

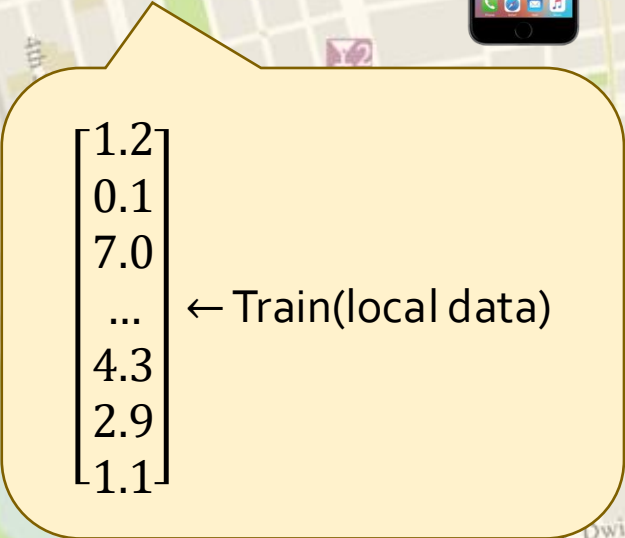
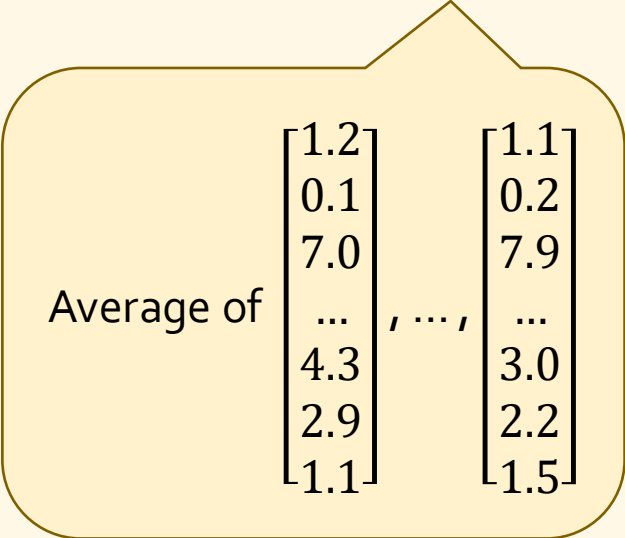
Private aggregation at scale with lightweight distributed trust

Yiping Ma
UC Berkeley

Trust in Decentralized Systems, Simons Institute, 2026

Motivation: Private aggregation for federated learning (FL)

Federated learning

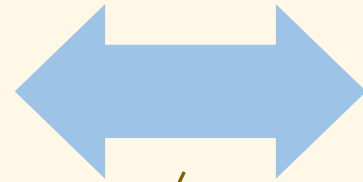


Motivation: Private aggregation for federated learning (FL)

Private aggregation



Server learns **only** $\sum \mathbf{x}_i$



Client i holds input \mathbf{x}_i

Many instances
of aggregation!

Problem: FL setting has stringent system constraints

- A single untrusted server



re:Invent | Discover AWS | Products | Solutions | Pricing | Resources

automation and the use of large distributed datasets. With increased access to data, ML has the potential to provide

AWS Blogs

Home

Blogs ▾

Editions ▾

To address this issue, federated learning (FL) is a decentralized and collaborative ML training technique that offers data privacy while maintaining accuracy and fidelity. Unlike traditional ML training, FL training occurs within an isolated client location using an independent secure session. The client only shares its output model parameters with a centralized server, known as the training coordinator or aggregation server, and not the actual data used to train the model. This approach alleviates many data privacy concerns while enabling effective collaboration on model training.



Google Cloud

Overview | Solutions | Products | Pricing | Resources | Contact us

Docs | Support | Contact

Federated learning: a guide to what it is and how it works

What is federated learning?

Federated learning versus machine learning

The different types of federated learning

How does federated learning work?

Federated learning works through an iterative process involving a central coordinator (typically a server) and multiple participating clients (devices or organizations). The general workflow can be broken down into these key steps:

1. Initial model distribution

The process begins with a central server initializing a global machine learning model. This model serves as the starting point for the collaborative training. The server then distributes this global model to a selected subset of participating client devices.

Problem: FL setting has stringent system constraints

- **A single untrusted server**
 - Restrict the private solution to work with only one server!

Problem: FL setting has stringent system constraints

- A single untrusted server
 - Restrict the private solution to work with only one server!
- **Many mobile clients**
 - Weak computational power
 - Unstable in execution

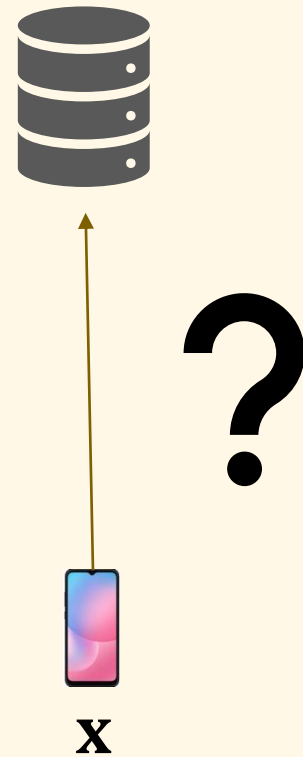
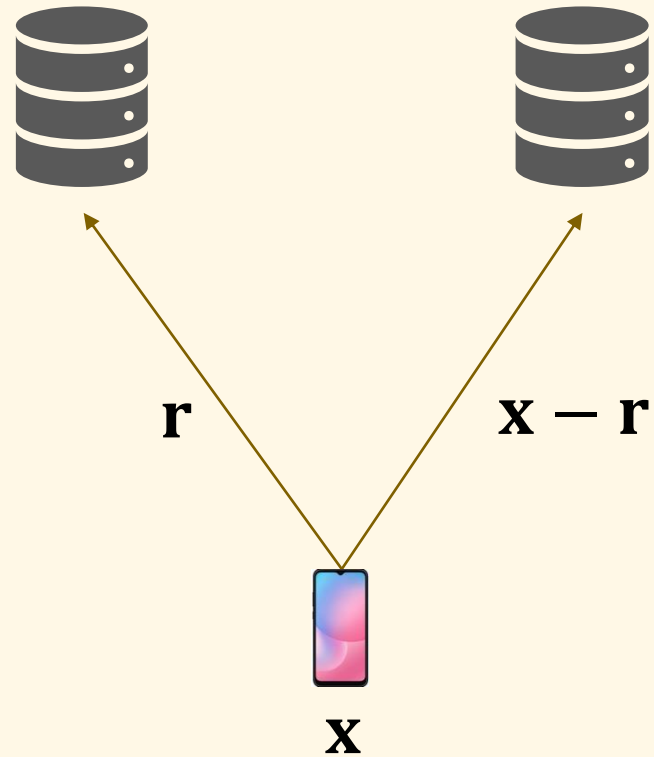
Problem: FL setting has stringent system constraints

- A single untrusted server
 - Restrict the private solution to work with only one server!
- **Many mobile clients**
 - Weak computational power → Client part must be lightweight
 - Unstable in execution

Problem: FL setting has stringent system constraints

- A single untrusted server
 - Restrict the private solution to work with only one server!
- **Many mobile clients**
 - Weak computational power → Client part must be lightweight
 - Unstable in execution → Protocol needs to handle failures

Problem: Classical multi-server solutions do not work



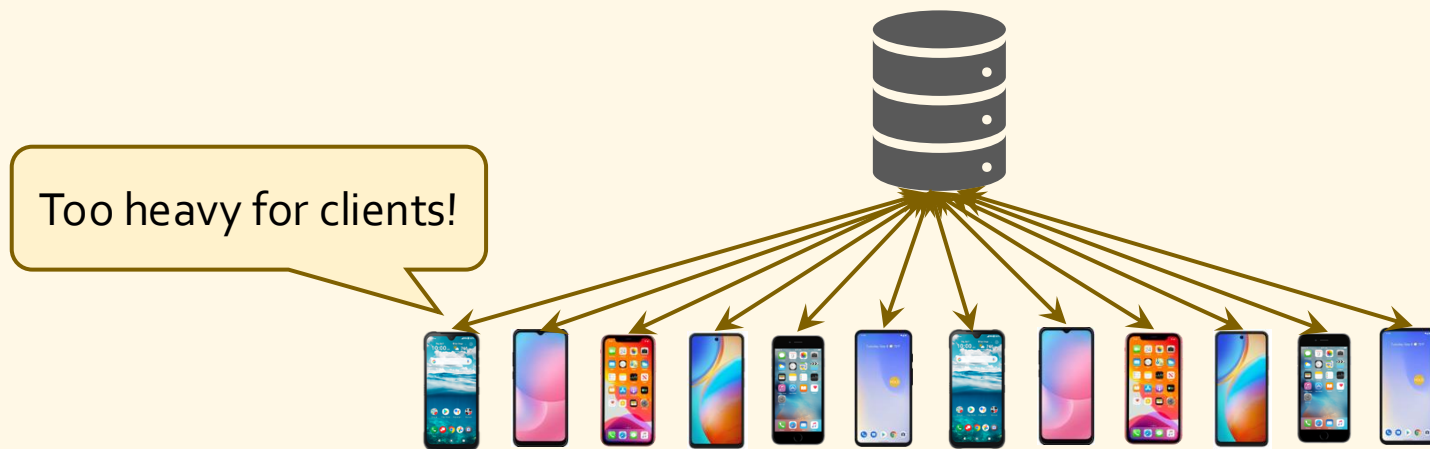
Classical idea: Distribute trust among clients

- **A naïve solution:** A generic MPC among clients
 - Clients have point-to-point channels
 - Require each client to talk to every other client



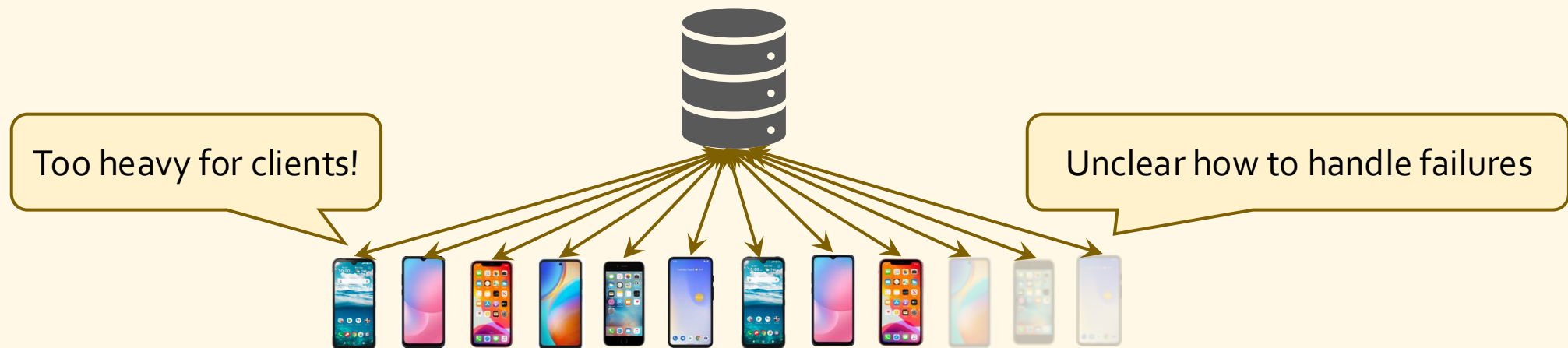
Classical idea: Distribute trust among clients

- **A naïve solution:** A generic MPC among clients
 - ~~Clients have point-to-point channels~~ Client communicate through the server
 - Require each client to talk to every other client



Classical idea: Distribute trust among clients

- **A naïve solution:** A generic MPC among clients
 - ~~Clients have point-to-point channels~~ Client communicate through the server
 - Require each client to talk to every other client



Prior work has greatly improved client cost

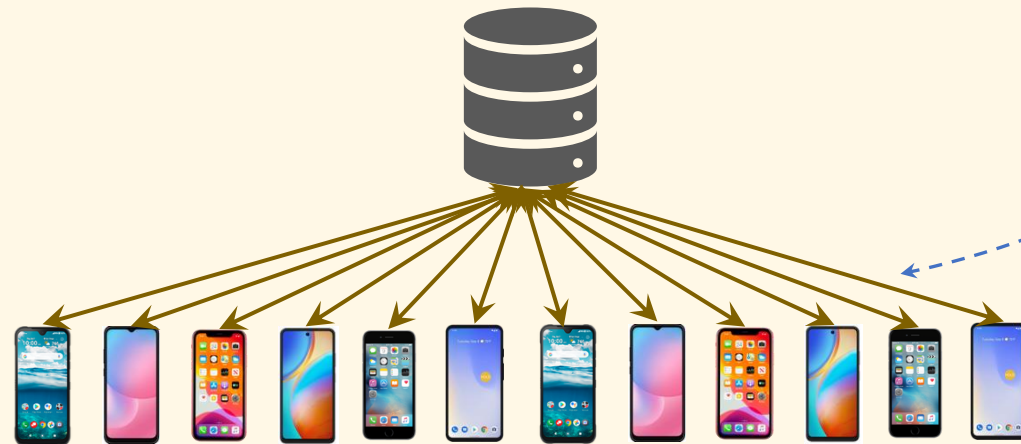
Parameters: input length ℓ , number of clients n .

	Handle failures?	Client comp.	Client comm.	Rounds
Naïve MPC	✗	$O(\ell \cdot n)$	$O(\ell \cdot n)$	2
Bonawitz et al. 2017	✓	$O(\ell \cdot n)$	$O(\ell + n)$	6

Prior work has greatly improved client cost

Parameters: input length ℓ , number of clients n .

	Handle failures?	Client comp.	Client comm.	Rounds
Naïve MPC	✗	$O(\ell \cdot n)$	$O(\ell \cdot n)$	2
Bonawitz et al. 2017	✓	$O(\ell \cdot n)$	$O(\ell + n)$	6
Bell et al. 2020	✓	$O(\ell + \text{polylog } n)$	$O(\ell + \text{polylog } n)$	6



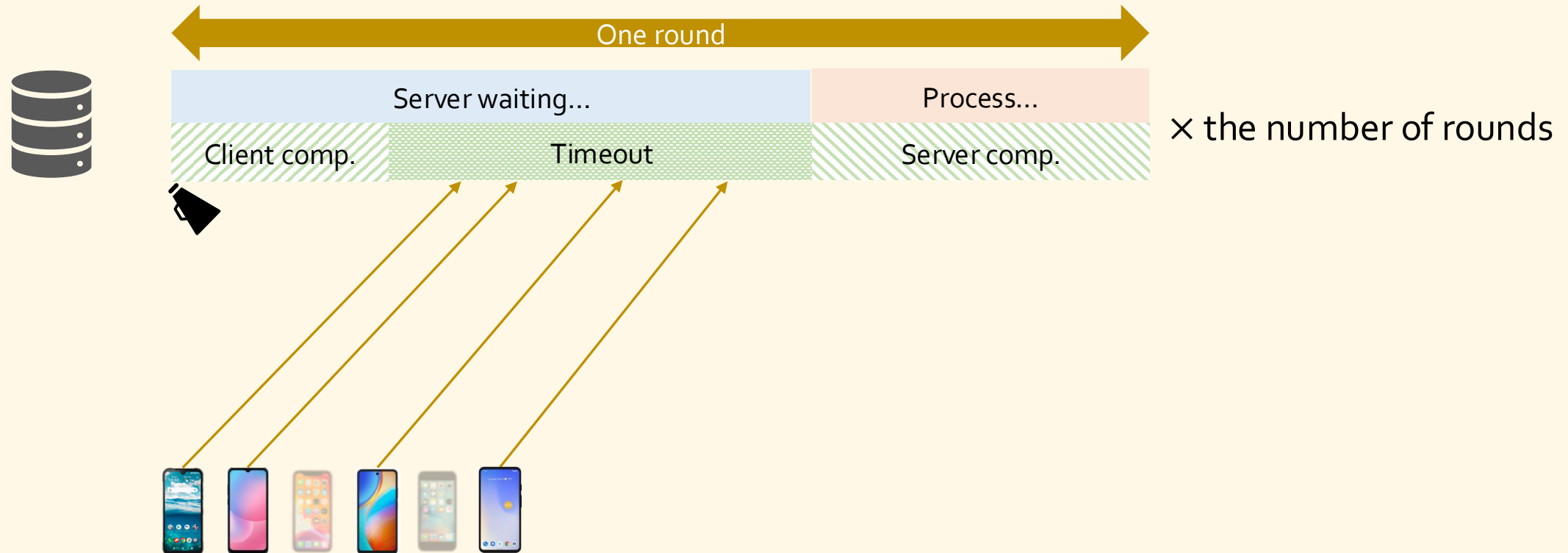
Problem: End-to-end performance is bottlenecked by rounds

- Not only the number of rounds
- But each round the server needs to interact with all the clients
- **The key issue:** their design requires the server to wait for sufficiently many client responses to proceed

Problem: Bottlenecked by waiting on clients

Decreasing the timeout?

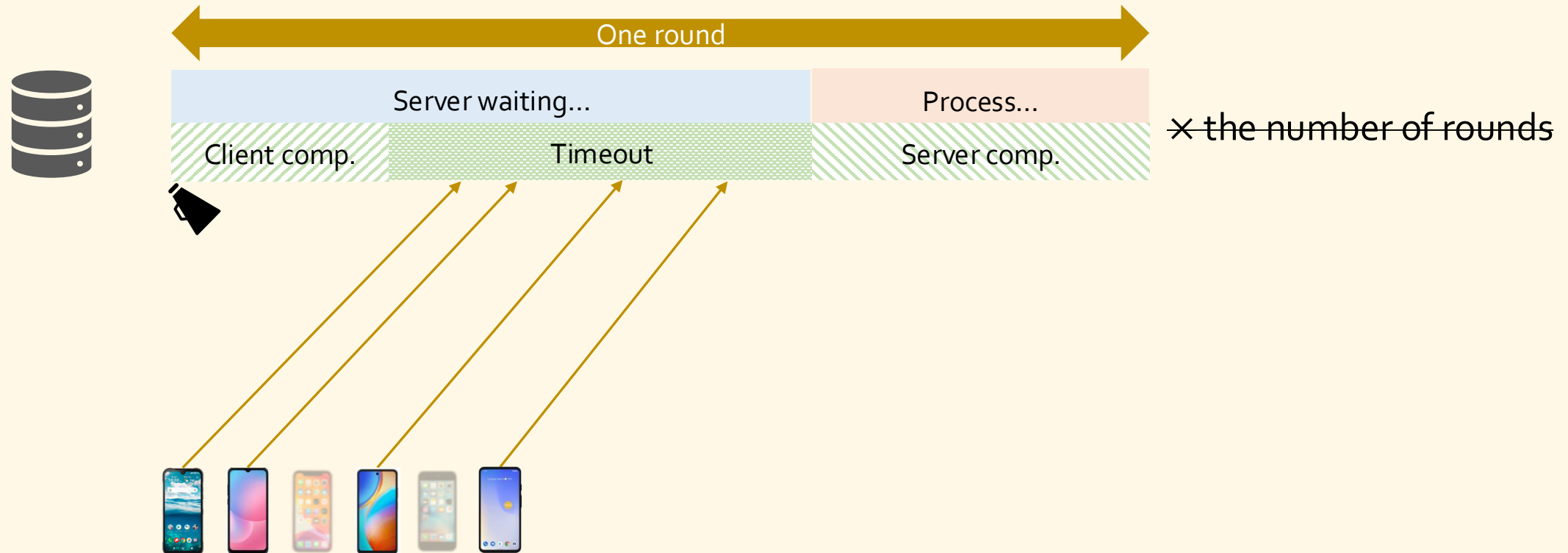
- Fewer clients included in the sum
- Can break the failure tolerance guarantee!



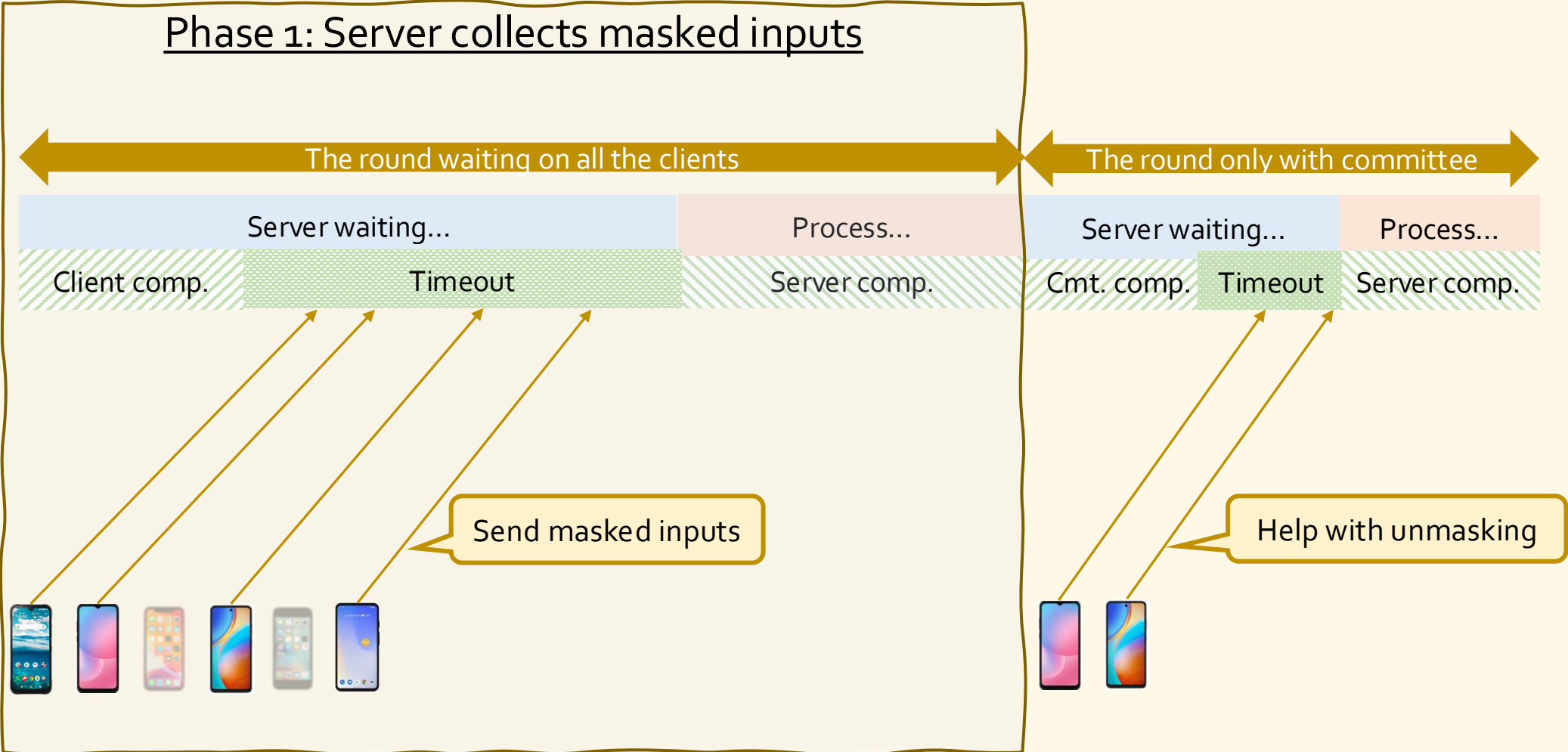
Observation

- Essentially only **one round** needs to wait on **all** the clients
- A round that asks all the clients to submit their inputs (but in masked form)
- After that, unmasking doesn't fundamentally need all-client involvement

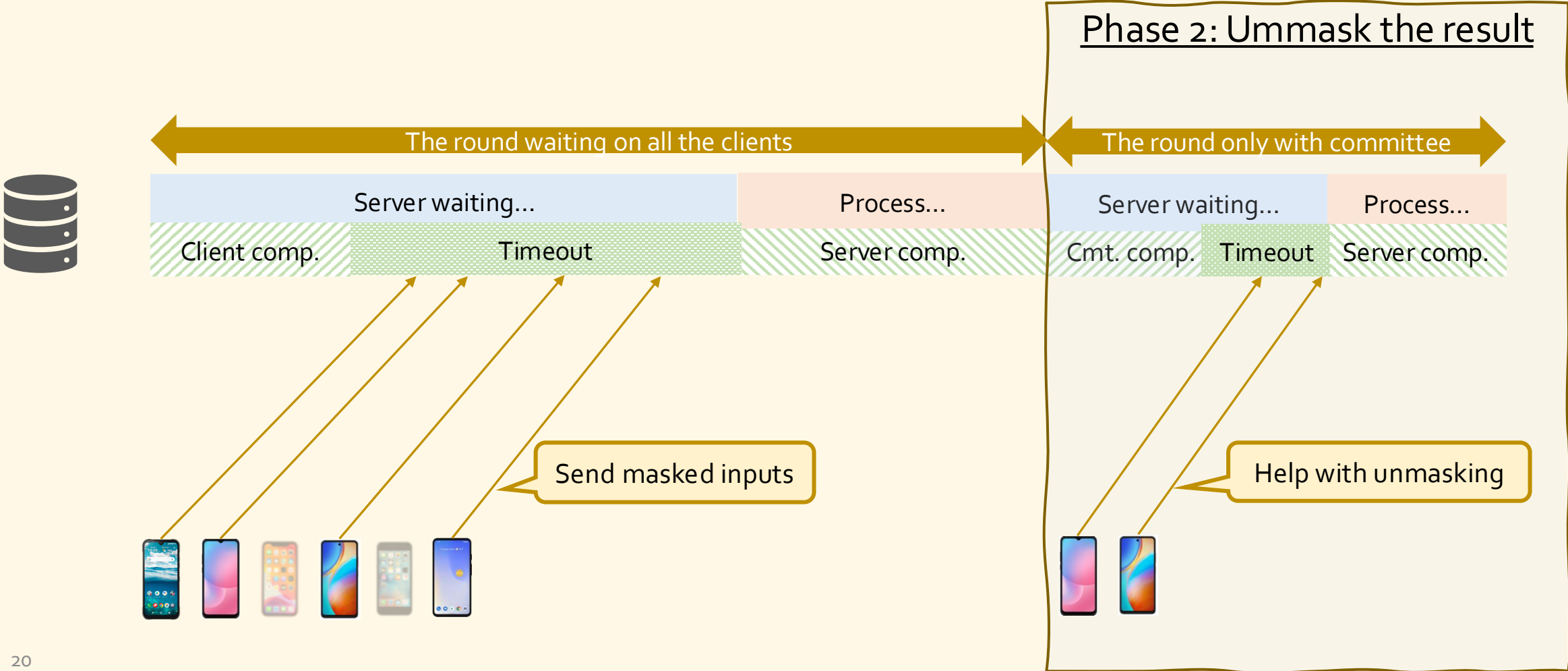
Our approach: A new aggregation paradigm



Our approach: A new aggregation paradigm



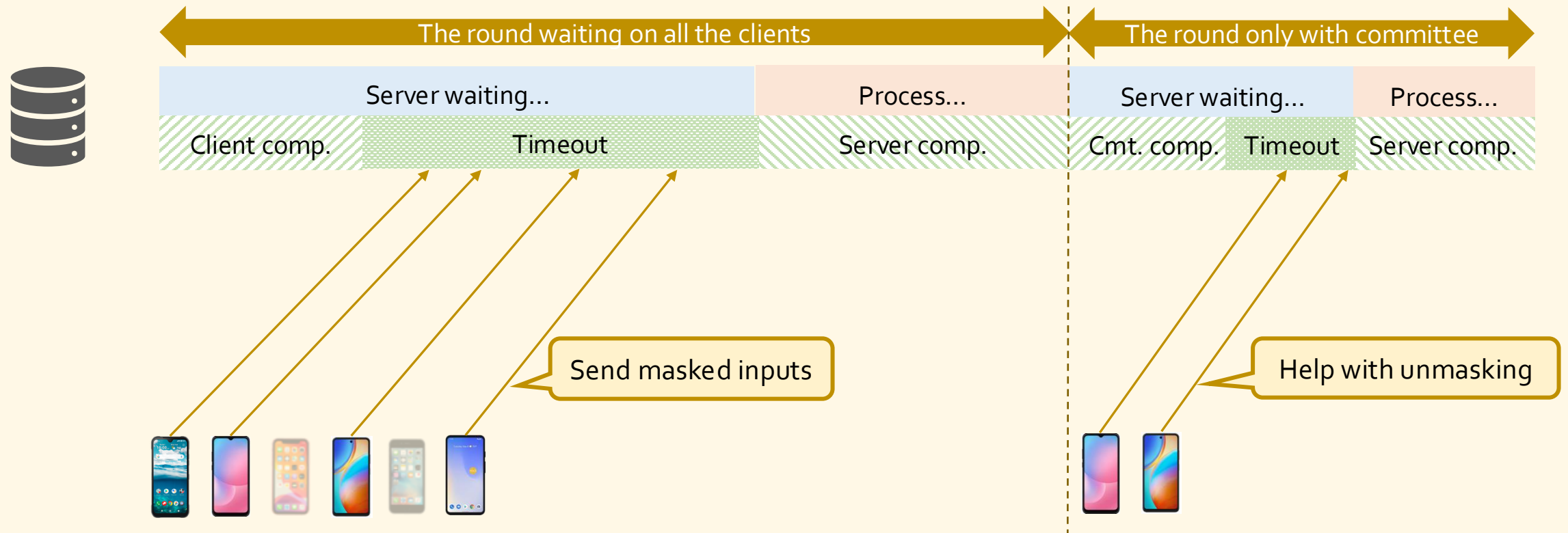
Our approach: A new aggregation paradigm



Our approach: A new aggregation paradigm

Decreasing the timeout?

- Not on the first round
- But on the rest of the rounds



Flamingo 2023



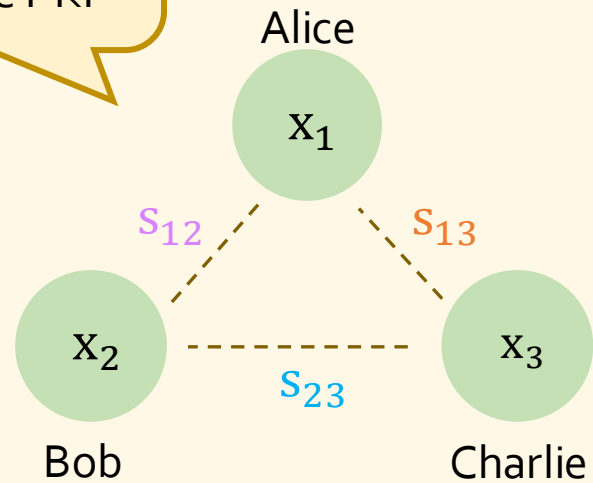
Adversary	Handle failures?	Regular client work	Committee work	Rounds
Semi-honest	✓	$O(\ell + \text{polylog } n)$	$O(n)$	1 + 1
Malicious	✓	$O(\ell + \text{polylog } n)$	$O(n)$	1 + 2

Parameters: input length ℓ , number of clients n .

Starting point: pairwise masking [Chaum88]

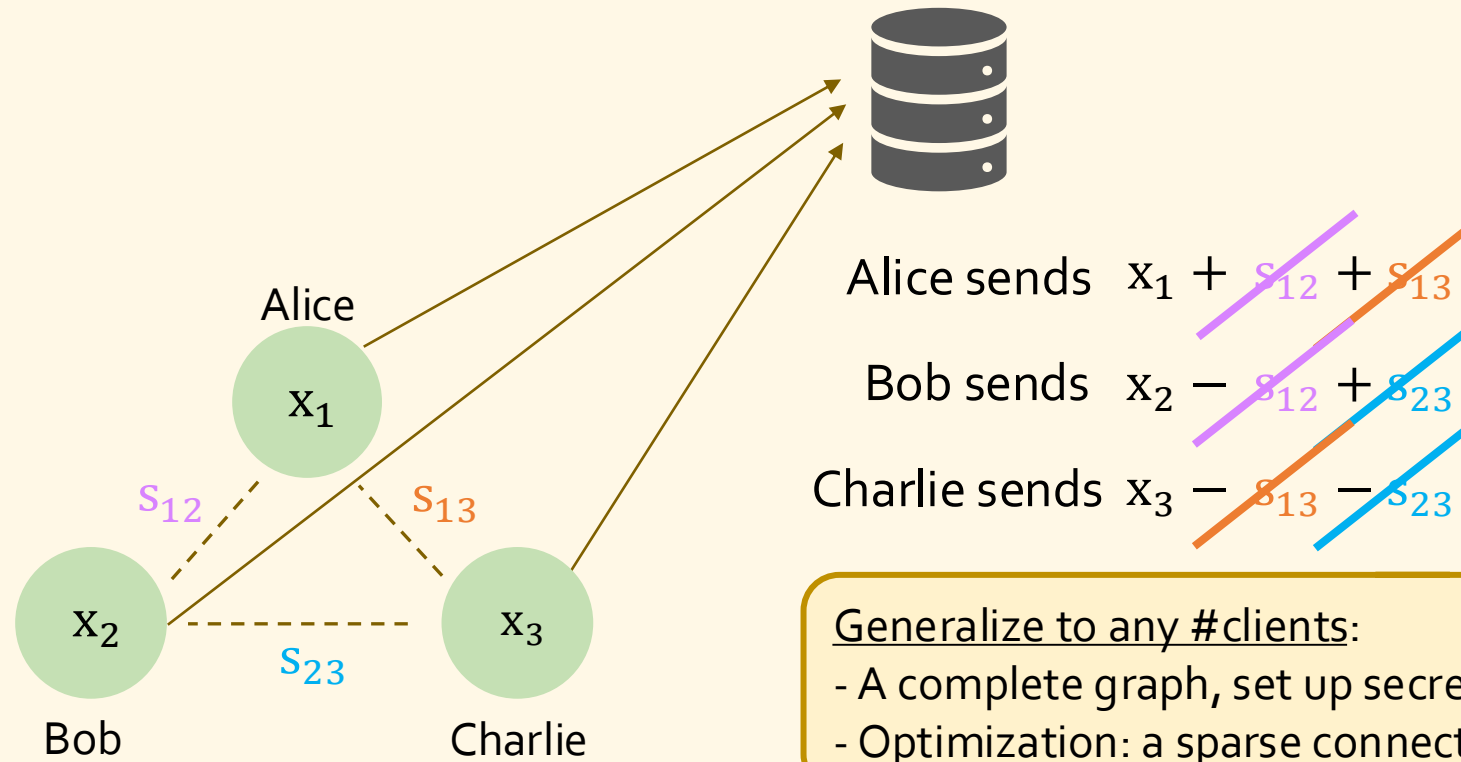
Step 1. Set up secrets

Set up via
DH key exchange
or assume PKI



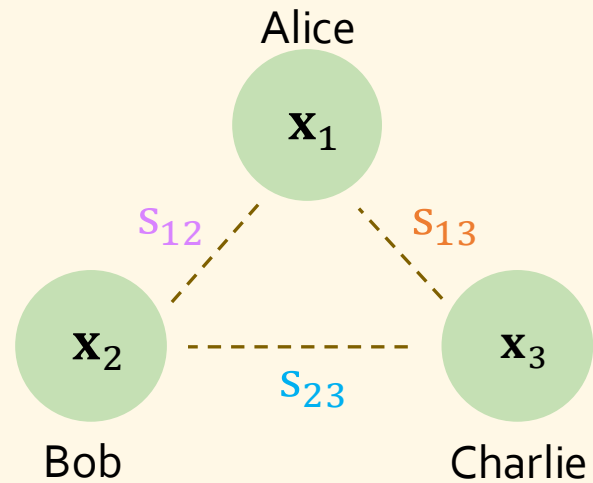
Starting point: pairwise masking [Chaum88]

Step 2. Send masked inputs



Starting point: pairwise masking [Chaum88]

Step 2. Send masked inputs

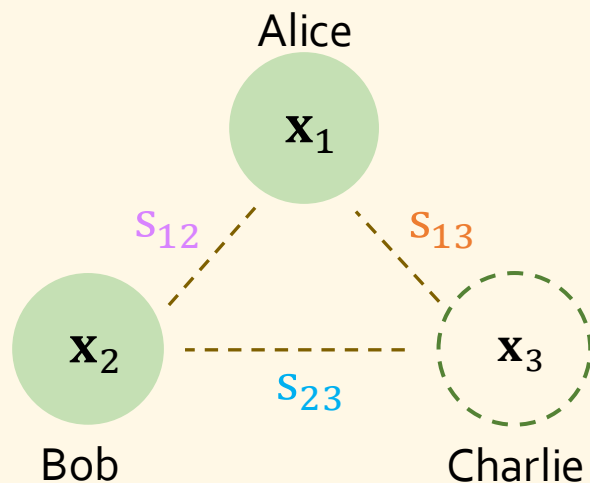


A diagram showing a vertical bar representing a masked input y_1 . The bar is split into two parts: a grey part on the left and a green part on the right. To the right of the bar is the equation $y_1 = x_1 + \text{PRG}(s_{12}) + \text{PRG}(s_{13})$. The terms s_{12} and s_{13} in the equation are highlighted with green boxes.

$$y_1 = x_1 + \text{PRG}(s_{12}) + \text{PRG}(s_{13})$$

Problem: Pairwise masking cannot handle failures

Step 2. Send masked inputs



Each input is still hidden

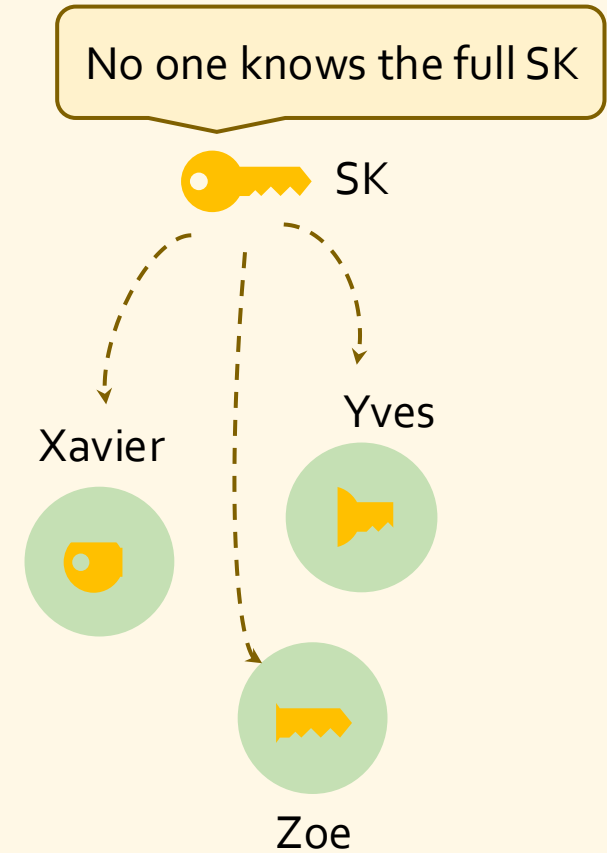
Reveal the secrets?

$$y_1 = x_1 + \text{PRG}(s_{12}) + \text{PRG}(s_{13})$$

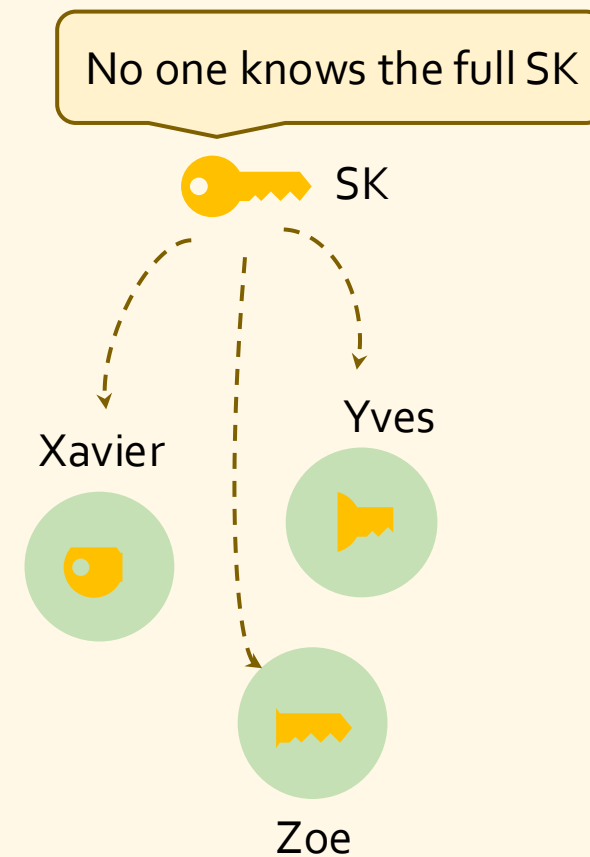
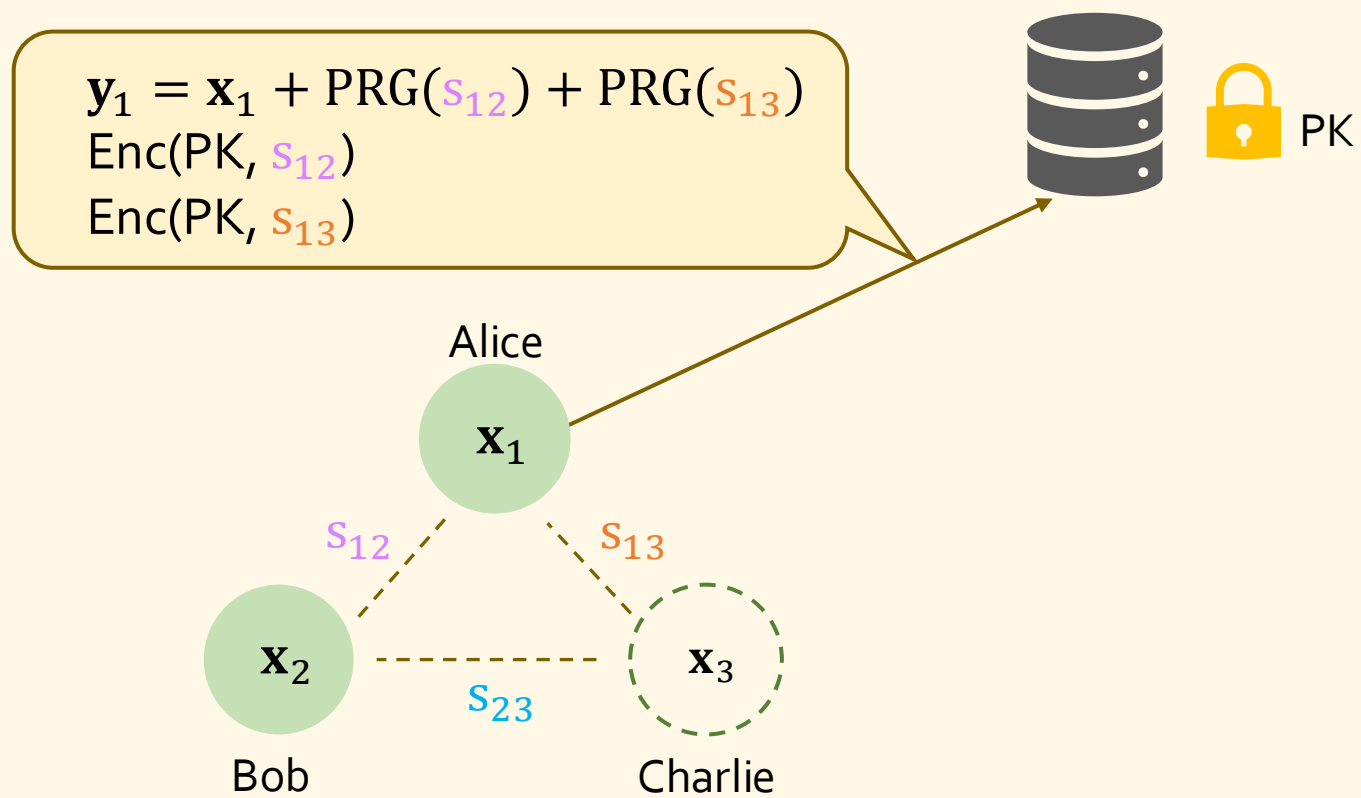
$$y_2 = x_2 - \text{PRG}(s_{12}) + \text{PRG}(s_{23})$$

Idea: Reveal offline-online pair via the committee

Tool: threshold decryption



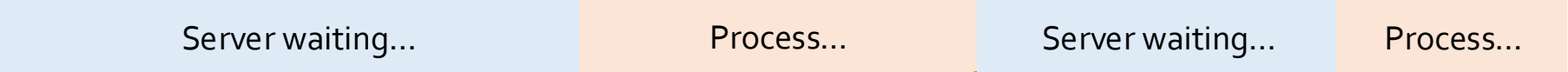
Idea: Reveal offline-online pair via the committee



Flamingo 2023: Workflow



Semi-honest version: 1+1 rounds
 Malicious version: 1+2 rounds



Process...

$$y \leftarrow y_1 + y_2$$

Server waiting...

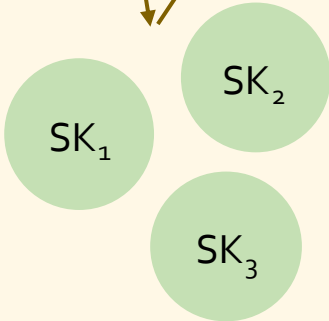
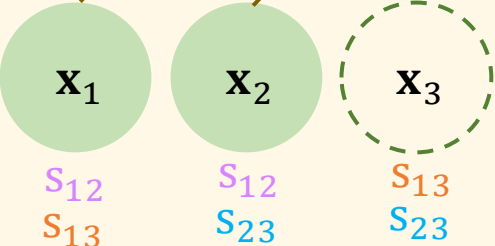
Process...

$$y = \text{PRG}(s_{13}) + \text{PRG}(s_{23})$$

y_1
 $\text{Enc}(\text{PK}, s_{12})$
 $\text{Enc}(\text{PK}, s_{13})$

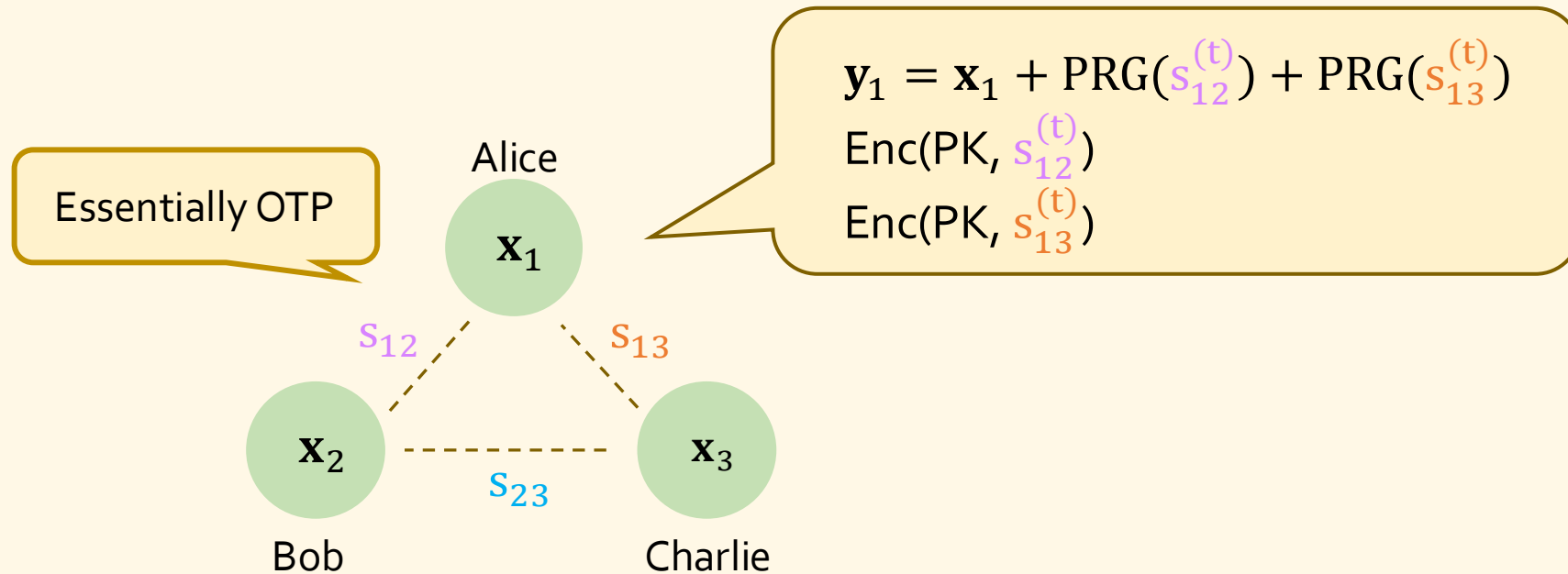
y_2
 $\text{Enc}(\text{PK}, s_{12})$
 $\text{Enc}(\text{PK}, s_{23})$

$\text{Enc}(\text{PK}, s_{13})$
 $\text{Enc}(\text{PK}, s_{23})$



$$y_1 = x_1 + \text{PRG}(s_{12}) + \text{PRG}(s_{13})$$


Reusing secrets from setup



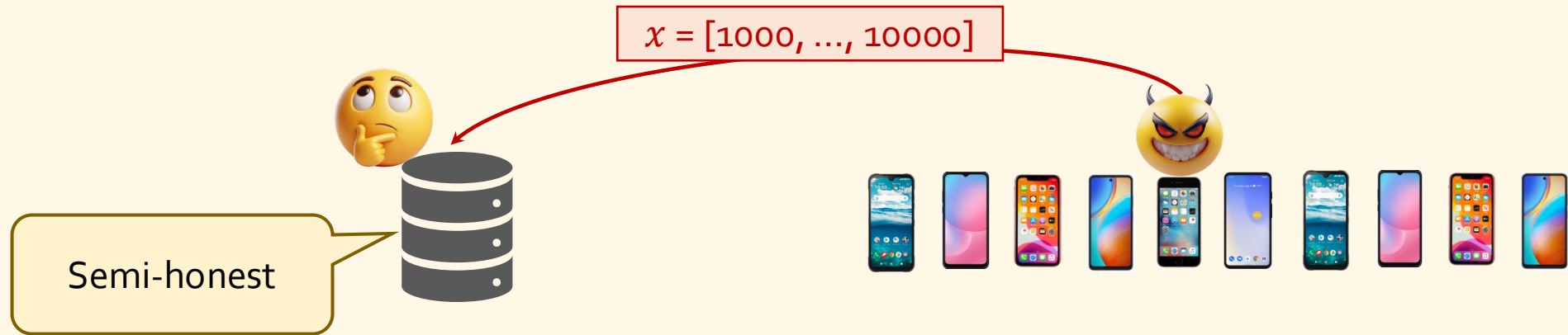
Iteration t : $s_{12}^{(t)} = \text{PRF}(s_{12}, t)$

Performance: Comparison for wait time

#Clients/Iteration	Flamingo Wait	BBGLR Wait	Wait Speedup
128	5.77s	43.66s	7.6×
256	6.11s	41.39s	6.8×
512	5.84s	48.10s	8.2×

- 
- Round 1 (~5s): wait for 90% of all clients (461/512)
 - Round 2 (<1s): wait for 2/3 committee (40/60)
 - Round 3 (<0.2s): wait for 1/3 committee (20/60)

Armadillo 2025



Robustness: even when some clients misbehave, the server can always learn $\sum \mathbf{x}_i$ for those **valid** \mathbf{x}_i 's.

This work: L_2, L_∞ norms

A generic recipe: Proof of correct execution

- Protocol execution in round 1:
 - Client local computation
 - Send message to server
- Computation: A sequence of $+$ and \times on private and public inputs
- Throw in a zero-knowledge proof (ZKP) for that sequence
- ZKP is expensive, so we want the sequence to be “simple”

Our approach: Co-design aggregation and proof

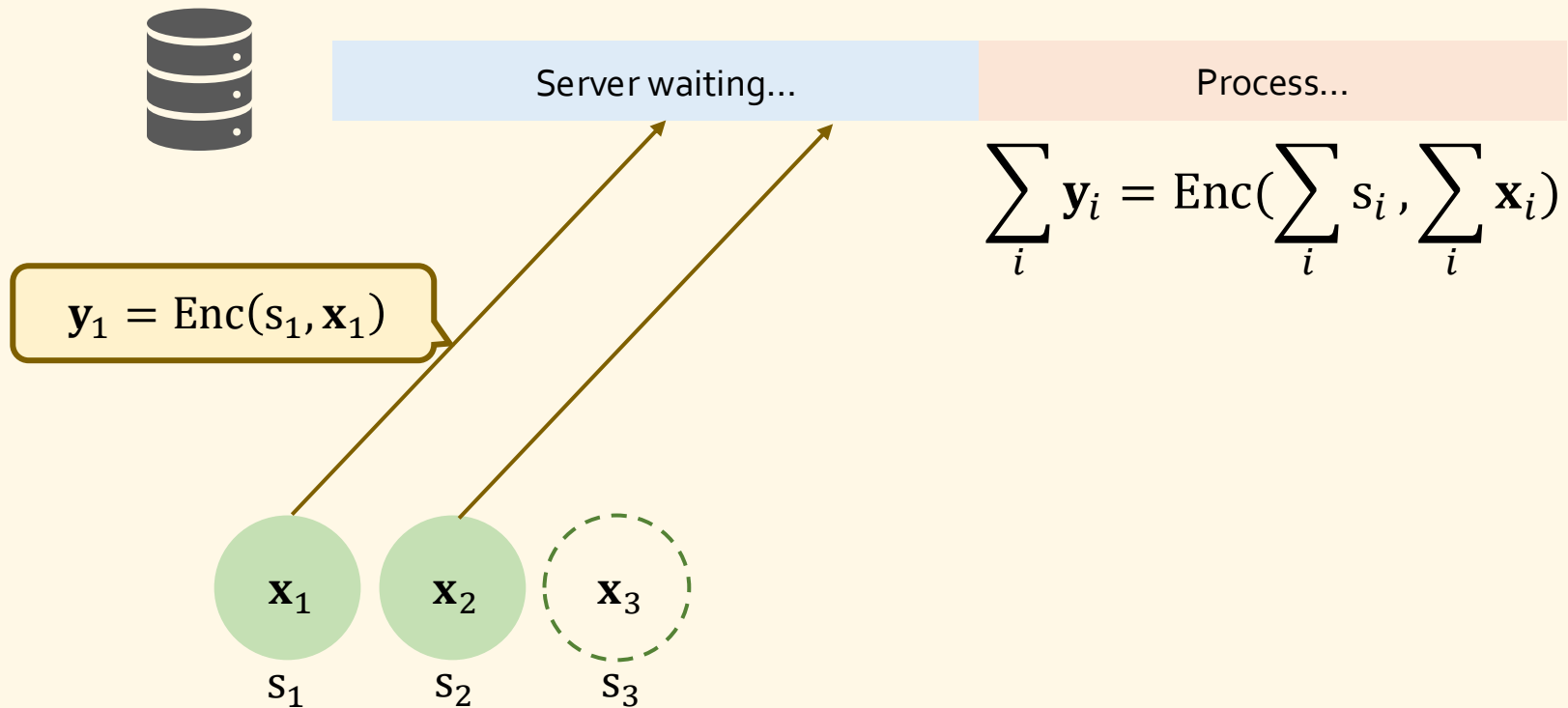
- A re-design of aggregation: **very simple arithmetic computation**
- Each client can write proof statement as **an inner-product relation**

Our approach: Co-design aggregation and proof

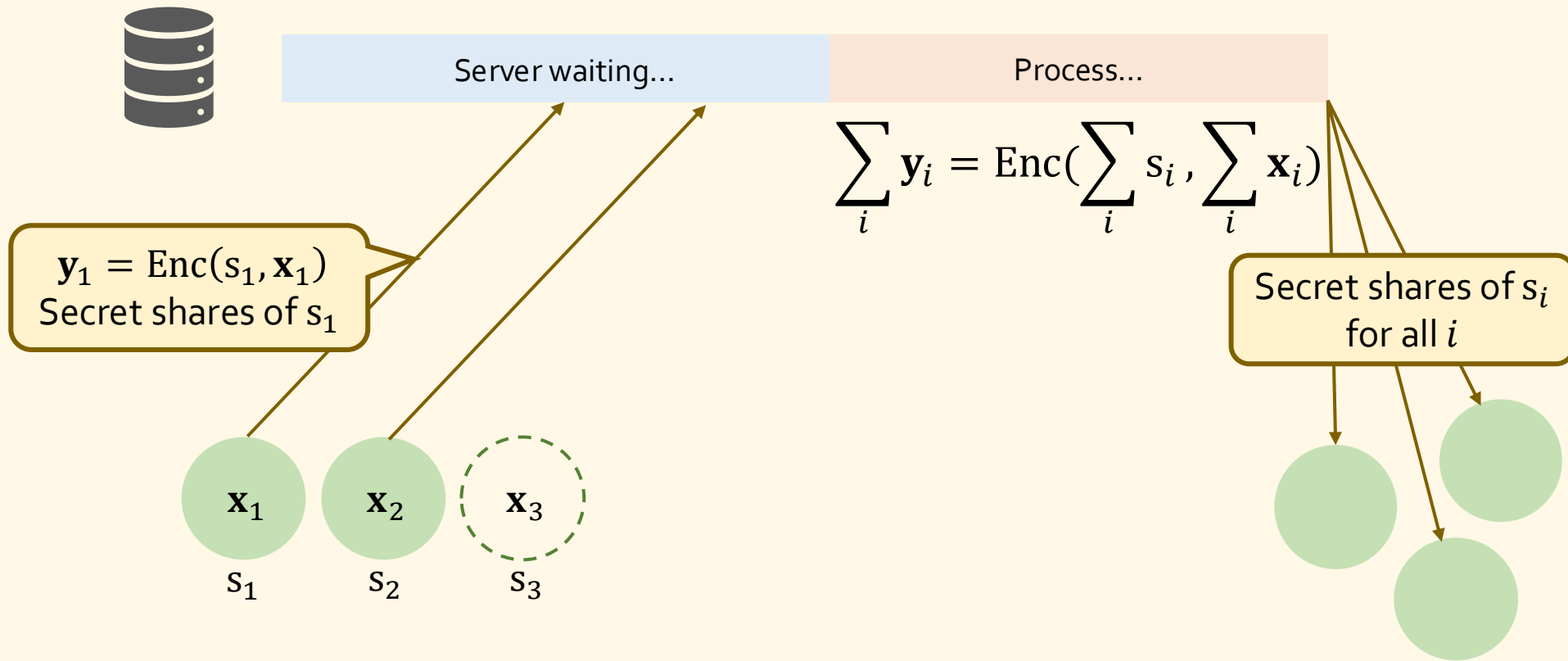
- A re-design of aggregation: **very simple arithmetic computation**
- Tool: Key-and-message-homomorphic encryption

$$\text{Enc}(s_1, \mathbf{x}_1) + \text{Enc}(s_2, \mathbf{x}_2) = \text{Enc}(s_1 + s_2, \mathbf{x}_1 + \mathbf{x}_2)$$

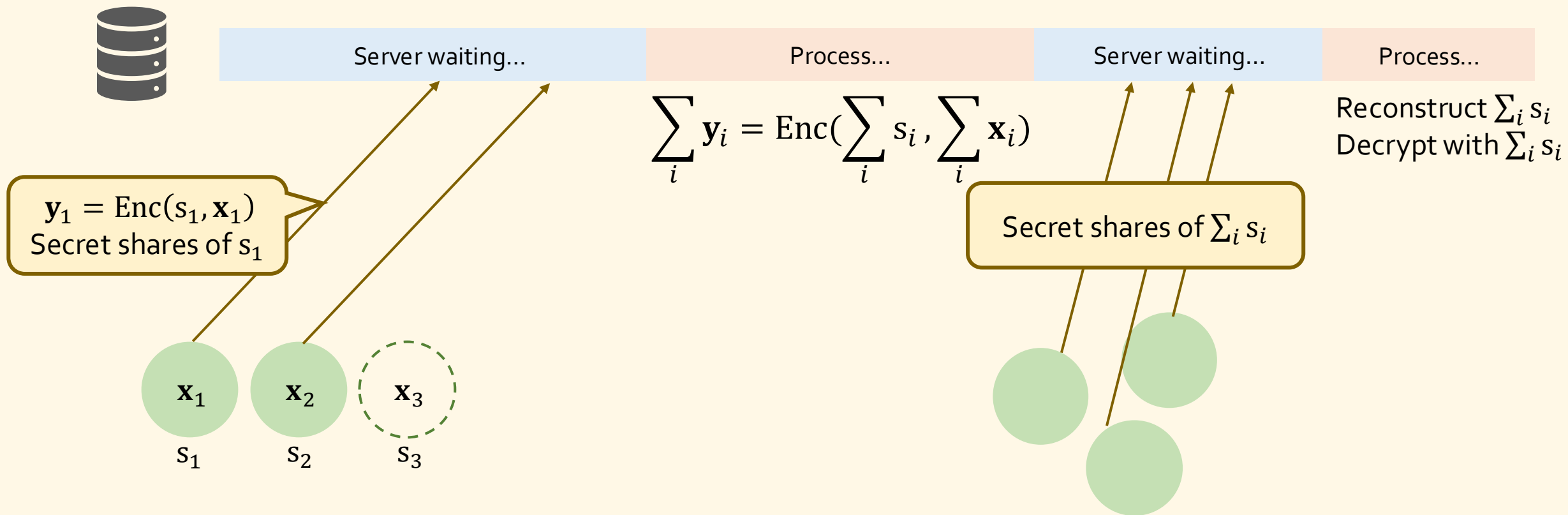
Our protocol: Two-layer aggregation



Our protocol: Two-layer aggregation



Our protocol: Two-layer aggregation



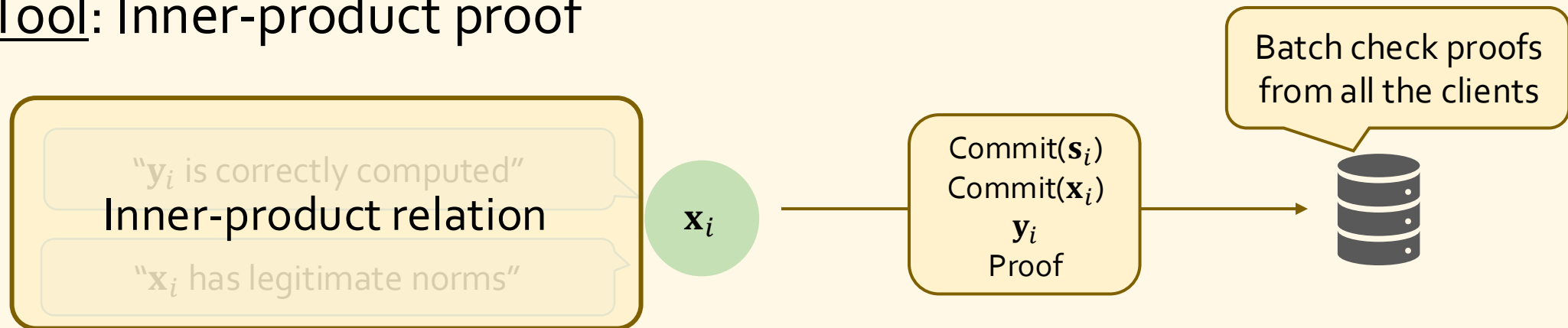
Our approach: Co-design aggregation and proof

- A re-design of aggregation: very simple arithmetic computation
- Each client can write proof statement as **an inner-product relation**
- Tool: Inner-product proof

“Given public value c , $\text{Commit}(\mathbf{a})$ and $\text{Commit}(\mathbf{b})$, prove that $\langle \mathbf{a}, \mathbf{b} \rangle = c$ ”

Our approach: Co-design aggregation and proof

- A re-design of aggregation: very simple arithmetic computation
- Each client can write proof statement as **an inner-product relation**
- Tool: Inner-product proof



The progress made so far

The progress made so far

	Dropouts	Robustness	Client comp.	Client comm.	Rounds		
Naïve MPC	✓	✗	$O(\ell \cdot n)$	$O(\ell \cdot n)$	2		
<p>And many more under the committee model: Lerna, OPA, Willow, WillowFold, ...</p>							
Flamingo 2023	✓	✗	$O(\ell + \text{polylog } n)$	$O(n)$	$O(\ell + \text{polylog } n)$	$O(n)$	1 + 2
Armadillo 2025	✓	✓	Same asymptotics as Flamingo				

The progress made so far

	Dropouts	Robustness	Client comp.	Client comm.	Rounds		
Naïve MPC	✓	✗	$O(\ell \cdot n)$	$O(\ell \cdot n)$	2		
<p>And many more under the committee model: Lerna, OPA, Willow, WillowFold, ...</p>							
Flamingo 2023	✓	✗	$O(\ell + \text{polylog } n)$	$O(n)$	$O(\ell + \text{polylog } n)$	$O(n)$	1 + 2
Armadillo	A new result in batch threshold decryption		Same asymptotics as Flamingo				
Lighthouse 2026	✓	✗	$O(\ell + \text{polylog } n)$	$O(1)$	$O(\ell + \text{polylog } n)$	$O(1)$	1 + 2

Open questions

- Research direction 1: How small can the committee cost be?
- Research direction 2: What if the adversary is adaptive?
- Research direction 3: Verification for inputs, how “sufficient” can norms be?
- Research direction 4: Richer functionality class
 - GMPC by Bar Alon, Moni Naor, Eran Omri, Uri Stemmer

Thank you!