RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID: CRYP-F01

Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks

Mark Abspoel
CWI Amsterdam
Philips Research Eindhoven

Niek J. Bouman
Eindhoven University of Technology

Joint work with **Niels de Vreede** and **Berry Schoenmakers**

Motivation: A Scenario Involving a Trusted Third Party

Running Example: Second-Price Sealed-Bid Auction

- Bidders convey their bids in private to the Auctioneer
- Auctioneer sells item to the highest bidder for the price of the second-highest bid

Motivation: A Scenario Involving a Trusted Third Party

Running Example: Second-Price Sealed-Bid Auction

- Bidders convey their bids in private to the Auctioneer
- Auctioneer sells item to the highest bidder for the price of the second-highest bid

Problems

Privacy Auctioneer could leak bids to other bidders

Correctness Auctioneer could cheat, e.g. by increasing second-highest bid

Secure Multiparty Computation (MPC)

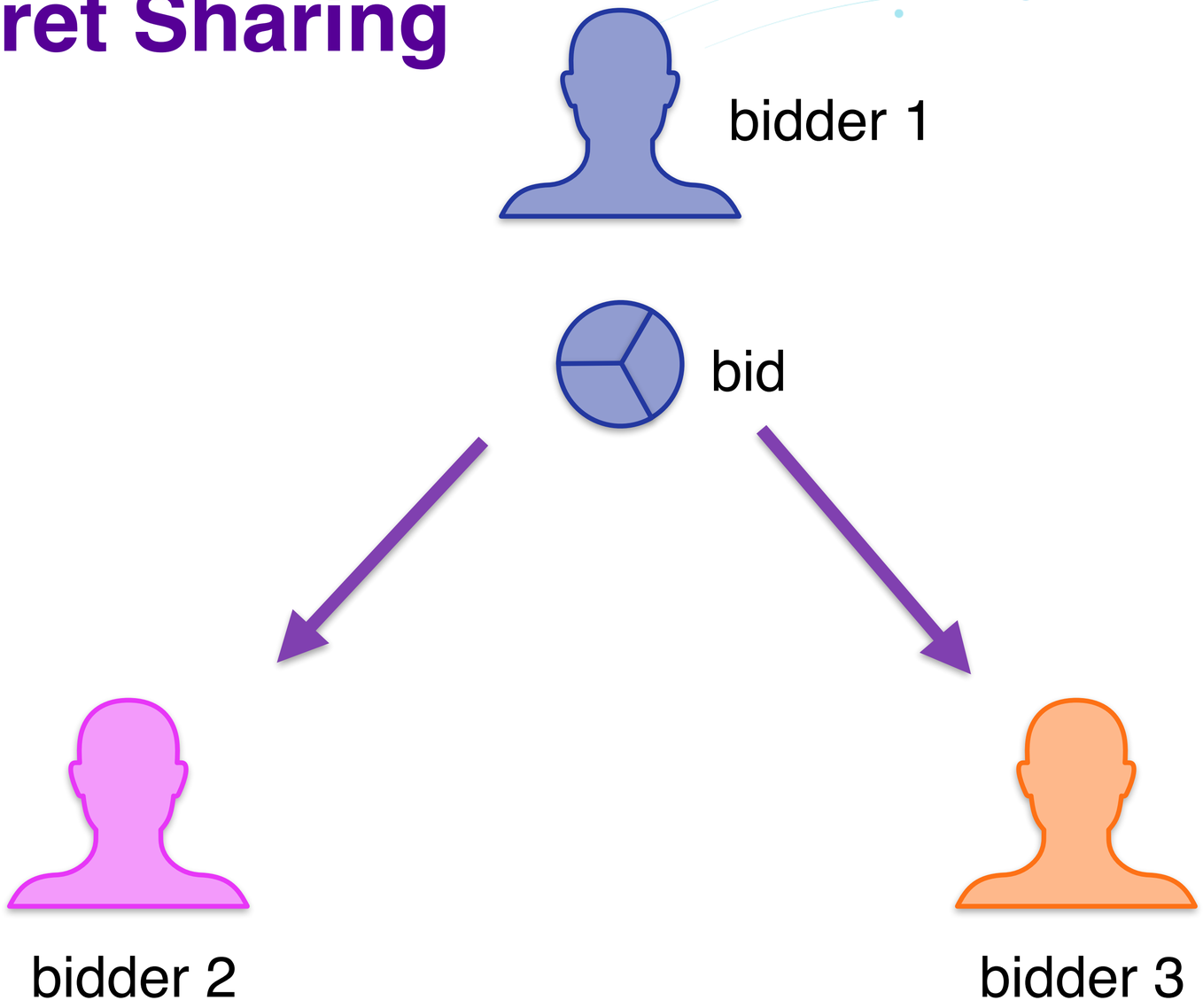
- Bidders jointly emulate the Auctioneer,
  by means of a distributed computation
- Bidders input their bids into their own computer
- Bids are distributed using secret sharing
- Computers execute a protocol (local computations + message exchange)
  to compute winning bidder and second price

Secure Multiparty Computation (MPC)

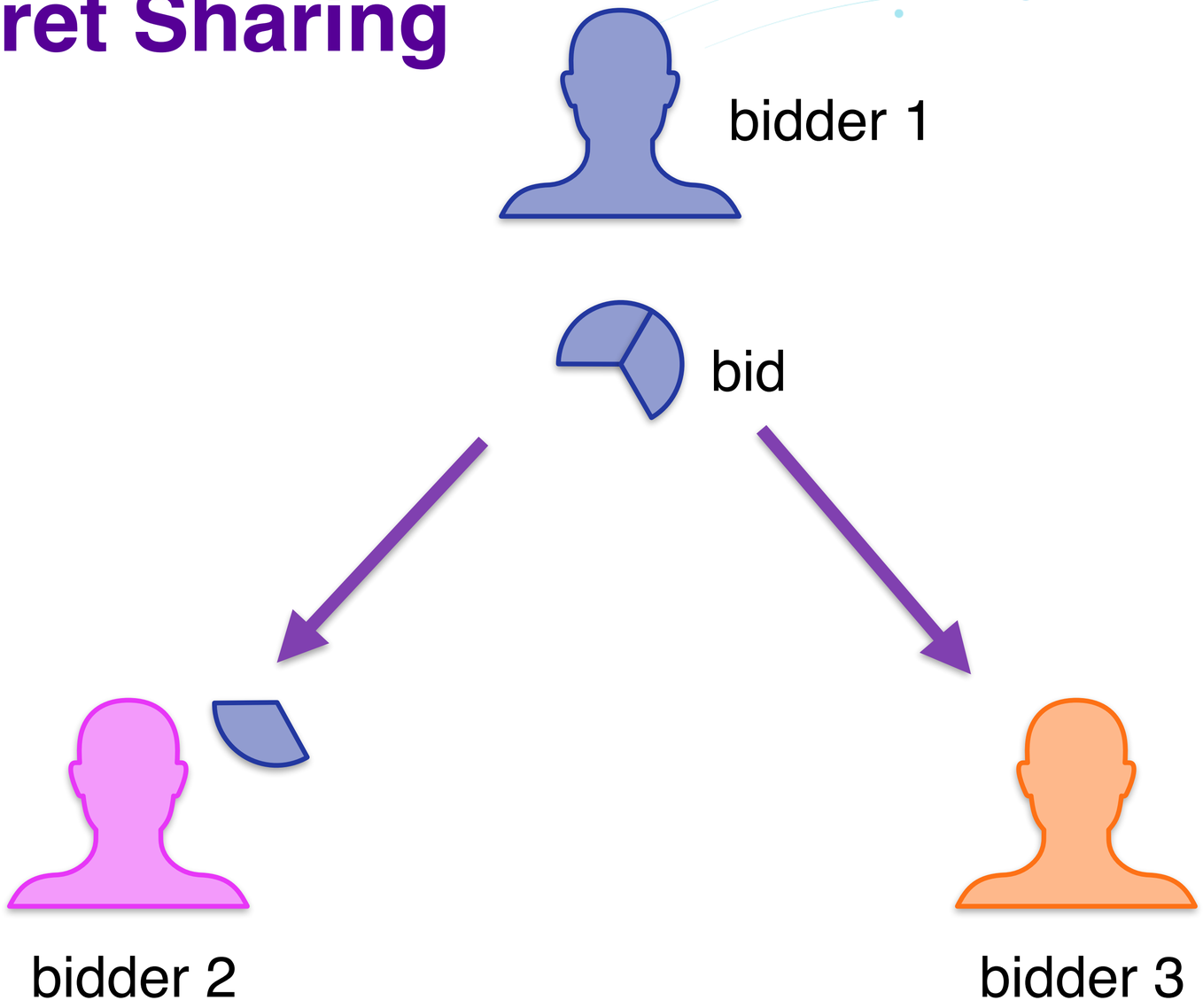
- Bidders jointly emulate the Auctioneer,
  by means of a distributed computation
- Bidders input their bids into their own computer
- Bids are distributed using secret sharing
- Computers execute a protocol (local computations + message exchange)
  to compute winning bidder and second price

…Achieve privacy and correctness through cryptography!

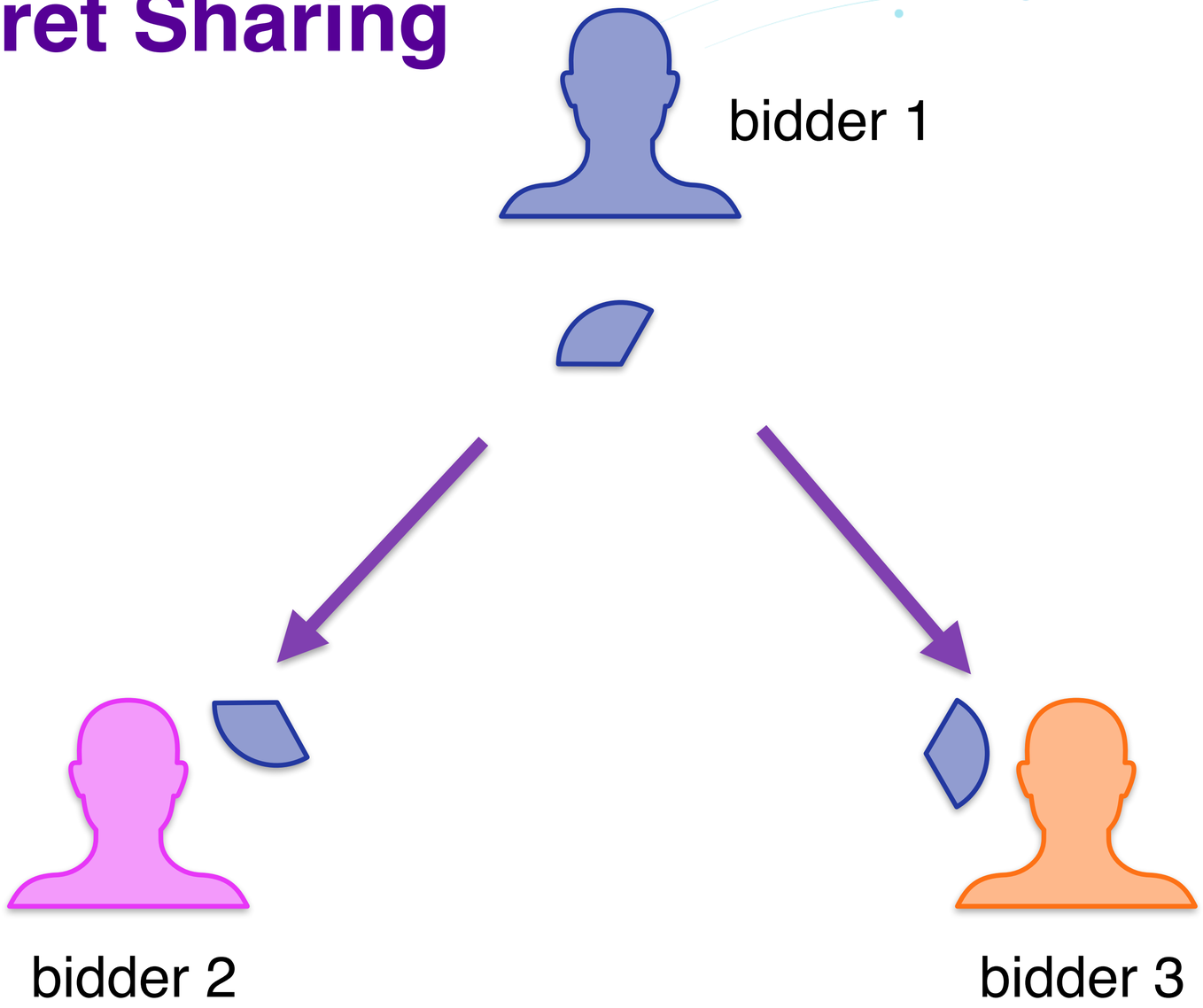
Secret Sharing



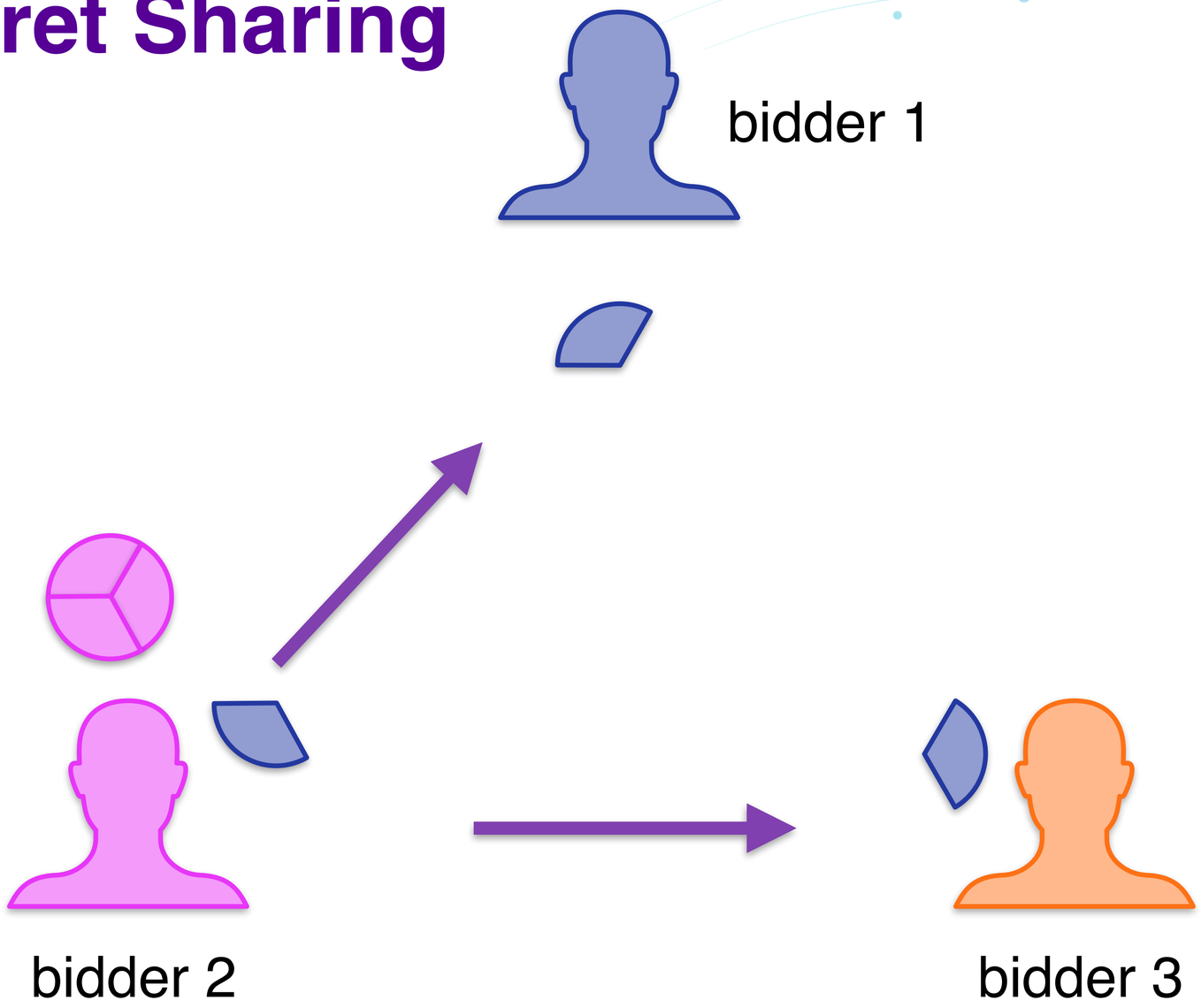
Secret Sharing



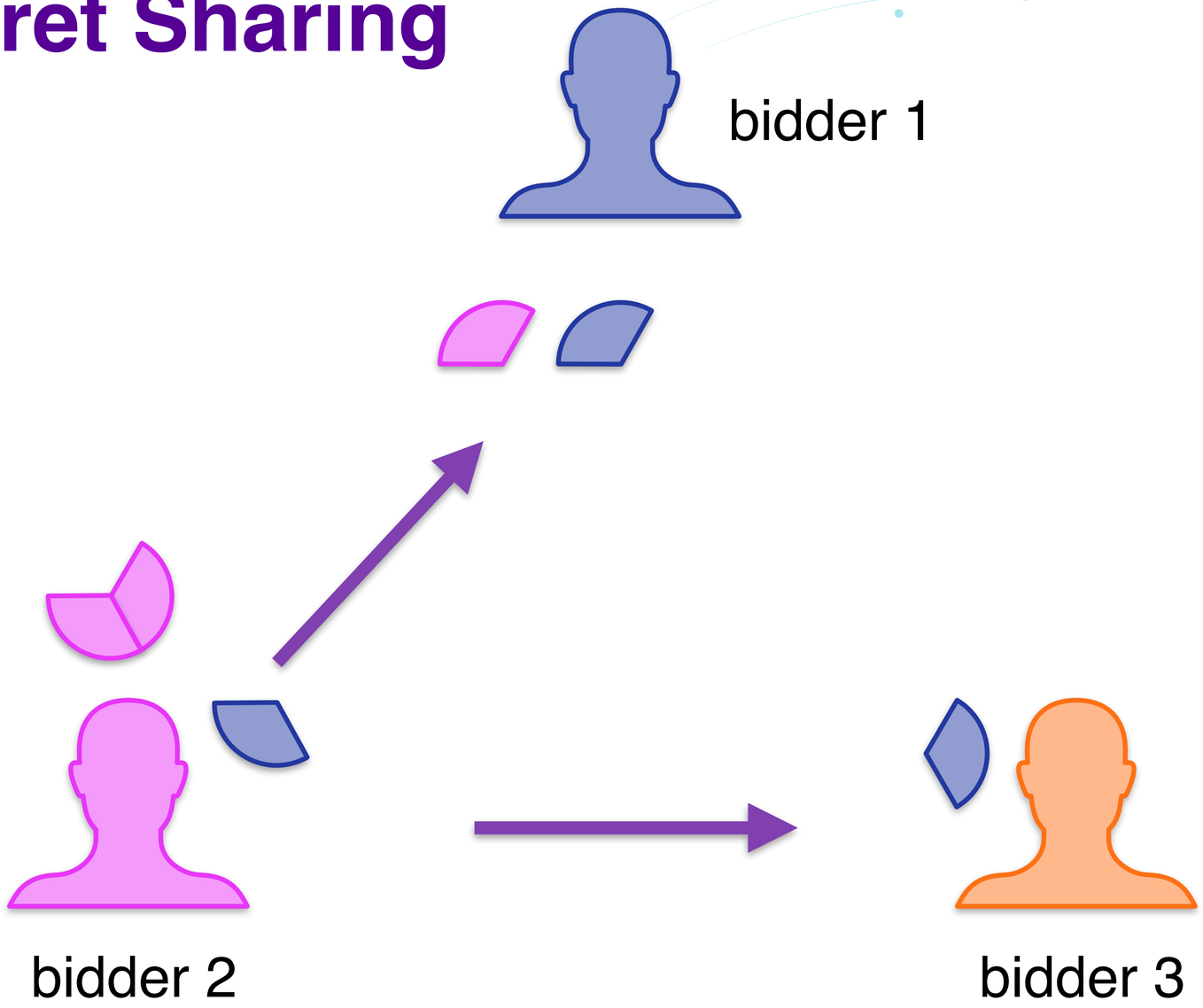
Secret Sharing



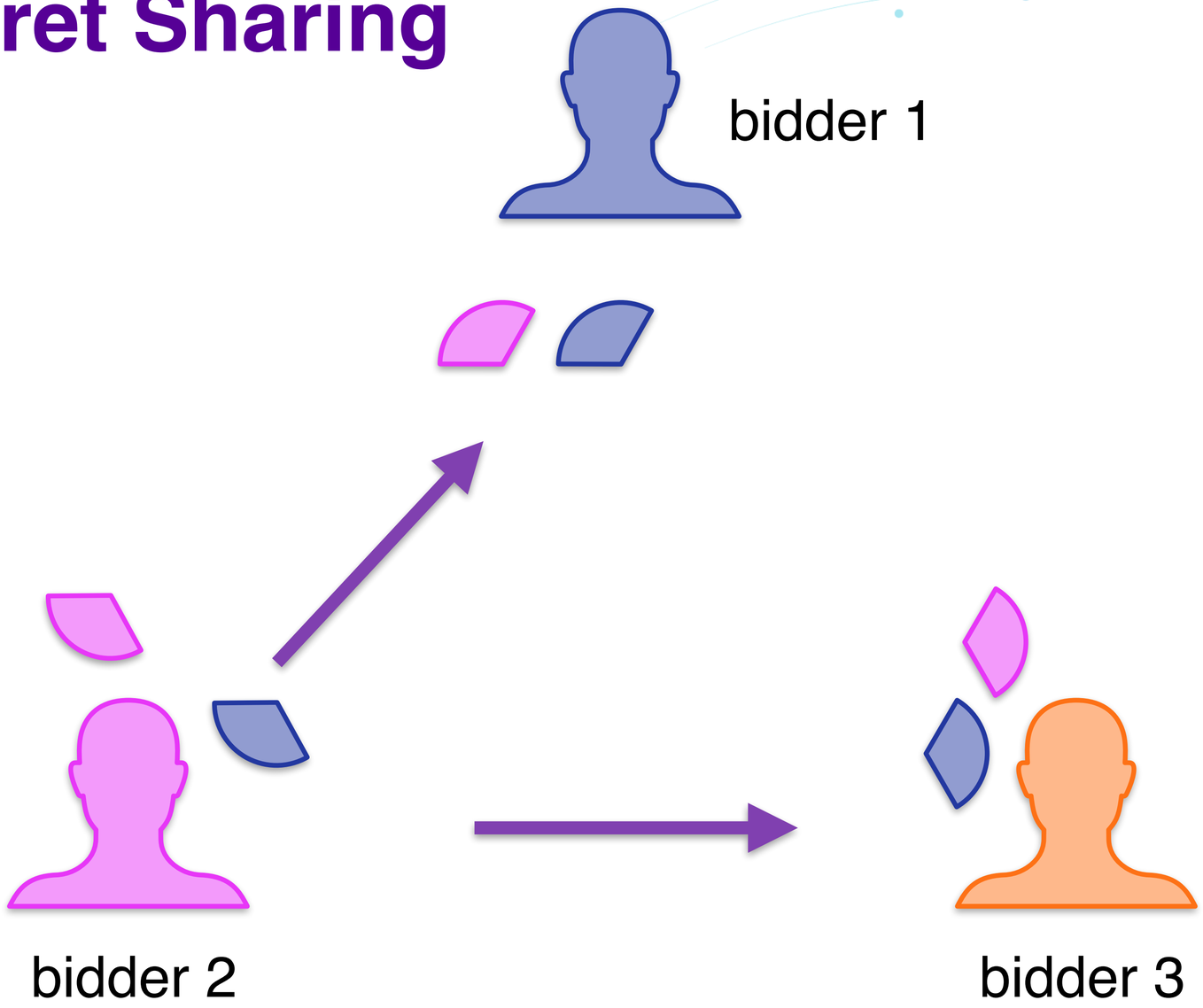
Secret Sharing



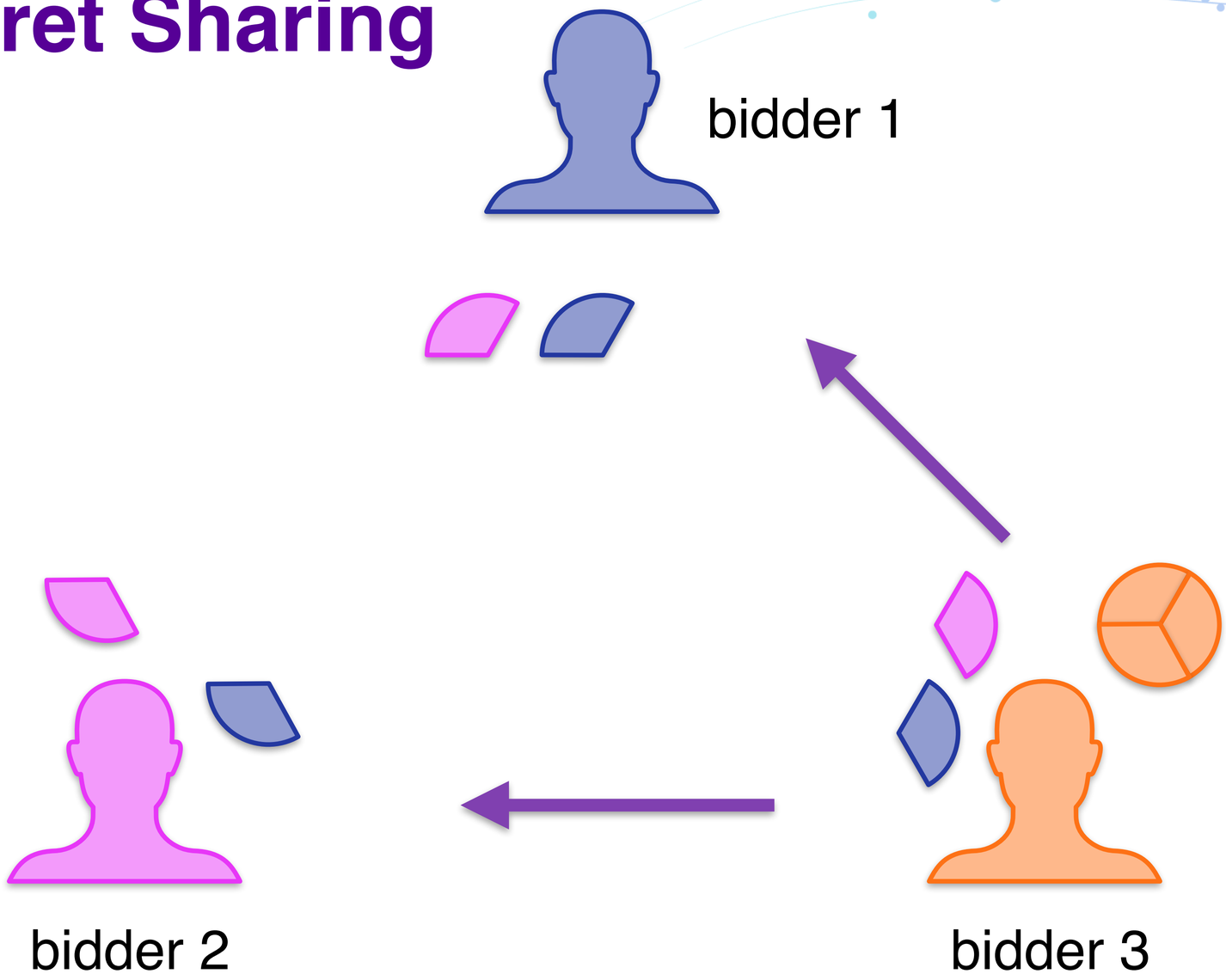
Secret Sharing



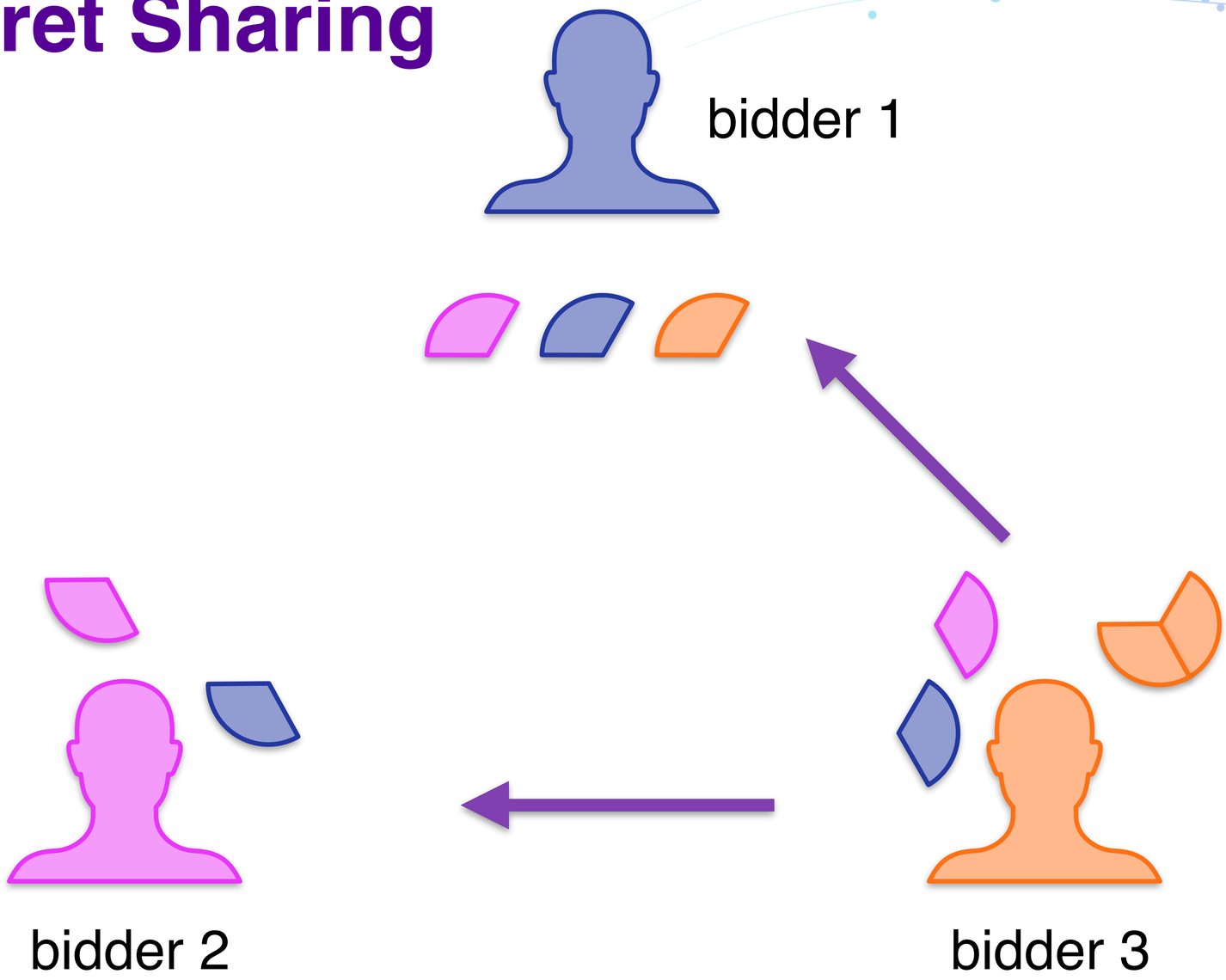
Secret Sharing



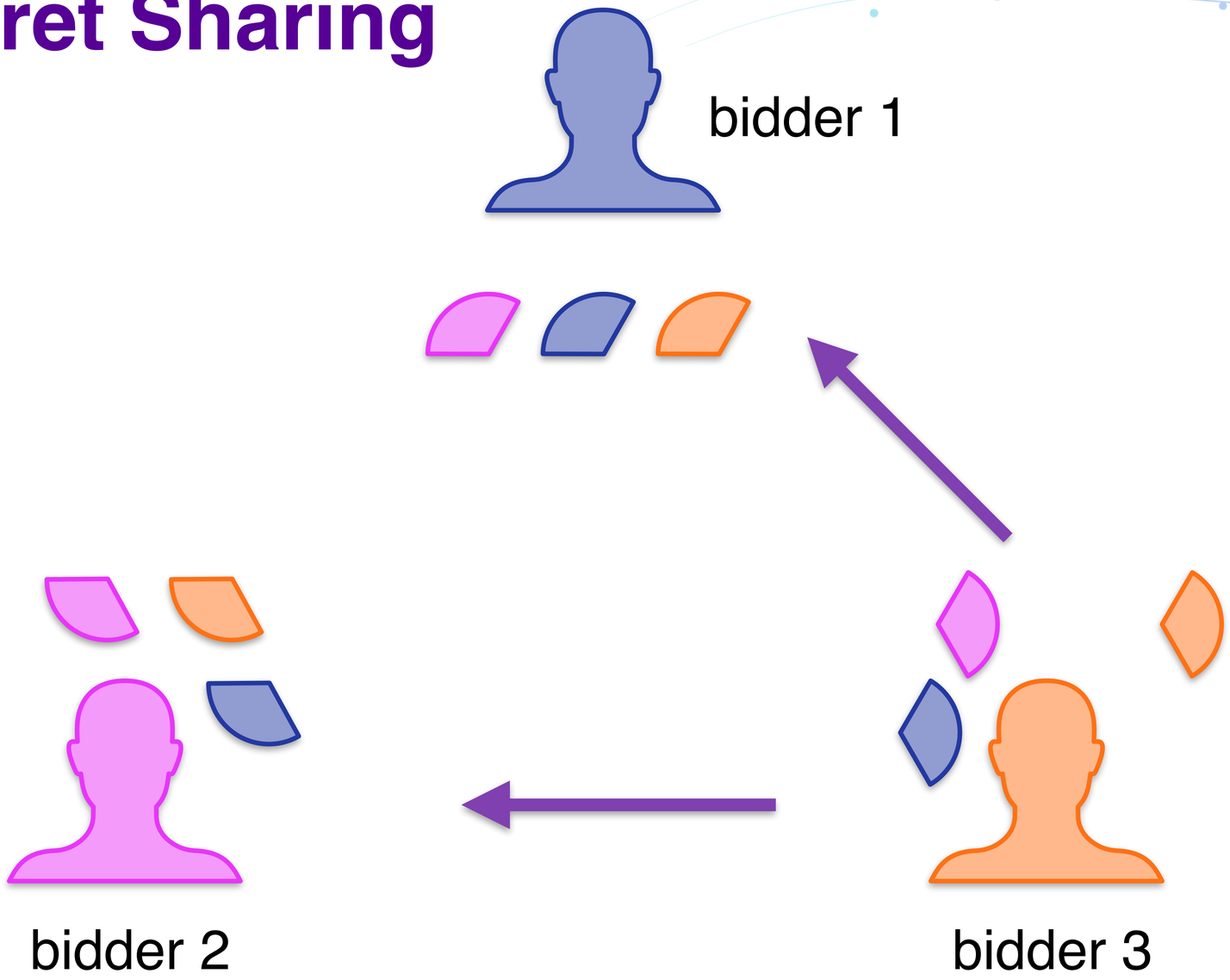
Secret Sharing



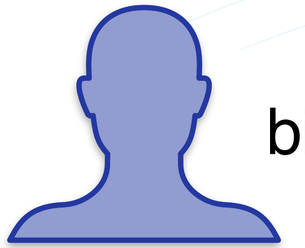
Secret Sharing



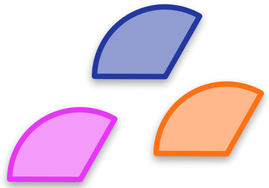
Secret Sharing

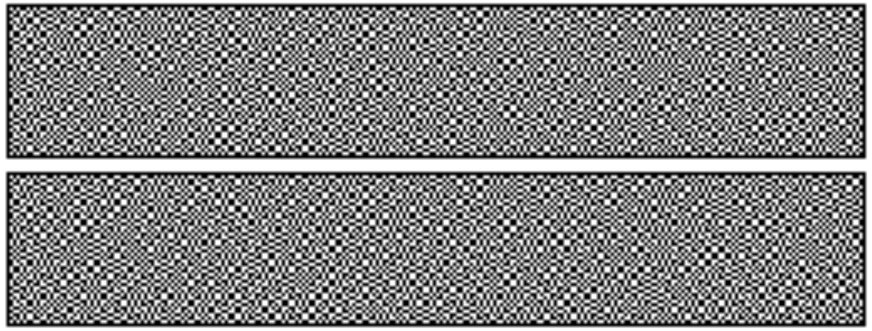


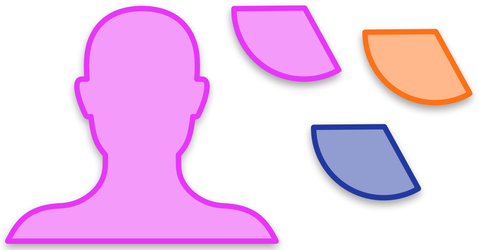
Secret Sharing



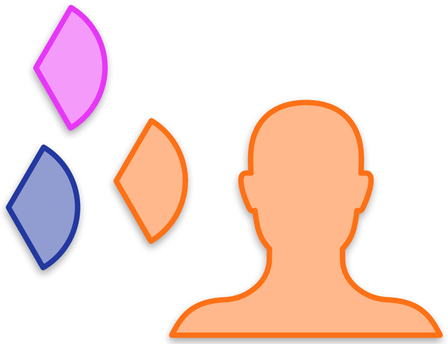
bidder 1







bidder 2



bidder 3

Secret Sharing

Let $s \in G$ be a secret input bid (e.g. $G = \mathbb{Z}_p$).

Pick random $r_1, \ldots, r_n \in G$ such that $s = r_1 + r_2 + \cdots + r_n$.

Send each r_i, **the *i*-th share**, to the i-th participant.

Here:

- n shares together reconstruct the secret.
- $n - 1$ shares or fewer give **zero information** about the secret.

Computing on Secret-Shared Data

We denote $[\![s]\!]$ for a secret-shared value.

- Addition: $[\![x + y]\!] = [\![x]\!] + [\![y]\!]$ (local operation)
- Multiplication $[\![x \cdot y]\!]$ requires interaction (exchange of messages)

Computing on Secret-Shared Data

We denote $[\![s]\!]$ for a secret-shared value.

- Addition: $[\![x + y]\!] = [\![x]\!] + [\![y]\!]$ (local operation)
- Multiplication $[\![x \cdot y]\!]$ requires interaction (exchange of messages)

This work:

Comparison $$[\![x \leq y]\!] = \begin{cases} [\![\ \ 1]\!] & \text{if } [\![x]\!] \leq [\![y]\!] \\ [\![-1]\!] & \text{if } [\![x]\!] > [\![y]\!] \end{cases}$$

Secure Comparison in MPC

$$[\![x]\!] := [\![a]\!] - [\![b]\!]$$

- Compute $\mathrm{sgn}(x)$ or $\mathrm{bsgn}(x)$

$$\mathrm{sgn}(z) := \begin{cases} 1 & \text{if } z > 0, \\ 0 & \text{if } z = 0, \\ -1 & \text{if } z < 0. \end{cases} \qquad \mathrm{bsgn}(z) := \begin{cases} 1 & \text{if } z \geq 0, \\ -1 & \text{if } z < 0. \end{cases}$$

Using the Legendre Symbol to Compute the Sign

For infinitely many primes p, there exists $d = \Omega(\log p)$ such that

$$\left(x \mid p\right) = \operatorname{sgn}(x) \qquad \forall x \in [-d, d]$$

Idea goes back to [Feige et al., 1994], developed in [Yu, 2011].

Using the Legendre Symbol to Compute the Sign

For infinitely many primes p, there exists $d = \Omega(\log p)$ such that

$$\left(x \mid p\right) = \operatorname{sgn}(x) \qquad \forall x \in [-d, d]$$

Idea goes back to [Feige et al., 1994], developed in [Yu, 2011].

Example: $p = 311$, $\lceil \log_2 p \rceil = 9$, $d = 10$

x	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
$\left(x \mid p\right)$	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
$\left(x \mid p\right)$	0	1	1	1	1	1	1	1	1	1	1	-1	1	1	

Securely Computing the Legendre Symbol

Assume *arithmetic black box* $[\![\cdot]\!]$ over $\mathbb{Z}_p$

The Legendre symbol is completely multiplicative:
$(a \mid p)(b \mid p) = (ab \mid p)$ for any $a, b \in \mathbb{Z}$.

Securely Computing the Legendre Symbol

Assume *arithmetic black box* $[\![\cdot]\!]$ over $\mathbb{Z}_p$

The Legendre symbol is completely multiplicative:
$(a \mid p)(b \mid p) = (ab \mid p)$ for any $a, b \in \mathbb{Z}$.

Protocol Legendre$[\![x]\!]$

Given pre-processed pair $([\![r]\!], [\![s]\!])$ where $r \leftarrow_R \mathbb{Z}_p^*$ and $s = (r \mid p)$

1: $c \leftarrow [\![x]\!] \cdot [\![r]\!]$
2: $[\![z]\!] \leftarrow (c \mid p) \cdot [\![s]\!]$
3: **return** $[\![z]\!]$

Securely Computing the Legendre Symbol

Assume *arithmetic black box* $[\![\cdot]\!]$ over $\mathbb{Z}_p$

The Legendre symbol is completely multiplicative:
$(a \mid p)(b \mid p) = (ab \mid p)$ for any $a, b \in \mathbb{Z}$.

Protocol Legendre$[\![x]\!]$

Given pre-processed pair $([\![r]\!], [\![s]\!])$ where $r \leftarrow_R \mathbb{Z}_p^*$ and $s = (r \mid p)$

1: $c \leftarrow [\![x]\!] \cdot [\![r]\!]$
2: $[\![z]\!] \leftarrow (c \mid p) \cdot [\![s]\!]$
3: **return** $[\![z]\!]$

Cannot securely evaluate $(0 \mid p)$! (But we ignore this in this talk)

Fast (Low-Latency) Secure Comparisons

Legendre-based comparison requires a single round in the online phase!

Fast (Low-Latency) Secure Comparisons

Legendre-based comparison requires a single round in the online phase!

- Caveat: Comparison range limited to *d*

Fast (Low-Latency) Secure Comparisons

Legendre-based comparison requires a single round in the online phase!

- Caveat: Comparison range limited to d

Can we increase d while keeping p fixed?

New Idea: Correcting “Errors” via Majority Vote

- Let’s take again $p = 311$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	…
$(x \mid p)$	0	1	1	1	1	1	1	1	1	1	1	-1	1	1	1	1	1	-1	1	-1	1	…

New Idea: Correcting “Errors” via Majority Vote

- Let’s take again $p = 311$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	…
$(x \mid p)$	0	1	1	1	1	1	1	1	1	1	1	-1	1	1	1	1	1	-1	1	-1	1	…

- Inspect neighboring Legendre symbols, compute sign of

$$y(x) = (x-1 \mid p) + (x \mid p) + (x+1 \mid p) \in [-3, +3] \subset \mathbb{Z}$$

- We can compute sign of $y(x)$ using a low-degree polynomial

In General (Window Length $2k+1$, Arbitrary k)

Definition

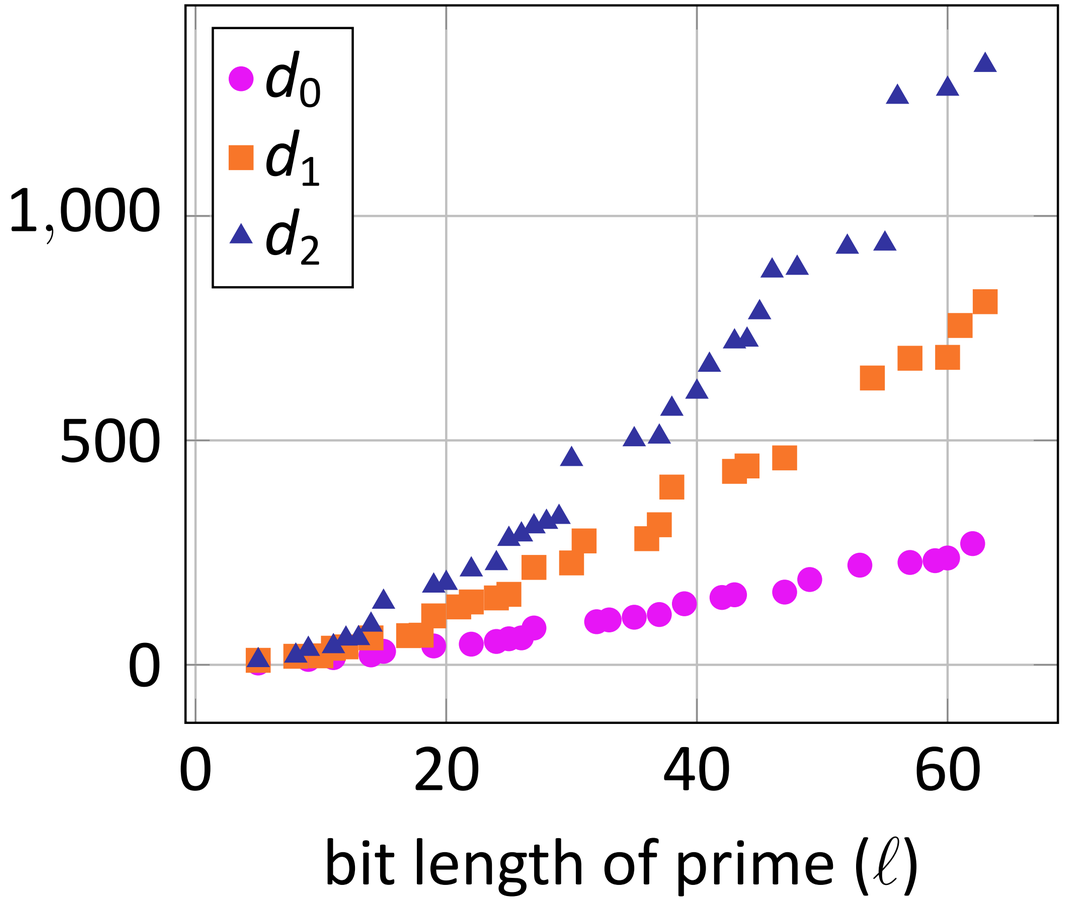
Let $k \in \mathbb{Z}_{\geq 0}$, and let $p > 2k+1$ be a Blum prime.

The *k-range of p*, denoted $d_k(p)$, is the largest integer d such that

$$\sum_{i=-k}^{k} (x+i \mid p) > 0 \quad \text{holds for all } x \text{ with } 1 \leq x \leq d, \tag{1}$$

and we set $d_k(p) := 0$ if no such d exists.

$k > 0$ Increases the Range of the Comparison



Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

We can do better…

Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

We can do better…

- By multiplicativity $(ab \,|\, p) = (a \,|\, p)(b \,|\, p)$: to calculate $d_0(p)$ only need to check $(q \,|\, p) = 1$ for $q = 2, 3, 5, 7, 11, \ldots$.

Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

We can do better…

- By multiplicativity $(ab \mid p) = (a \mid p)(b \mid p)$: to calculate $d_0(p)$ only need to check $(q \mid p) = 1$ for $q = 2, 3, 5, 7, 11, \ldots$.
- Quadratic reciprocity: $(p \mid q) = (-1 \mid q)(-1 \mid p)(q \mid p)$, so $(q \mid p)$ only depends on residue class modulo q.
  We precompute $(a \mid q)$ for each $0 \leq a < q$.

Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

We can do better…

- By multiplicativity $(ab \mid p) = (a \mid p)(b \mid p)$: to calculate $d_0(p)$ only need to check $(q \mid p) = 1$ for $q = 2, 3, 5, 7, 11, \ldots$.
- Quadratic reciprocity: $(p \mid q) = (-1 \mid q)(-1 \mid p)(q \mid p)$, so $(q \mid p)$ only depends on residue class modulo q.
  We precompute $(a \mid q)$ for each $0 \leq a < q$.
- Use *wheel data structure* to enumerate p that have high $d_0(p)$

Finding Primes p with High $d_0(p)$

Naive search: try all primes $p = 2, 3, 5, 7, 11, \ldots$ and calculate $d_0(p)$.

We can do better…

- By multiplicativity $(ab \mid p) = (a \mid p)(b \mid p)$: to calculate $d_0(p)$ only need to check $(q \mid p) = 1$ for $q = 2, 3, 5, 7, 11, \ldots$.
- Quadratic reciprocity: $(p \mid q) = (-1 \mid q)(-1 \mid p)(q \mid p)$, so $(q \mid p)$ only depends on residue class modulo q.
  We precompute $(a \mid q)$ for each $0 \leq a < q$.
- Use *wheel data structure* to enumerate p that have high $d_0(p)$

For $k > 0$: more involved. Details in paper, code on Github.

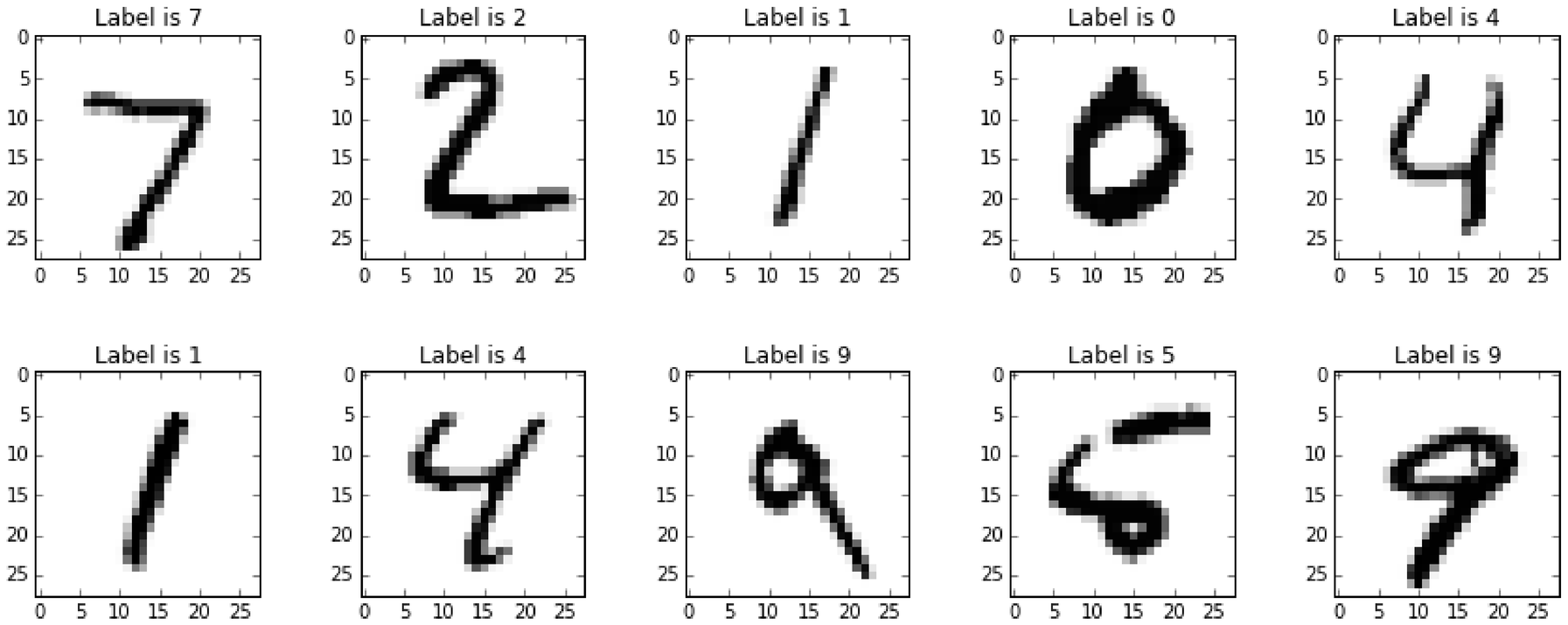
Results

- Our protocol for $k = 1$: single-round online phase, 3 secure multiplications in parallel.
- For $k = 1, 2$, we have searched all[1] primes up to 64 bits.

k	best d_k	range improvement (w.r.t. d_0)
0	270	
1	809	3.0×
2	1336	4.9×

[1] With conditions $d_0(p) \geq 64$, and for $k = 2$ also $d_1(p) \geq 400$.

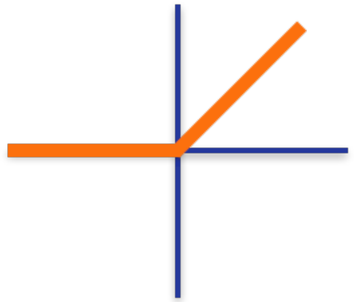
Application: Secure Neural Network Evaluation



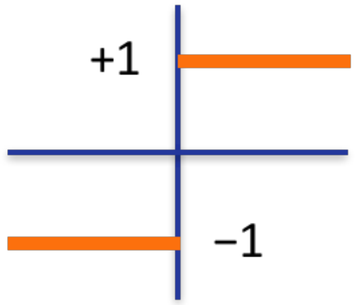
Application: Secure Neural Network Evaluation

- Handwritten digit recognition “MNIST”
- Binarized multi-layer perceptron
  - Neuron outputs are in $\{-1, +1\}$
  - Uses bsgn as activation function
- Implementation in the MPyC framework

ReLU Activation Function



bsgn Activation Function



Questions?

References

Feige, U., Kilian, J., and Naor, M. (1994).
A minimal model for secure computation (extended abstract).
In *Proceedings STOC '94*, pages 554–563.

Yu, C.-H. (2011).
Sign modules in secure arithmetic circuits.
Cryptology ePrint Archive, Report 2011/539.
`http://eprint.iacr.org/2011/539`.