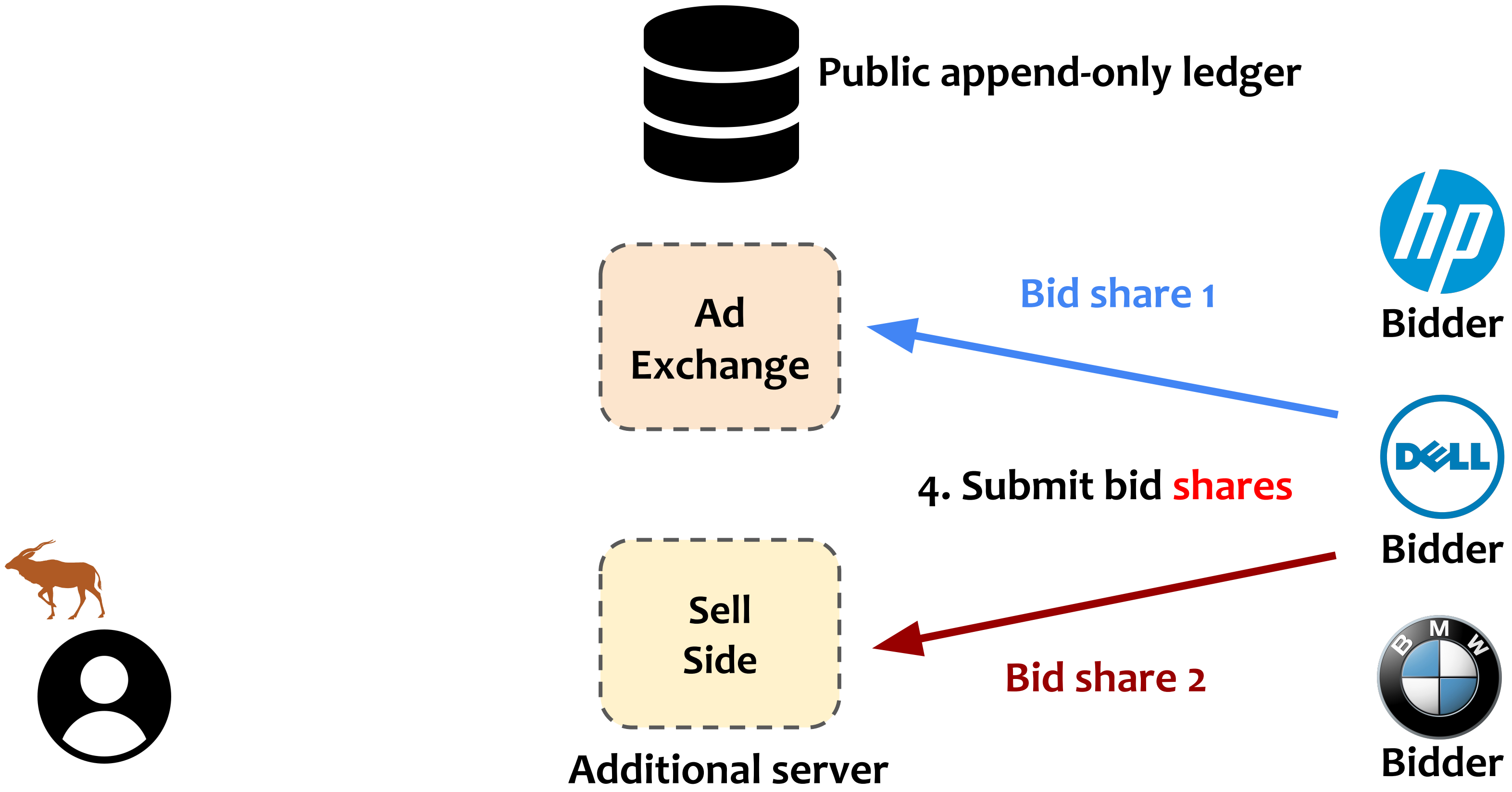
Bidder

Bidder

Architecture of Addax



Architecture of Addax



Public append-only ledger

Ad Exchange

Bid share 1

Bidder

4. Submit bid shares

Bidder

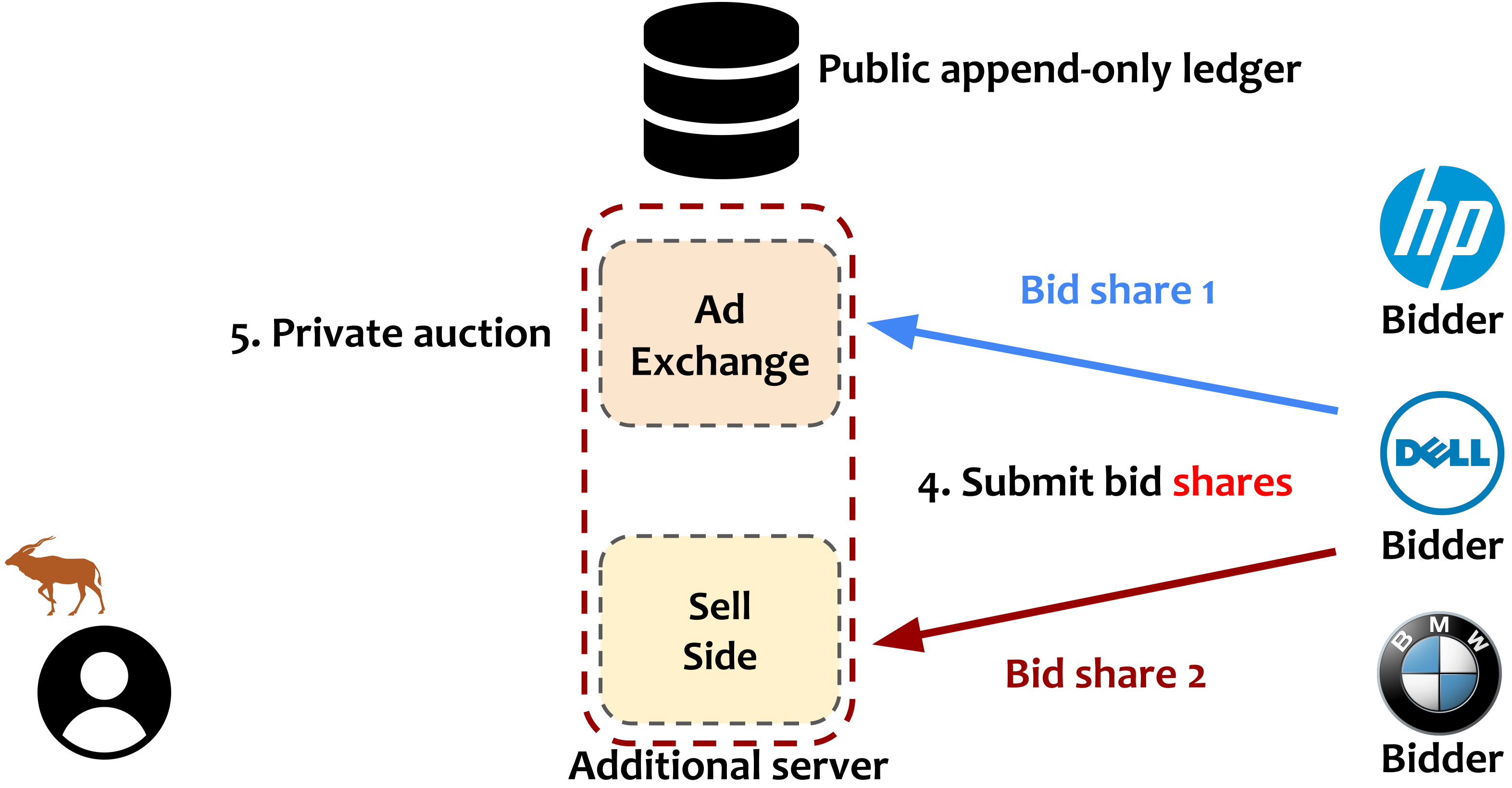
Sell Side

Bid share 2

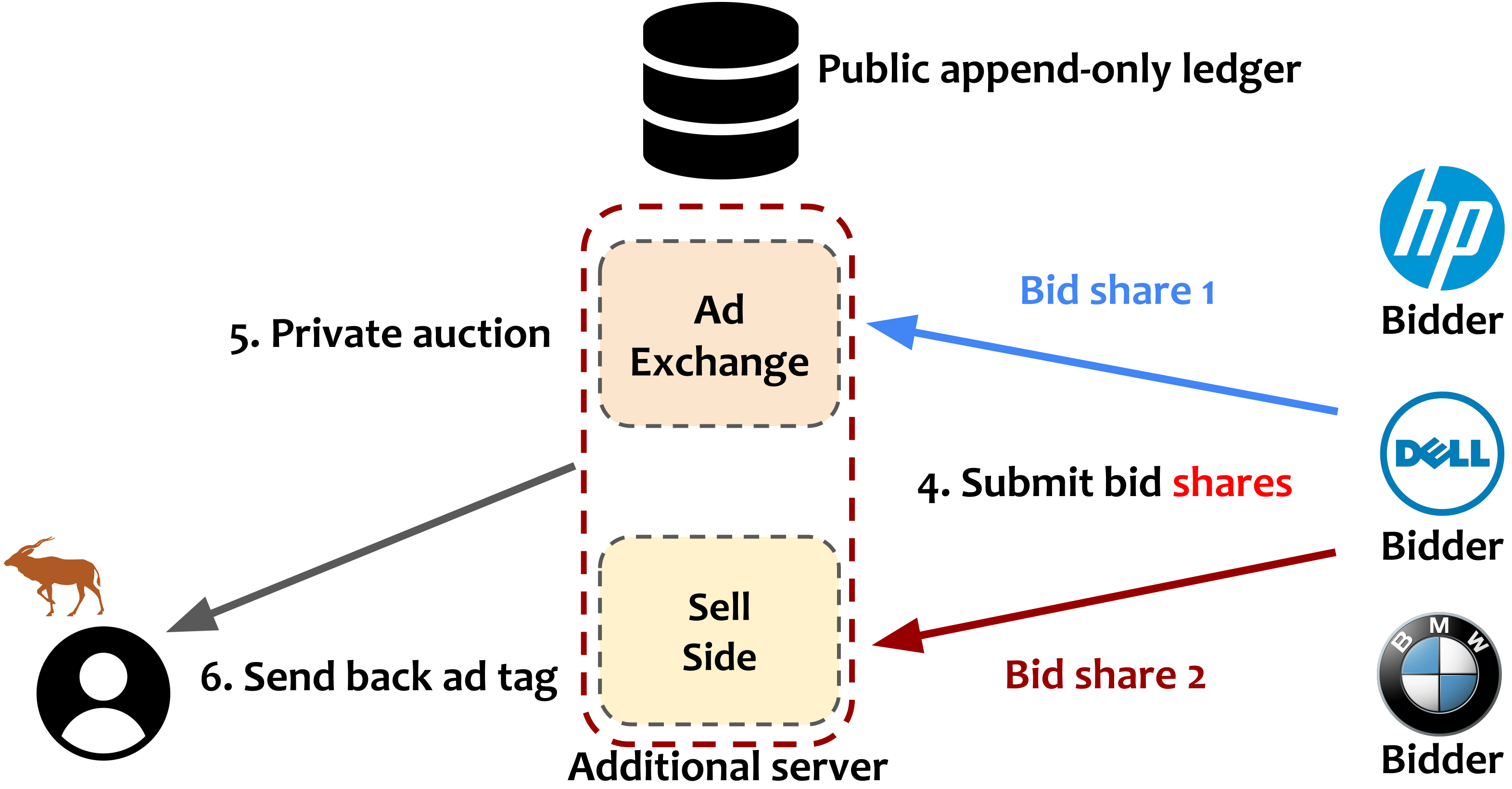
Bidder

Additional server

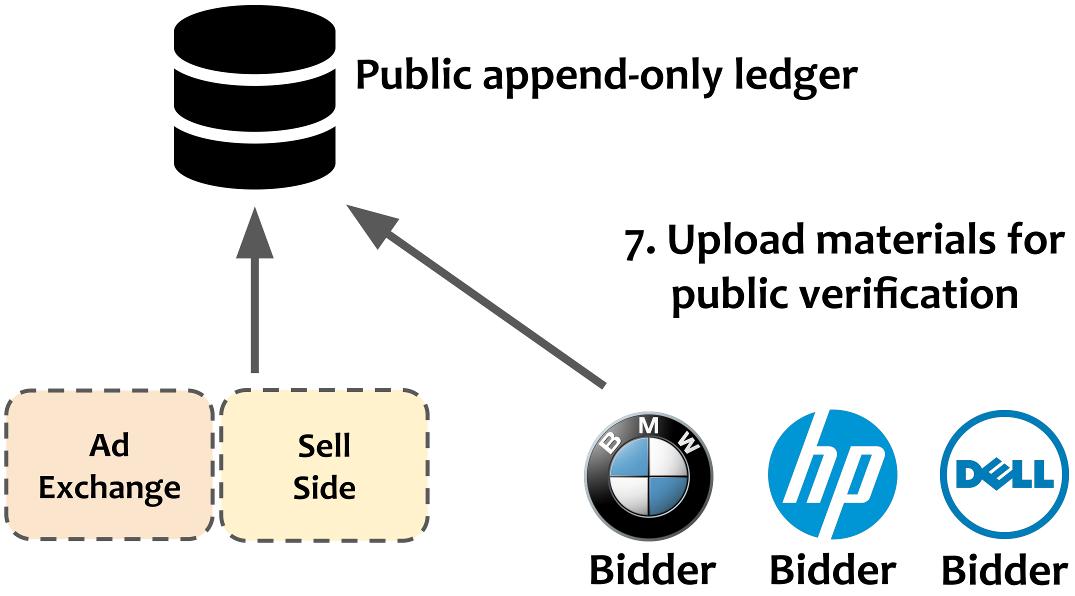
Architecture of Addax



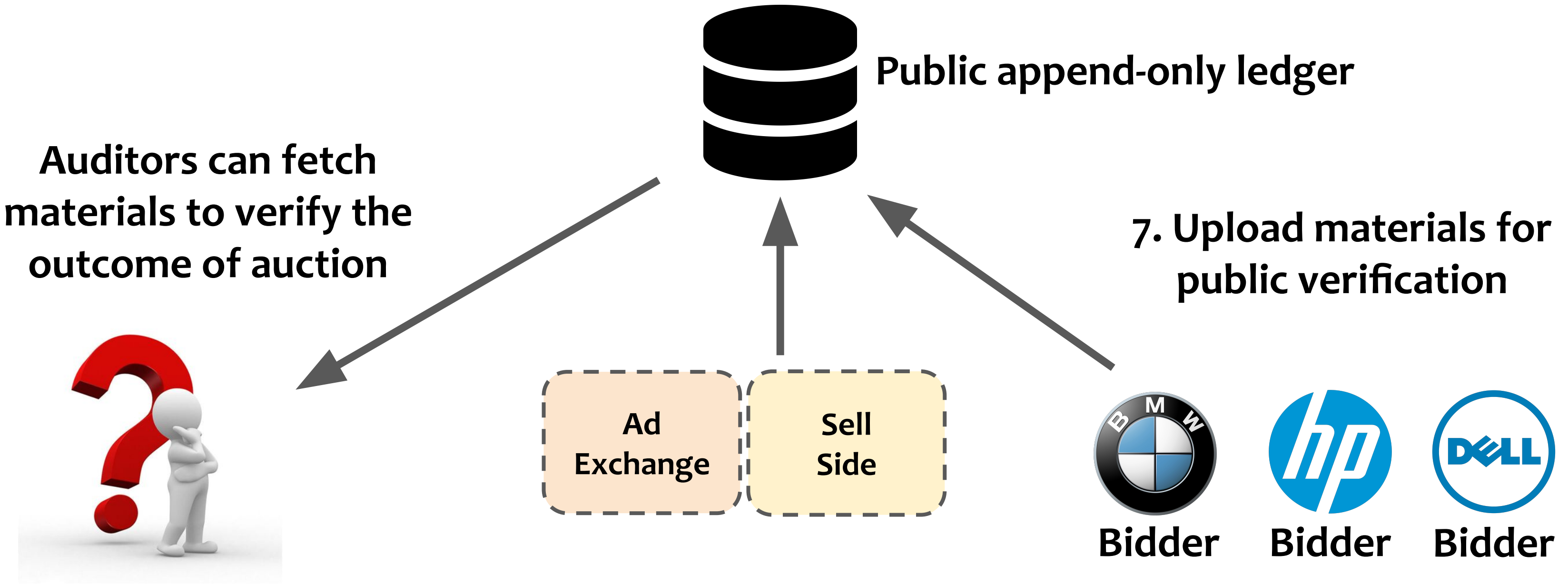
Architecture of Addax



Architecture of Addax


Public append-only ledger
7. Upload materials for public verification
Ad Exchange
Sell Side
Bidder
Bidder
Bidder

Architecture of Addax



Public append-only ledger

Auditors can fetch materials to verify the outcome of auction

7. Upload materials for public verification

Ad Exchange

Sell Side

Bidder

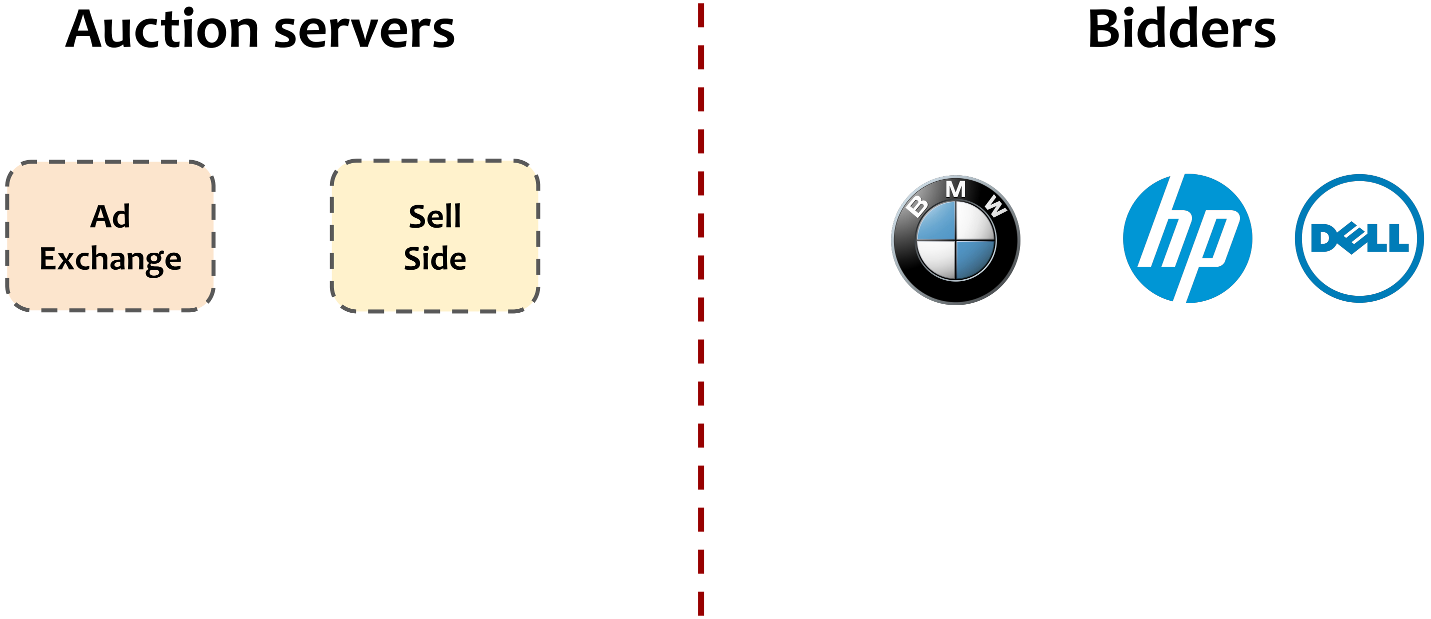
Bidder

Bidder

Rest of this talk

- Overview of Addax
- Private auction protocol
- Make auction verifiable
- Experimental evaluation

Threat model



Auction servers

Bidders

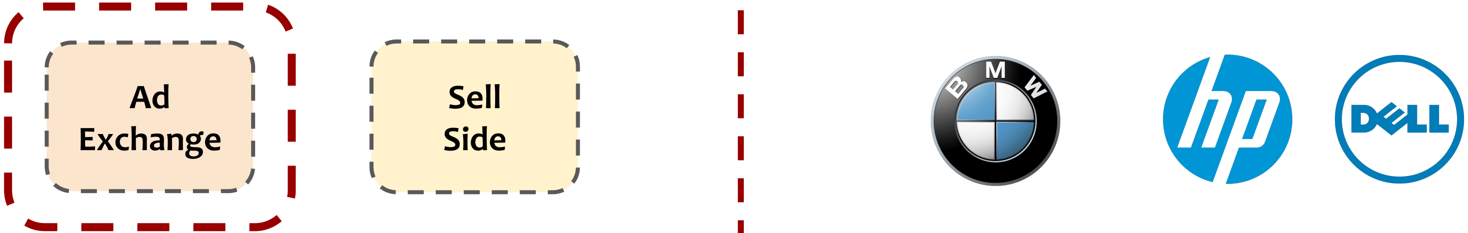
Ad Exchange

Sell Side

Threat model

Auction servers

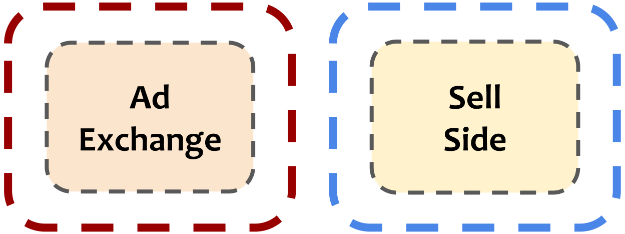
Bidders



One server could **deviate arbitrarily**

Threat model

Auction servers

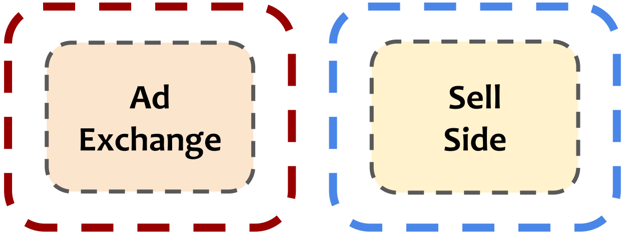


One server could deviate arbitrarily but another server is honest

Bidders

Threat model

Auction servers



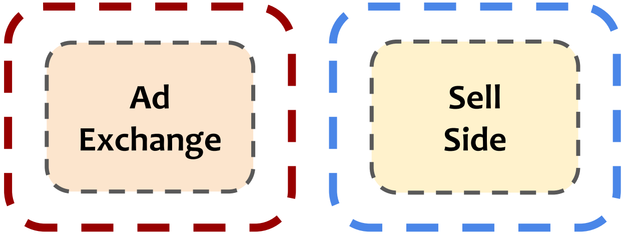
One server could deviate arbitrarily but another server is honest (the honest server can be any one)

Bidders

BMW hp DELL

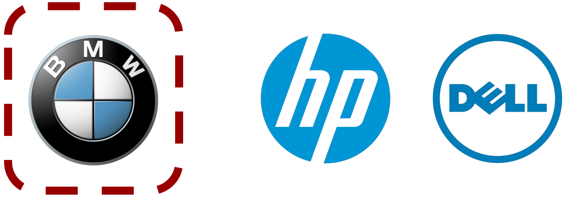
Threat model

Auction servers



One server could deviate arbitrarily but another server is honest (the honest server can be any one)

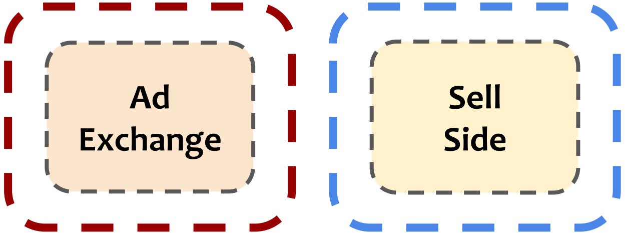
Bidders



Some Bidders can deviate arbitrarily

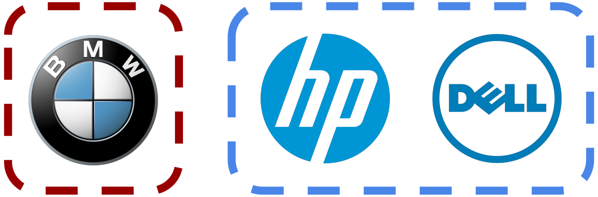
Threat model

Auction servers



One server could deviate arbitrarily but another server is honest (the honest server can be any one)

Bidders



Some Bidders can deviate arbitrarily and the others are honest

A non-private auction (strawman)

	Bids
hp Bidder	3
DELL Bidder	2
BMW Bidder	1

A non-private auction (strawman)

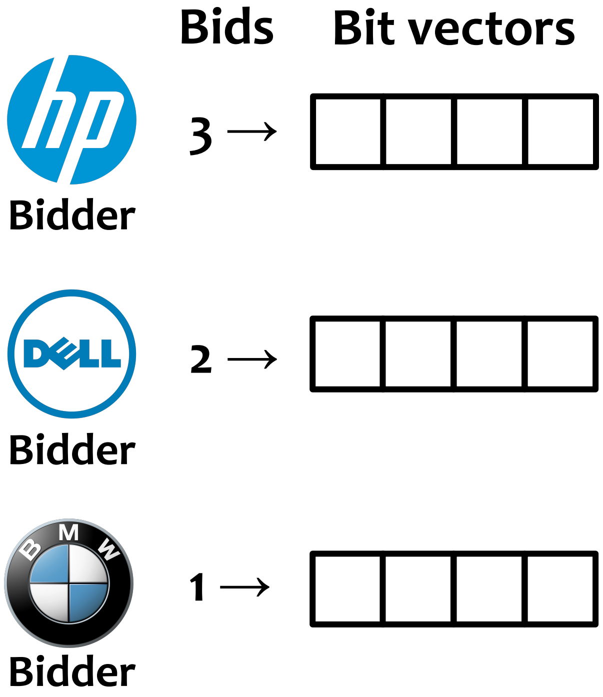
	Bids
Bidder	3
Bidder	2
Bidder	1

ℓ=4 in this example

↓

Given the upper bound of bids ℓ

A non-private auction (strawman)



Bids Bit vectors

Bidder 3 →

Bidder 2 →

Bidder 1 →

ℓ=4 in this example

↓

Given the upper bound of bids ℓ

A non-private auction (strawman)

	Bids	Bit vectors			
hp Bidder	3 →	1	1	1	0
DELL Bidder	2 →	1	1	0	0
BMW Bidder	1 →	1	0	0	0

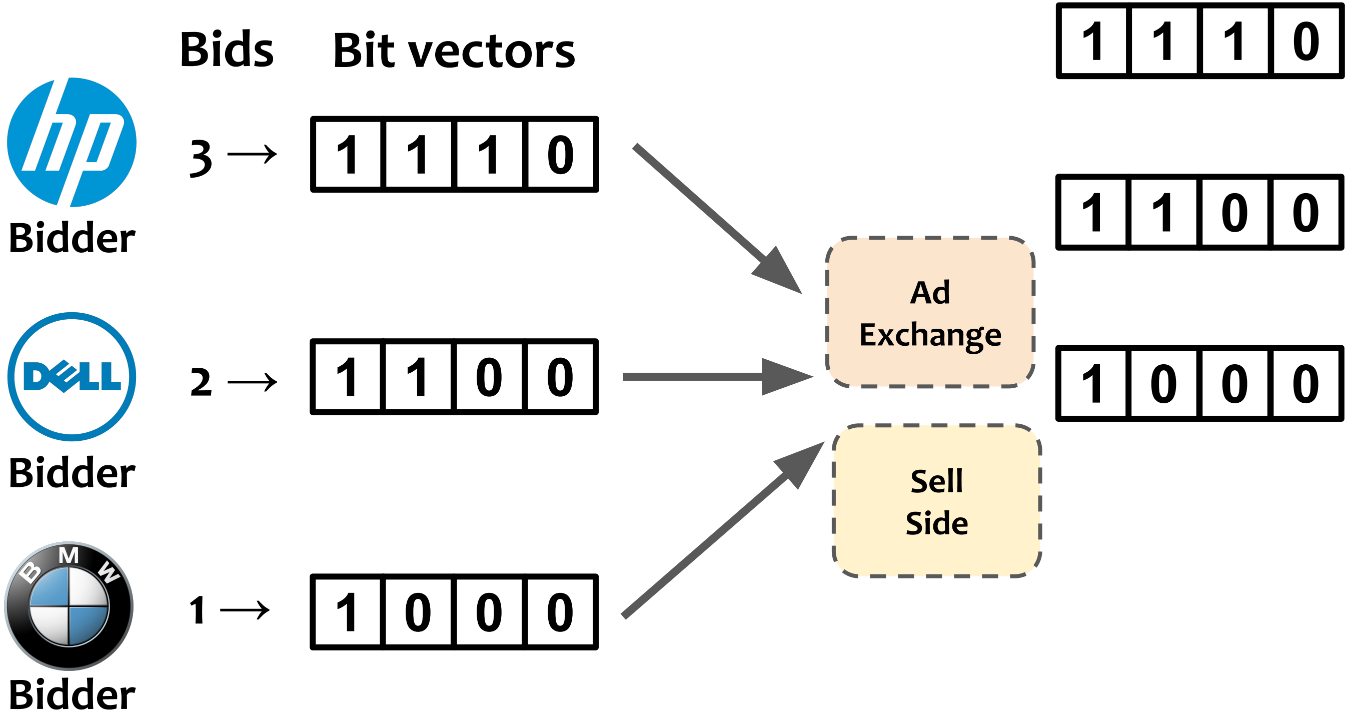
For a bid x, the first x bits are set to 1

ℓ=4 in this example

↓

Given the upper bound of bids ℓ

A non-private auction (strawman)



	Bids	Bit vectors
Bidder	3 →	1 1 1 0
Bidder	2 →	1 1 0 0
Bidder	1 →	1 0 0 0

Ad Exchange

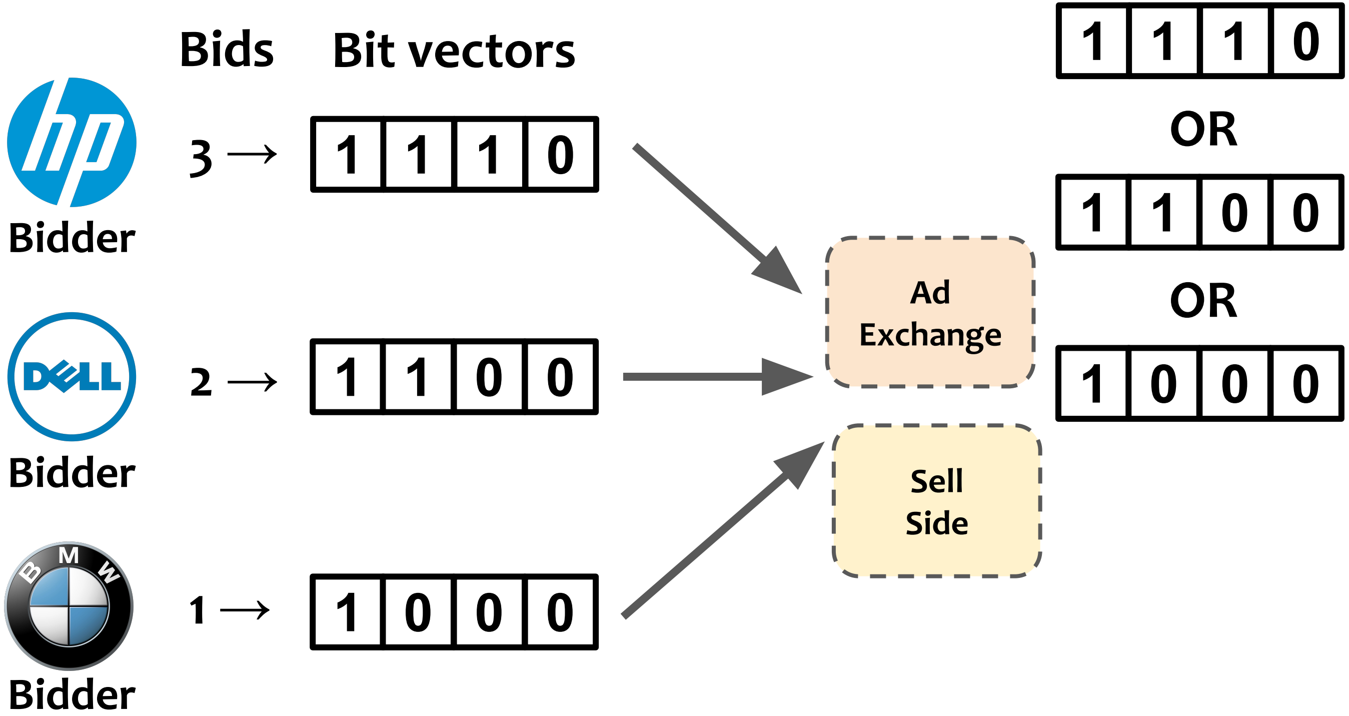
Sell Side

1 1 1 0

1 1 0 0

1 0 0 0

A non-private auction (strawman)



Bids Bit vectors

Bidder 3 → 1 1 1 0

Bidder 2 → 1 1 0 0

Bidder 1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

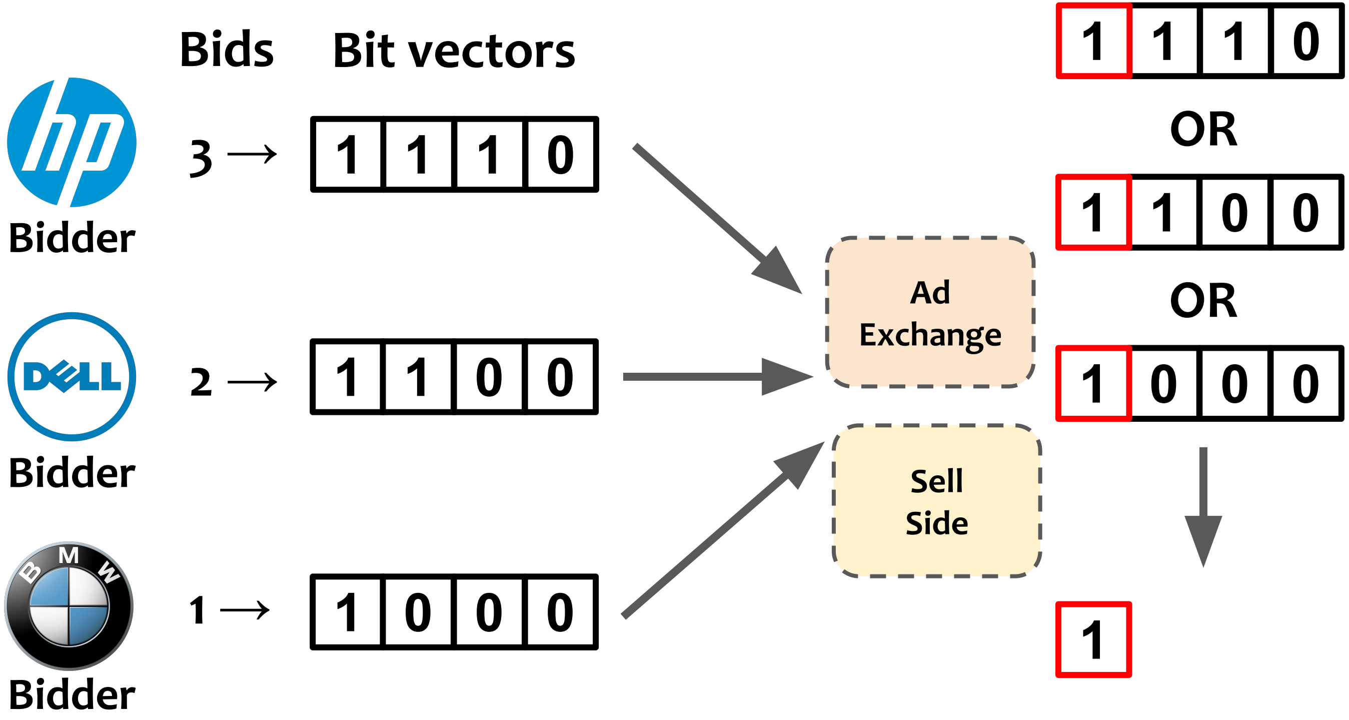
OR

1 1 0 0

OR

1 0 0 0

A non-private auction (strawman)



Bids

Bit vectors

Bidder 3 → 1 1 1 0

Bidder 2 → 1 1 0 0

Bidder 1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

OR

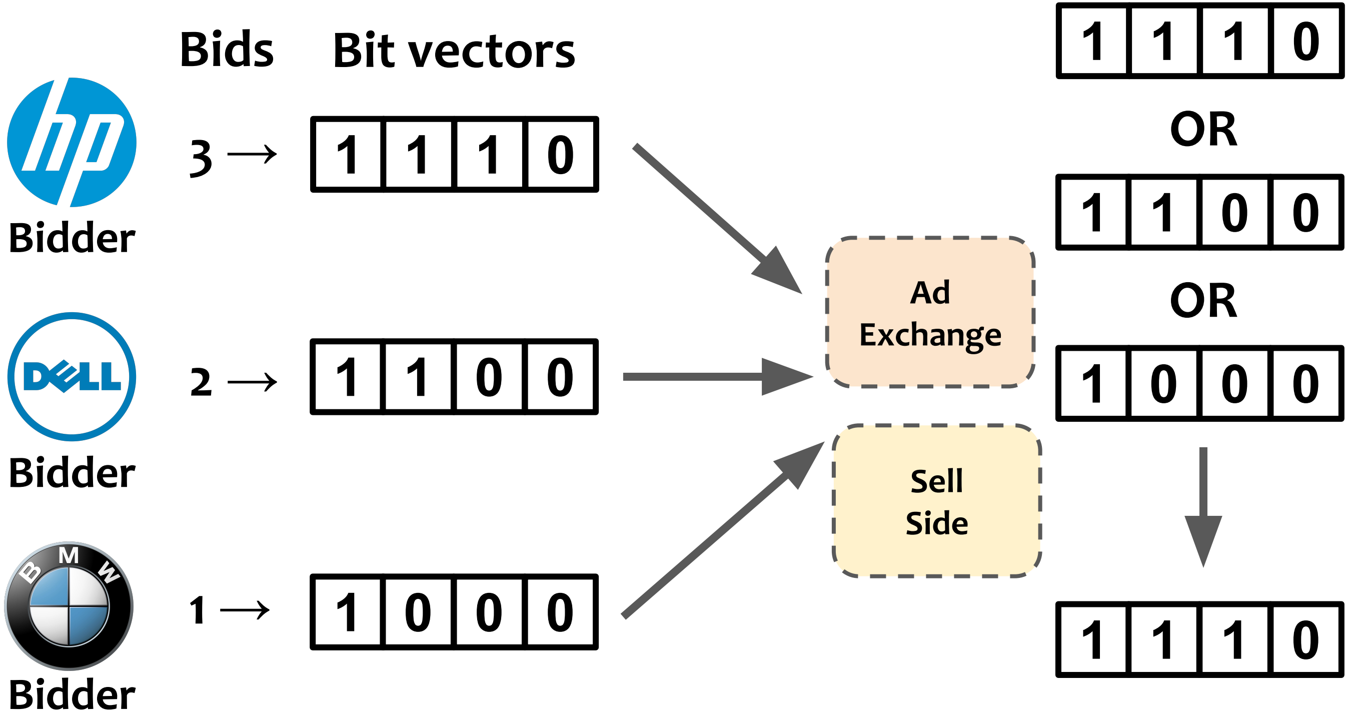
1 1 0 0

OR

1 0 0 0

1

A non-private auction (strawman)



	Bids	Bit vectors			
Bidder	3 →	1	1	1	0
Bidder	2 →	1	1	0	0
Bidder	1 →	1	0	0	0

Ad Exchange

Sell Side

1	1	1	0

OR

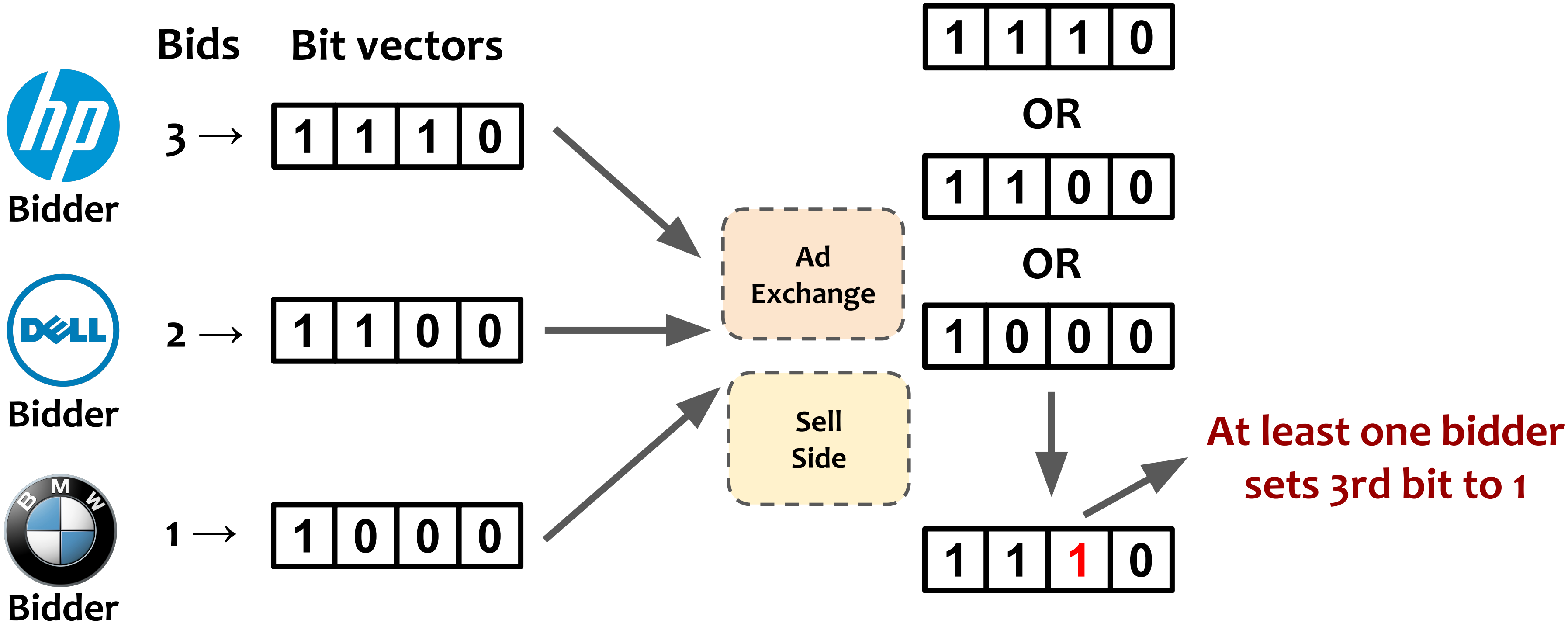
1	1	0	0

OR

1	0	0	0

1	1	1	0

A non-private auction (strawman)



Bids

Bit vectors

Bidder 3 → 1 1 1 0

Bidder 2 → 1 1 0 0

Bidder 1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

OR

1 1 0 0

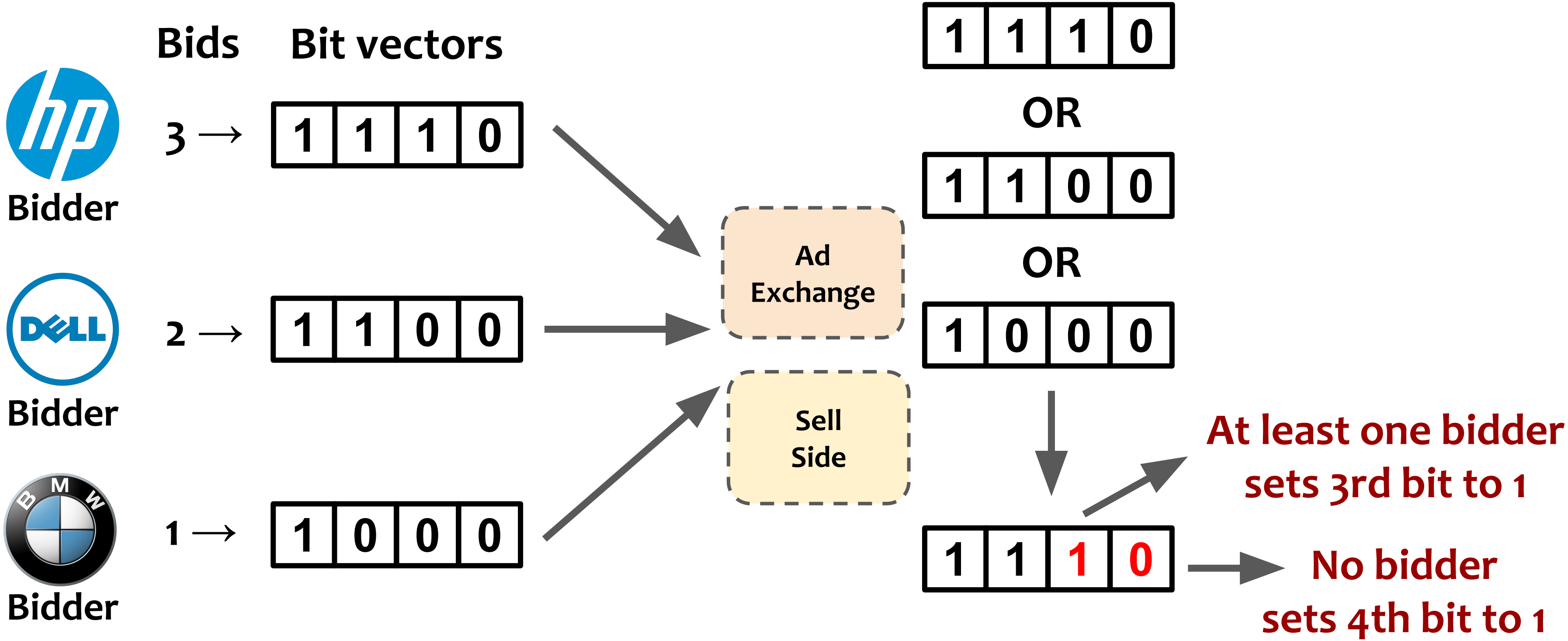
OR

1 0 0 0

1 1 1 0

At least one bidder sets 3rd bit to 1

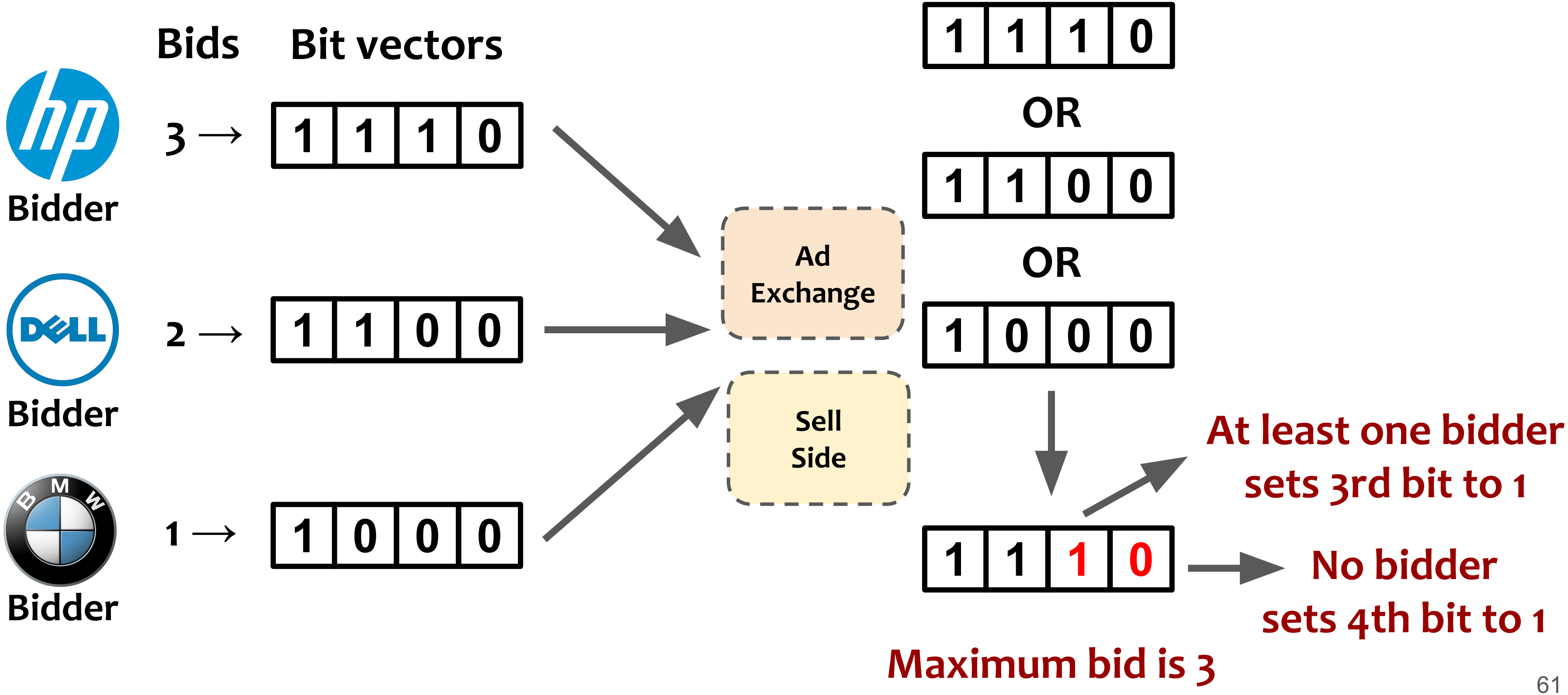
A non-private auction (strawman)



At least one bidder sets 3rd bit to 1

No bidder sets 4th bit to 1

A non-private auction (strawman)



Bids

Bit vectors

Bidder

3 → 1 1 1 0

Bidder

2 → 1 1 0 0

Bidder

1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

OR

1 1 0 0

OR

1 0 0 0

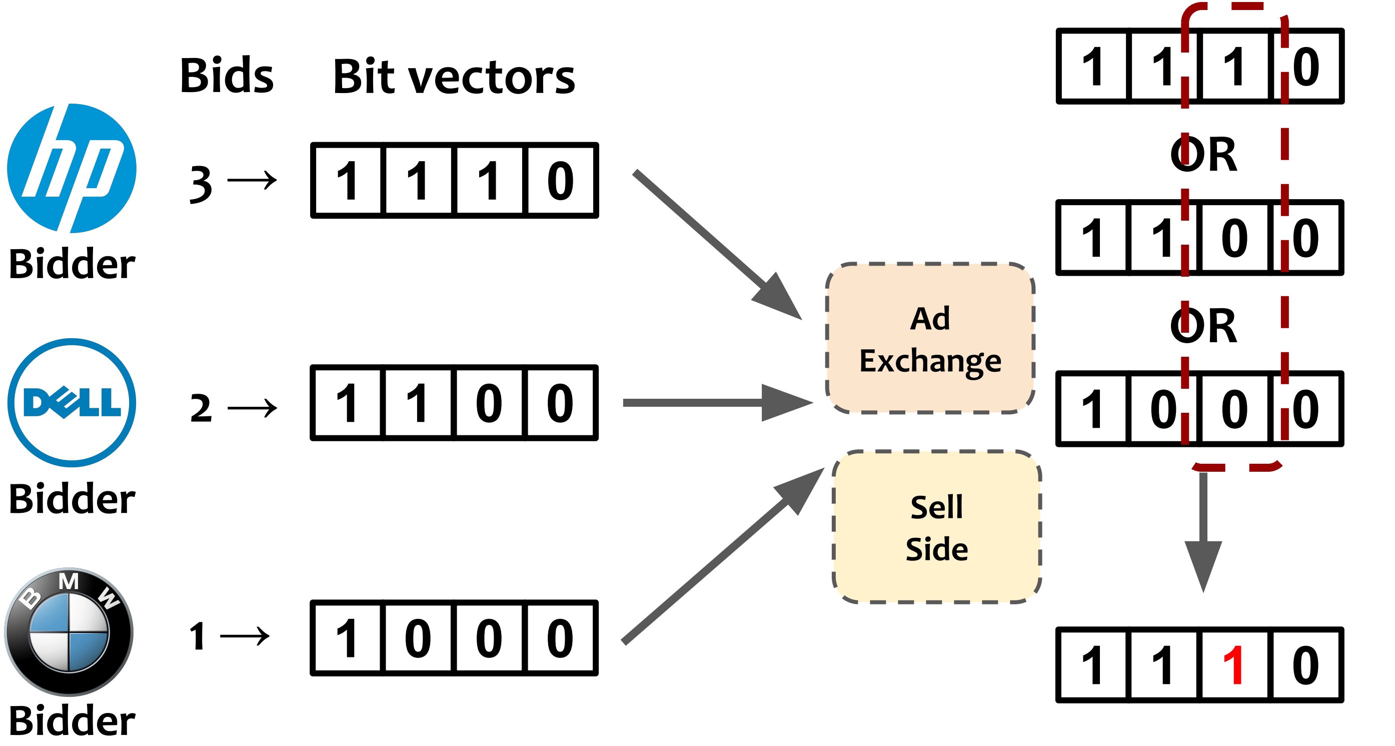
1 1 1 0

At least one bidder sets 3rd bit to 1

No bidder sets 4th bit to 1

Maximum bid is 3

A non-private auction (strawman)



Bids Bit vectors

Bidder 3 → 1 1 1 0

Bidder 2 → 1 1 0 0

Bidder 1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

OR

1 1 0 0

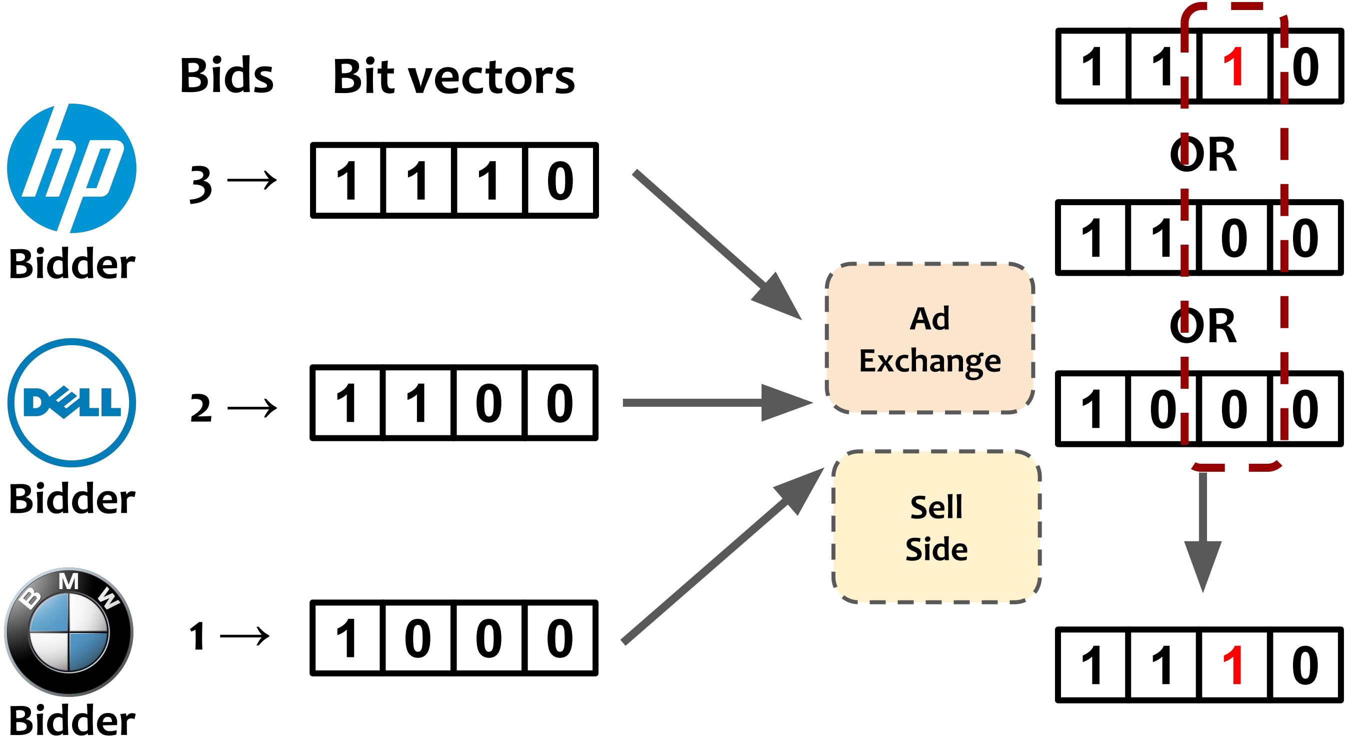
OR

1 0 0 0

1 1 1 0

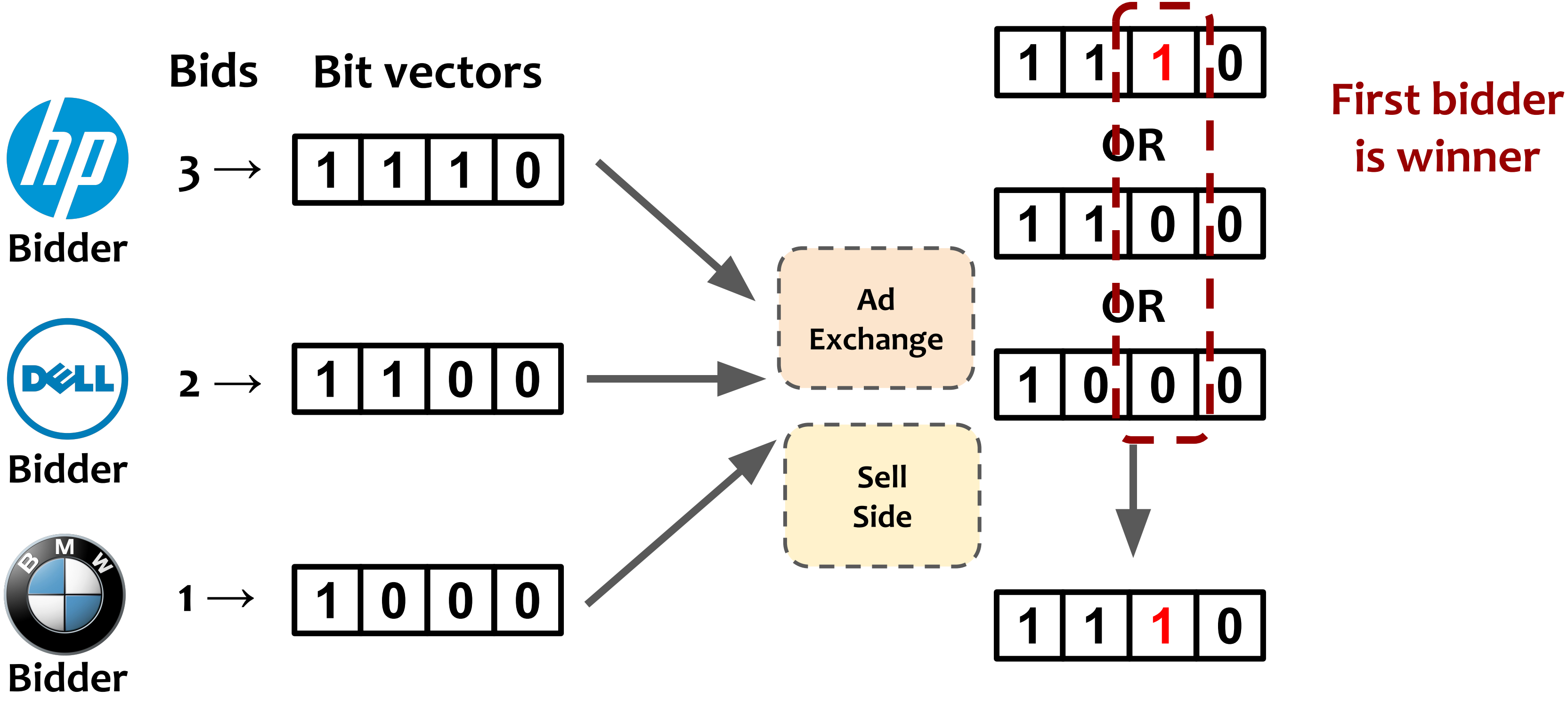
Maximum bid is 3

A non-private auction (strawman)



Maximum bid is 3

A non-private auction (strawman)



Bids Bit vectors

Bidder 3 → 1 1 1 0

Bidder 2 → 1 1 0 0

Bidder 1 → 1 0 0 0

Ad Exchange

Sell Side

1 1 1 0

OR

1 1 0 0

OR

1 0 0 0

1 1 1 0

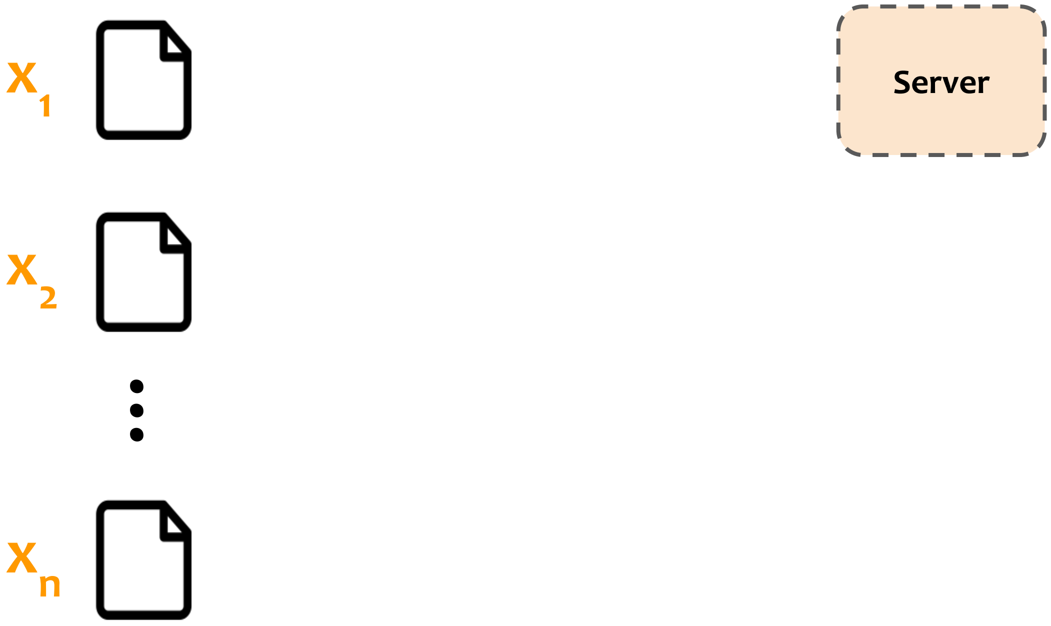
First bidder is winner

Maximum bid is 3

A non-private auction (strawman)

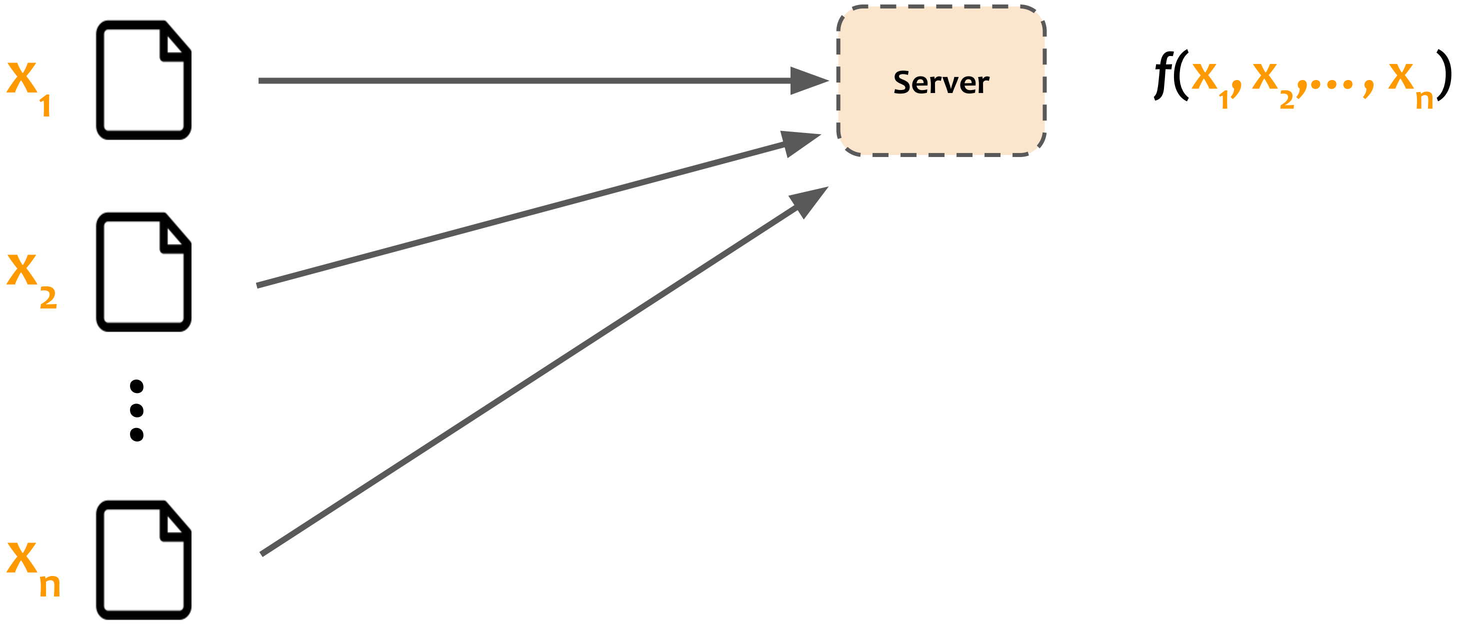
Key challenge to address:
privately compute bit-wise OR operations

Affine Aggregatable Encodings [Prio, NSDI’17*]



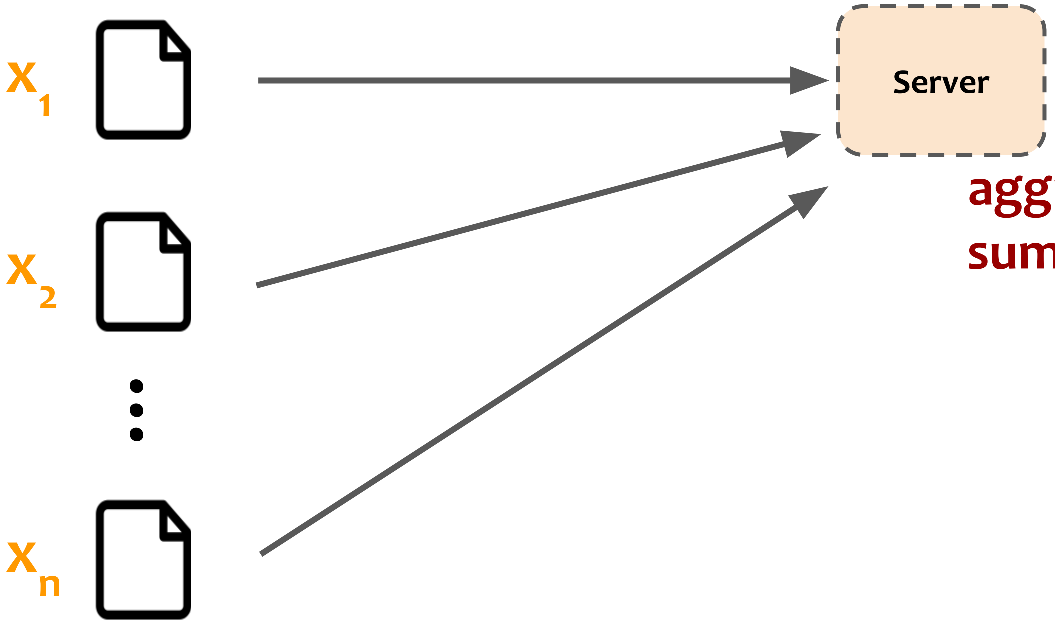
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI’17). Henry Corrigan-Gibbs and Dan Boneh

Affine Aggregatable Encodings [Prio, NSDI’17*]



*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI’17). Henry Corrigan-Gibbs and Dan Boneh

Affine Aggregatable Encodings [Prio, NSDI'17*]

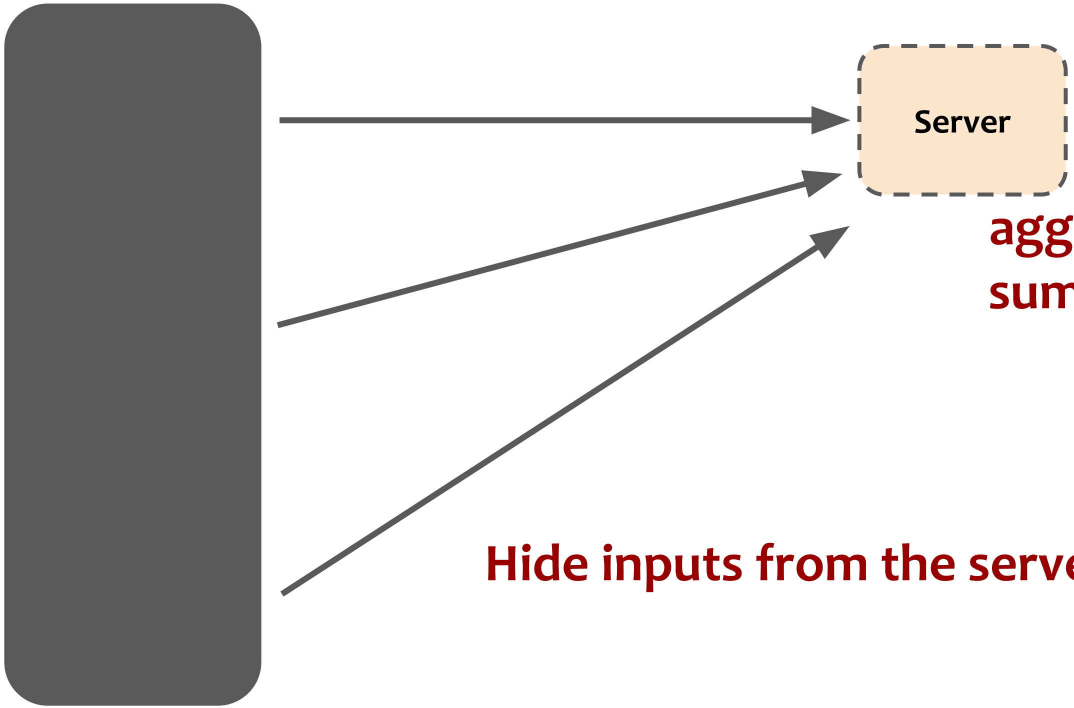


$f(x_1, x_2, \ldots, x_n)$

aggregate function *f* include sum, average, min, max, etc.

*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

Affine Aggregatable Encodings [Prio, NSDI'17*]



$f(x_1, x_2, \ldots, x_n)$

aggregate function *f* include sum, average, min, max, etc.

Hide inputs from the server

*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with a single server (non-private)

x_1

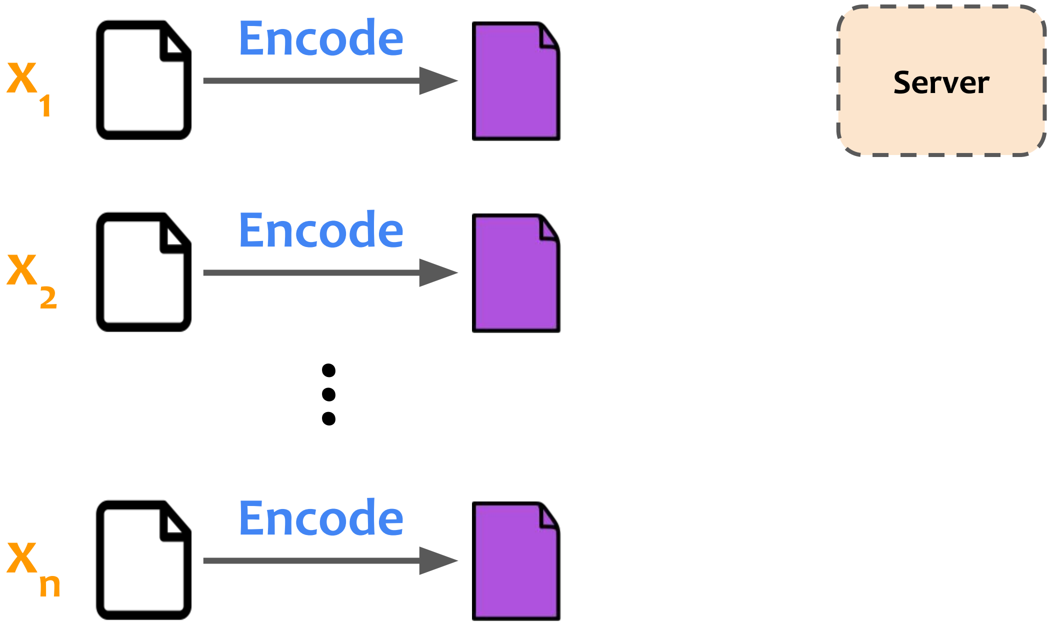
x_2

x_n

Server

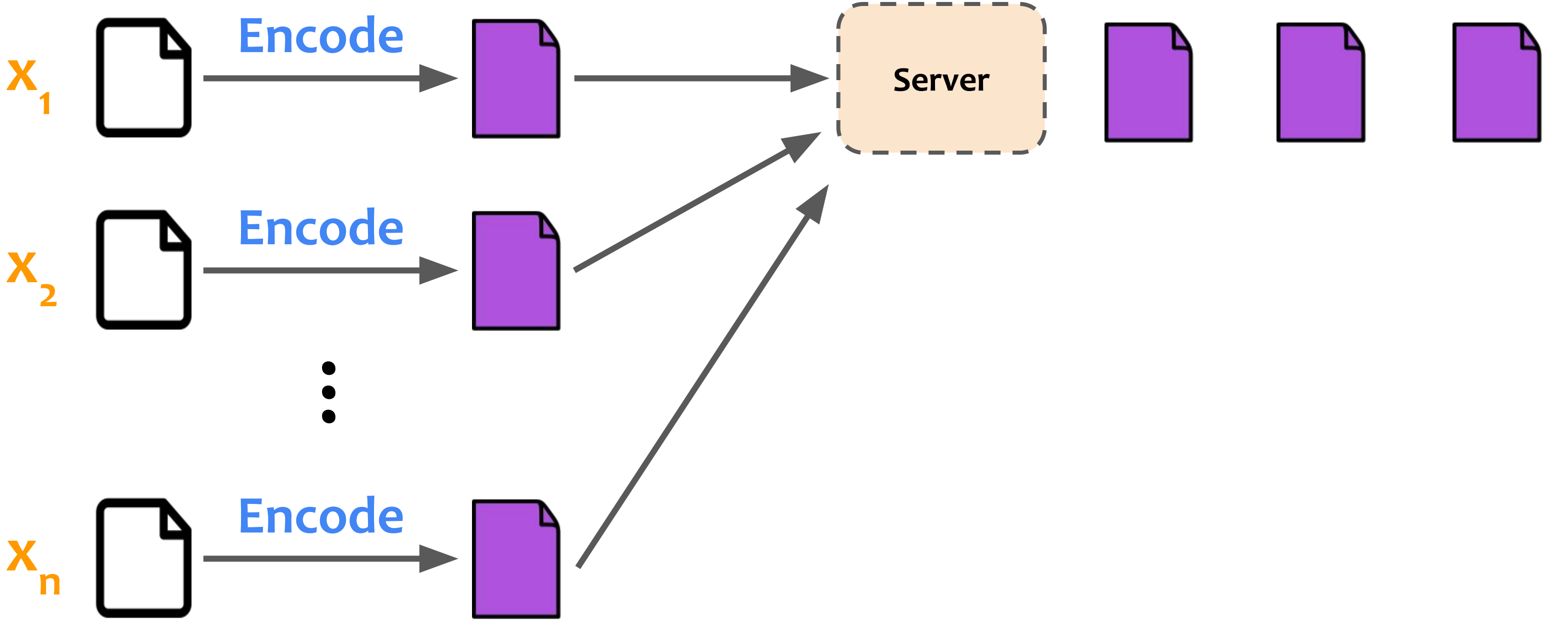
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE[*] with a single server (non-private)



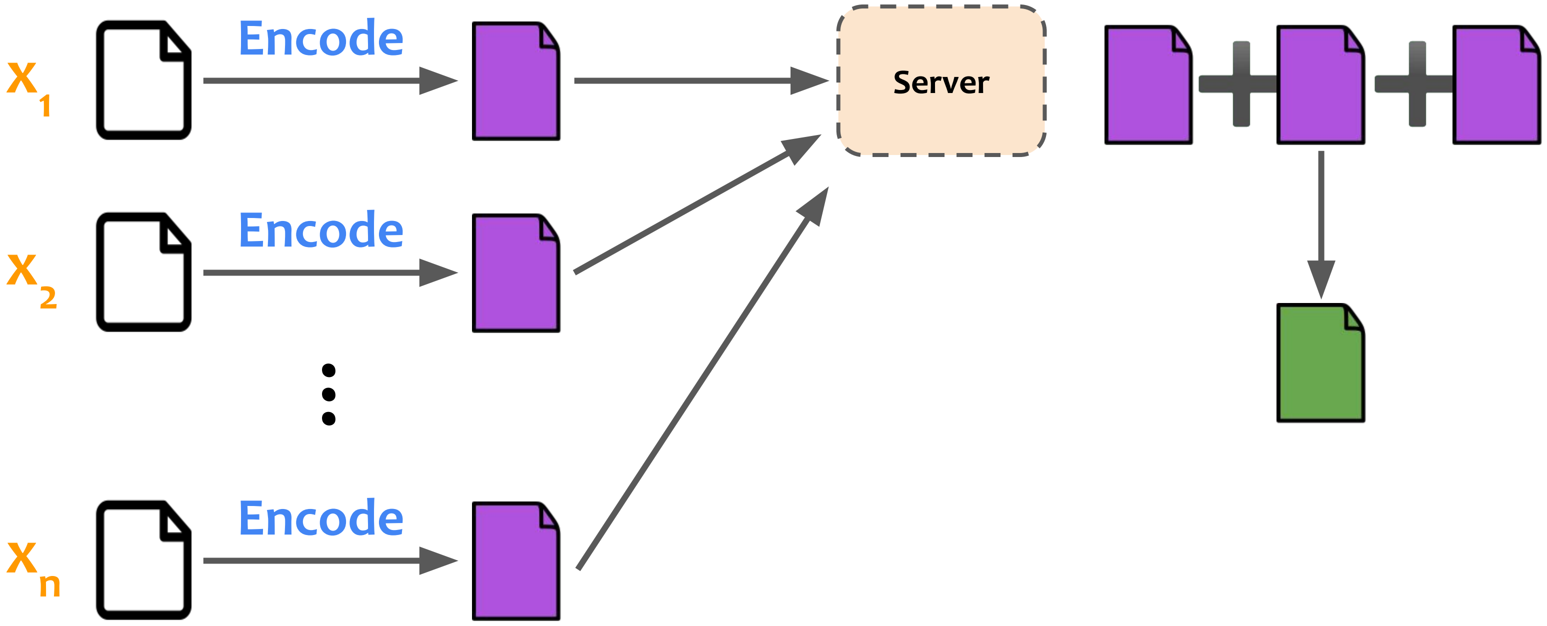
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with a single server (non-private)



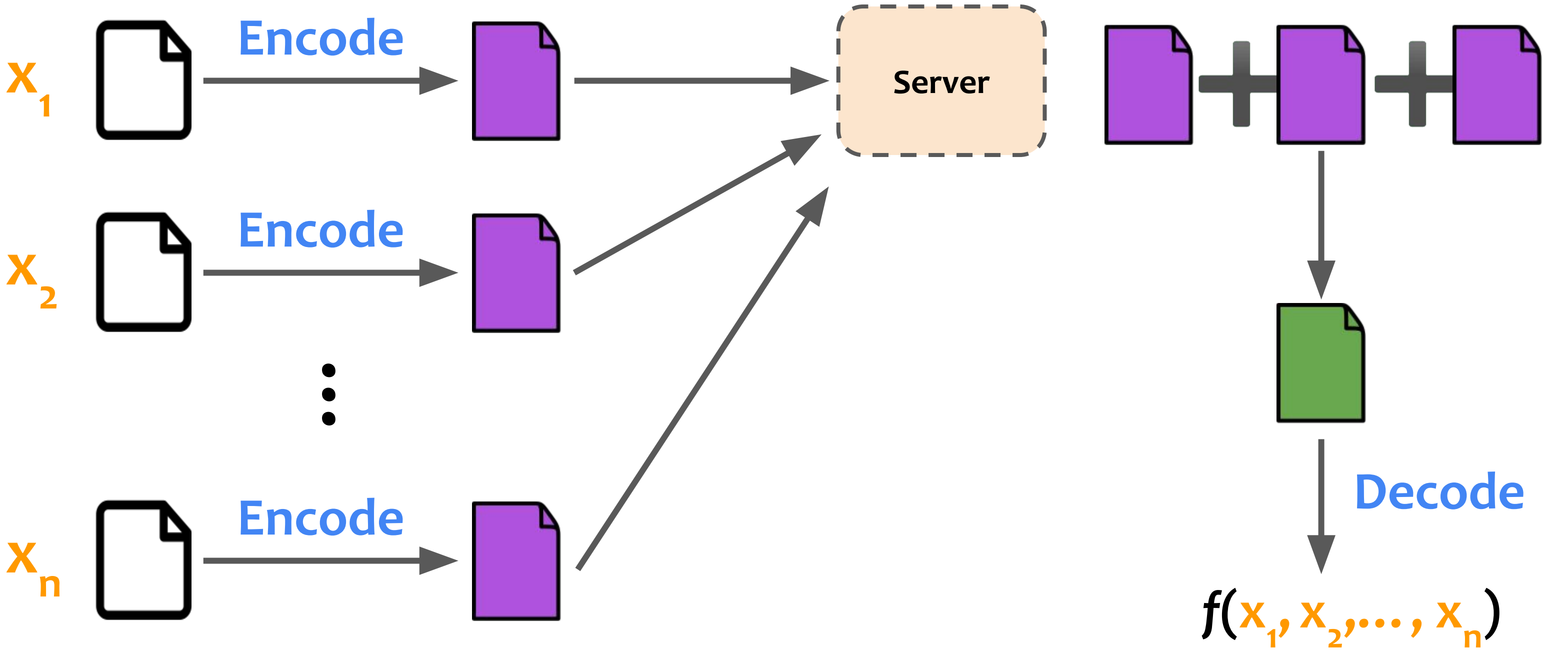
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI’17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with a single server (non-private)



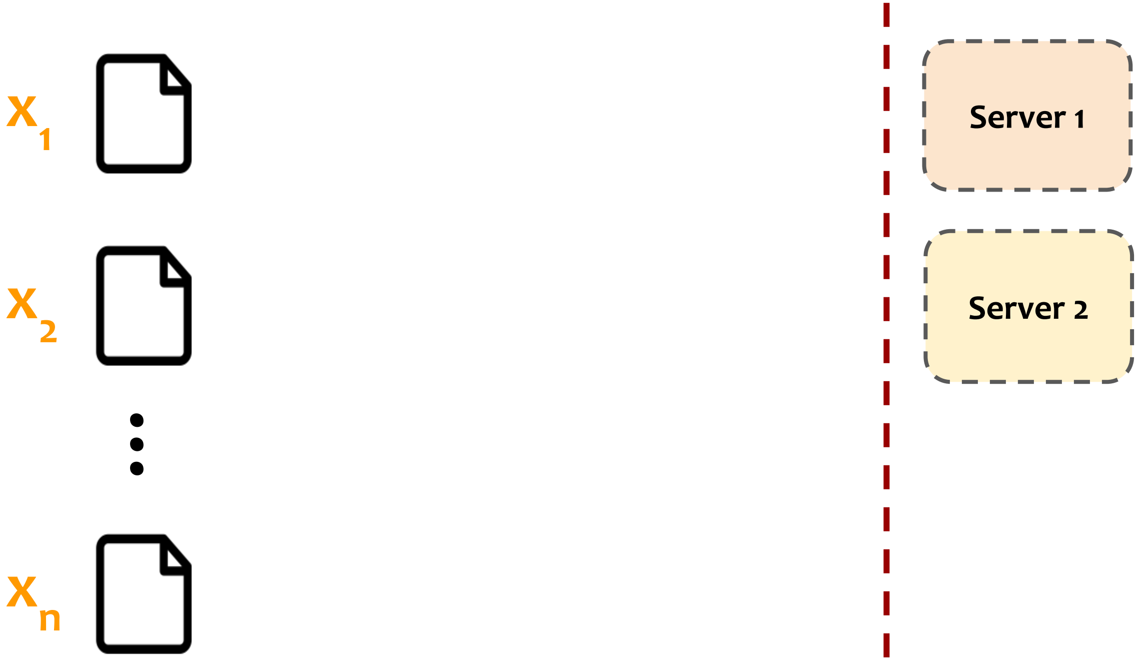
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with a single server (non-private)



*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



x_1

x_2

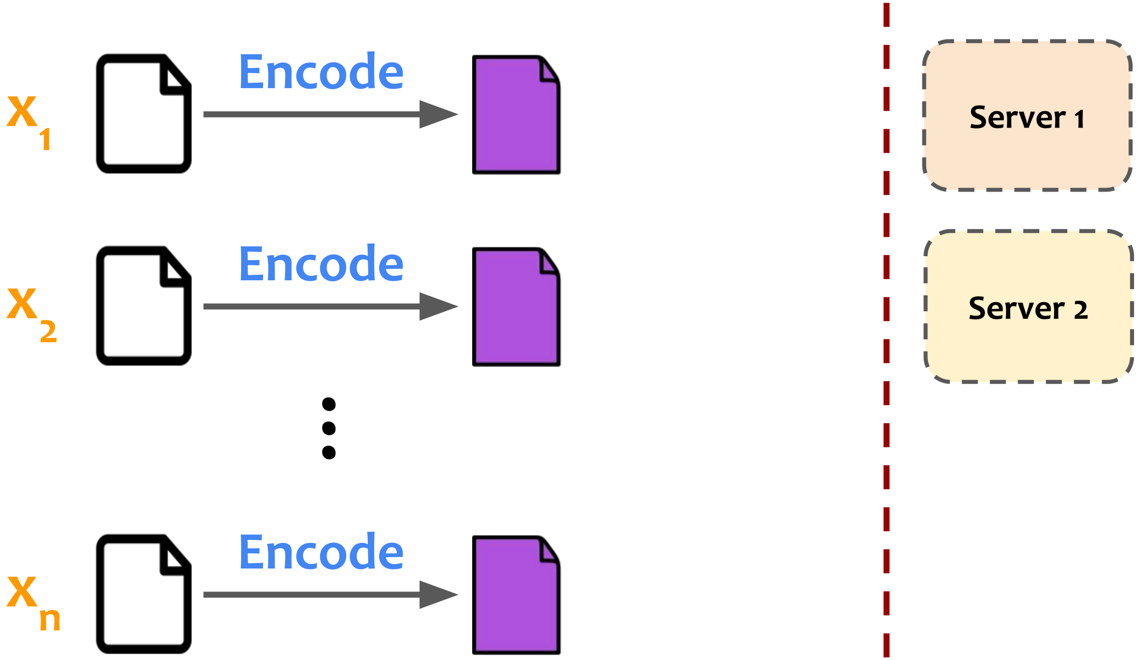
x_n

Server 1

Server 2

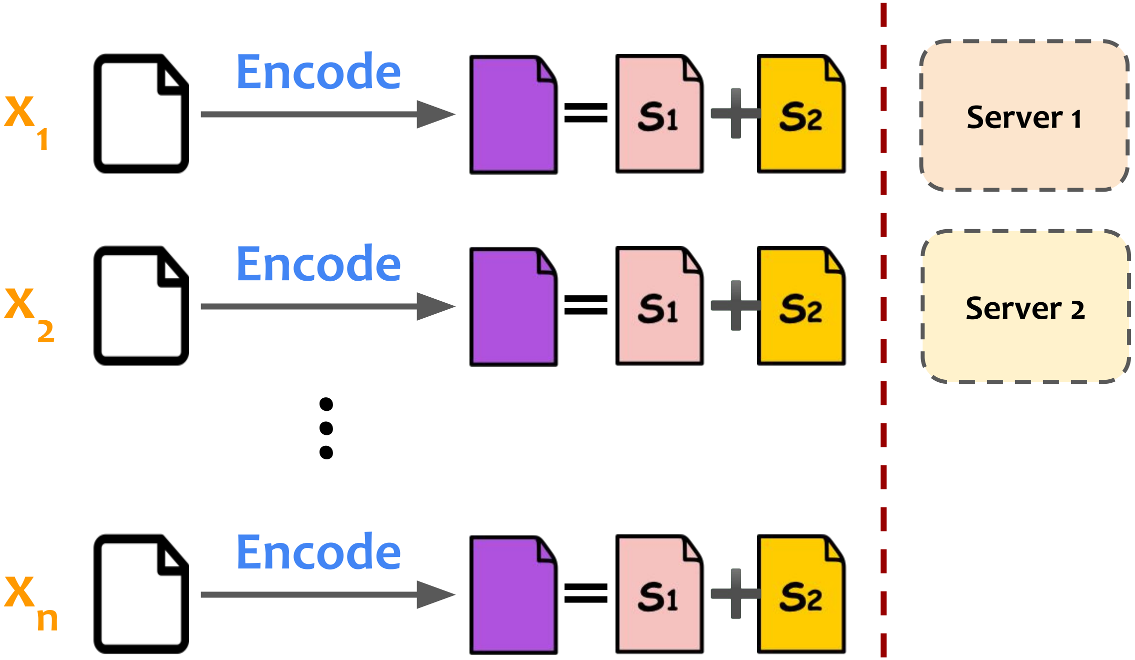
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



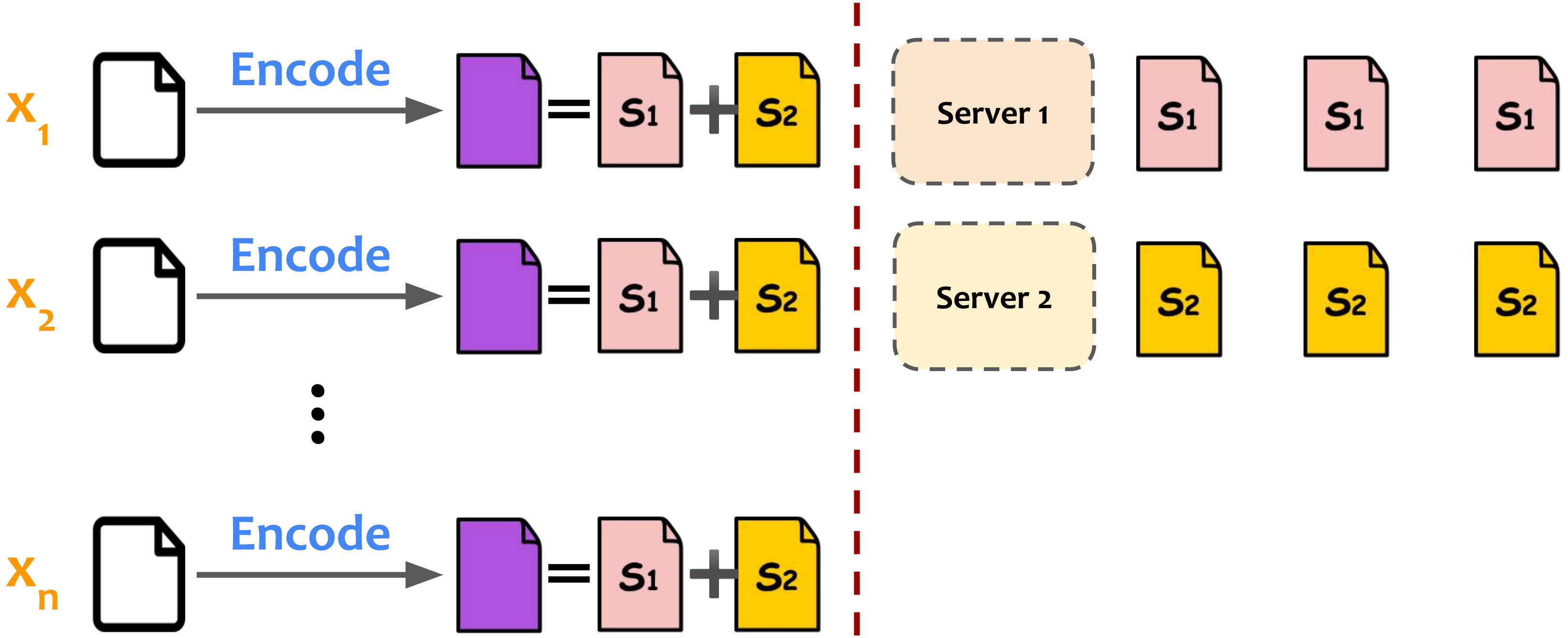
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



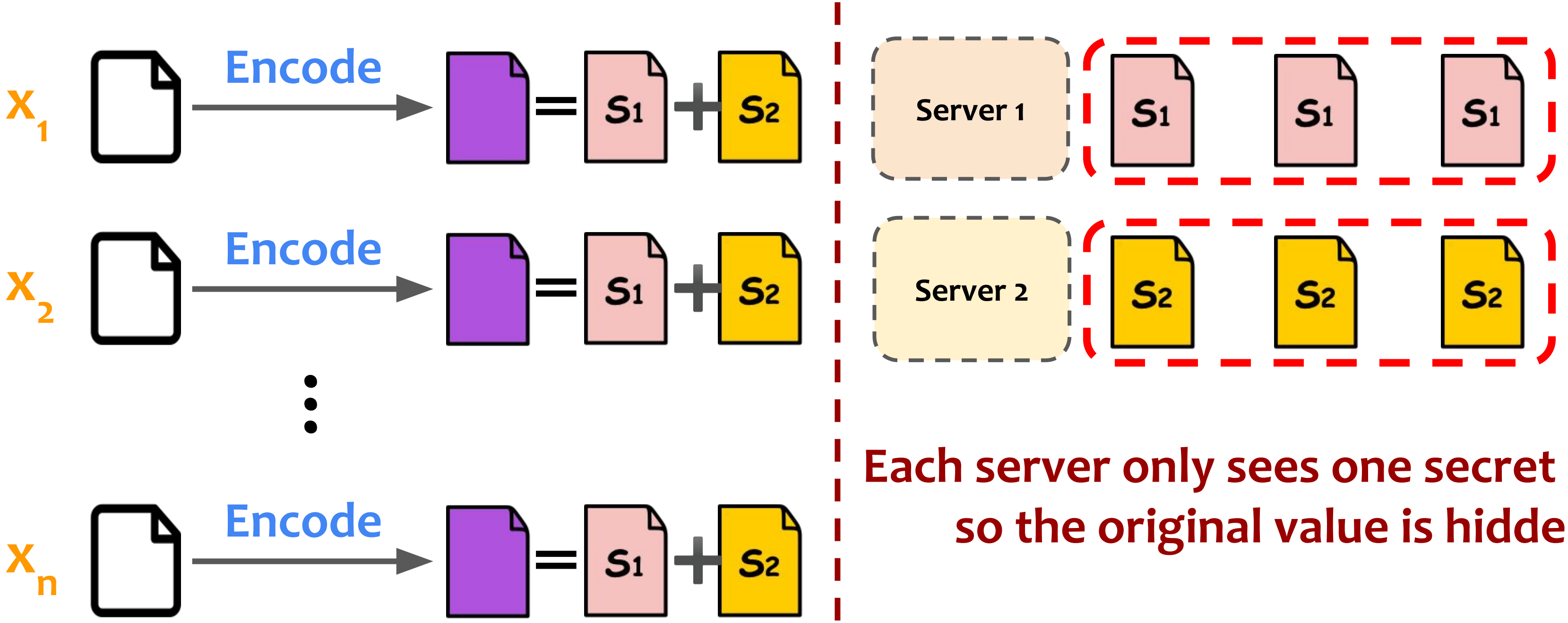
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

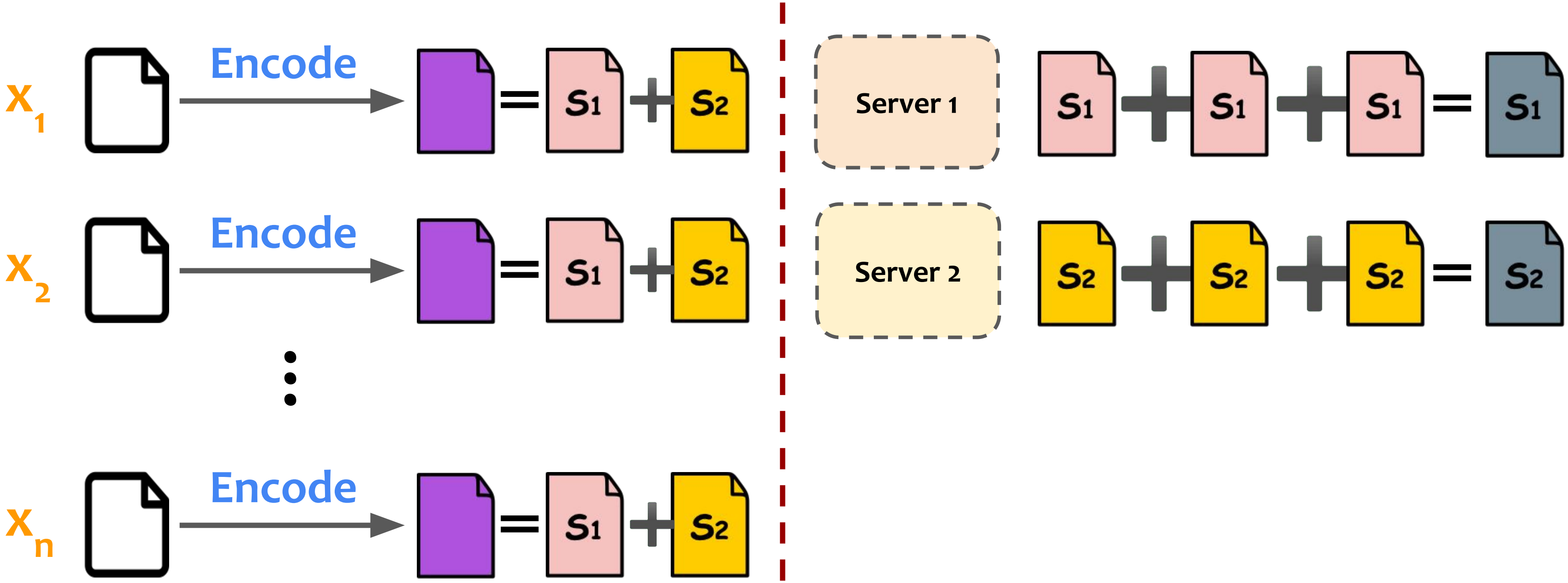
AFE* with multiple servers (private)



Each server only sees one secret share so the original value is hidden

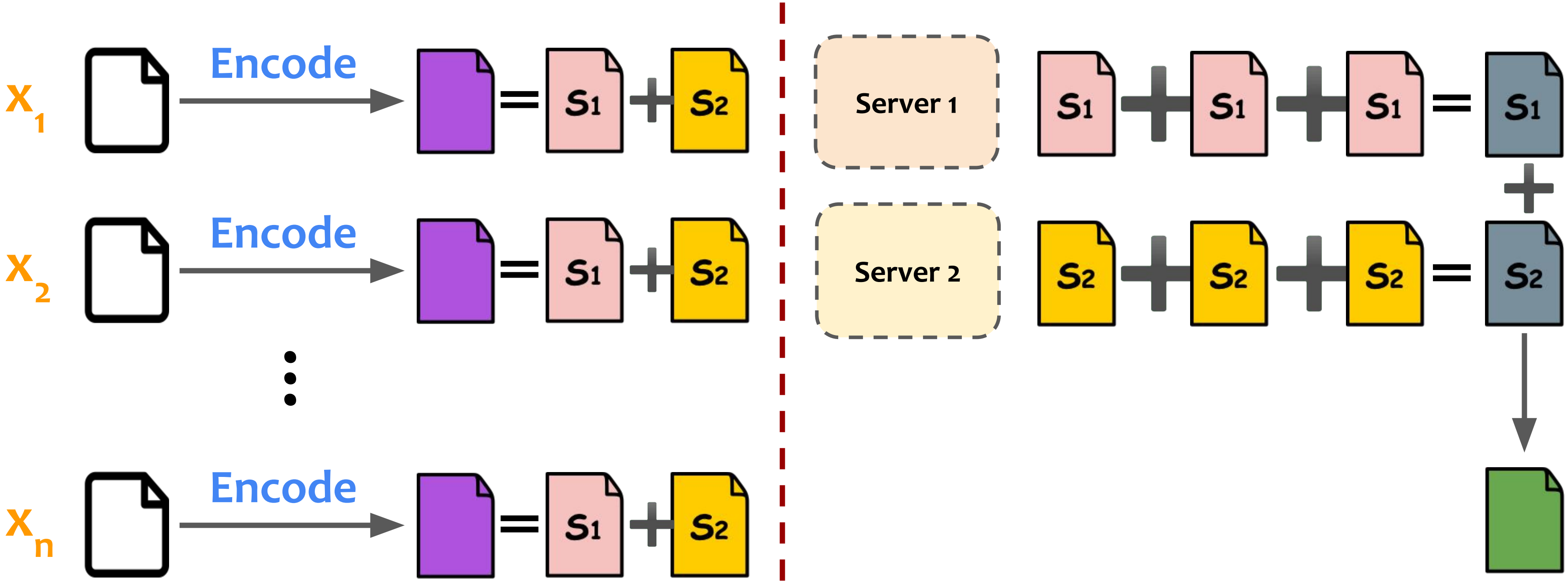
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



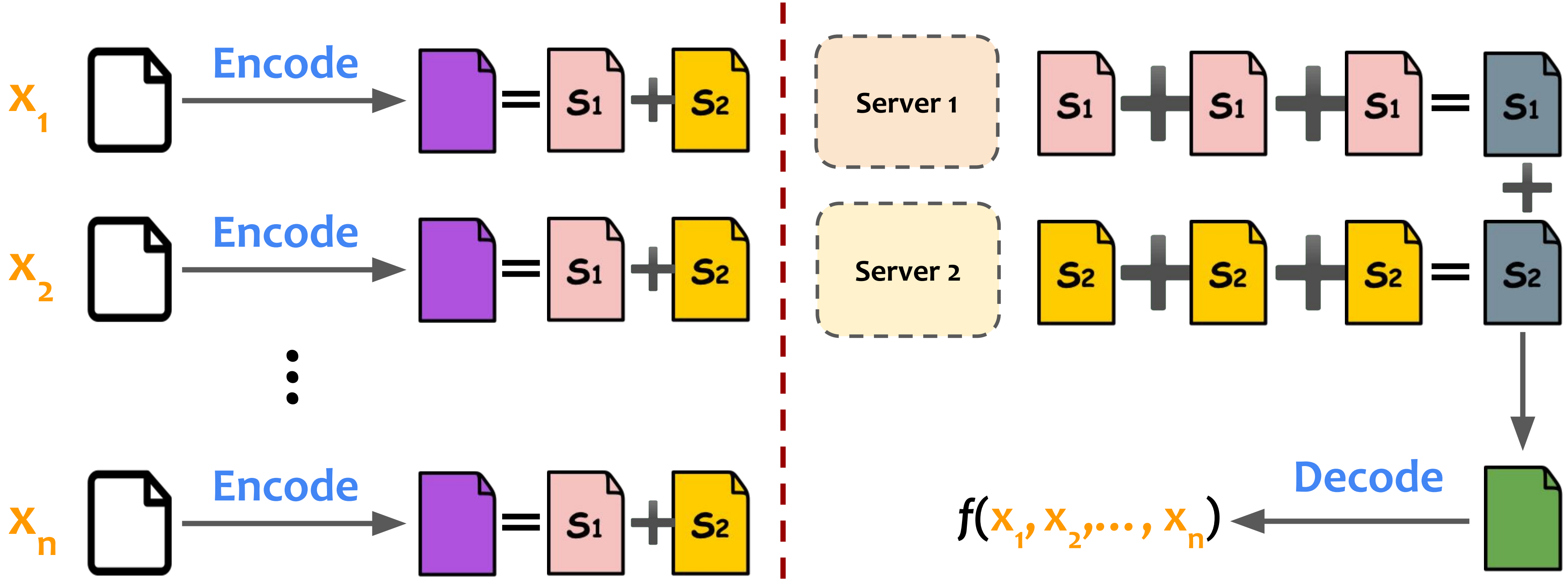
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI’17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



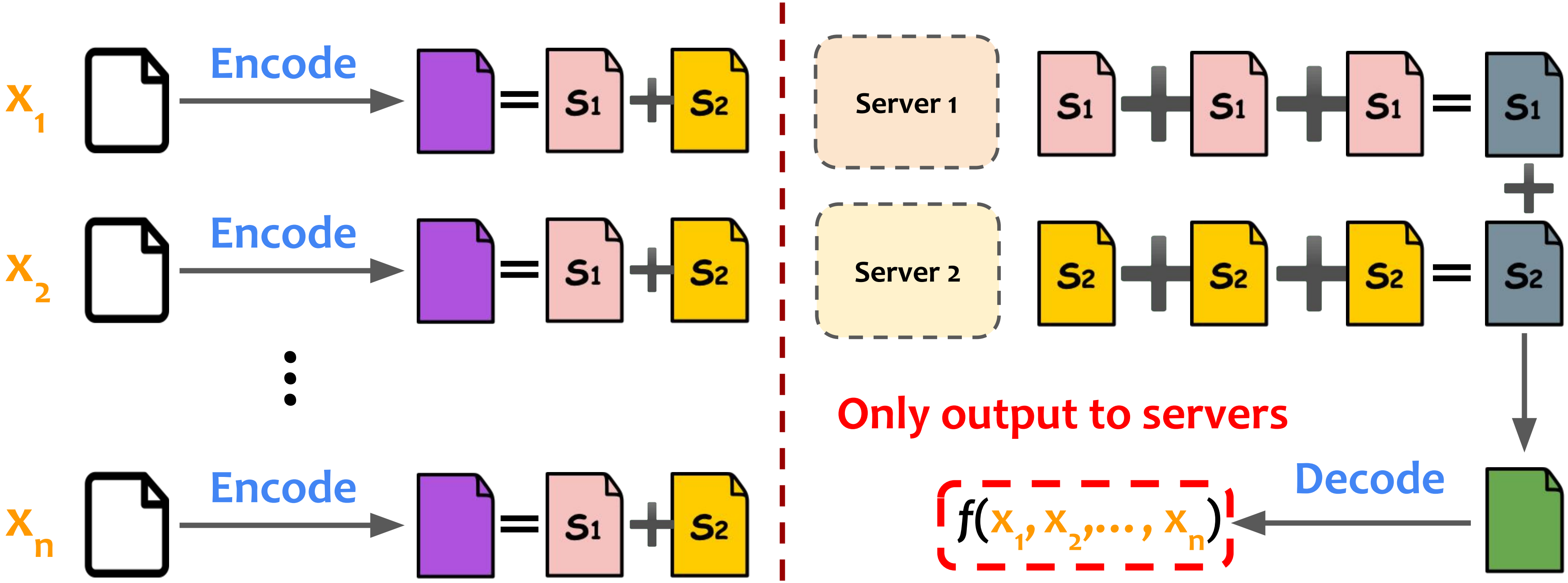
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



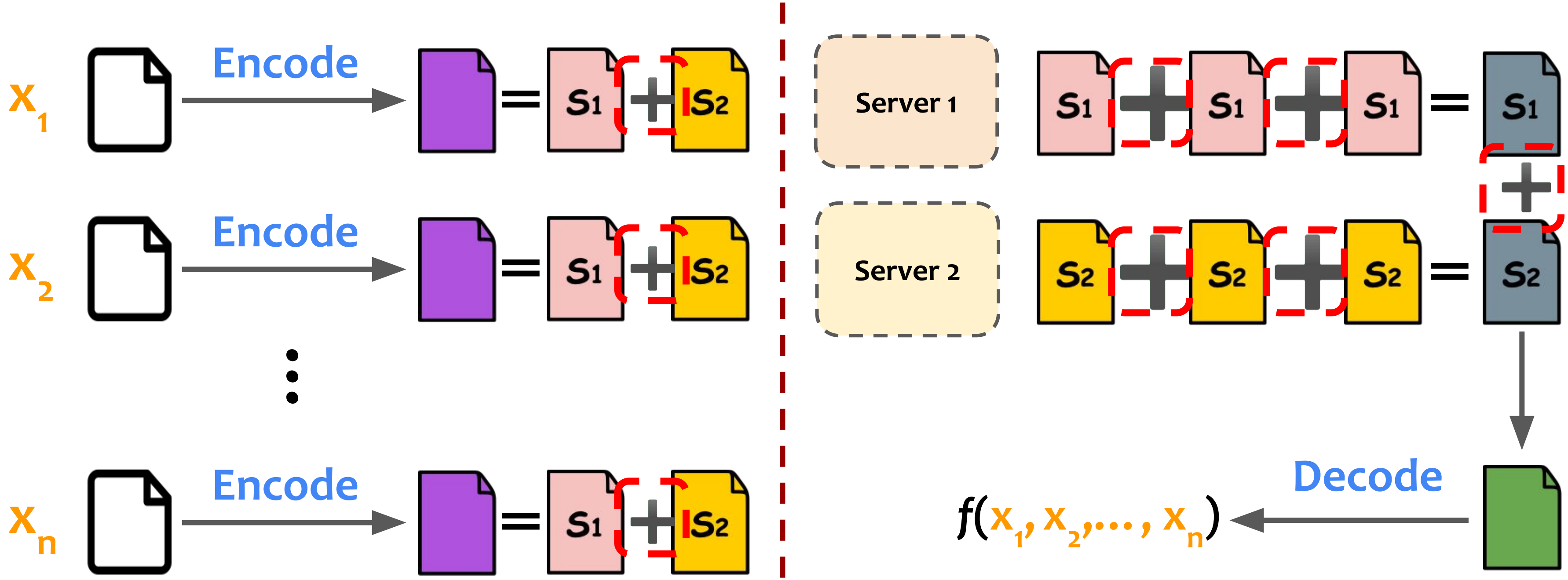
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



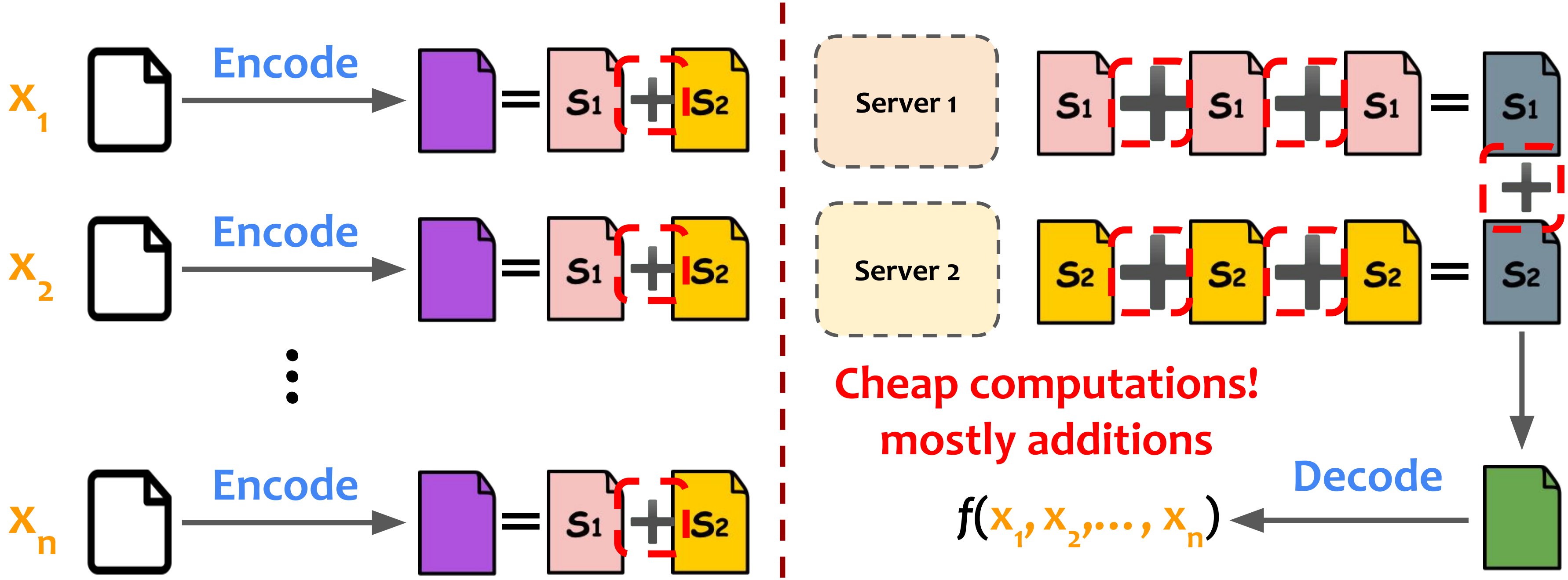
*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

AFE* with multiple servers (private)



Cheap computations!
mostly additions

*Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (NSDI'17). Henry Corrigan-Gibbs and Dan Boneh

Addax’s AFE for OR over bits

Input space: x in {0, 1}

Addax’s AFE for OR over bits

Input space: x in {0, 1} **Encoding output space: e in $\boldsymbol{Z}_p$**

Addax’s AFE for OR over bits

Input space: x in {0, 1}

Encoding output space: e in $\boldsymbol{Z}_p$

↓

Integers from 0 to p-1

Addax's AFE for OR over bits

Input space: x in {0, 1}   Encoding output space: e in $\boldsymbol{Z}_p$

Encode-OR(x):

Addax's AFE for OR over bits

Input space: x in {0, 1}

Encoding output space: e in $\mathbb{Z}_p$

```
Encode-OR(x):
        return e
```

Addax’s AFE for OR over bits

Input space: x in $\{0, 1\}$

Encoding output space: e in $\mathbb{Z}_p$

Encode-OR(x):

$$\text{return } e \begin{cases} 0 & \text{if } x = 0 \\ & \end{cases}$$

Addax’s AFE for OR over bits

Input space: x in {0, 1}

Encoding output space: e in Z_p

Encode-OR(x):

$$\text{return } e \begin{cases} 0 & \text{if } x = 0 \\ \text{a random element in } Z_p & \text{if } x = 1 \end{cases}$$

Addax's AFE for OR over bits

Input space: x in {0, 1}   Encoding output space: e in $\mathbb{Z}_p$

Encode-OR(x):

$$\text{return } e = \begin{cases} 0 & \text{if } x = 0 \\ \text{a random element in } \mathbb{Z}_p & \text{if } x = 1 \end{cases}$$

Decode-OR(S):

Addax's AFE for OR over bits

Input space: x in {0, 1}

Encoding output space: e in $\boldsymbol{Z}_p$

Encode-OR(x):

return e $\begin{cases} 0 & \text{if } x = 0 \\ \text{a random element in } \boldsymbol{Z}_p & \text{if } x = 1 \end{cases}$

Decode-OR(S):

return y

Addax's AFE for OR over bits

Input space: x in $\{0, 1\}$ **Encoding output space: e in $\mathbb{Z}_p$**

Encode-OR(x):

$$\text{return } e \begin{cases} 0 & \text{if } x = 0 \\ \text{a random element in } \mathbb{Z}_p & \text{if } x = 1 \end{cases}$$

Decode-OR(S):

$$\text{return } y \begin{cases} 0 & \text{if } S = 0 \\ & \end{cases}$$

Addax's AFE for OR over bits

Input space: x in $\{0, 1\}$ **Encoding output space:** e in $\boldsymbol{Z}_p$

Encode-OR(x):

$$\text{return } e \begin{cases} 0 & \text{if } x = 0 \\ \text{a random element in } \boldsymbol{Z}_p & \text{if } x = 1 \end{cases}$$

Decode-OR(S):

$$\text{return } y \begin{cases} 0 & \text{if } S = 0 \\ 1 & \text{otherwise} \end{cases}$$

Addax's AFE for OR over bits

Toy example for p = 5:

Addax’s AFE for OR over bits

Toy example for p = 5:

In reality, p should be large enough to ensure
a negligible decoding failure probability
(we experimented with p of 192 bits)

Addax’s AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0

Encode-OR(0) → 0

Encode-OR(0) → 0

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0

Encode-OR(0) → 0

Encode-OR(0) → 0

Sum up encoding values

0+0+0 (mod 5) = 0

Addax's AFE for OR over bits

Toy example for $p = 5$:

Encode-OR(0) → 0

Encode-OR(0) → 0

Encode-OR(0) → 0

Sum up encoding values

$0+0+0 \pmod{5} = 0$

Decode-OR(0) → 0

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0

Encode-OR(0) → 0

Encode-OR(0) → 0

Sum up encoding values

0+0+0 (mod 5) = 0

Decode-OR(0) → 0

0 | 0 | 0 = 0

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0
Encode-OR(0) → 0
Encode-OR(0) → 0

Encode-OR(0) → 0
Encode-OR(1) → 4
Encode-OR(1) → 3

Sum up encoding values
0+0+0 (mod 5) = 0

Decode-OR(0) → 0
0 | 0 | 0 = 0

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0

Encode-OR(0) → 0

Encode-OR(0) → 0

Sum up encoding values

0+0+0 (mod 5) = 0

Decode-OR(0) → 0

0 | 0 | 0 = 0

Encode-OR(0) → 0

Encode-OR(1) → 4

Encode-OR(1) → 3

Sum up encoding values

0+4+3 (mod 5) = 2

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0
Encode-OR(0) → 0
Encode-OR(0) → 0

Sum up encoding values
0+0+0 (mod 5) = 0

Decode-OR(0) → 0
0 | 0 | 0 = 0

Encode-OR(0) → 0
Encode-OR(1) → 4
Encode-OR(1) → 3

Sum up encoding values
0+4+3 (mod 5) = 2

Decode-OR(2) → 1

Addax's AFE for OR over bits

Toy example for p = 5:

Encode-OR(0) → 0
Encode-OR(0) → 0
Encode-OR(0) → 0

Sum up encoding values
0+0+0 (mod 5) = 0

Decode-OR(0) → 0
0 | 0 | 0 = 0

Encode-OR(0) → 0
Encode-OR(1) → 4
Encode-OR(1) → 3

Sum up encoding values
0+4+3 (mod 5) = 2

Decode-OR(2) → 1
0 | 1 | 1 = 1

Addax’s private auction using AFE

	Bids
hp Bidder	3
DELL Bidder	2
BMW Bidder	1

Addax’s private auction using AFE

	Bids				
Bidder	3 →	1	1	1	0
Bidder	2 →	1	1	0	0
Bidder	1 →	1	0	0	0

Addax's private auction using AFE

Bids

Encode-OR in Z_5

Bidder $3 \rightarrow \boxed{1}\boxed{1}\boxed{1}\boxed{0} \rightarrow \boxed{4}\boxed{3}\boxed{2}\boxed{0}$

Bidder $2 \rightarrow \boxed{1}\boxed{1}\boxed{0}\boxed{0} \rightarrow \boxed{4}\boxed{3}\boxed{0}\boxed{0}$

Bidder $1 \rightarrow \boxed{1}\boxed{0}\boxed{0}\boxed{0} \rightarrow \boxed{3}\boxed{0}\boxed{0}\boxed{0}$

Addax's private auction using AFE

Bids

Encode-OR in Z_5

Bidder

$$3 \rightarrow \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 0 \\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline 4 & 3 & 2 & 0 \\ \hline\end{array}$$

Bidder

$$2 \rightarrow \begin{array}{|c|c|c|c|}\hline 1 & 1 & 0 & 0 \\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline 4 & 3 & 0 & 0 \\ \hline\end{array}$$

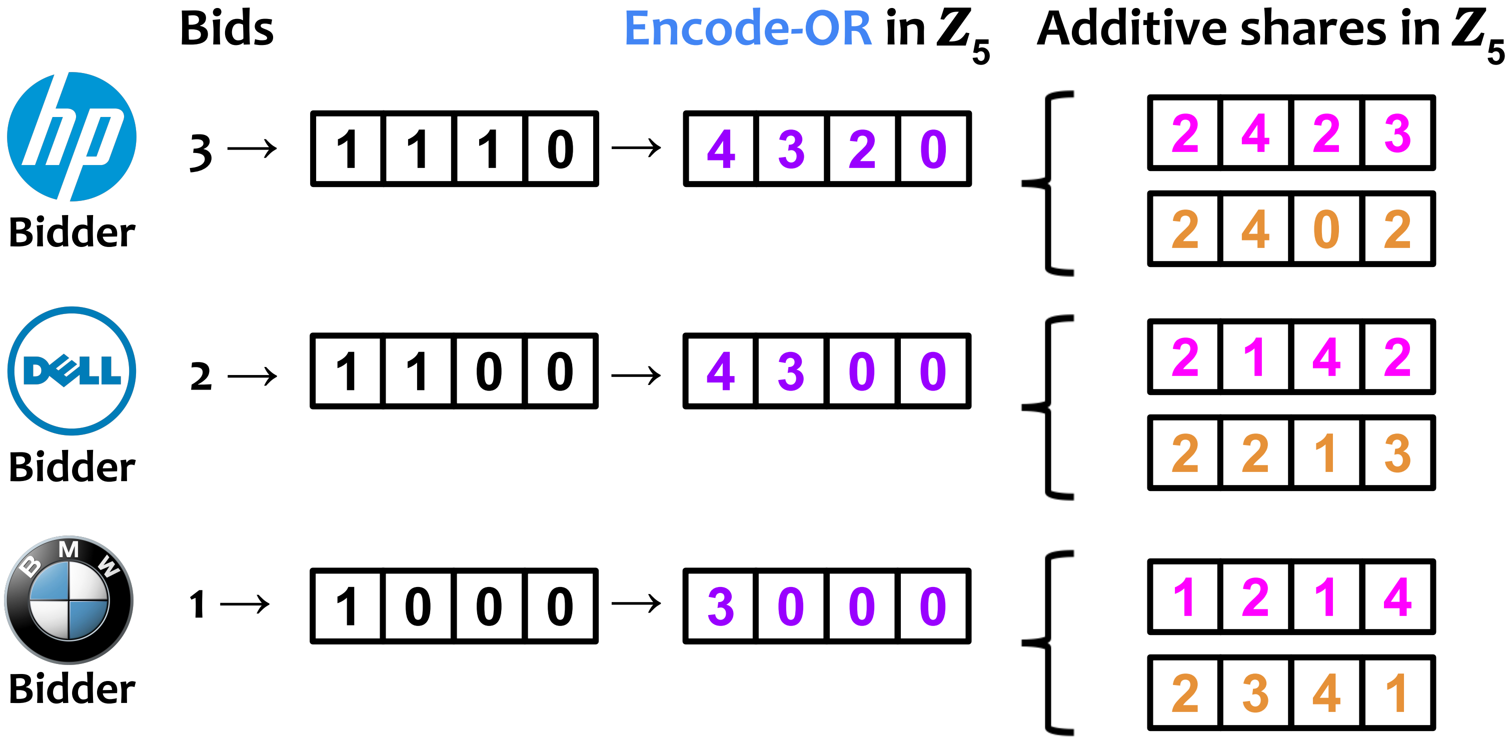
Bidder

$$1 \rightarrow \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 0 \\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline 3 & 0 & 0 & 0 \\ \hline\end{array}$$

Addax’s private auction using AFE

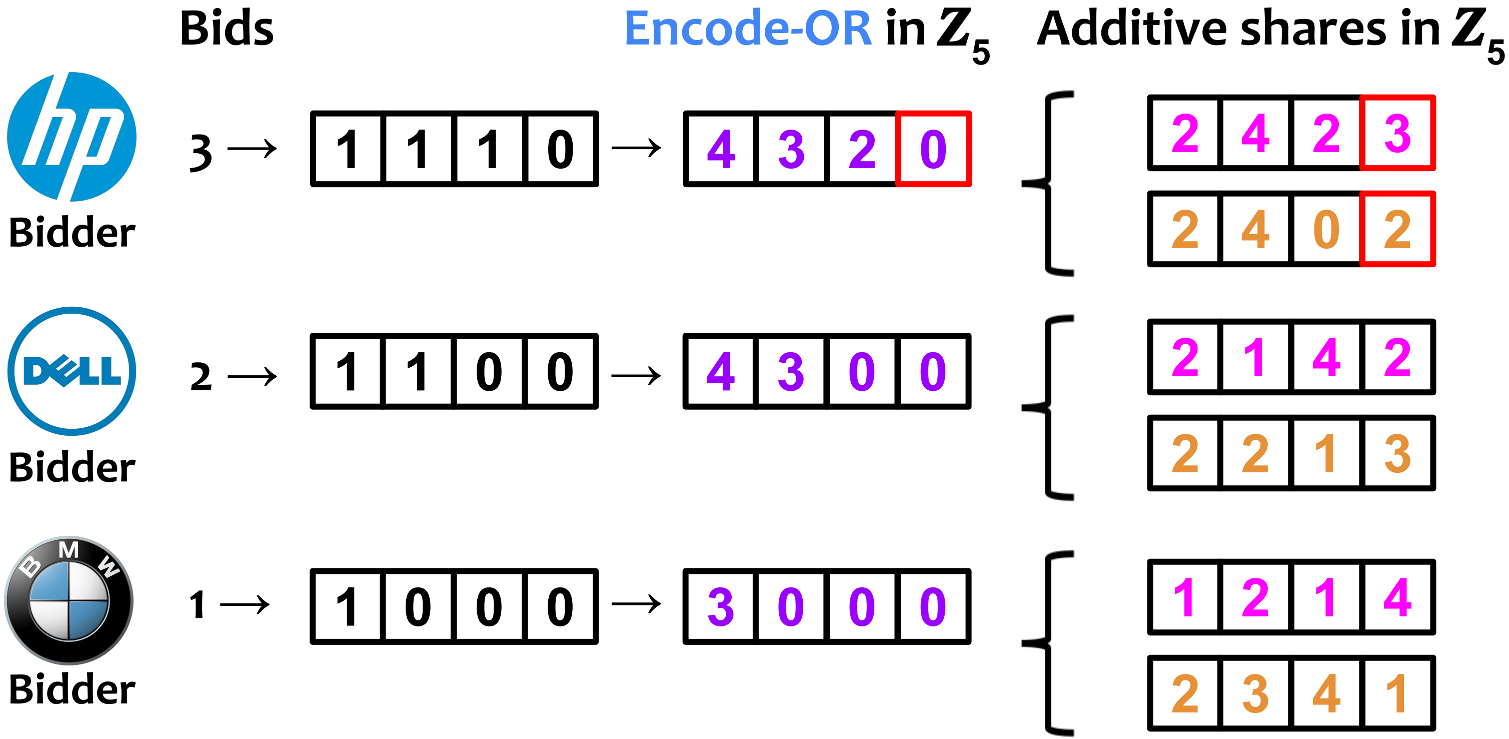
	Bids						Encode-OR in Z_5			
Bidder	3 →	1	1	1	0	→	4	3	2	0
Bidder	2 →	1	1	0	0	→	4	3	0	0
Bidder	1 →	1	0	0	0	→	3	0	0	0

Addax's private auction using AFE



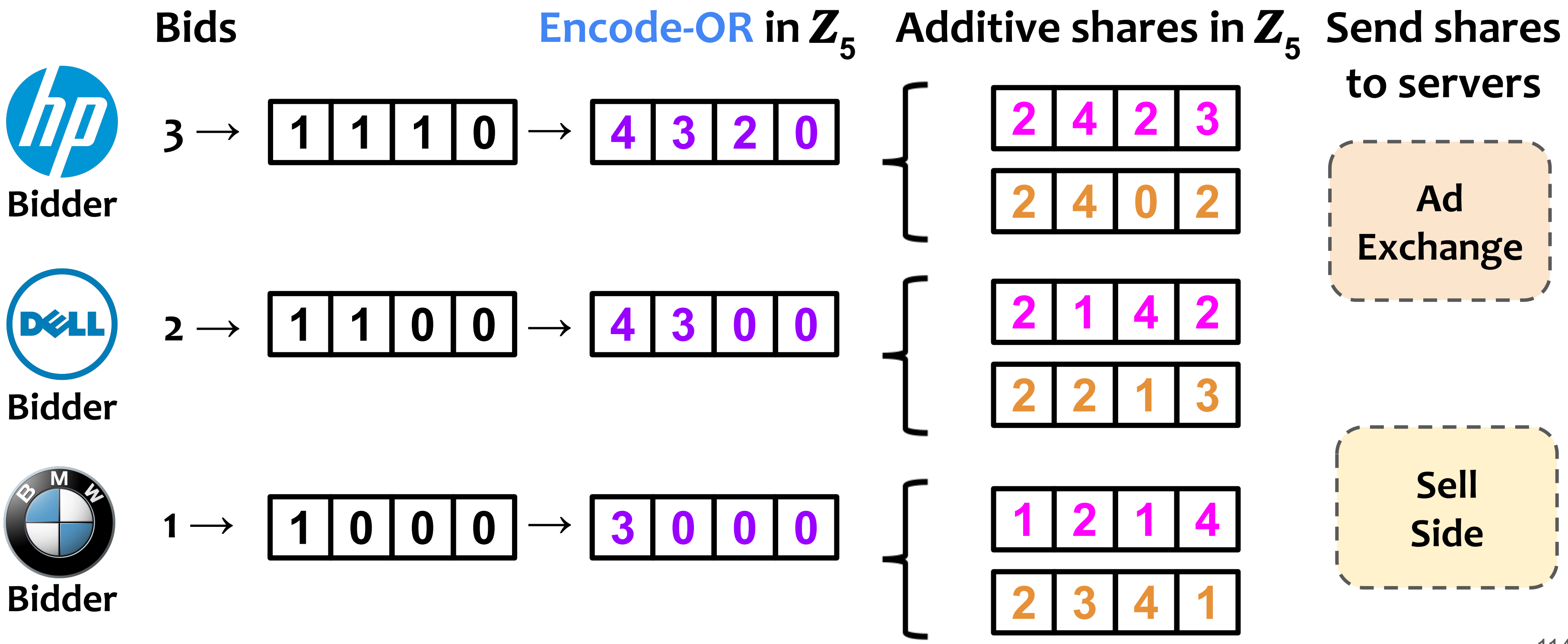
	Bids		Encode-OR in Z_5	Additive shares in Z_5
Bidder	3 →	1 1 1 0	→ 4 3 2 0	2 4 2 3 2 4 0 2
Bidder	2 →	1 1 0 0	→ 4 3 0 0	2 1 4 2 2 2 1 3
Bidder	1 →	1 0 0 0	→ 3 0 0 0	1 2 1 4 2 3 4 1

Addax's private auction using AFE



	Bids		Encode-OR in Z_5	Additive shares in Z_5
Bidder	3 →	1 1 1 0 →	4 3 2 0	2 4 2 3 2 4 0 2
Bidder	2 →	1 1 0 0 →	4 3 0 0	2 1 4 2 2 2 1 3
Bidder	1 →	1 0 0 0 →	3 0 0 0	1 2 1 4 2 3 4 1

Addax’s private auction using AFE



Bids

Encode-OR in Z_5

Additive shares in Z_5

Send shares to servers

Bidder

$3 \rightarrow$ 1 1 1 0 $\rightarrow$ 4 3 2 0

2 4 2 3

2 4 0 2

Bidder

$2 \rightarrow$ 1 1 0 0 $\rightarrow$ 4 3 0 0

2 1 4 2

2 2 1 3

Bidder

$1 \rightarrow$ 1 0 0 0 $\rightarrow$ 3 0 0 0

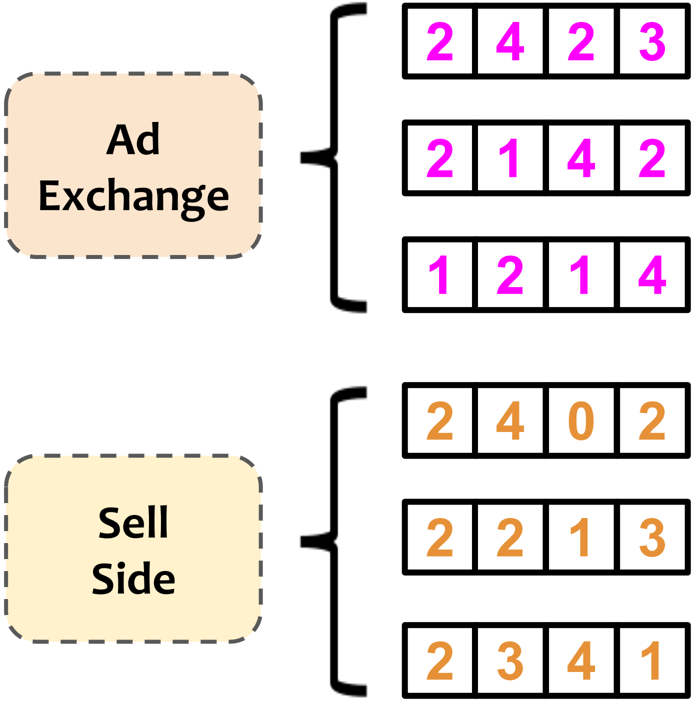
1 2 1 4

2 3 4 1

Ad Exchange

Sell Side

Addax’s private auction using AFE



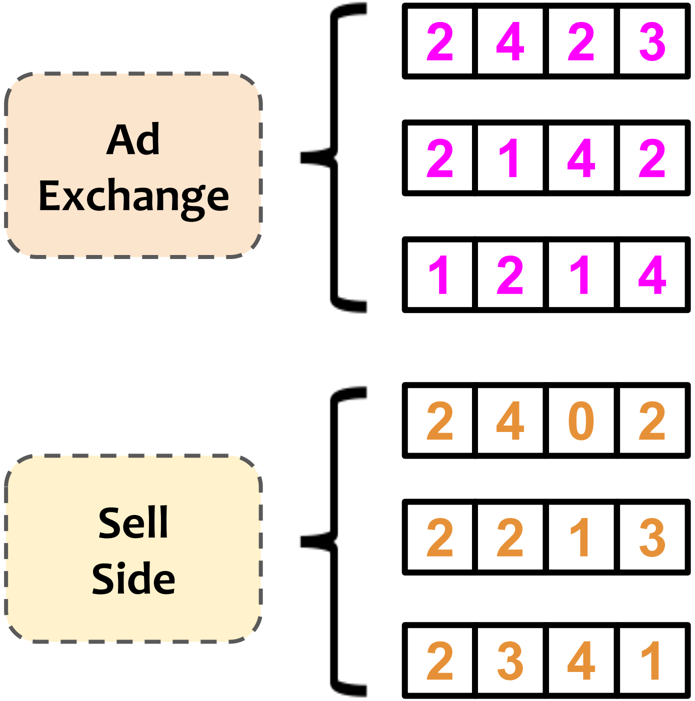
Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

Addax’s private auction using AFE



Ad Exchange

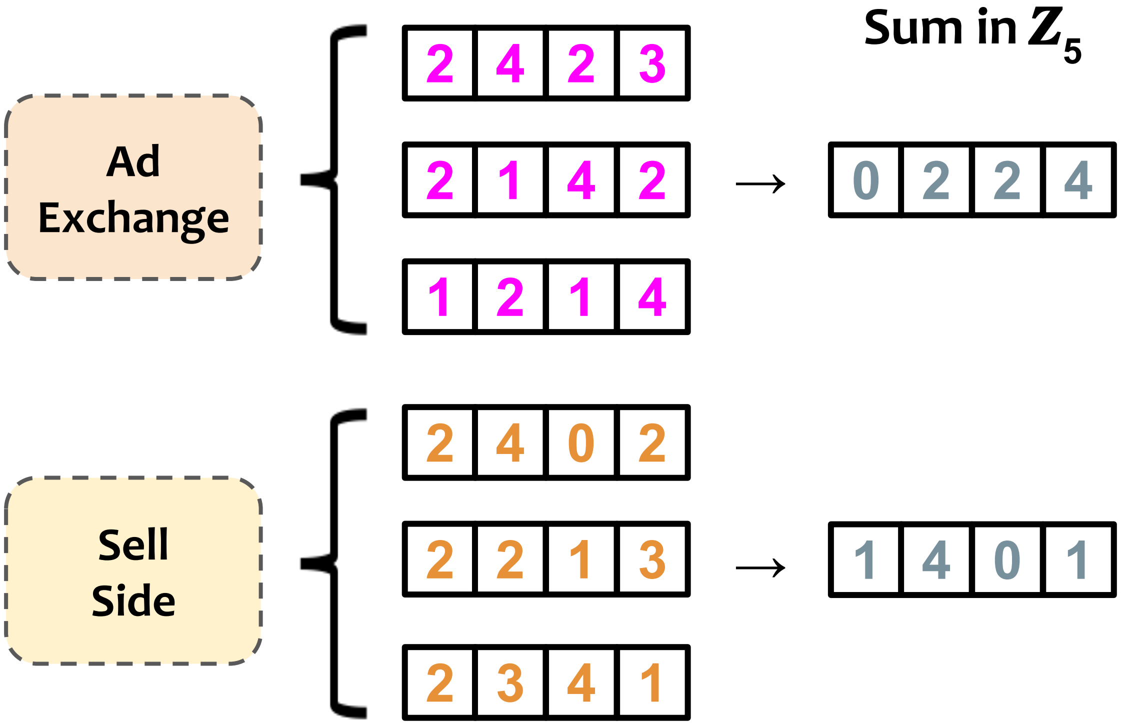
2	4	2	3
2	1	4	2
1	2	1	4

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

Compute the maximum bid

Addax's private auction using AFE



Sum in Z_5

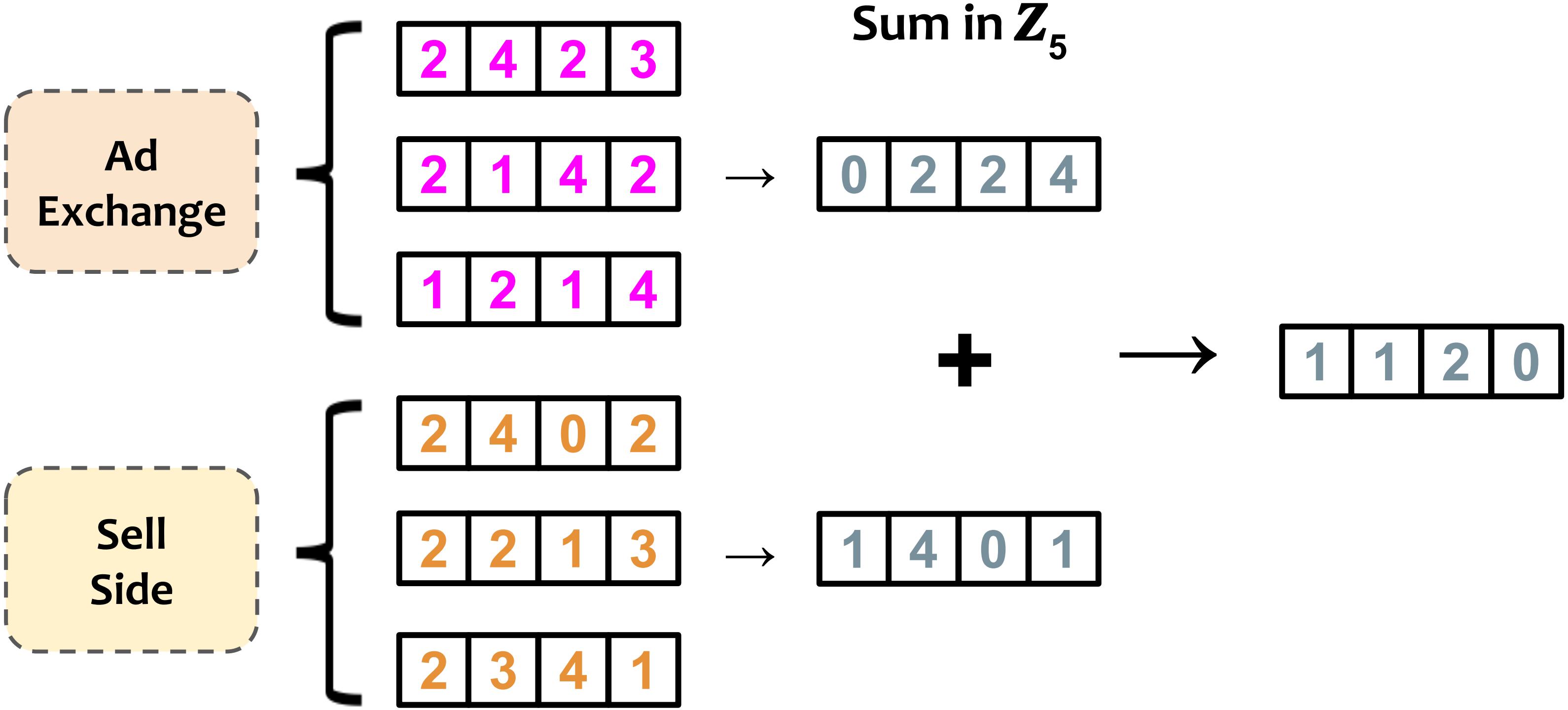
Ad Exchange

2	4	2	3						
2	1	4	2	→	0	2	2	4
1	2	1	4						

Sell Side

2	4	0	2						
2	2	1	3	→	1	4	0	1
2	3	4	1						

Addax’s private auction using AFE



Sum in Z_5

Ad Exchange

2 4 2 3

2 1 4 2 → 0 2 2 4

1 2 1 4

+ ⟶ 1 1 2 0

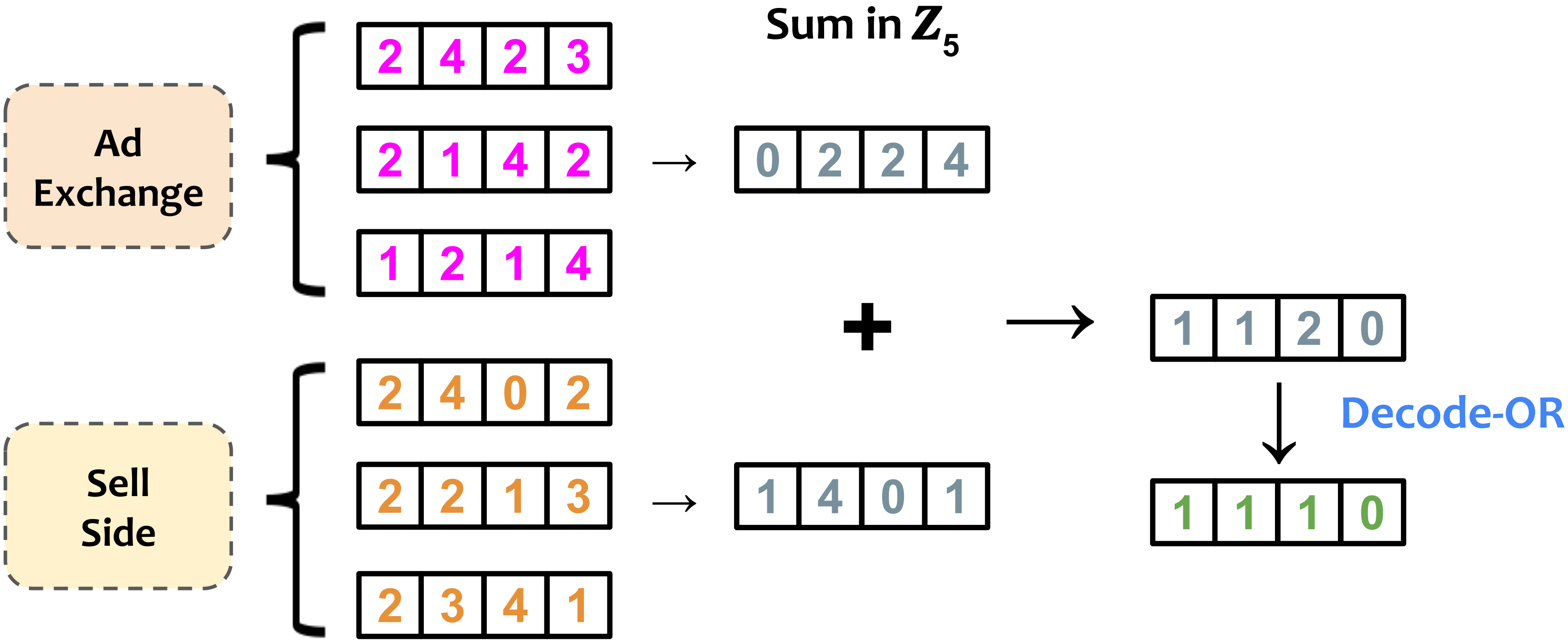
Sell Side

2 4 0 2

2 2 1 3 → 1 4 0 1

2 3 4 1

Addax’s private auction using AFE



Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

Sum in $\boldsymbol{Z}_5$

→ 0 2 2 4

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

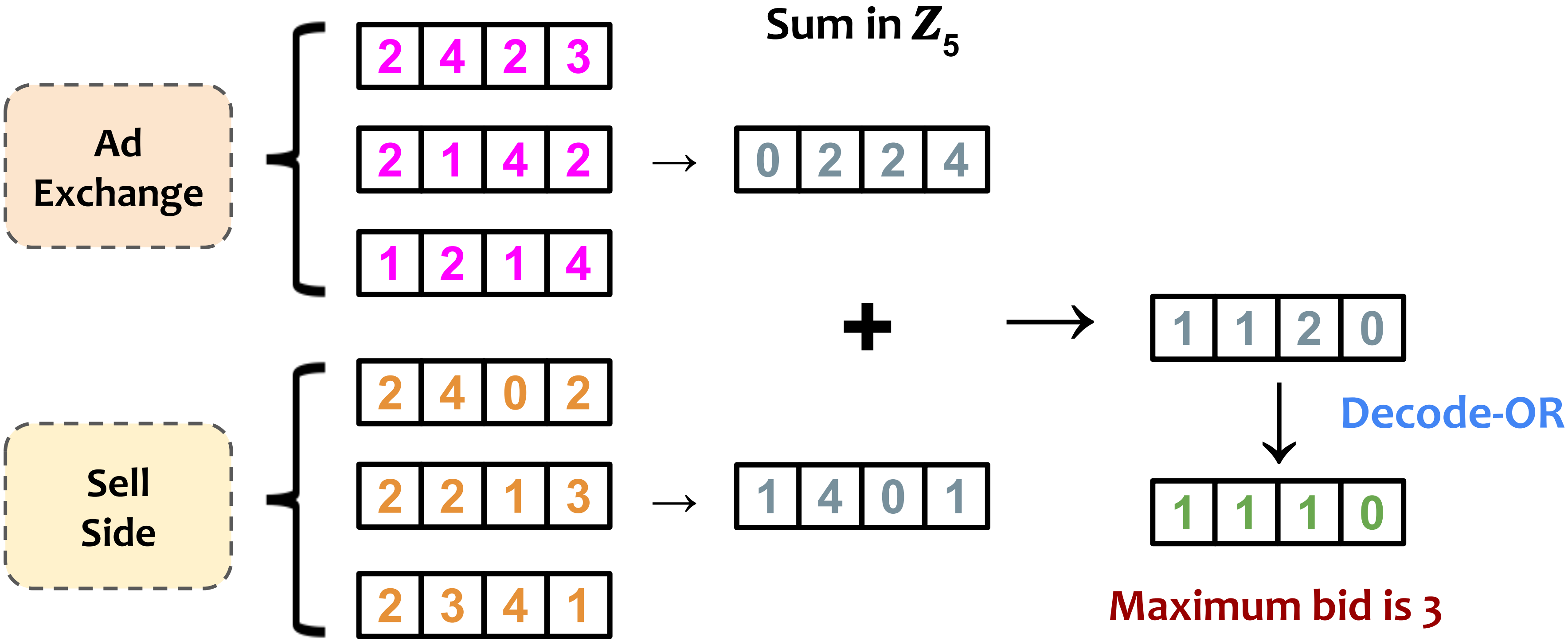
→ 1 4 0 1

+ → 1 1 2 0

Decode-OR

1 1 1 0

Addax’s private auction using AFE



Sum in Z_5

Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

→

0	2	2	4

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

→

1	4	0	1

+ →

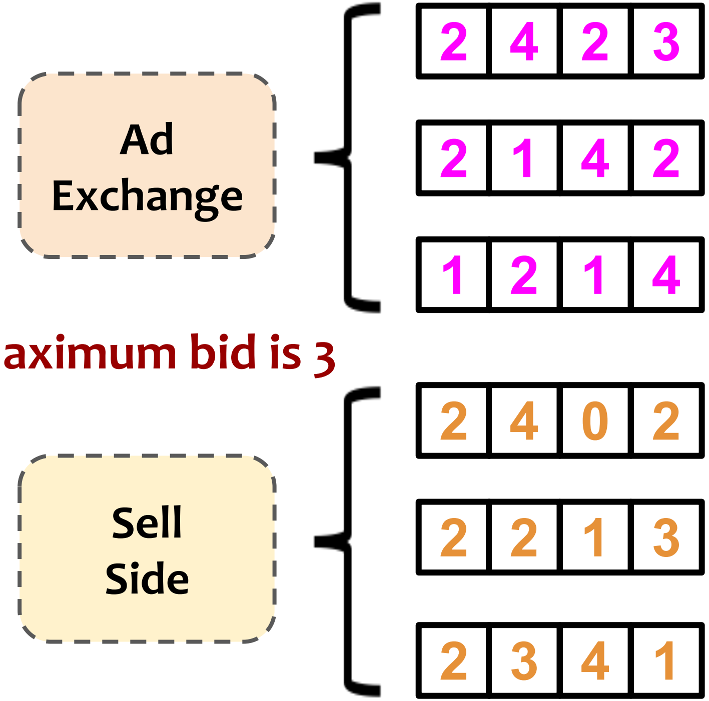
1	1	2	0

Decode-OR

1	1	1	0

Maximum bid is 3

Addax's private auction using AFE



Ad Exchange

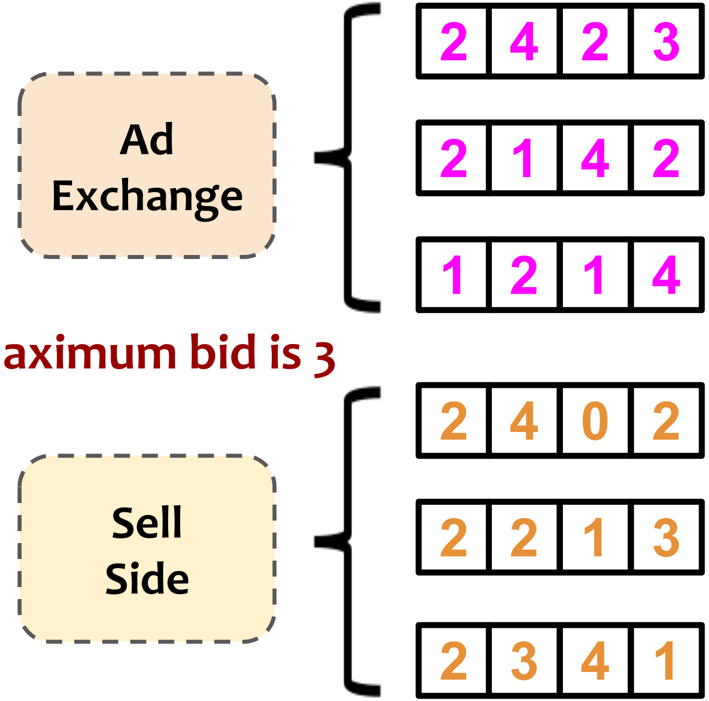
2	4	2	3
2	1	4	2
1	2	1	4

Maximum bid is 3

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

Addax's private auction using AFE



Ad Exchange

2 4 2 3

2 1 4 2

1 2 1 4

Maximum bid is 3

Sell Side

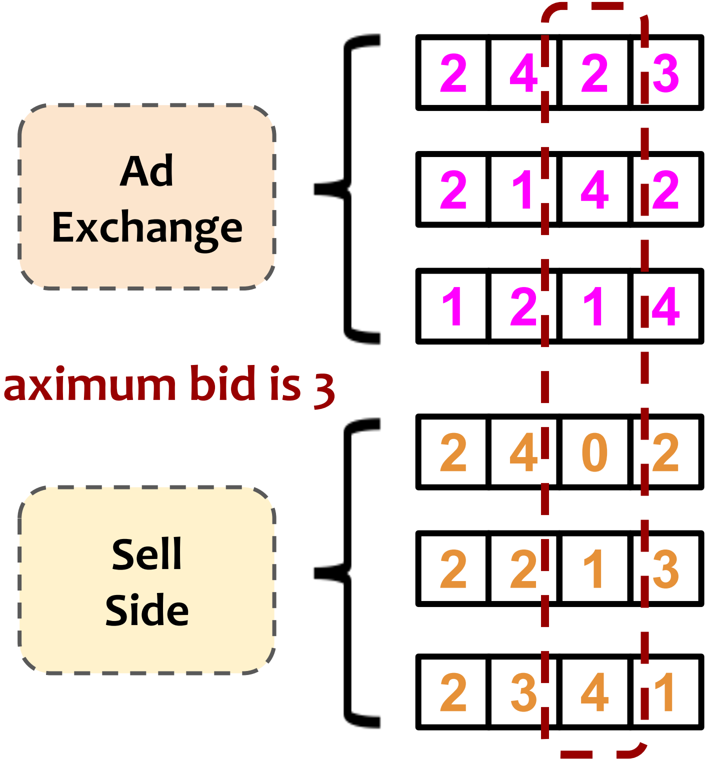
2 4 0 2

2 2 1 3

2 3 4 1

Find out the winner

Addax’s private auction using AFE



Ad Exchange

2 4 2 3

2 1 4 2

1 2 1 4

Maximum bid is 3

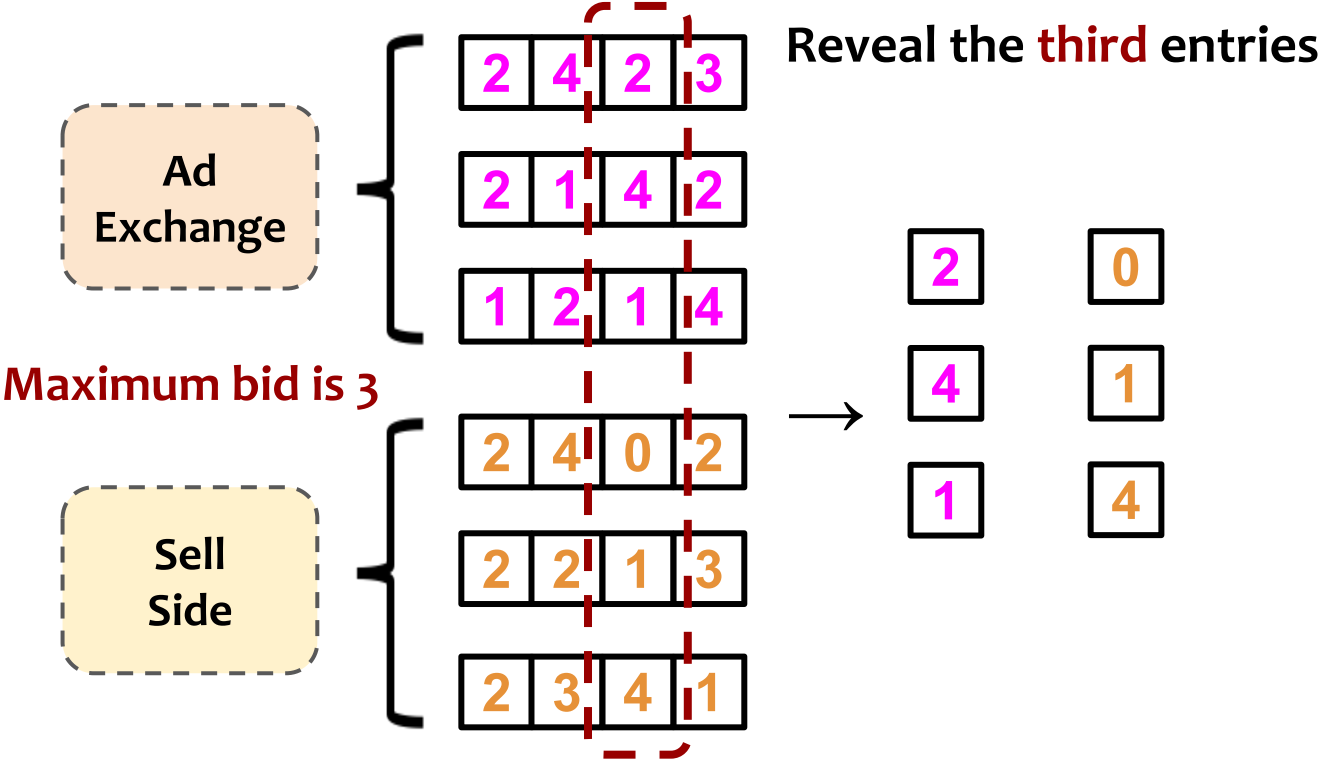
Sell Side

2 4 0 2

2 2 1 3

2 3 4 1

Addax’s private auction using AFE



Reveal the **third** entries

Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

Maximum bid is 3

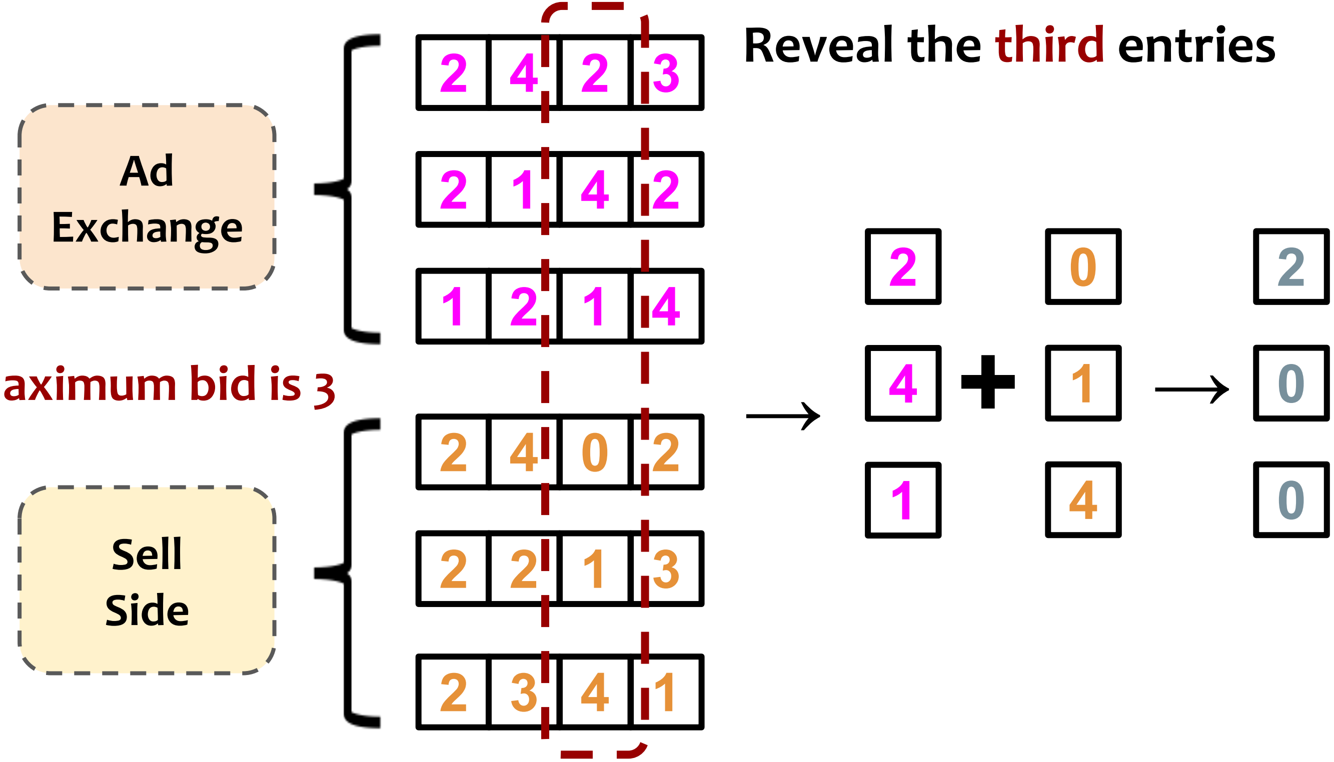
Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

→

2	0
4	1
1	4

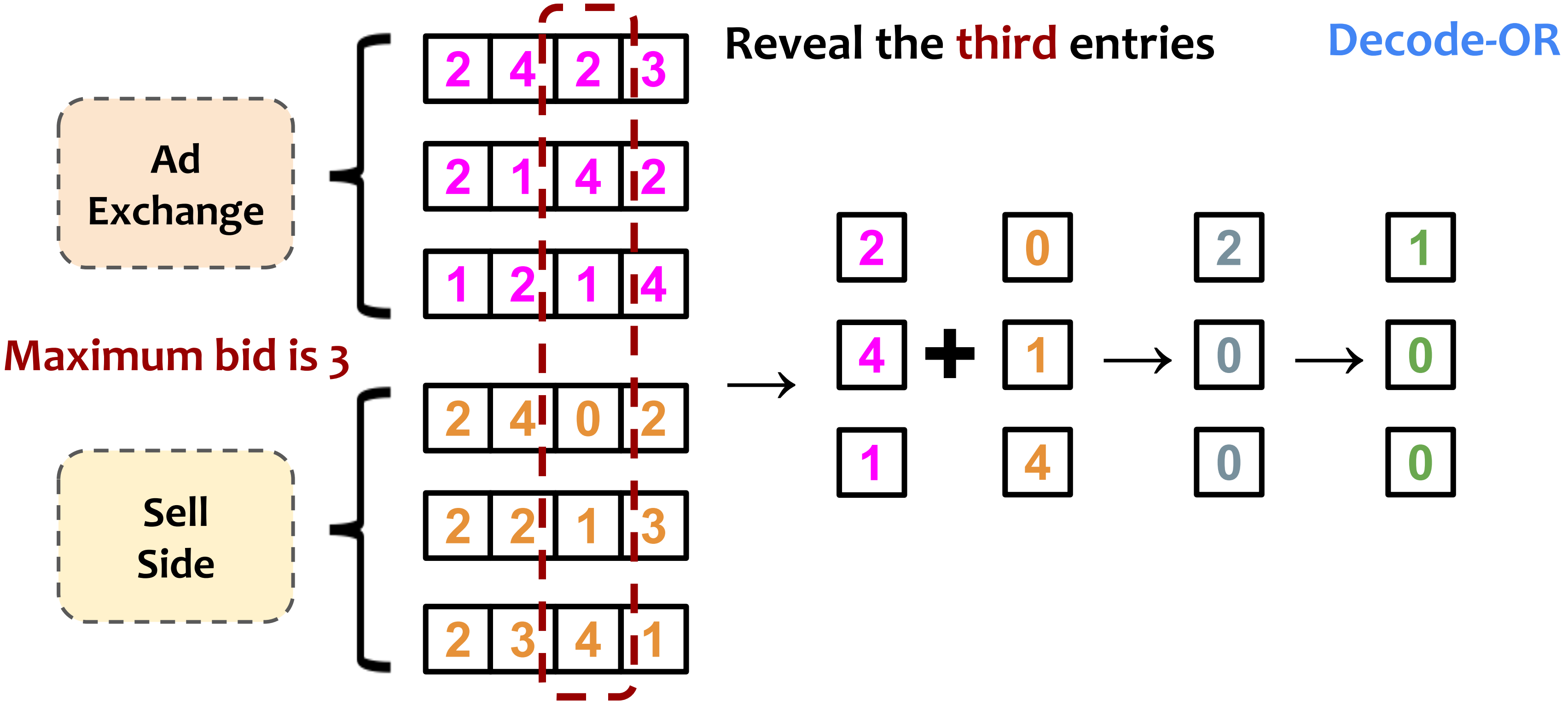
Addax's private auction using AFE



Reveal the third entries

Maximum bid is 3

Addax’s private auction using AFE



Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

Maximum bid is 3

Sell Side

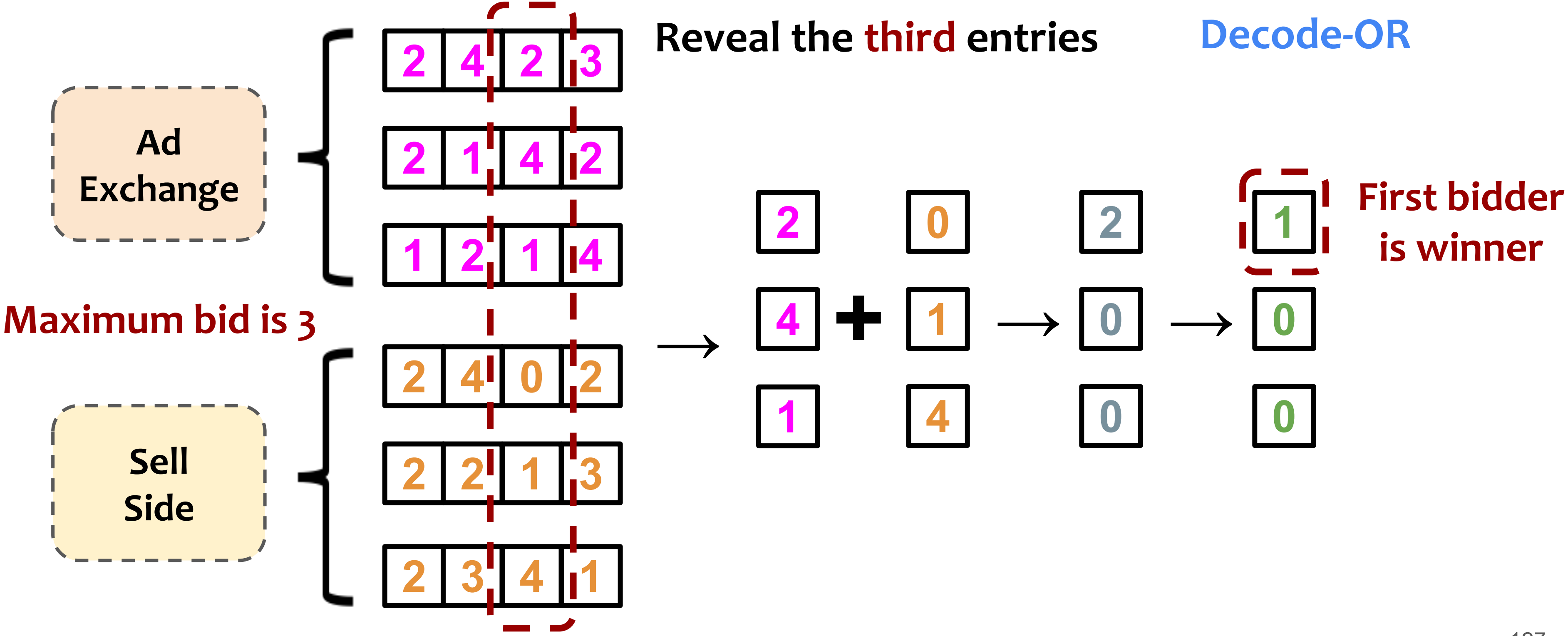
2	4	0	2
2	2	1	3
2	3	4	1

Reveal the third entries

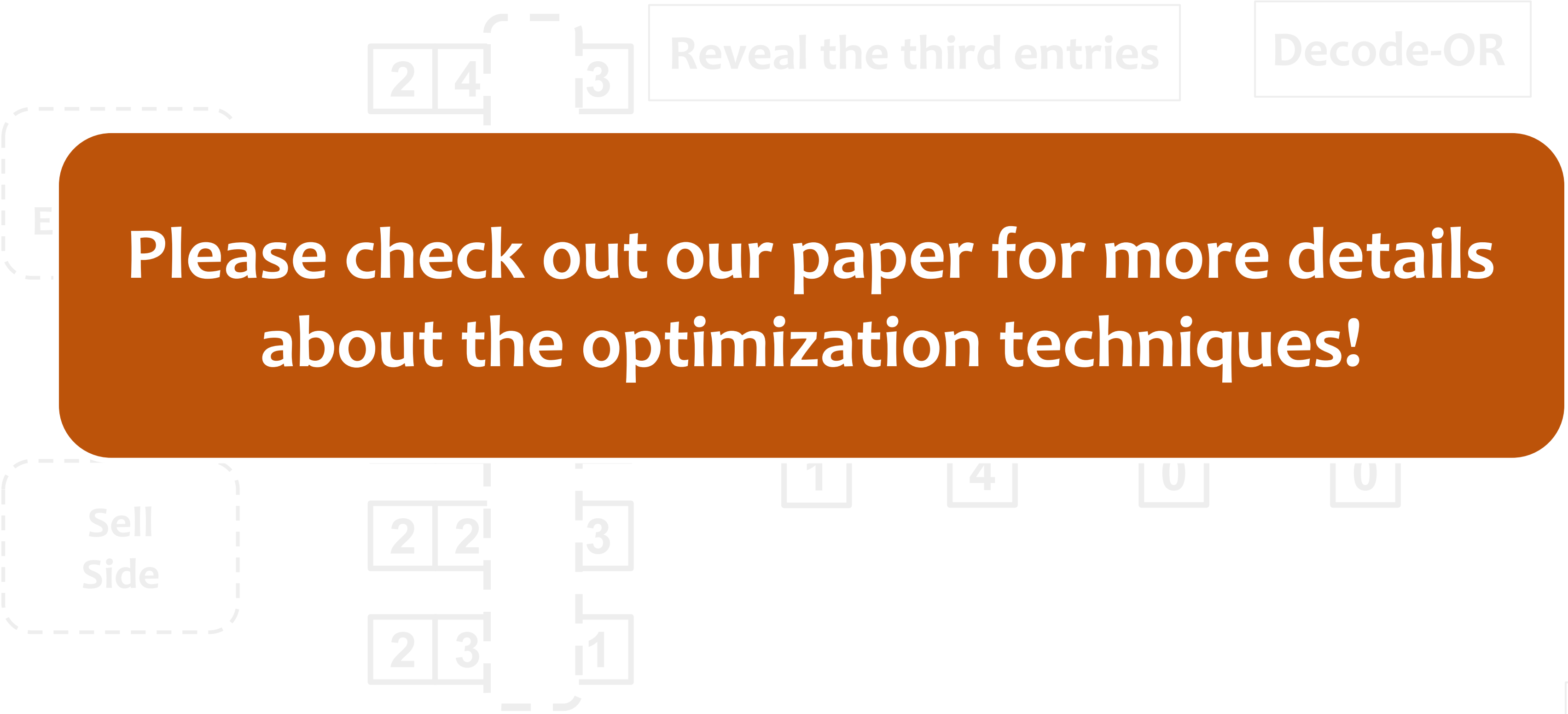
Decode-OR

2		0		2		1
4	+	1	→	0	→	0
1		4		0		0

Addax's private auction using AFE



Addax’s private auction using AFE

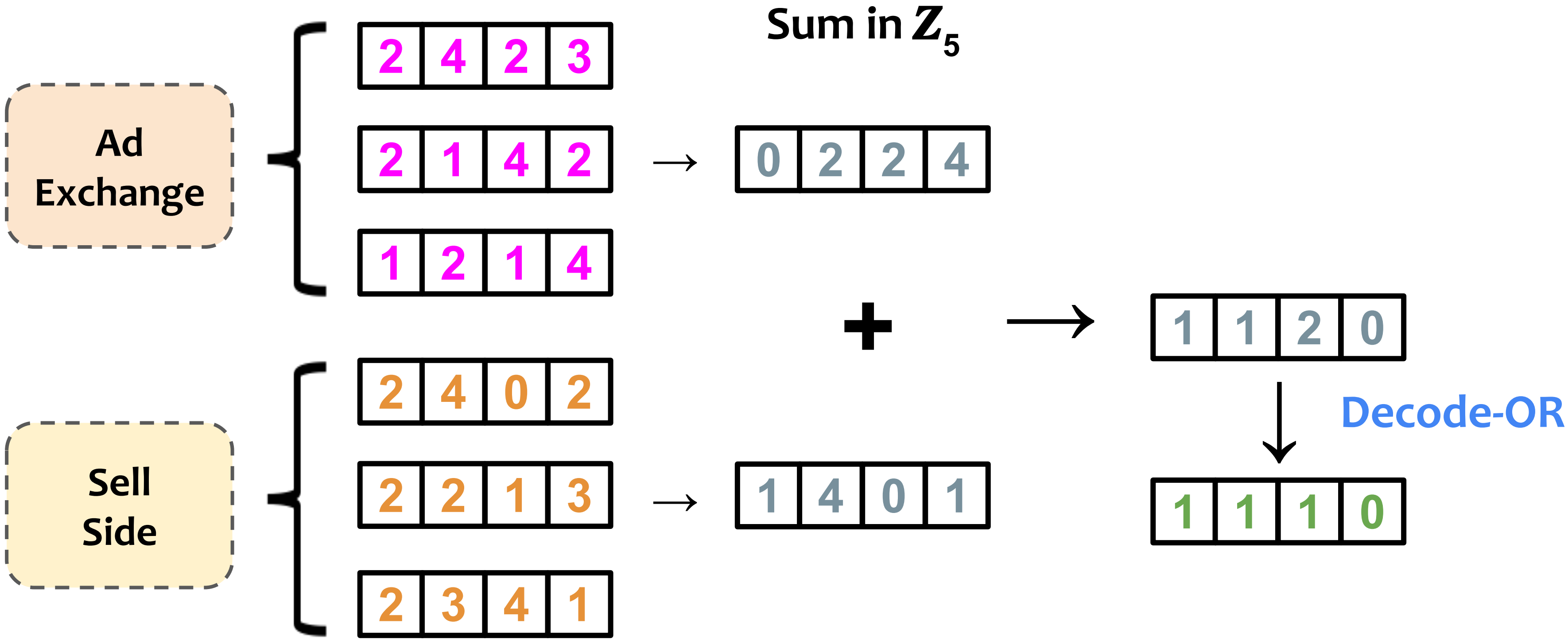


Please check out our paper for more details about the optimization techniques!

Rest of this talk

- Overview of Addax
- Private auction protocol
- **Make auction verifiable**
- Experimental evaluation

Private auction only uses additions



Ad Exchange

2 4 2 3

2 1 4 2

1 2 1 4

→ 0 2 2 4

Sell Side

2 4 0 2

2 2 1 3

2 3 4 1

→ 1 4 0 1

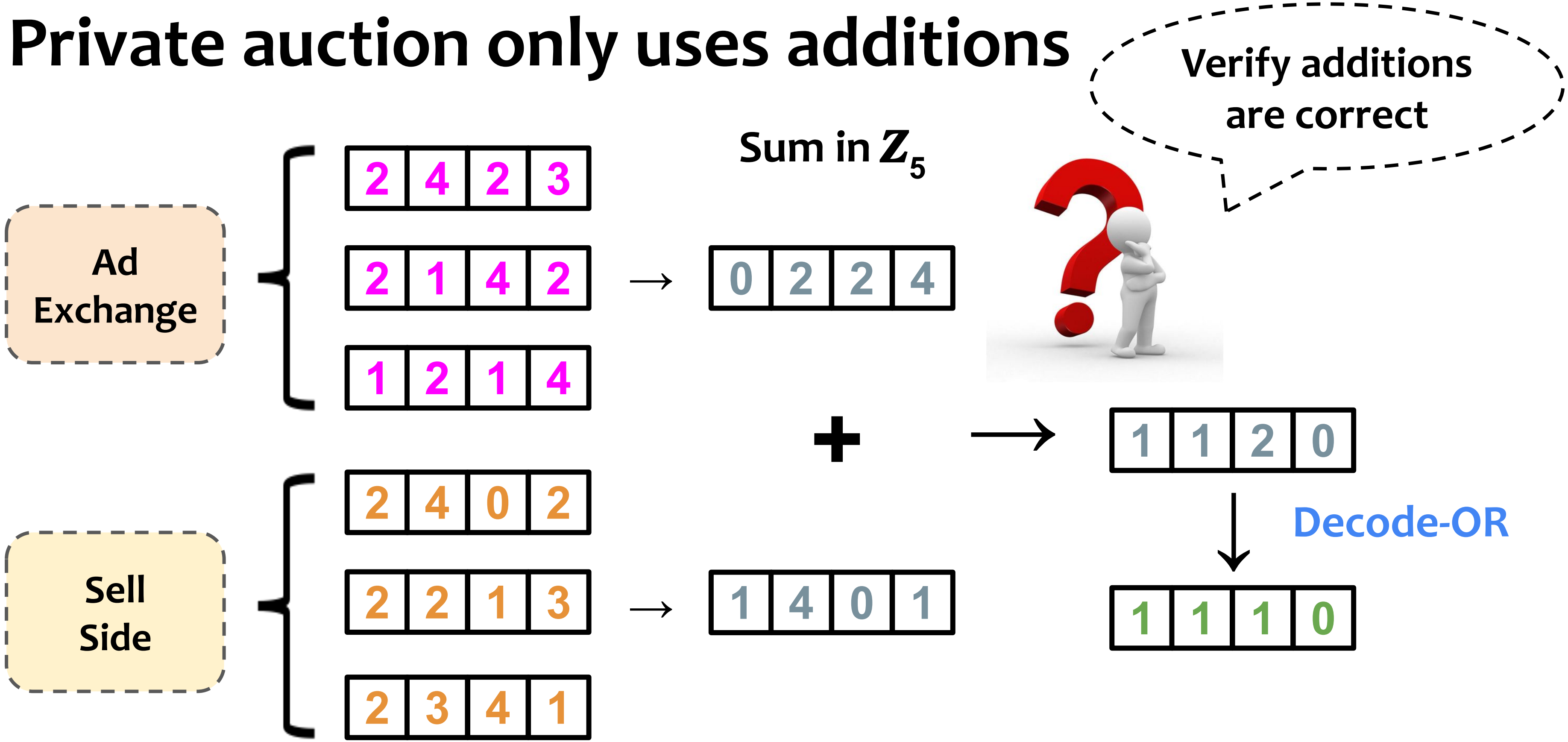
Sum in Z_5

+ → 1 1 2 0

Decode-OR

1 1 1 0

Private auction only uses additions



Verify additions are correct

Sum in Z_5

Ad Exchange

2	4	2	3
2	1	4	2
1	2	1	4

→

0	2	2	4

Sell Side

2	4	0	2
2	2	1	3
2	3	4	1

→

1	4	0	1

+ →

1	1	2	0

Decode-OR

1	1	1	0

Private auction only uses additions

Ad Exchange

2 4 2 3

2 1 4 2

1 2 1 4

Sell Side

2 4 0 2

2 2 1 3

2 3 4 1

Sum in Z_5

→ 0 2 2 4

→ 1 4 0 1

+ → 1 1 2 0

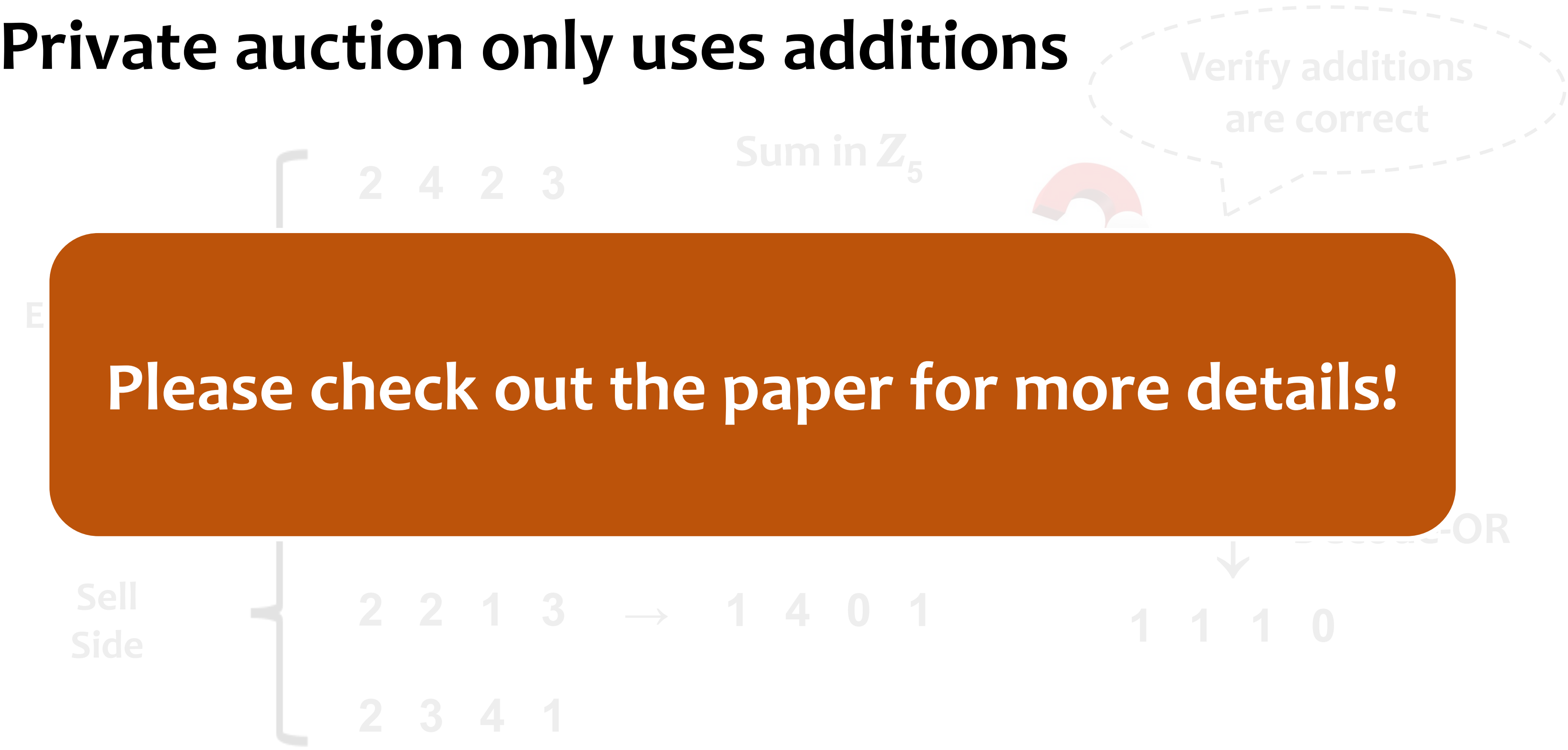
↓ Decode-OR

1 1 1 0

Verify additions are correct

Pedersen commitment!

Private auction only uses additions

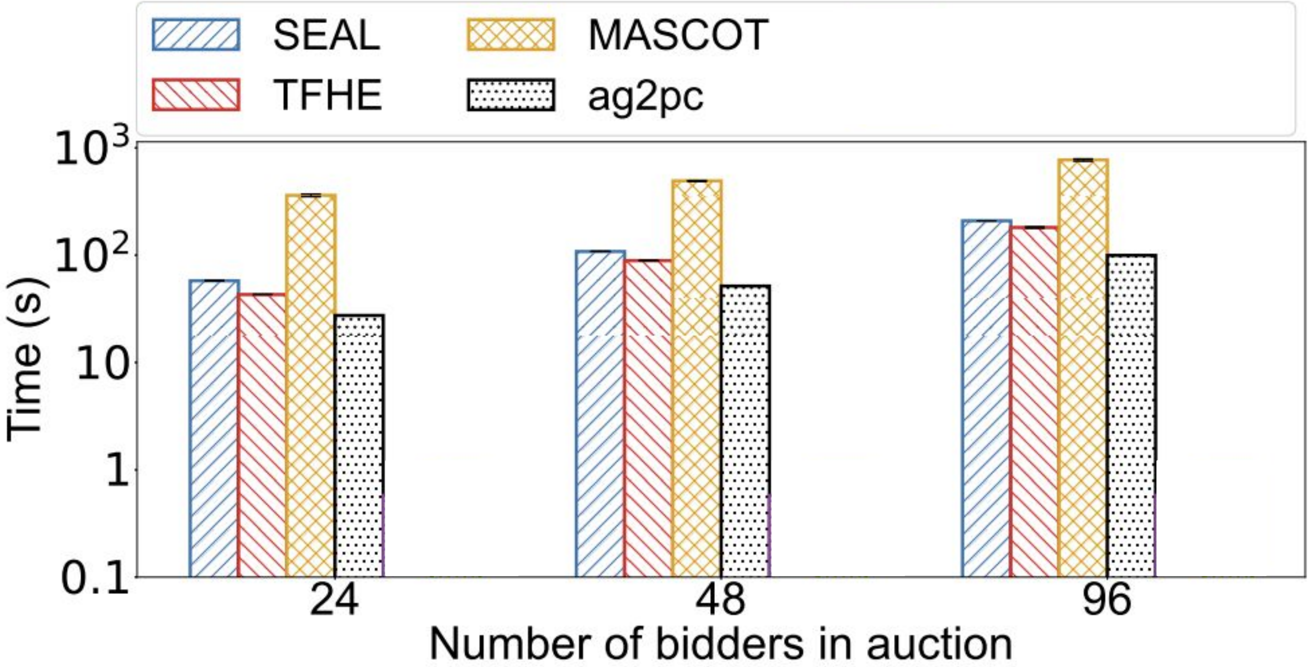


Please check out the paper for more details!

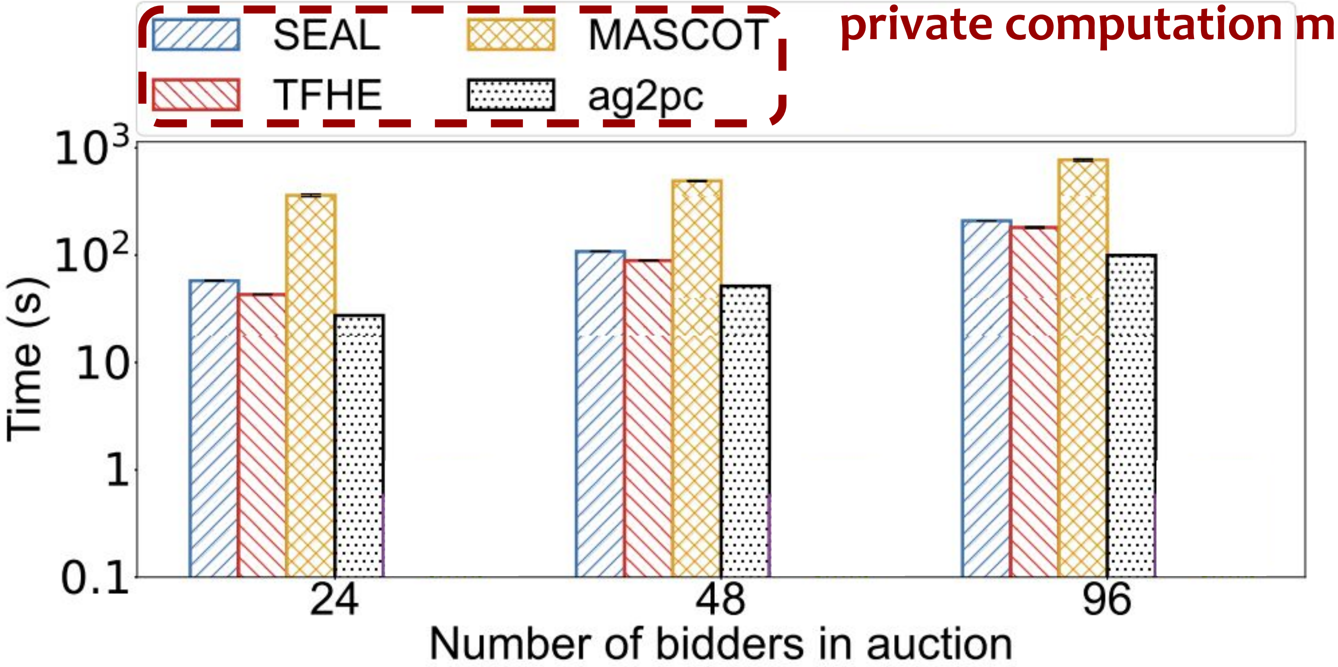
Rest of this talk

- Overview of Addax
- Private auction protocol
- Make auction verifiable
- **Experimental evaluation**

End-to-end latency over WAN

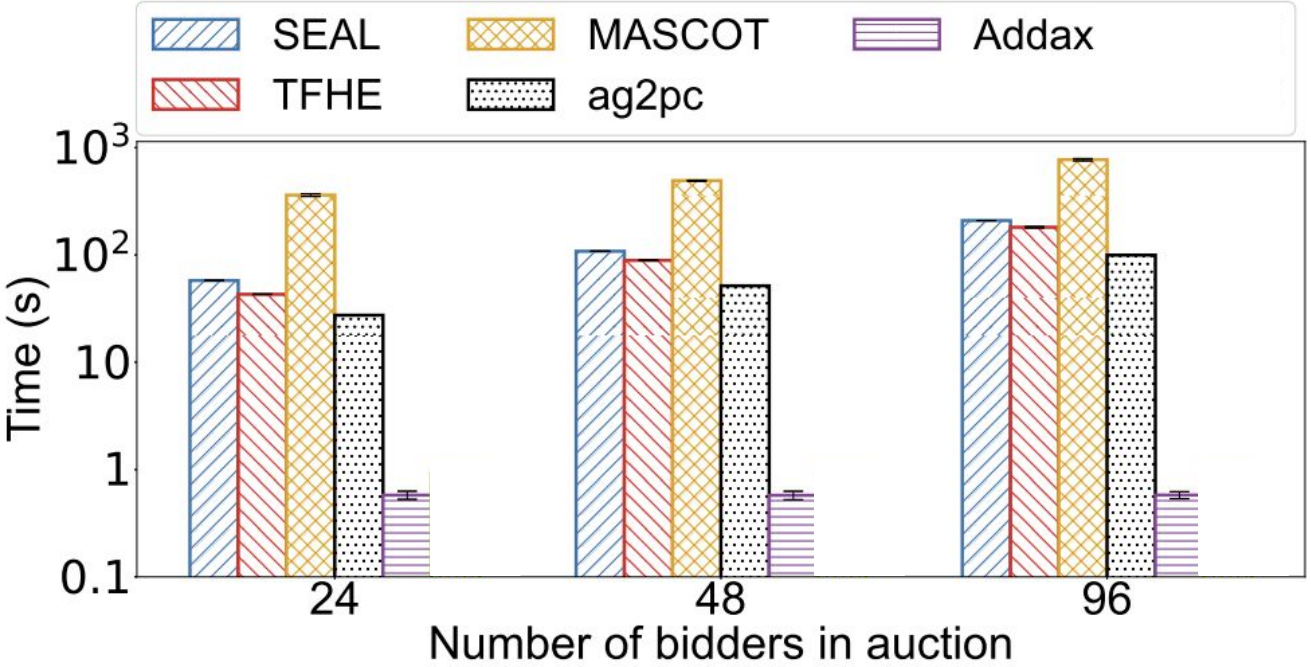


End-to-end latency over WAN



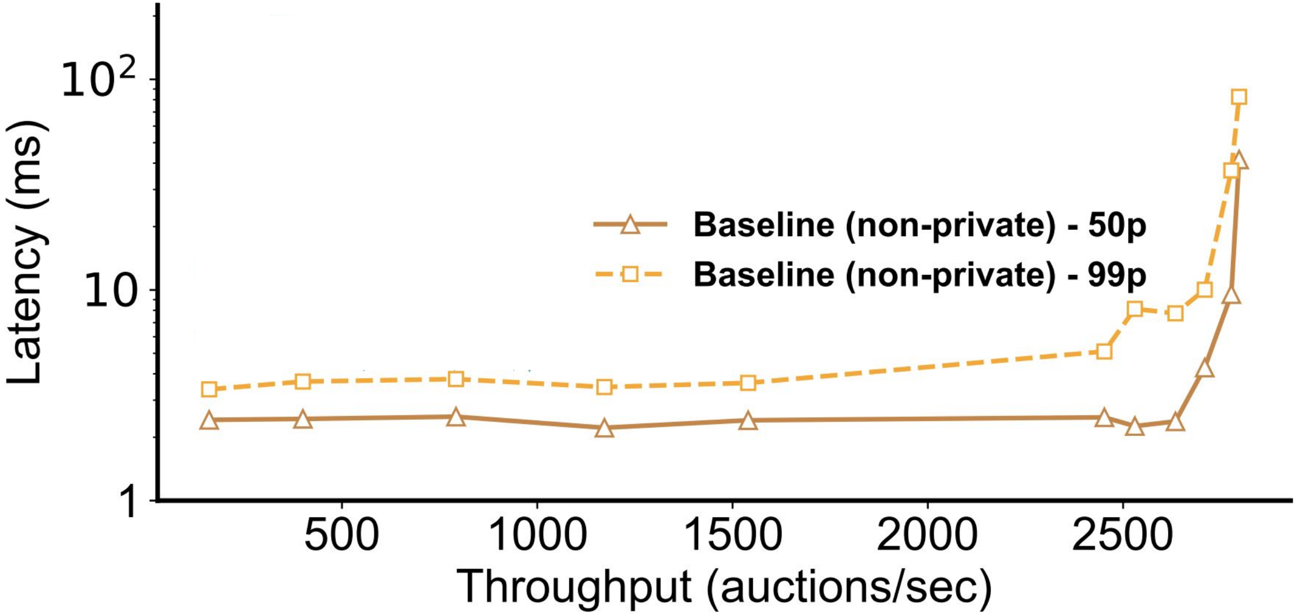
Baselines using other private computation methods

End-to-end latency over WAN

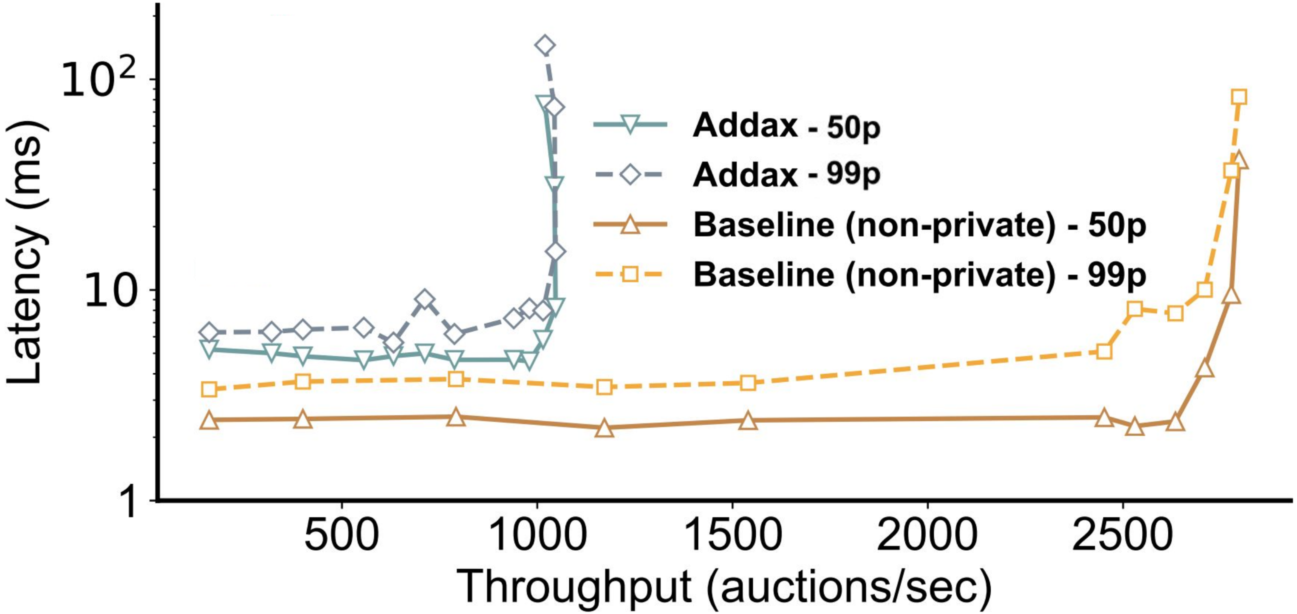


Auction can finish within 600 ms; enough to support real-time bidding

Throughput (no WAN)

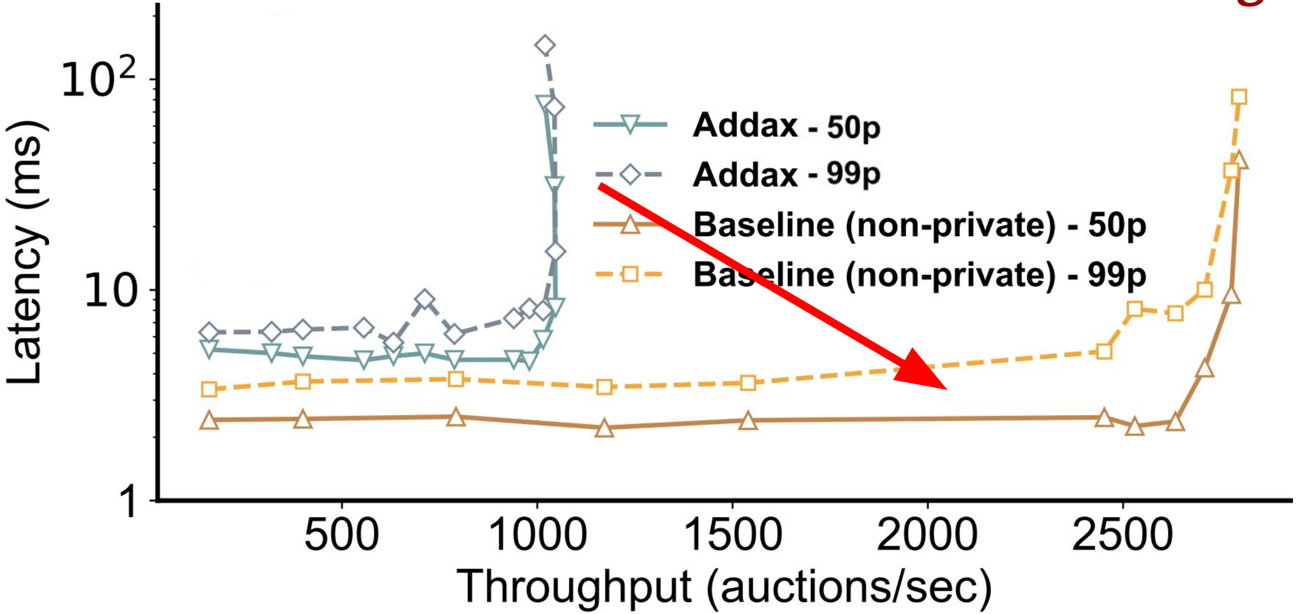


Throughput (no WAN)



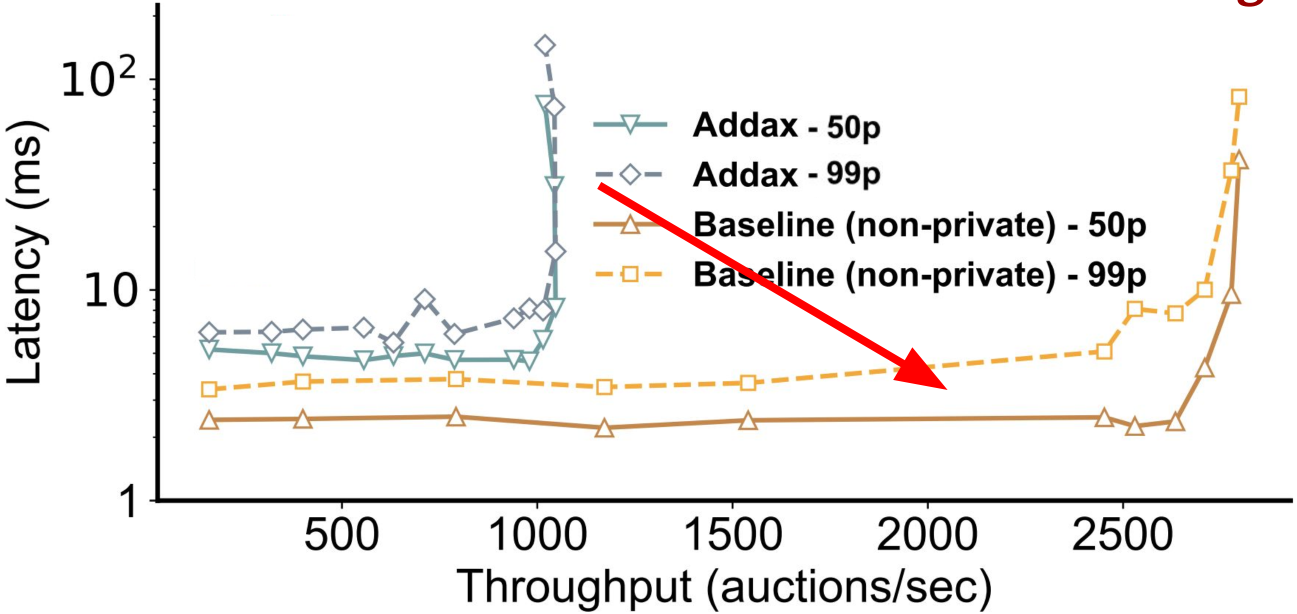
Throughput (no WAN)

The lower and to the right is better



Throughput (no WAN)

The lower and to the right is better



Addax can achieve roughly 40% throughput of a non-private baseline

Summary

- Addax: a fast, private, and accountable ad exchange infrastructure to help ad exchanges build up trust
  - Public verifiability for auction
  - Bids privacy for losing bidders
- Evaluation shows practicability for real-time bidding
  - Low end-to-end latency over WAN
  - High and reasonable throughput compared to non-private baseline

Thank you! Any questions?

- Addax: a fast, private, and accountable ad exchange infrastructure to help ad exchanges build up trust
  - Public verifiability for auction
  - Bids privacy for losing bidders
- Evaluation shows practicability for real-time bidding
  - Low end-to-end latency over WAN
  - High and reasonable throughput compared to non-private baseline