MPC With Delayed Parties Over Star-Like Networks

Mariana Gama
KU Leuven

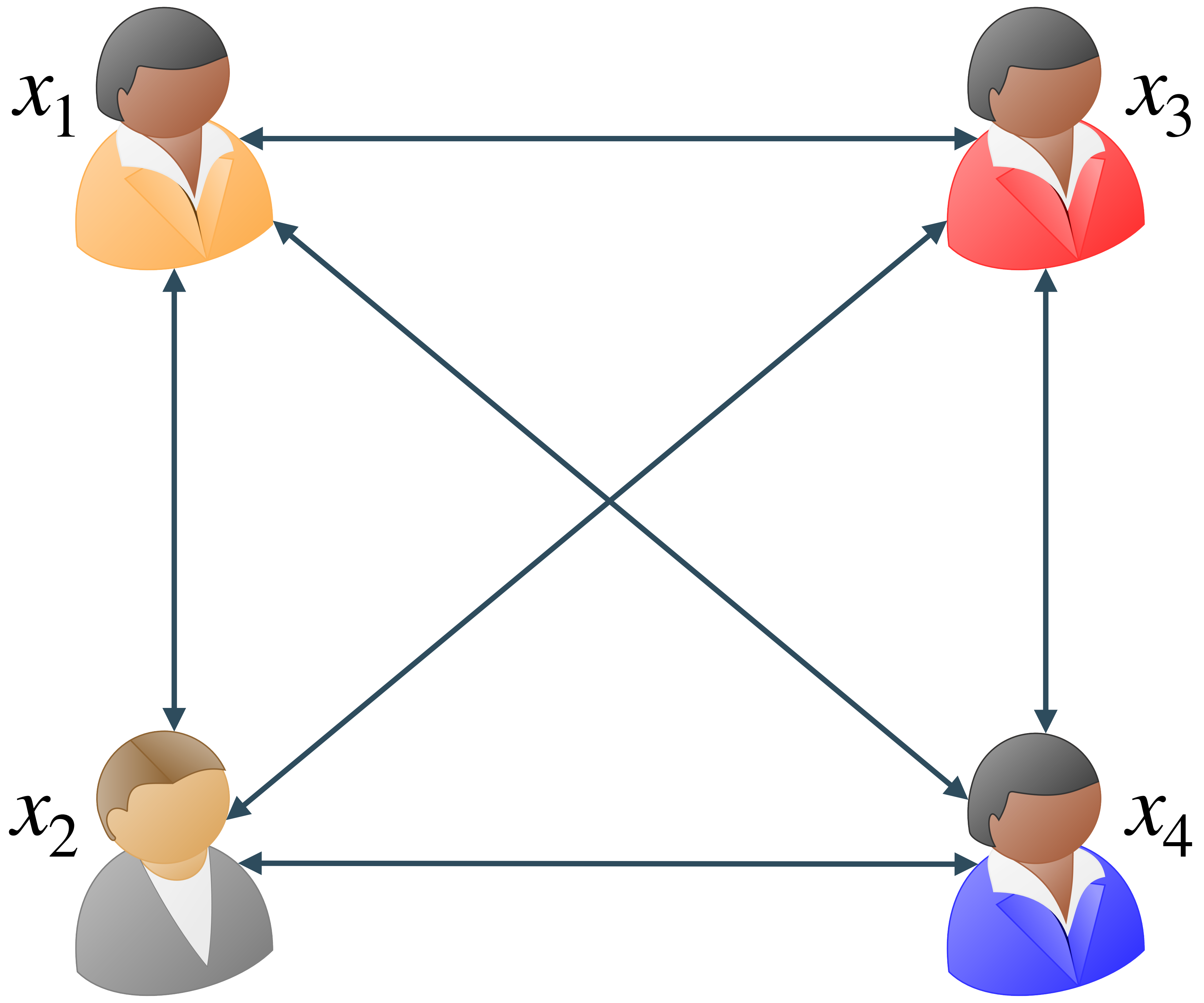
Emad Heydari Beni
KU Leuven
Nokia Bell Labs

Emmanuela Orsini
Uni. Bocconi

Nigel Smart
KU Leuven
Zama

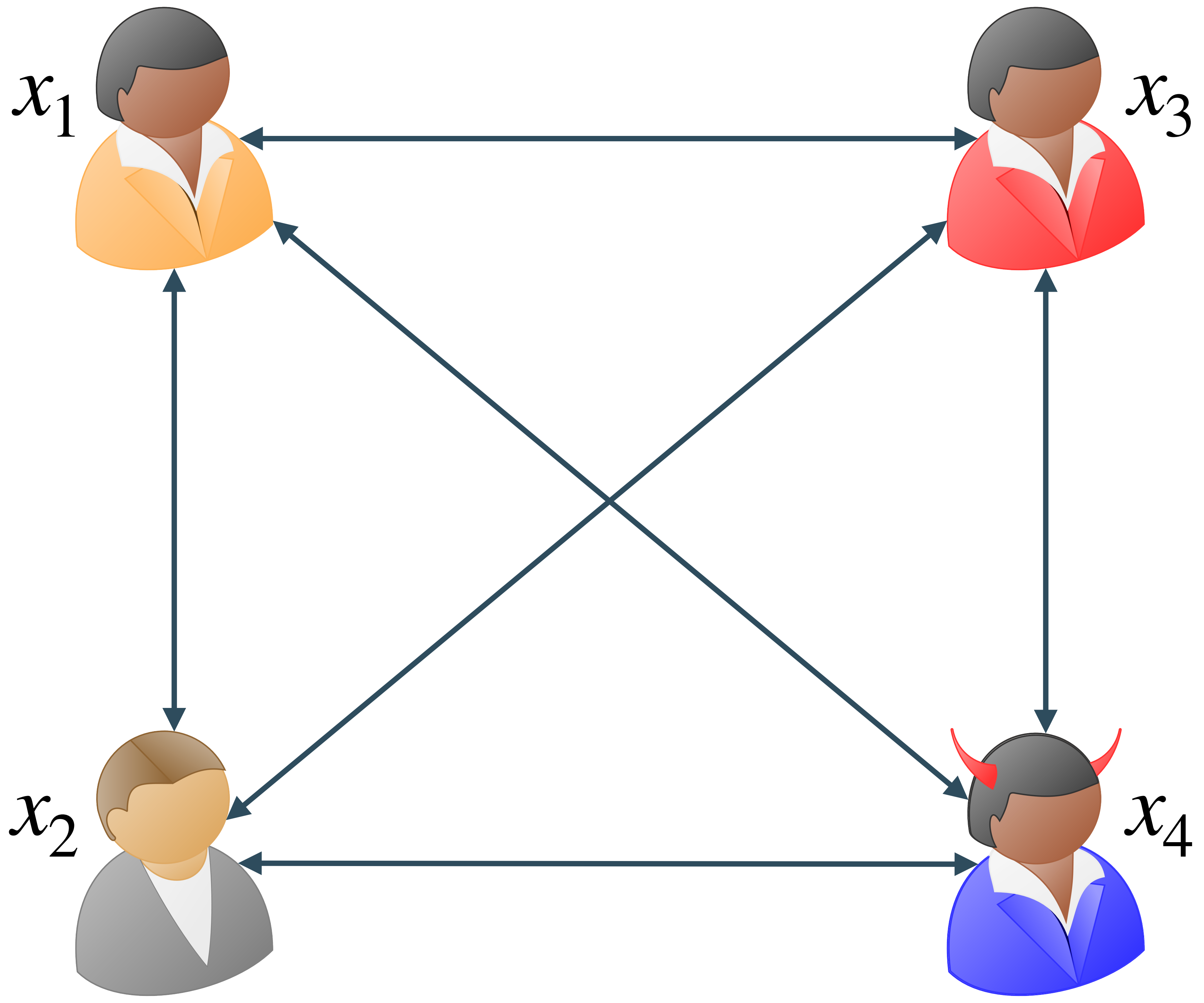
Oliver Zajonc
KU Leuven

Multiparty Computation



- Technique for computing over encrypted data.
- Achieves privacy by distributing the computation.

Multiparty Computation

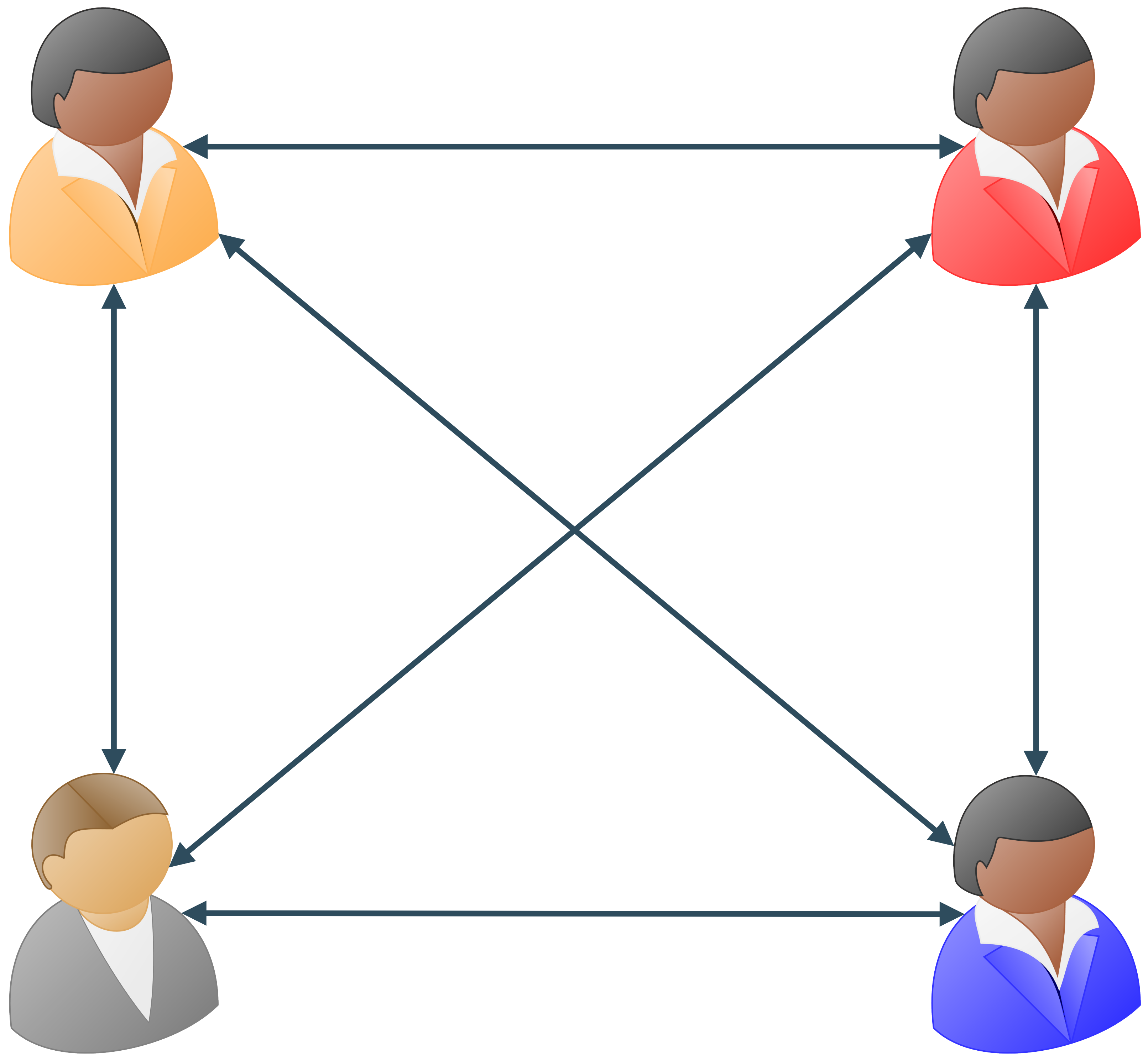


- Technique for computing over encrypted data.
- Achieves privacy by distributing the computation.

Adversary corrupting a percentage of the parties will still learn nothing but the output,

$$y = f(x_1, x_2, x_3, x_4)$$

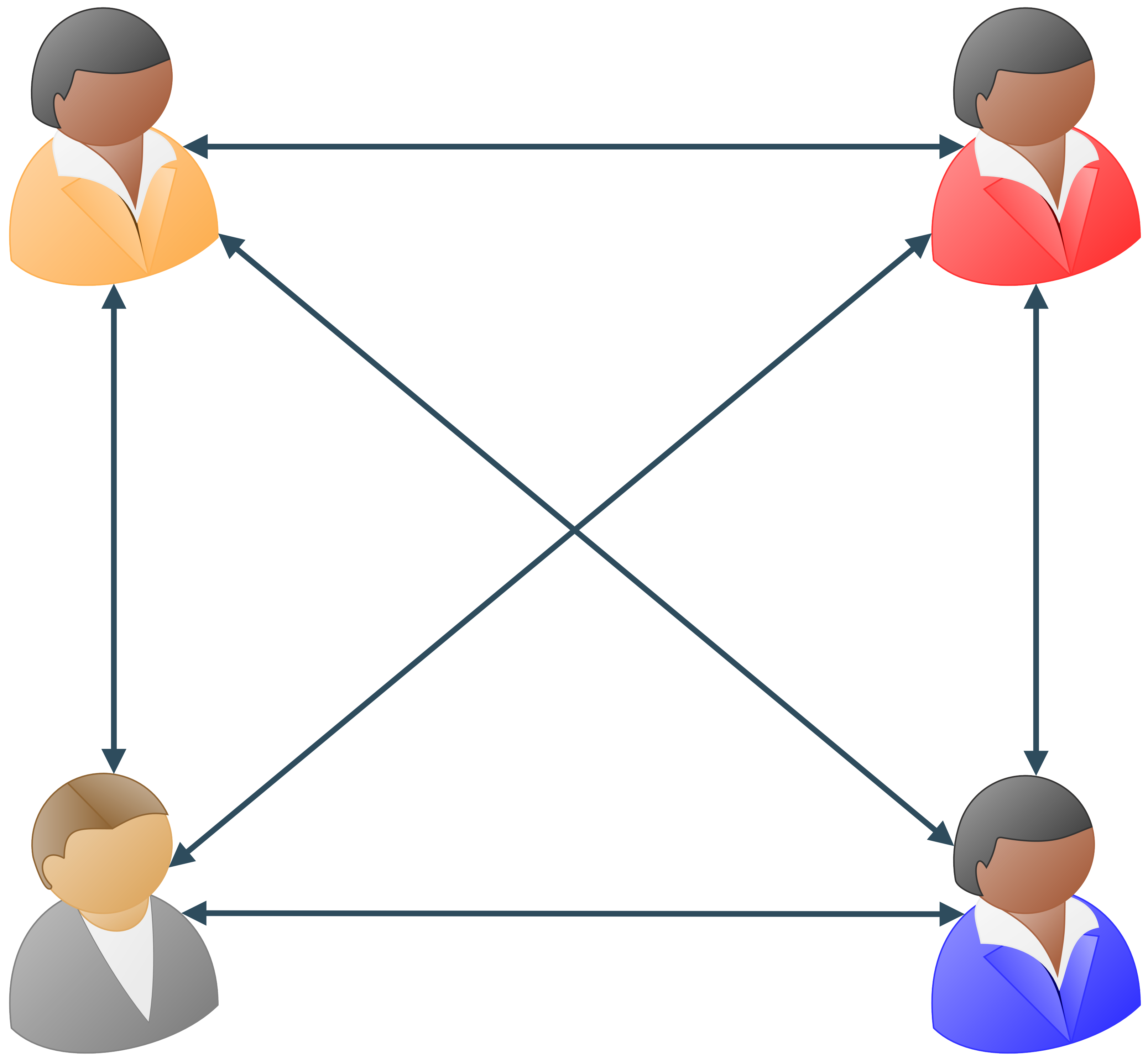
Multiparty Computation



Common assumptions

- Communication channels are direct and fast.

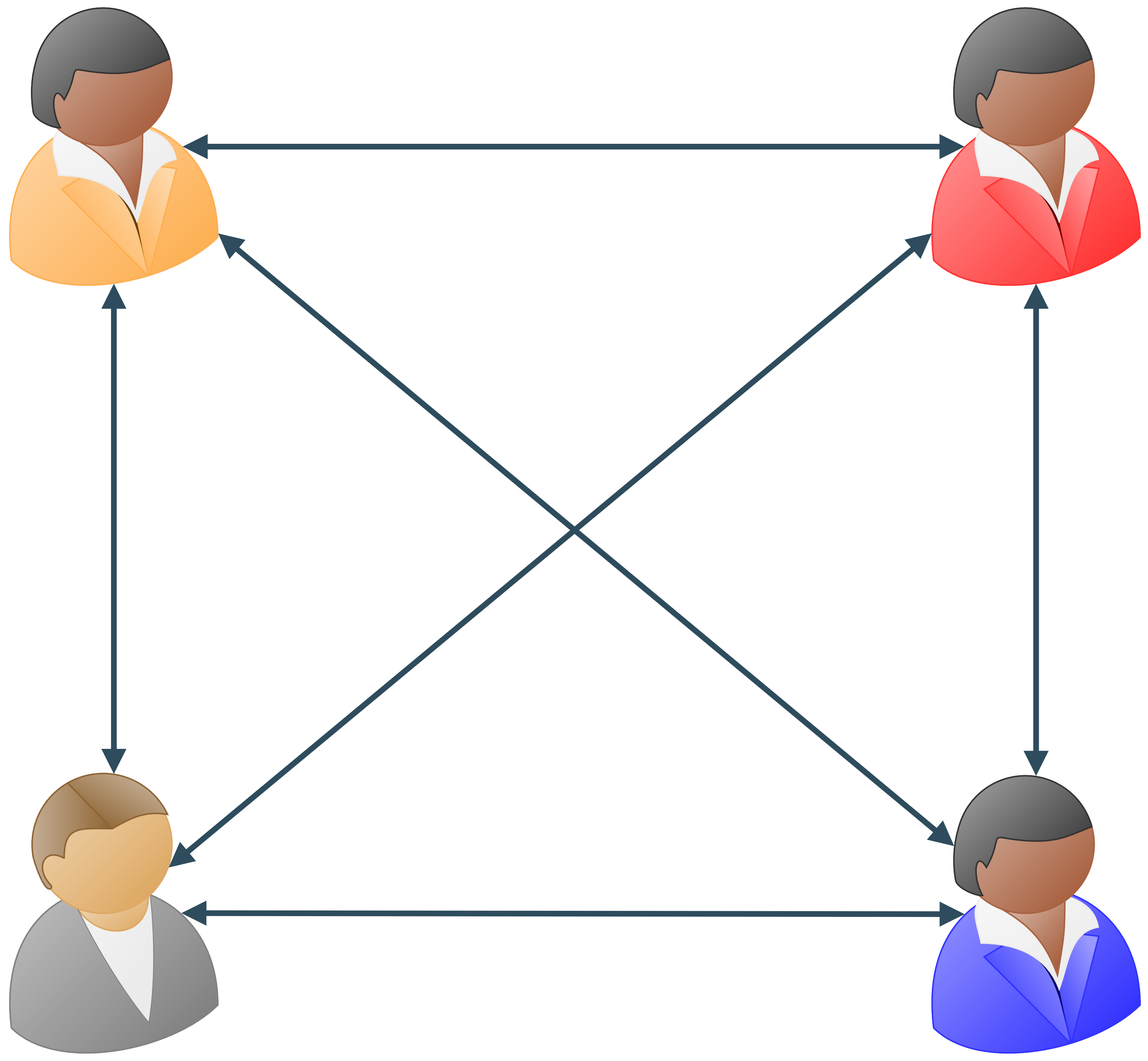
Multiparty Computation



Common assumptions

- Communication channels are direct and fast.
- Parties stay online during the whole computation.

Multiparty Computation

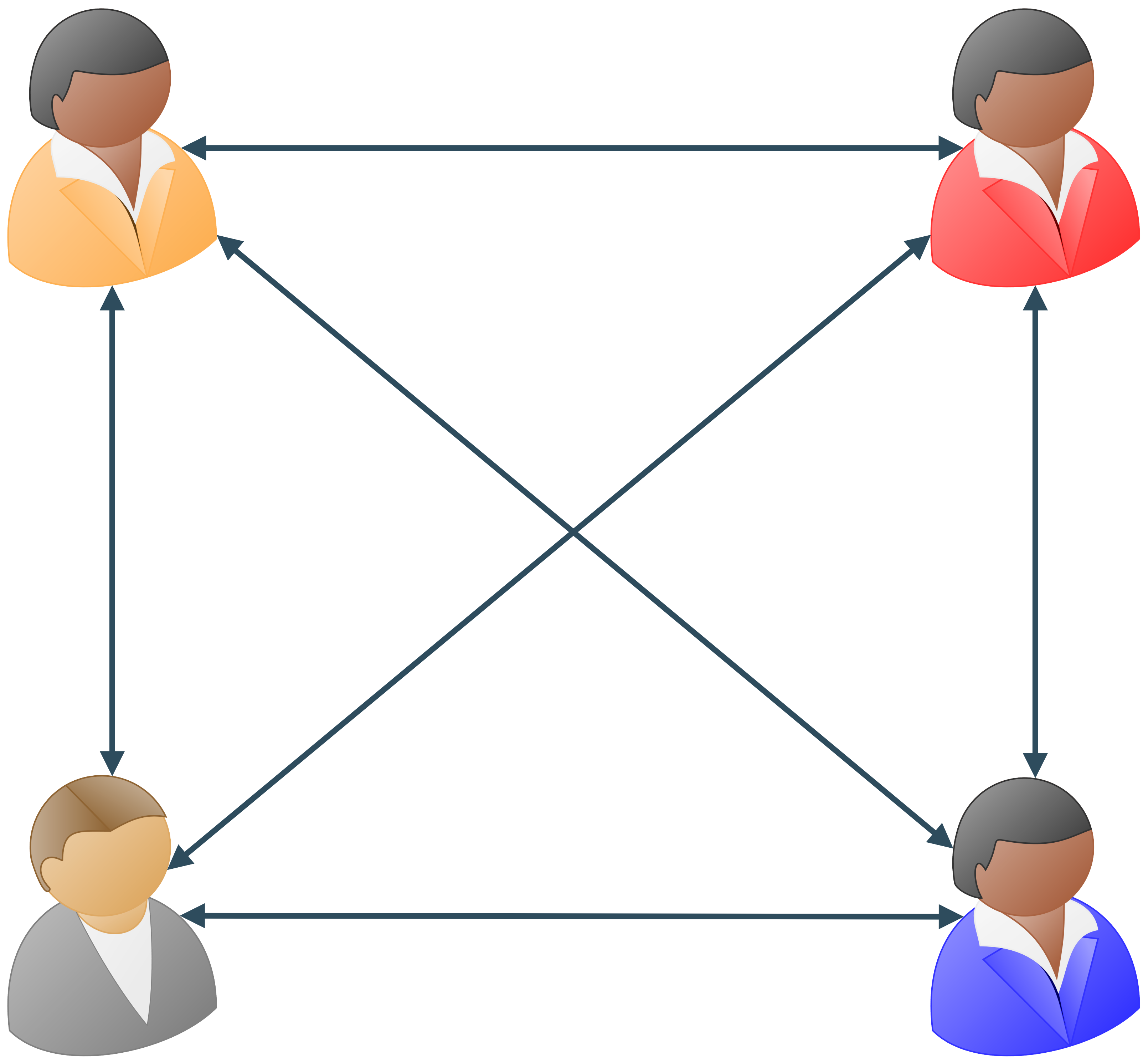


Common assumptions

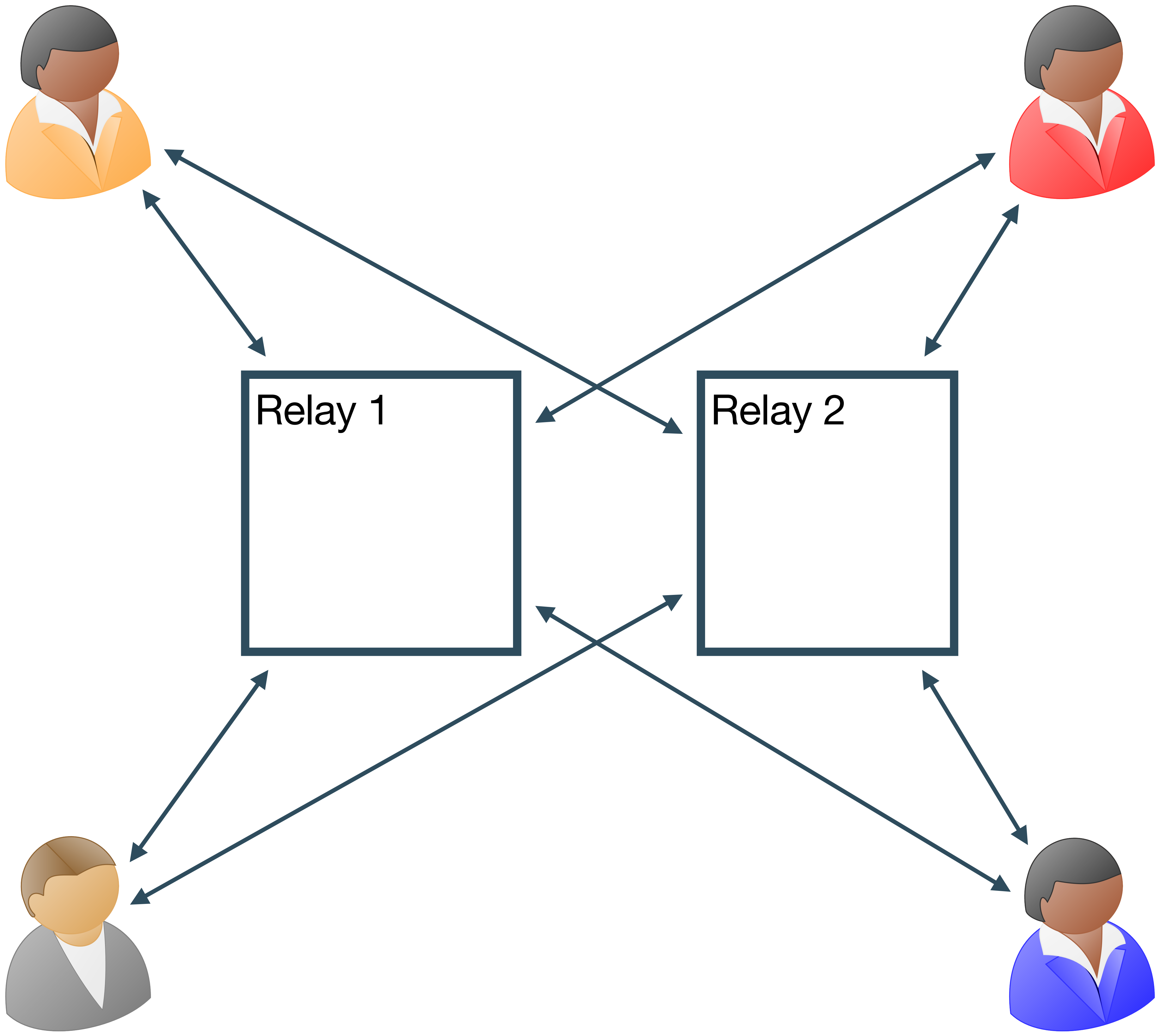
- Communication channels are direct and fast.
- Parties stay online during the whole computation.

Not the case in currently deployed systems!

This work

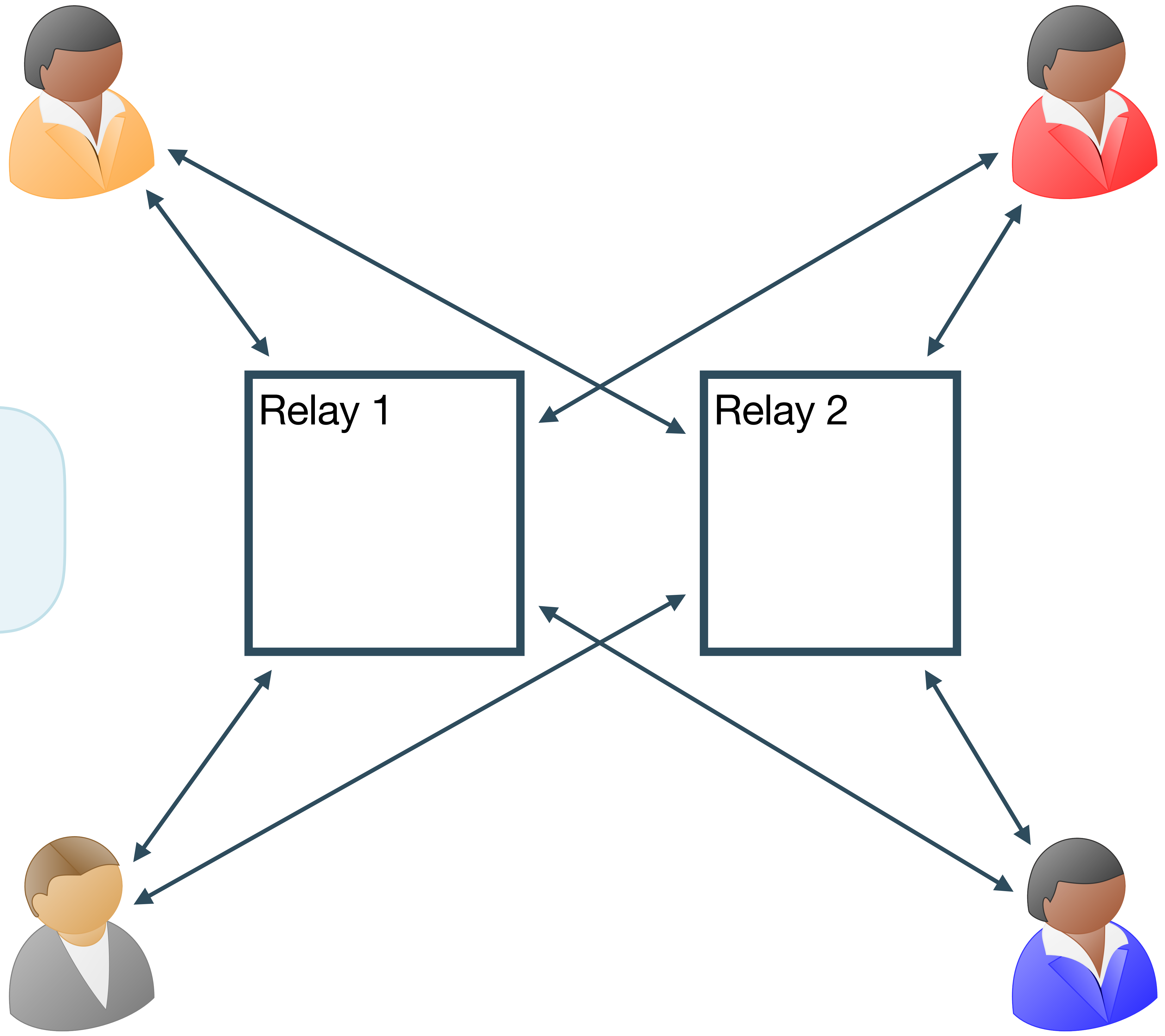


This work



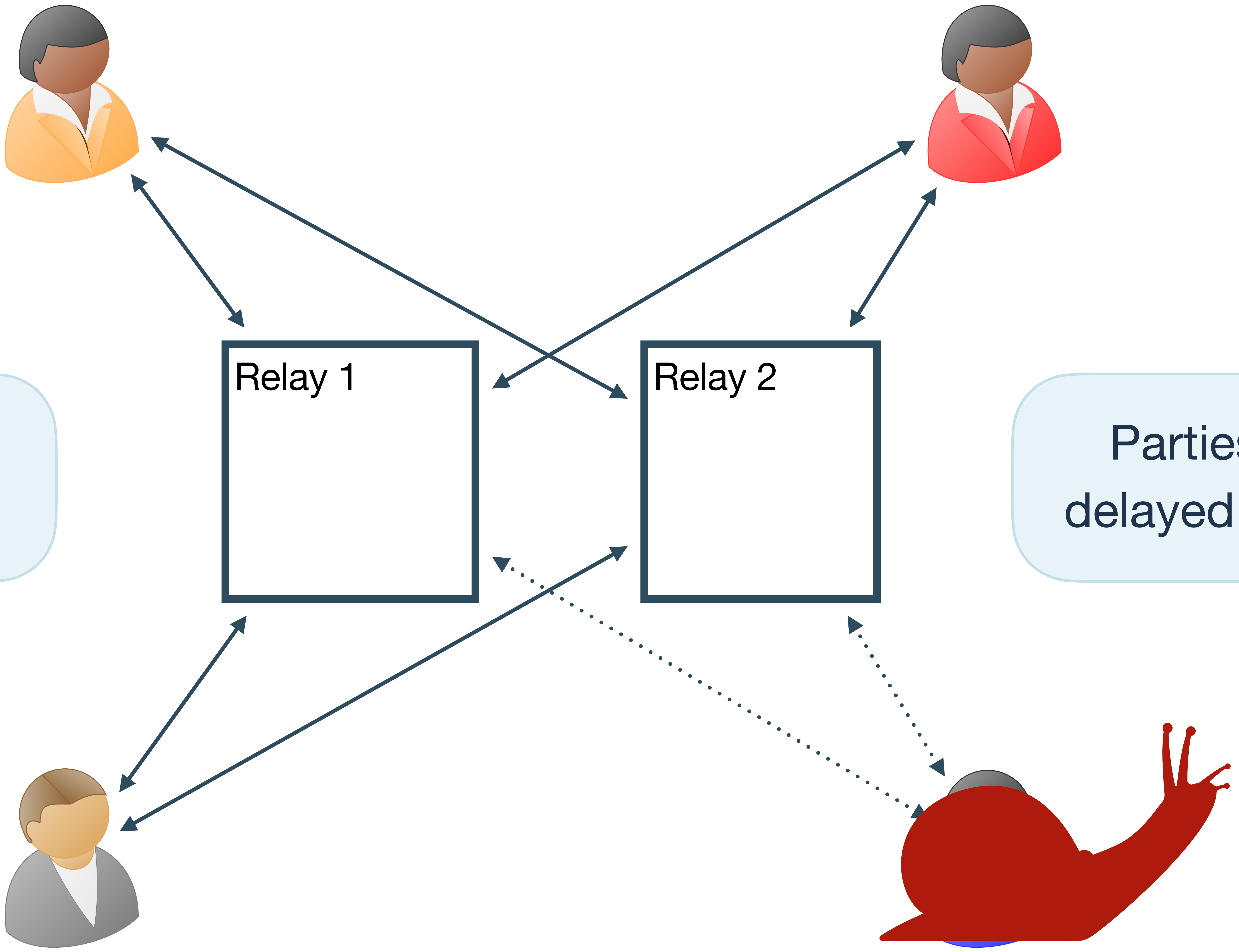
This work

Star-like communication topology with relay nodes



This work

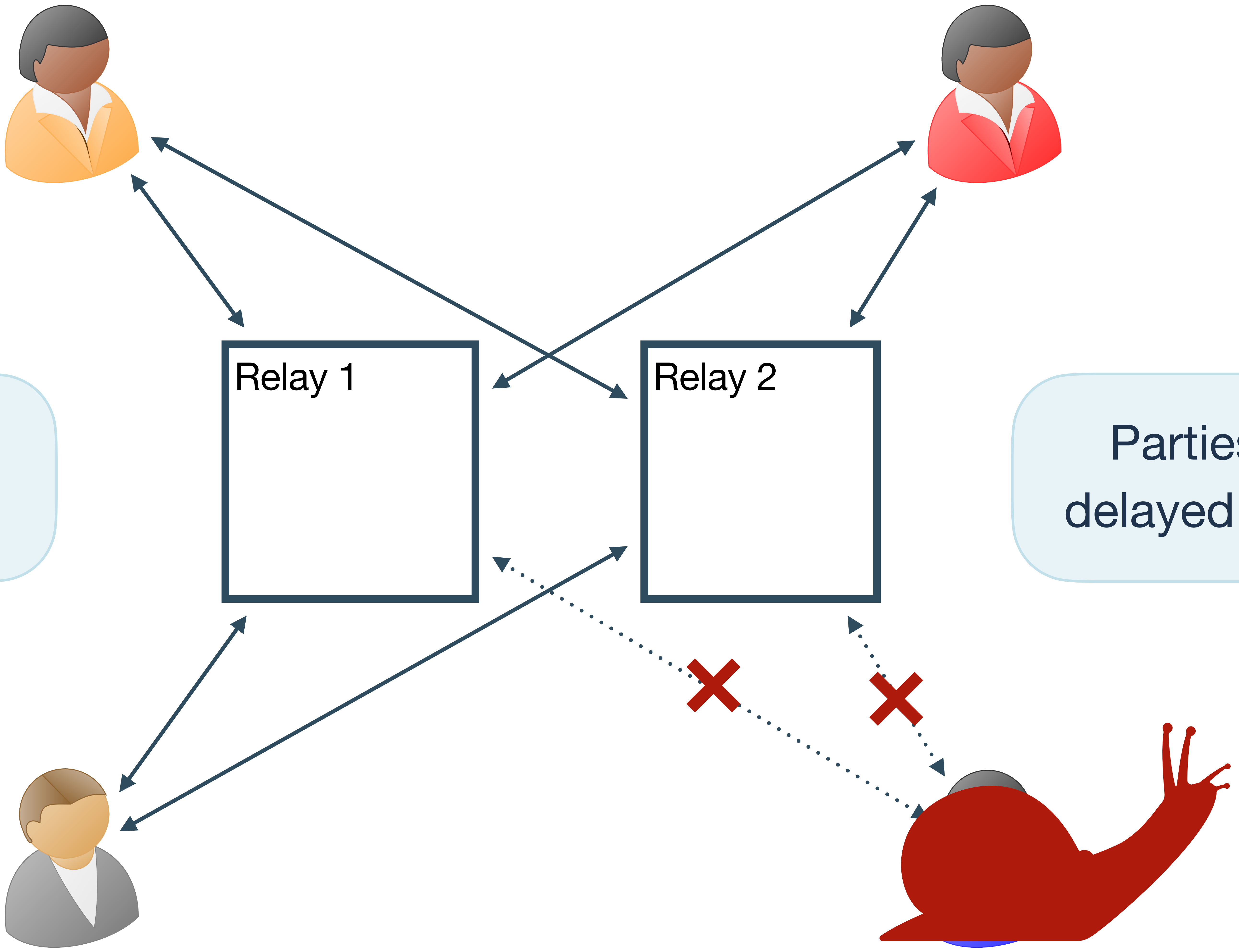
Star-like communication topology with relay nodes



Parties' actions can be delayed for up to δ rounds.

This work

Star-like communication topology with relay nodes



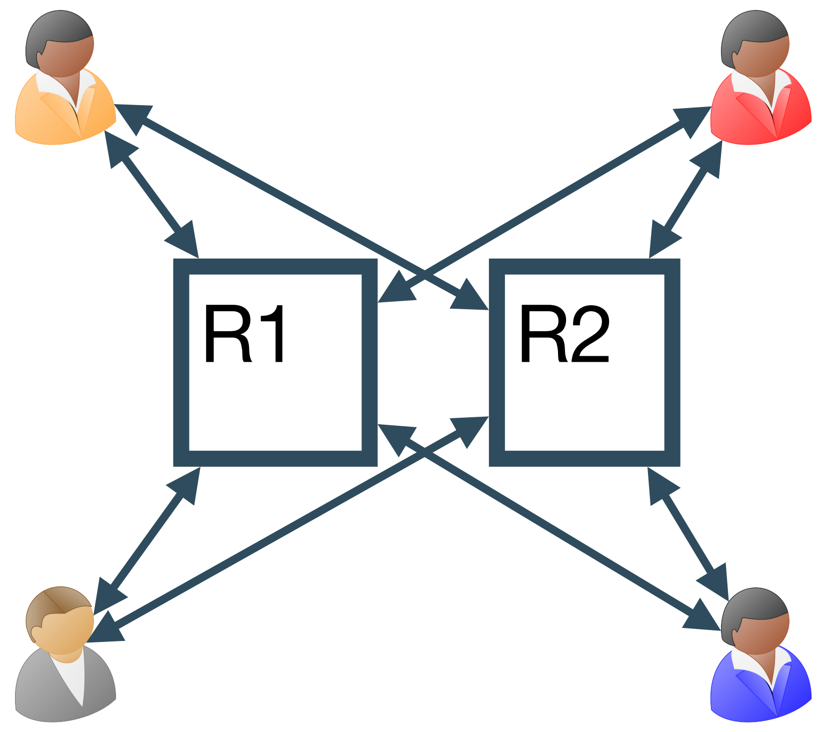
Parties' actions can be delayed for up to δ rounds.

Related work

Related work

Star-like topology

White-City: A framework for massive mpc with partial synchrony and partially authenticated channels
ZenGo technical report, 2020

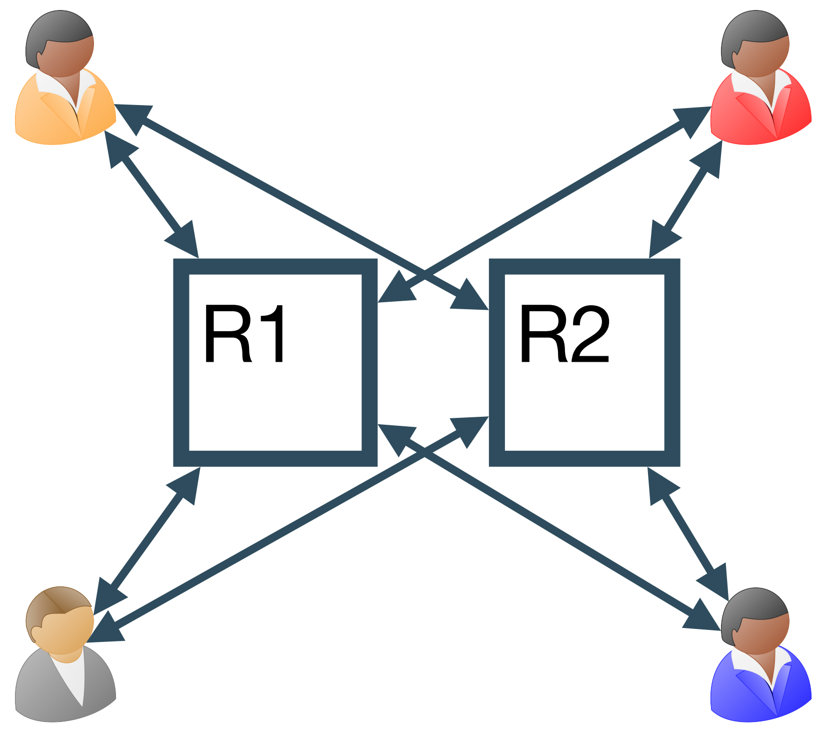


- Relays maintain consistency via a consensus protocol.
- Designed for threshold ECDSA signing -> no mechanism to limit the number of stored messages.

Related work

Star-like topology

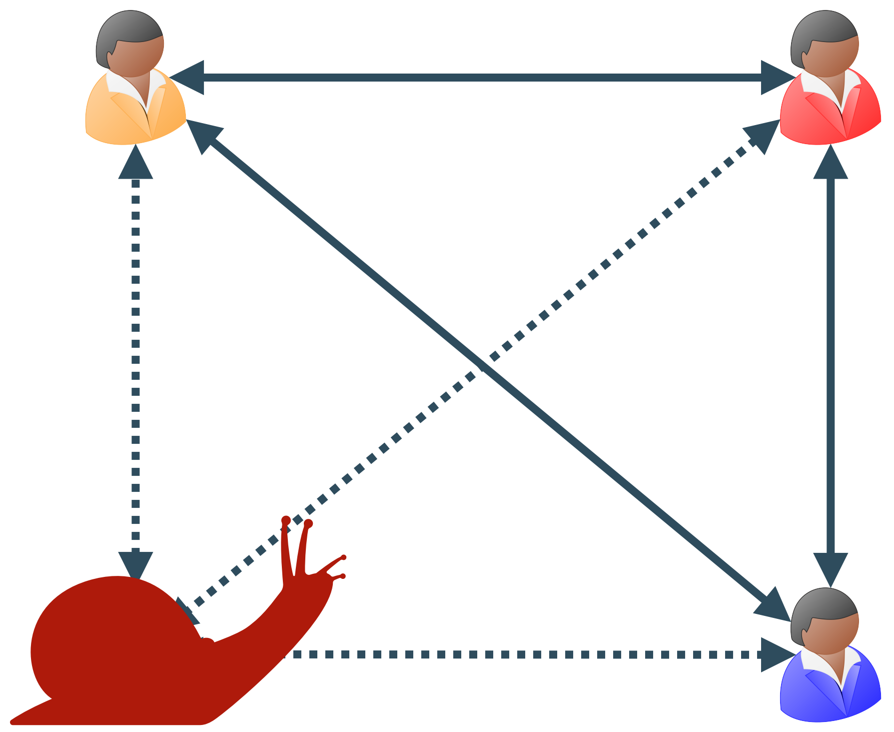
White-City: A framework for massive mpc with partial synchrony and partially authenticated channels
ZenGo technical report, 2020



- Relays maintain consistency via a consensus protocol.
- Designed for threshold ECDSA signing -> no mechanism to limit the number of stored messages.

MPC with delays

Generalized pseudorandom secret sharing and efficient straggler-resilient secure computation
Benhamouda et al. [BBG+21]

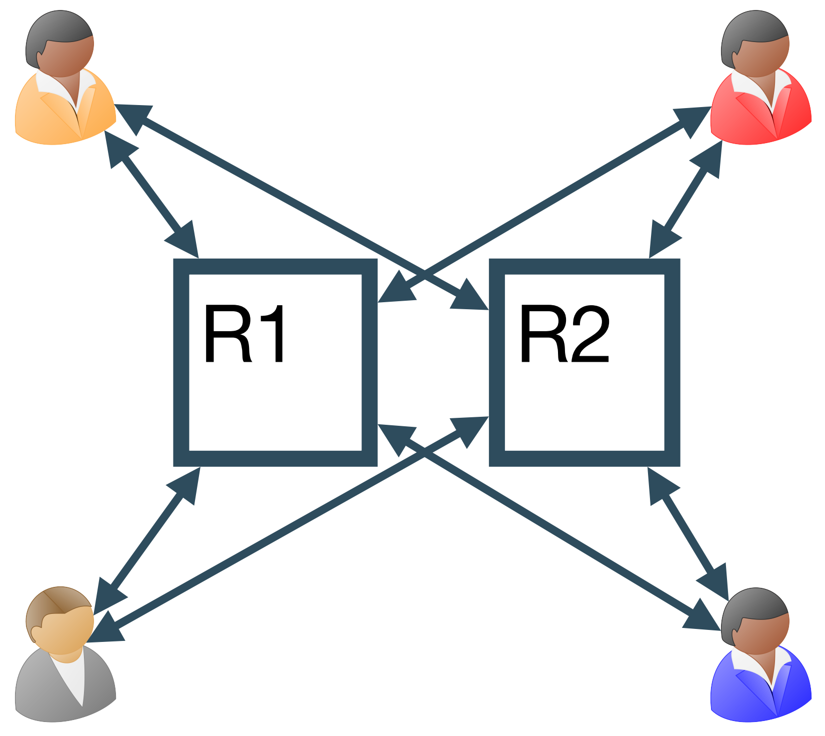


- Delays are caused by network channels instead of node failures
- Multiplication protocol introduces additional overhead.

Related work

Star-like topology

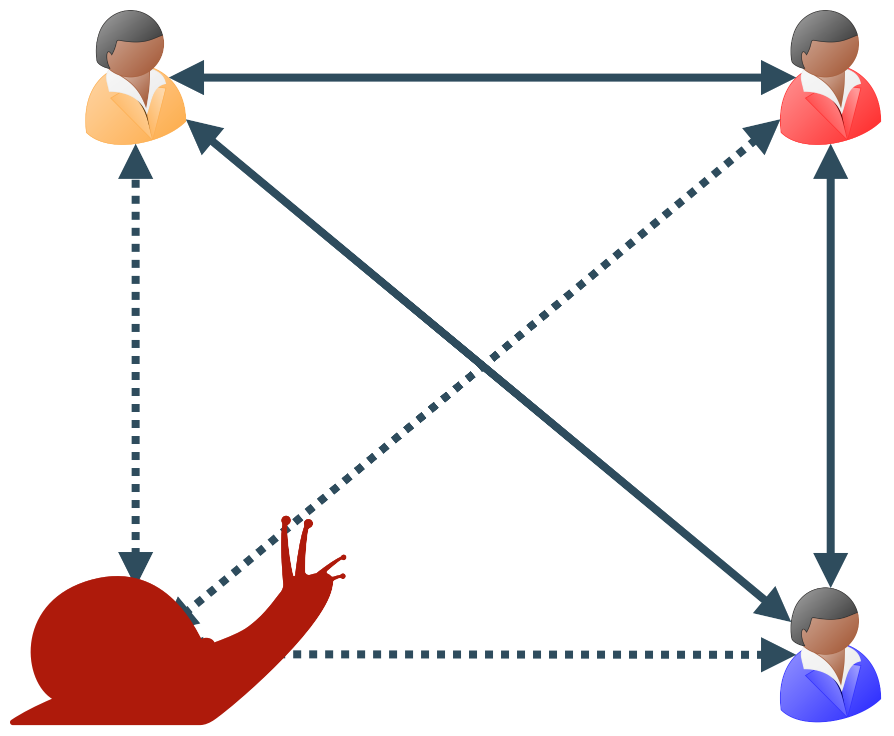
White-City: A framework for massive mpc with partial synchrony and partially authenticated channels
ZenGo technical report, 2020



- Relays maintain consistency via a consensus protocol.
- Designed for threshold ECDSA signing -> no mechanism to limit the number of stored messages.

MPC with delays

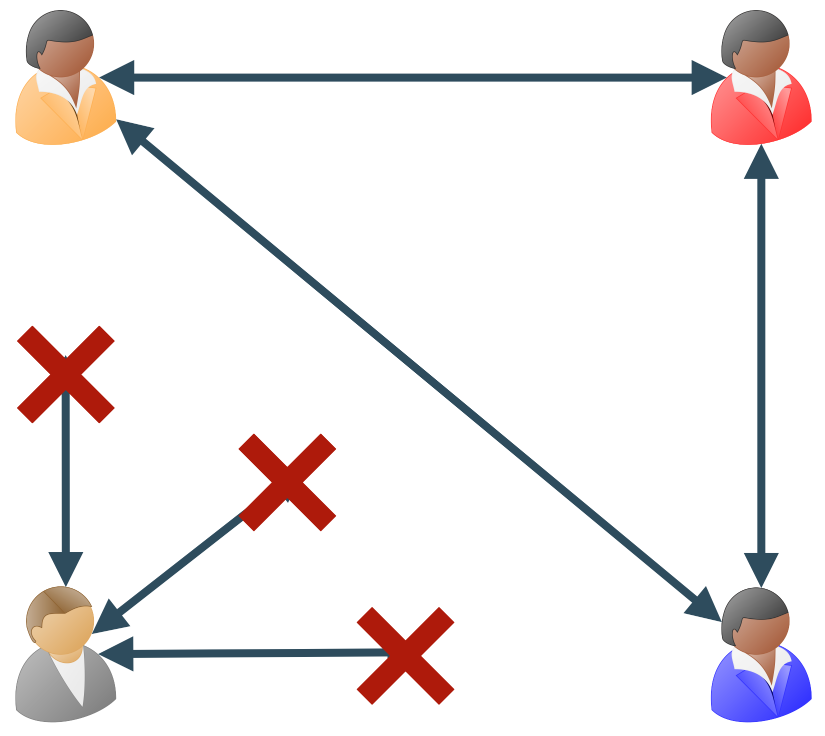
Generalized pseudorandom secret sharing and efficient straggler-resilient secure computation
Benhamouda et al. [BBG+21]



- Delays are caused by network channels instead of node failures
- Multiplication protocol introduces additional overhead.

Dynamic participation

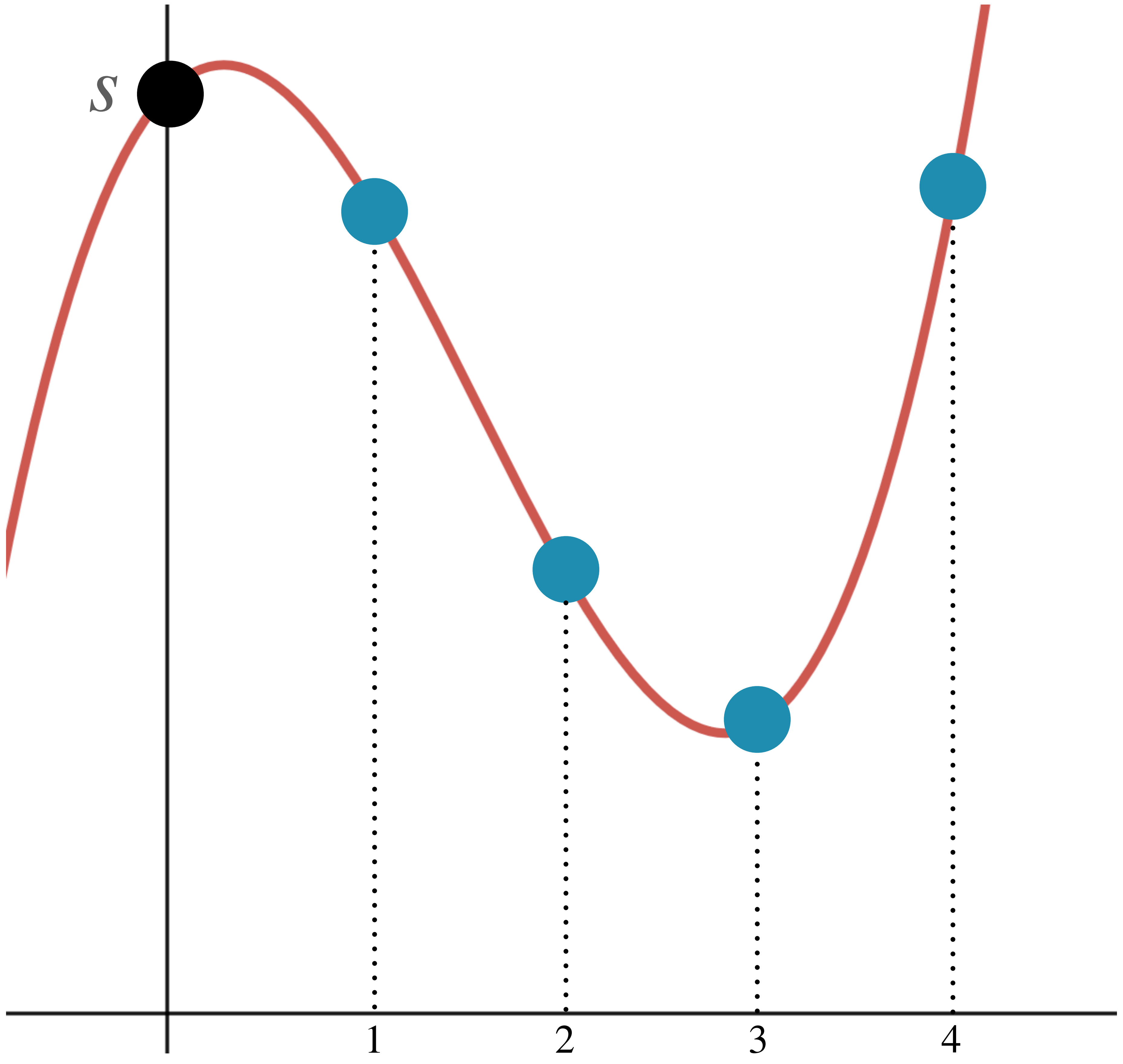
Phoenix: Secure computation in an unstable network with dropouts and comebacks
I. Damgård, D. Escudero, A. Polychroniadou, 2021

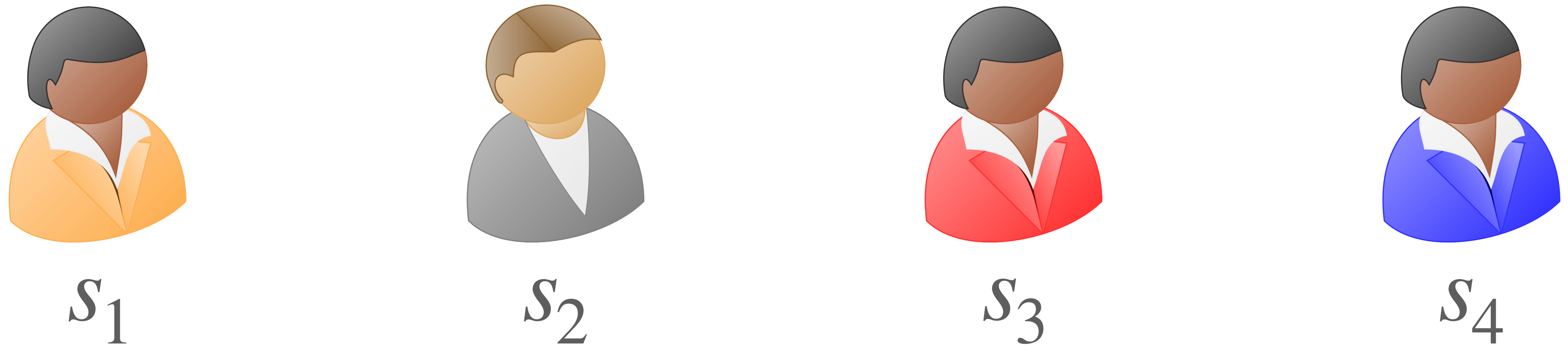


- Parties who dropout are not assumed to receive messages sent while they were offline.
- Requires a certain number of parties to be online from one round to the next one.

Multiparty Computation

Shamir secret sharing





Sharing a secret:

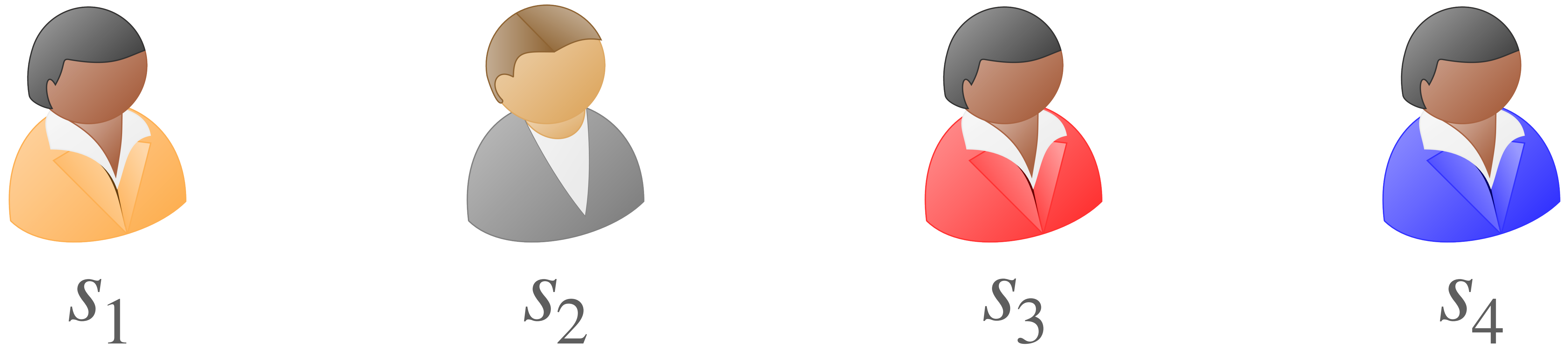
- Sample degree t polynomial such that $p(0) = s$.
- Evaluate $p(x)$ at n public points.
- Give $p(i) = s_i$ to party i.

Reconstructing a secret:

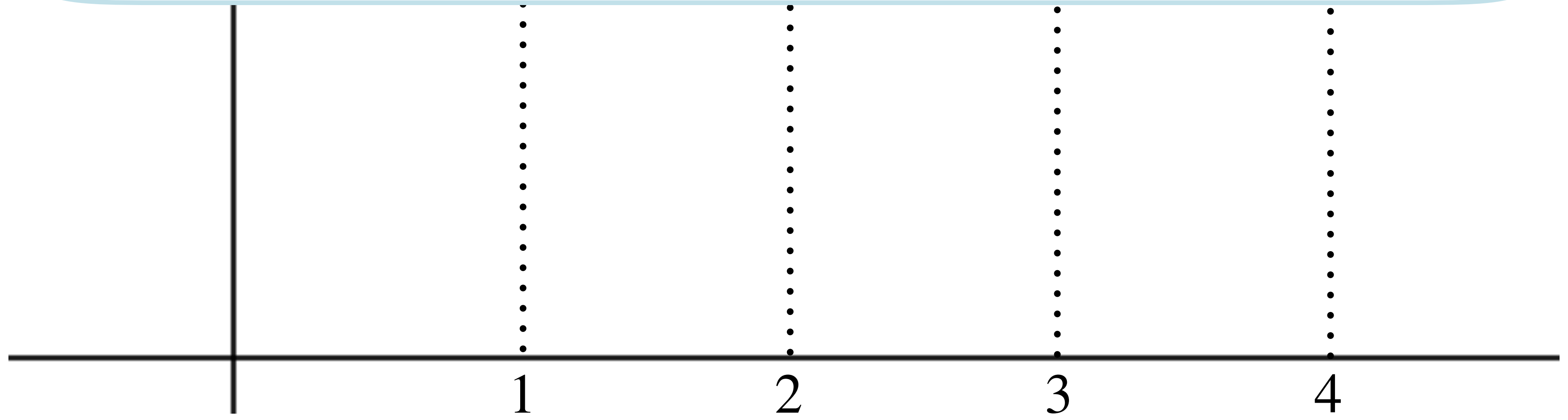
- Parties reveal their shares and reconstruct the polynomial.

Multiparty Computation

Shamir secret sharing



Can have at most t corrupt parties!



Sharing a secret:

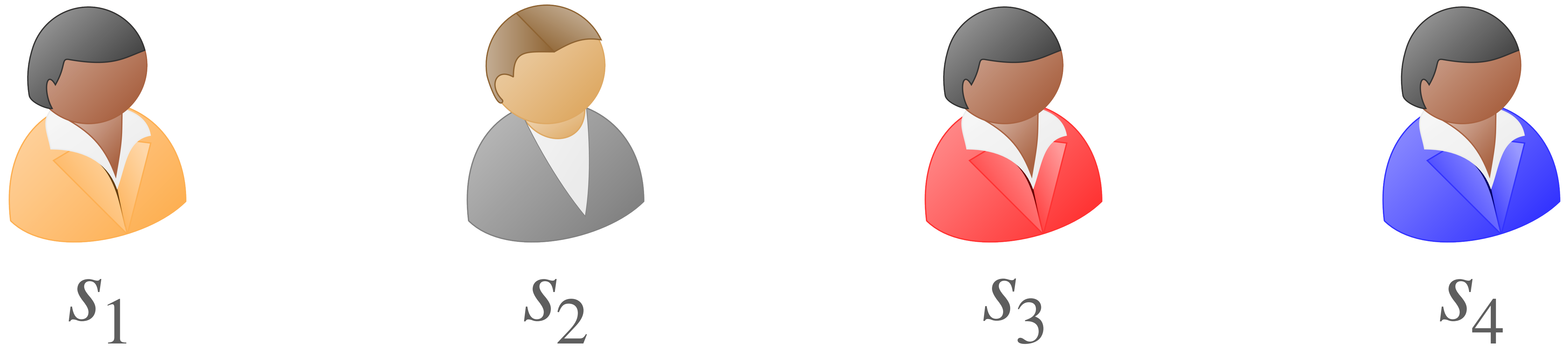
- Sample degree t polynomial such that $p(0) = s$.
- Evaluate $p(x)$ at n public points.
- Give $p(i) = s_i$ to party i.

Reconstructing a secret:

- Parties reveal their shares and reconstruct the polynomial.

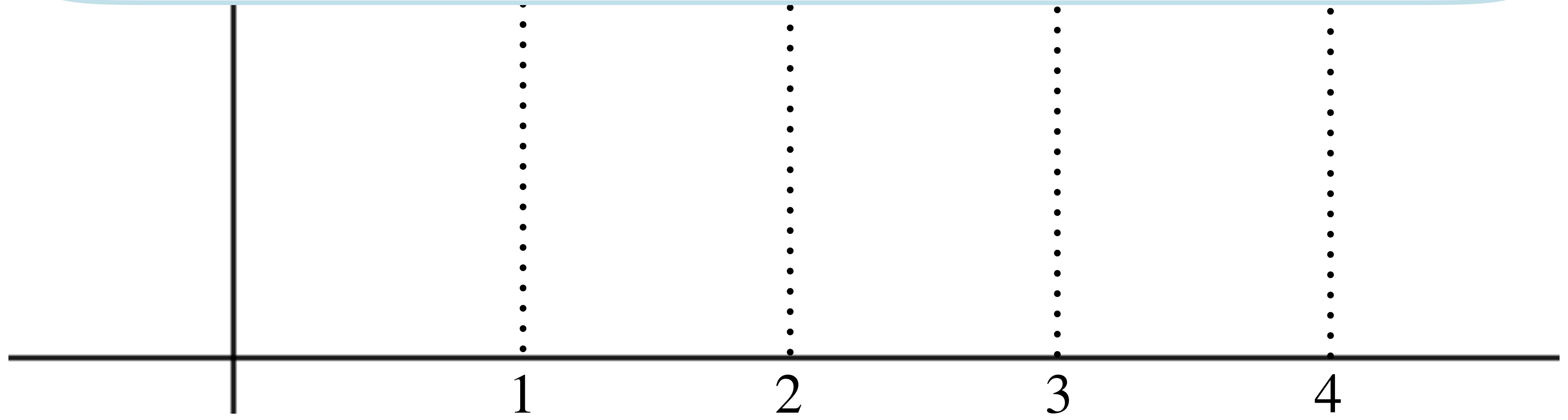
Multiparty Computation

Shamir secret sharing



Can have at most t corrupt parties!

Honest majority: $n \geq 2t + 1$



Sharing a secret:

- Sample degree t polynomial such that $p(0) = s$.
- Evaluate $p(x)$ at n public points.
- Give $p(i) = s_i$ to party i.

Reconstructing a secret:

- Parties reveal their shares and reconstruct the polynomial.

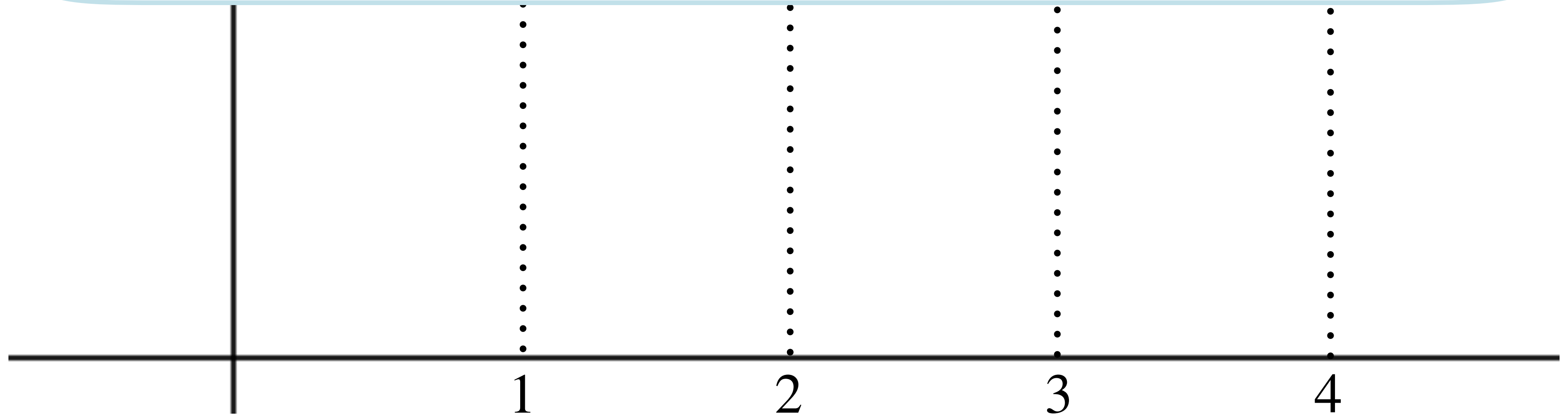
Multiparty Computation

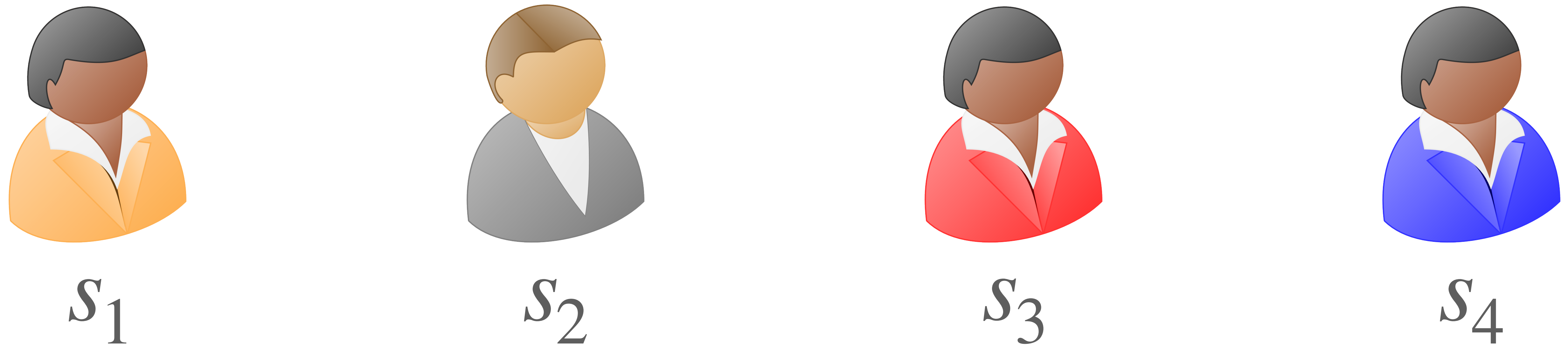
Shamir secret sharing

Can have at most t corrupt parties!

Honest majority: $n \geq 2t + 1$

Strong honest majority: $n > 2t + 1$





Sharing a secret:

- Sample degree t polynomial such that $p(0) = s$.
- Evaluate $p(x)$ at n public points.
- Give $p(i) = s_i$ to party i.

Reconstructing a secret:

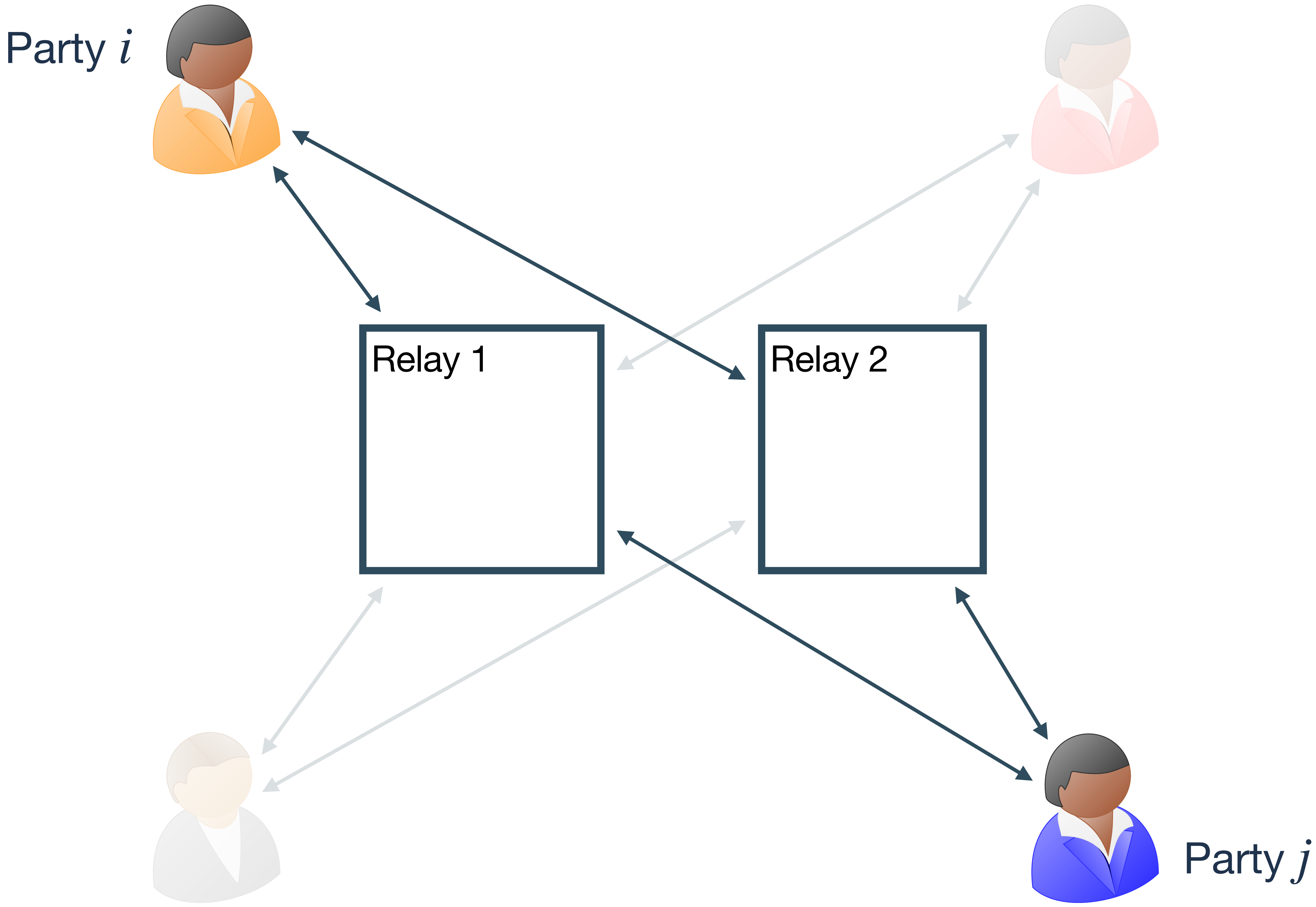
- Parties reveal their shares and reconstruct the polynomial.

The relays

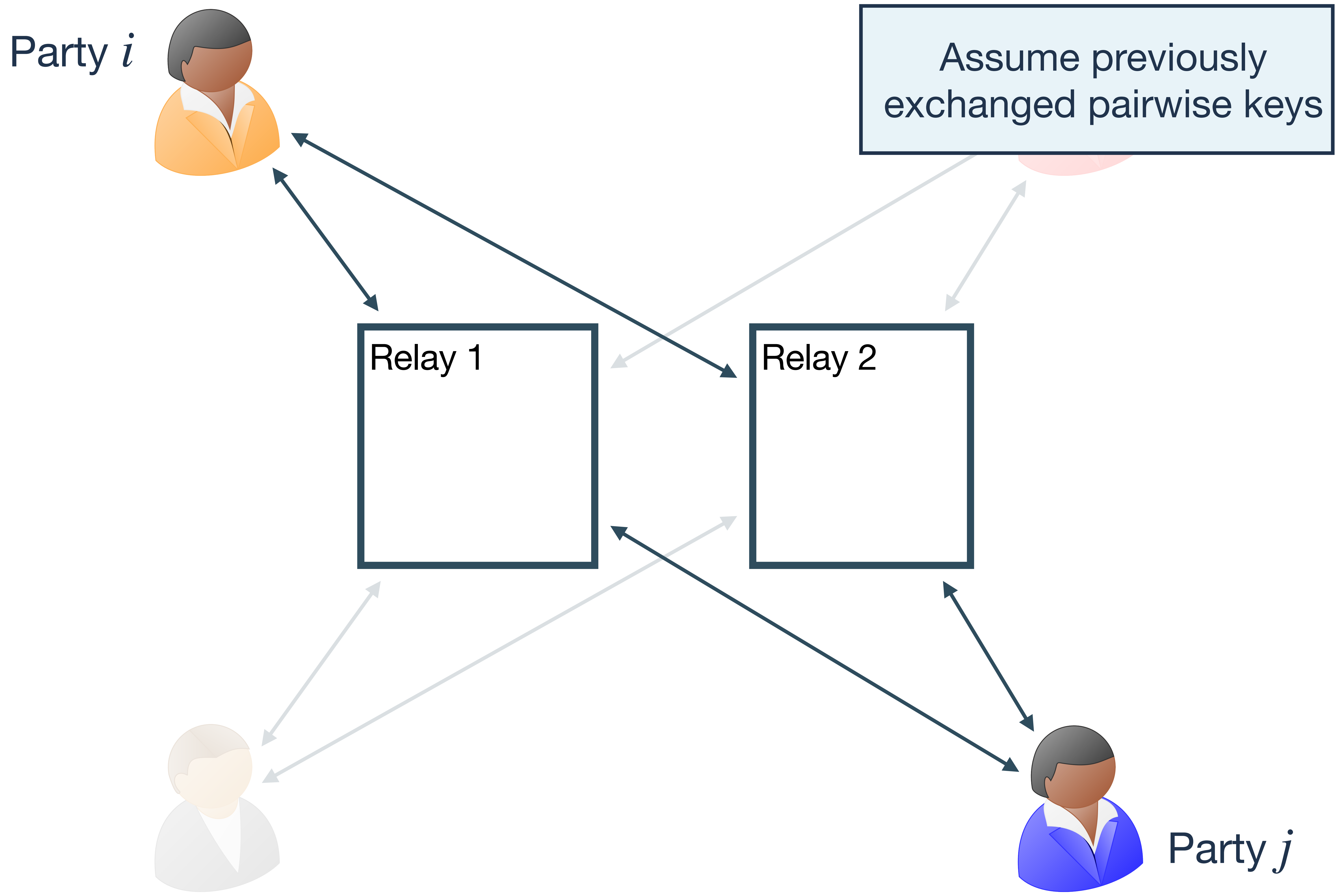
p2p messages

Broadcast messages

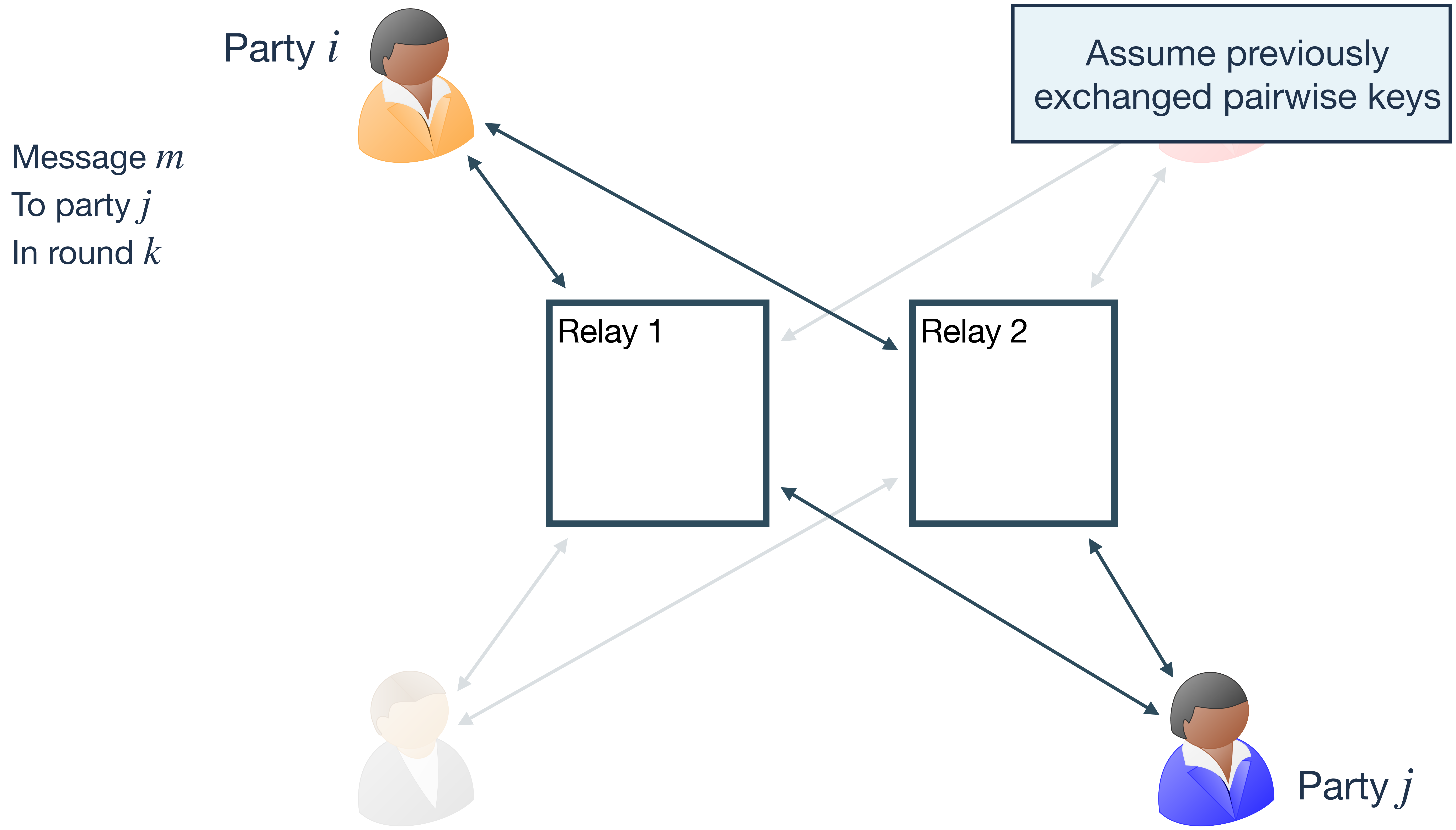
The relays: p2p messages



The relays: p2p messages



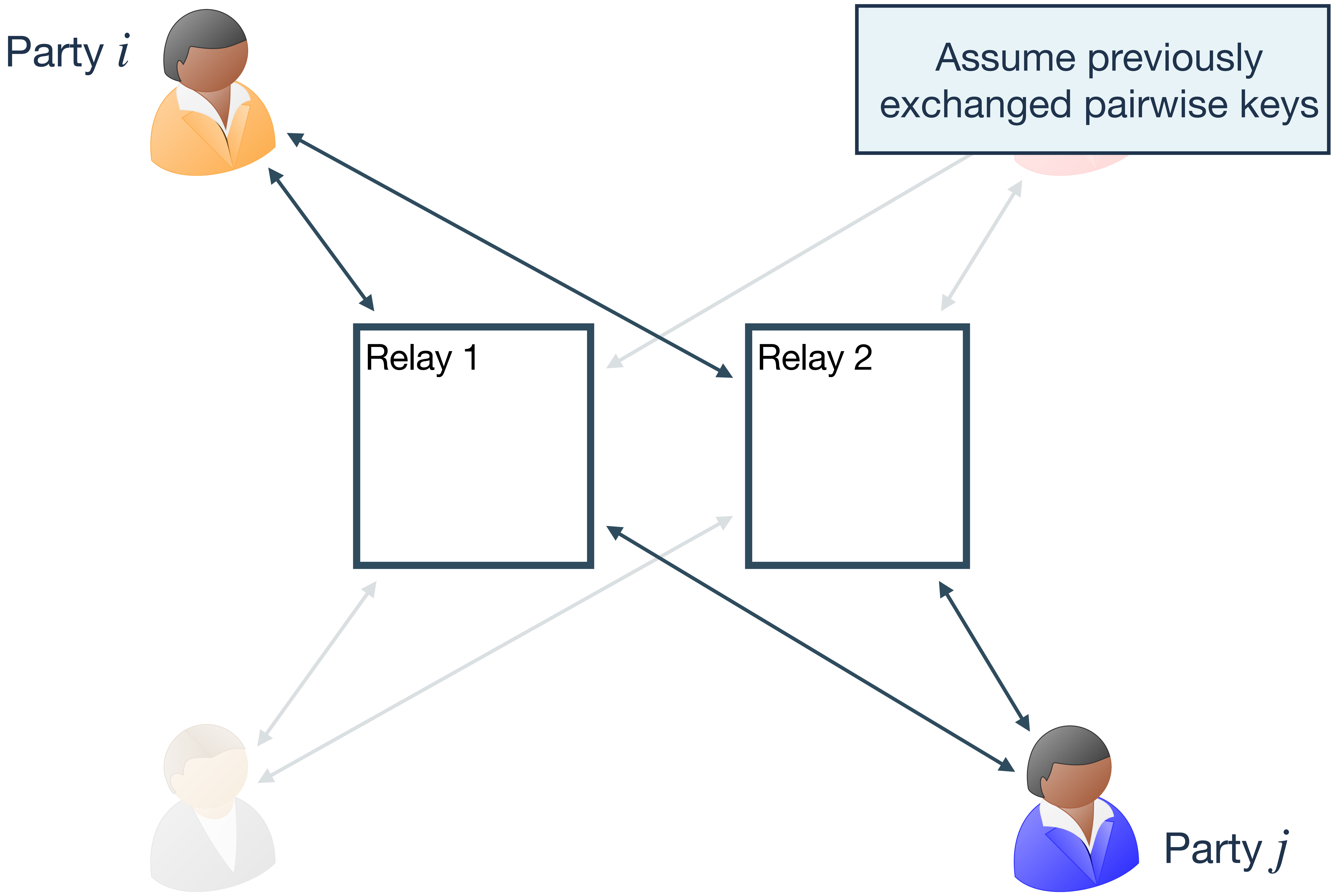
The relays: p2p messages



The relays: p2p messages

Message m
To party j
In round k

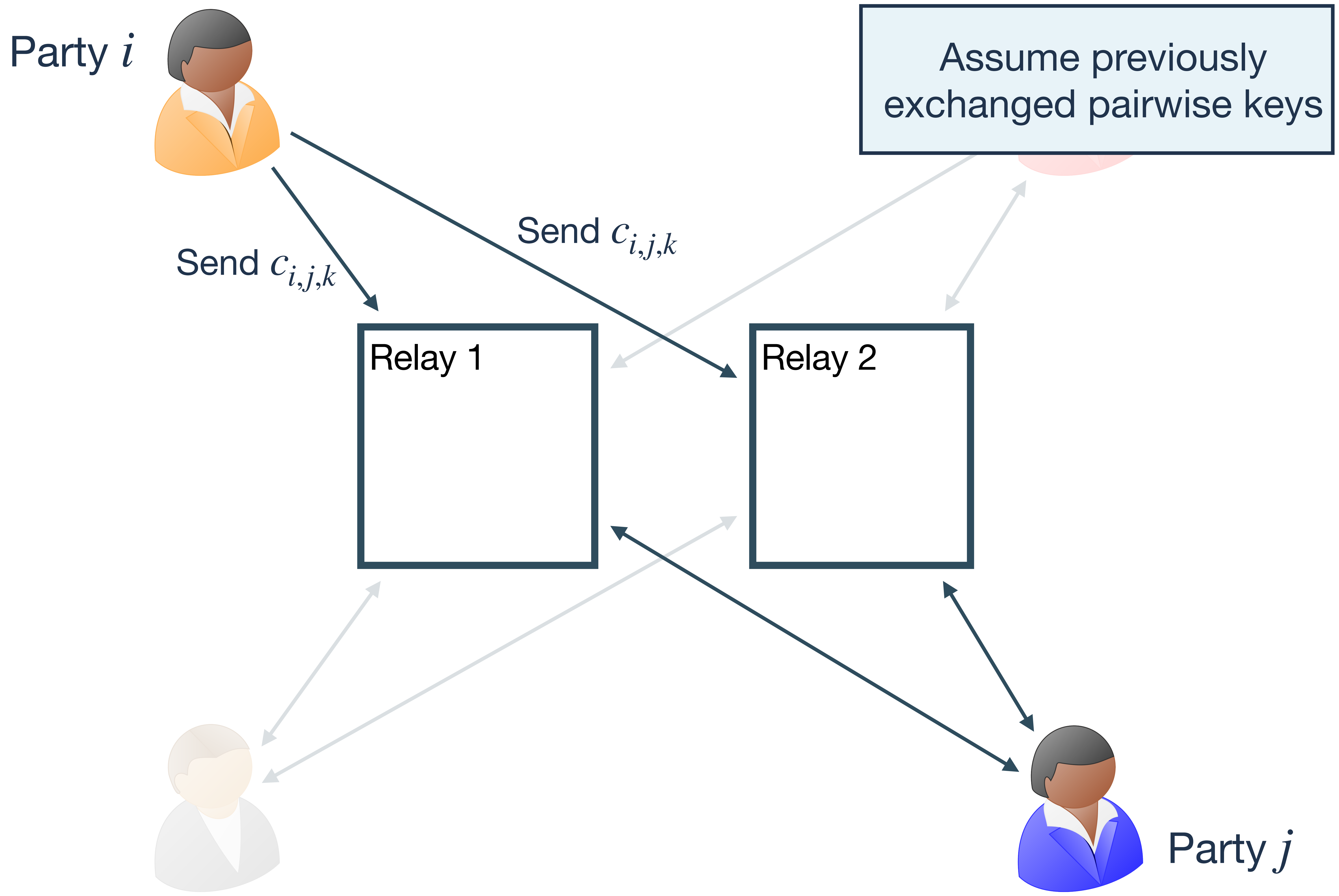
$c_{i,j,k} = \text{Enc}(m)$



The relays: p2p messages

Message m
To party j
In round k

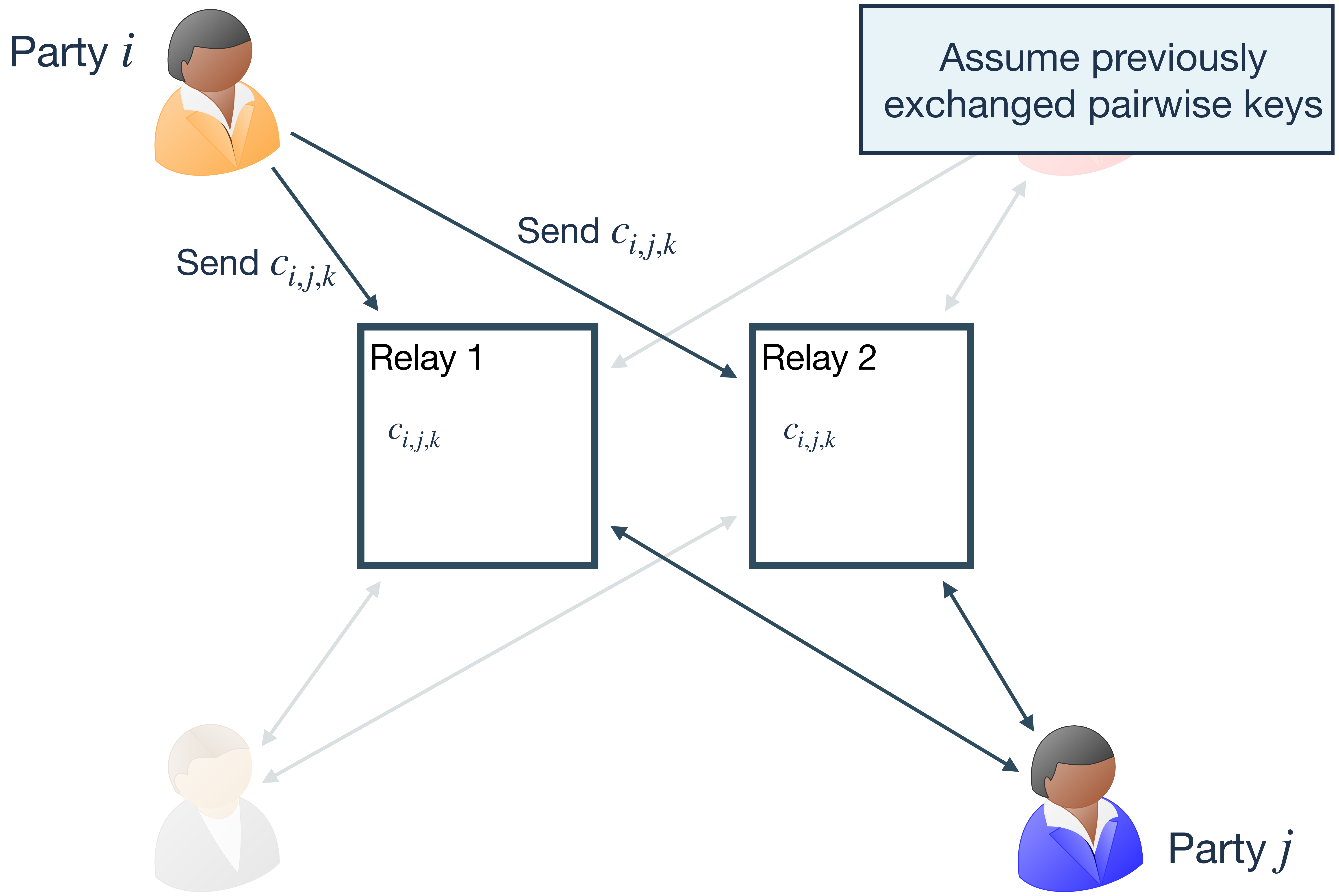
$c_{i,j,k} = \text{Enc}(m)$



The relays: p2p messages

Message m
To party j
In round k

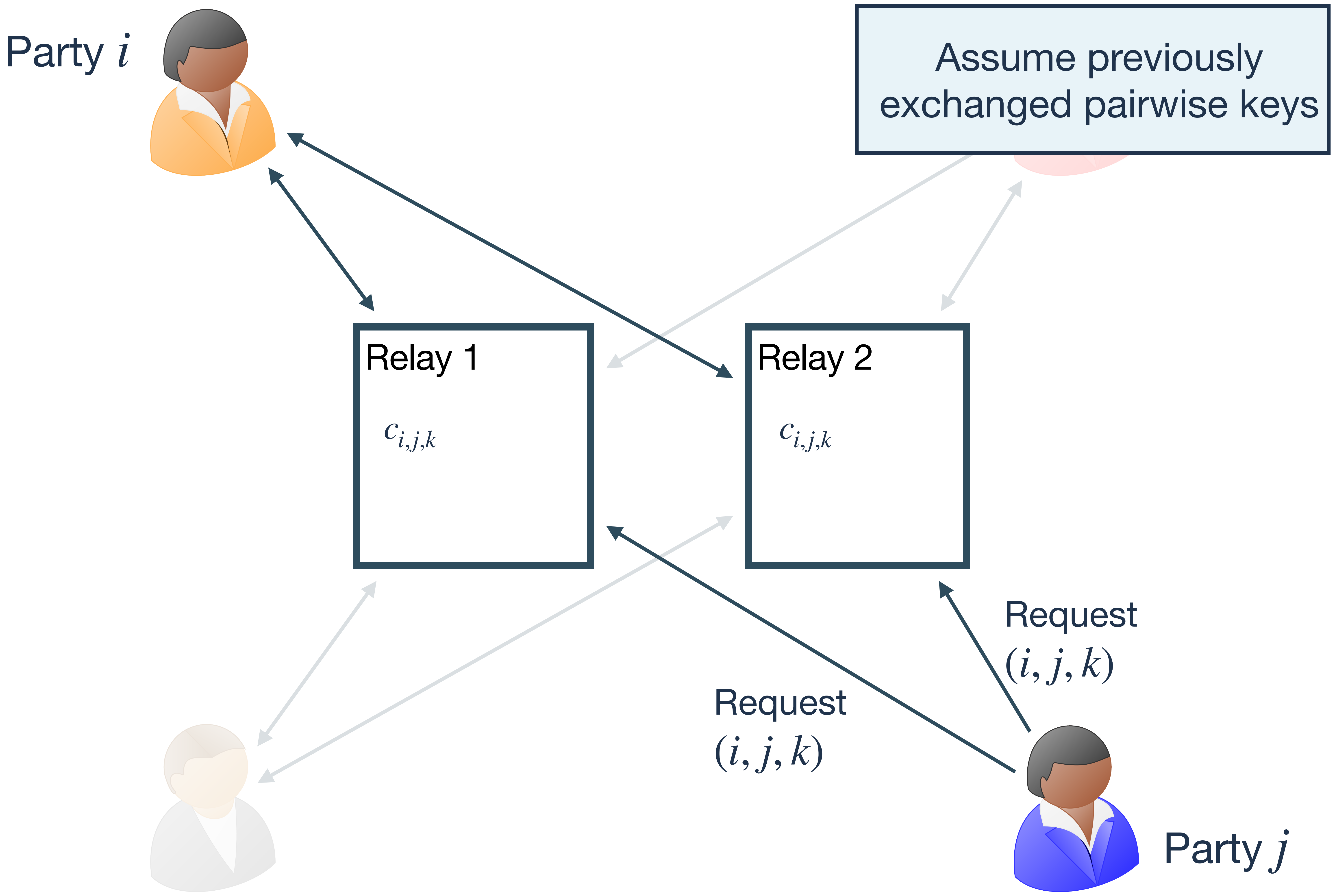
$c_{i,j,k} = \text{Enc}(m)$



The relays: p2p messages

Message m
To party j
In round k

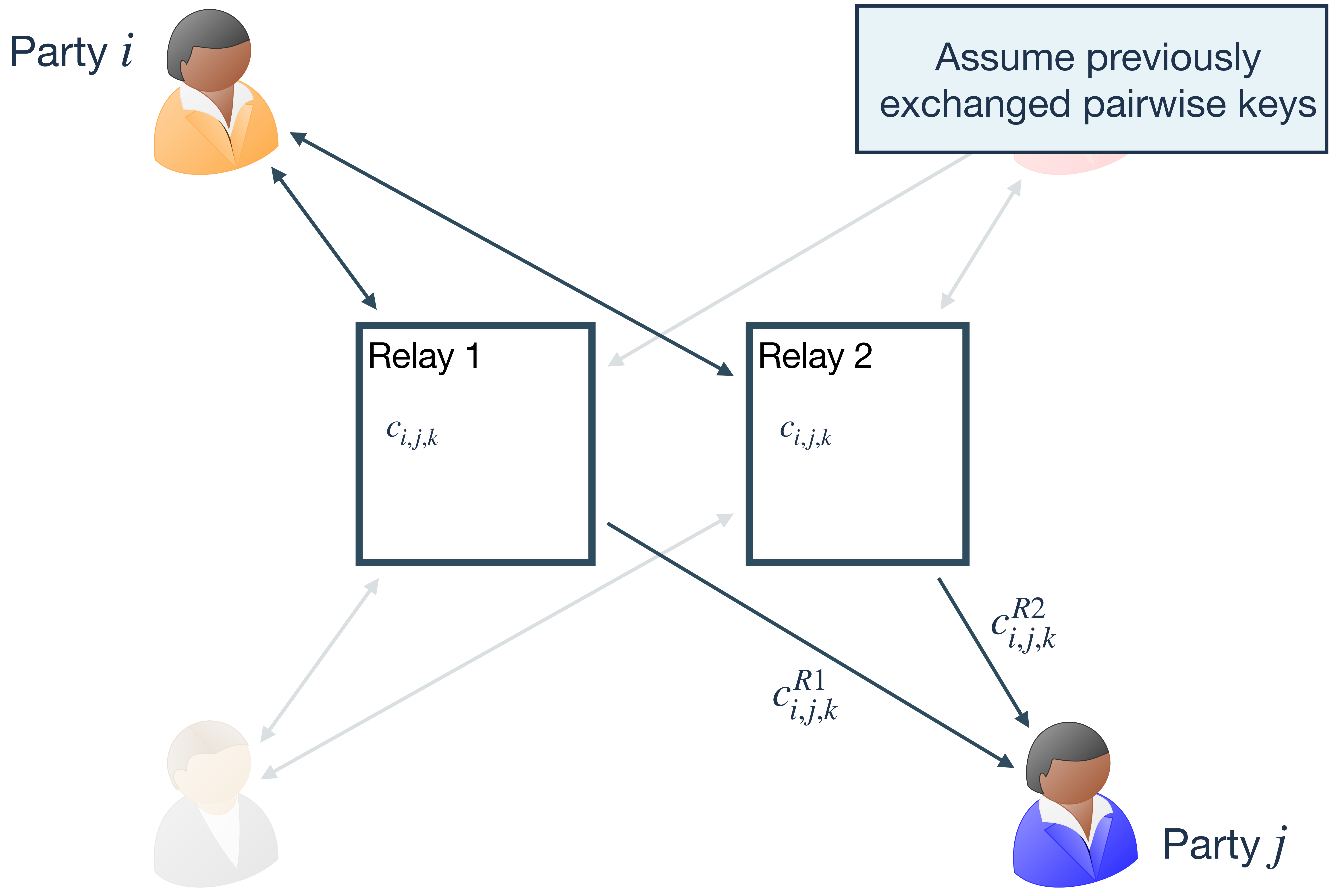
$c_{i,j,k} = \text{Enc}(m)$



The relays: p2p messages

Message m
To party j
In round k

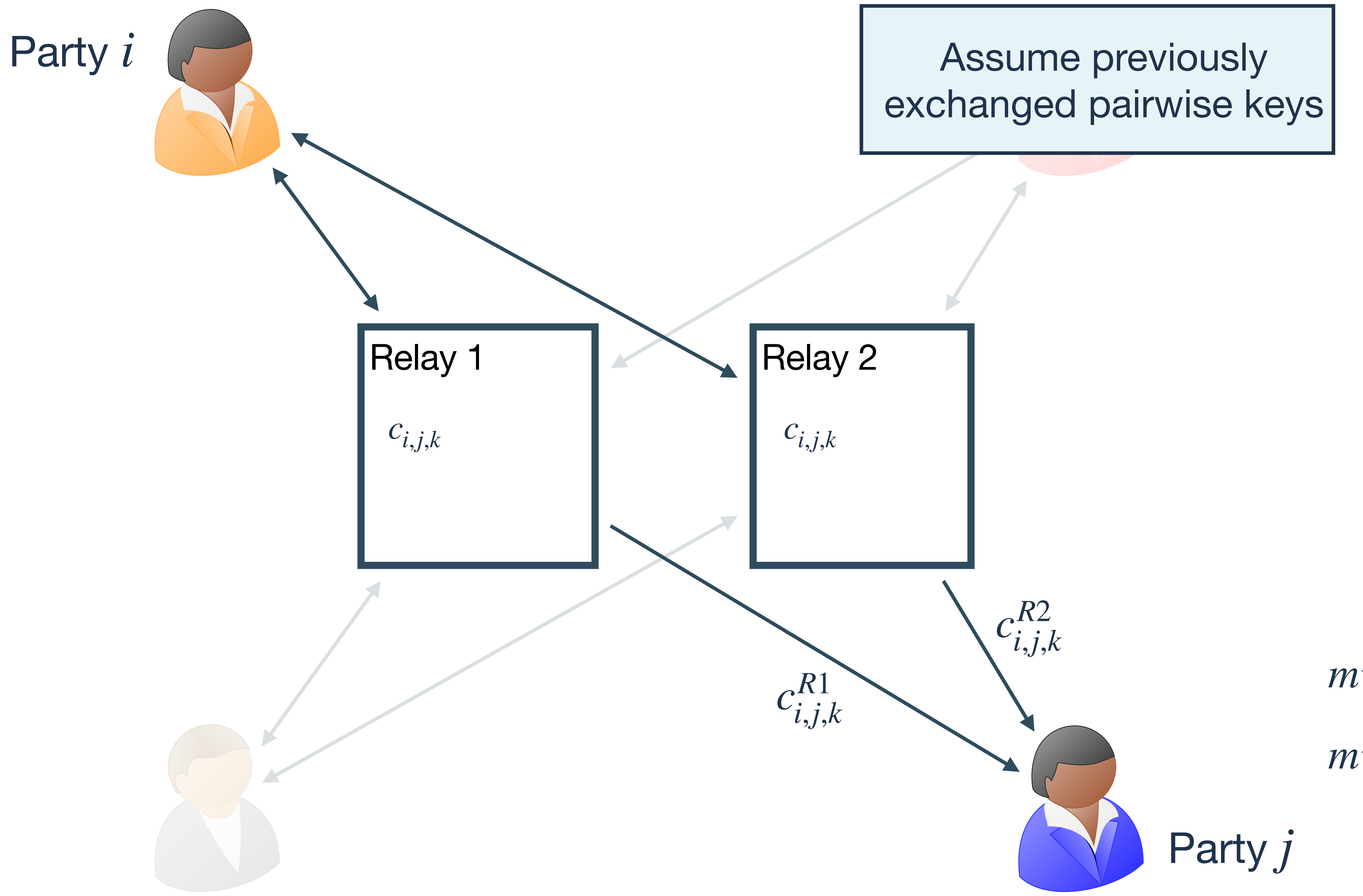
$c_{i,j,k} = \text{Enc}(m)$



The relays: p2p messages

Message m
To party j
In round k

$c_{i,j,k} = \text{Enc}(m)$



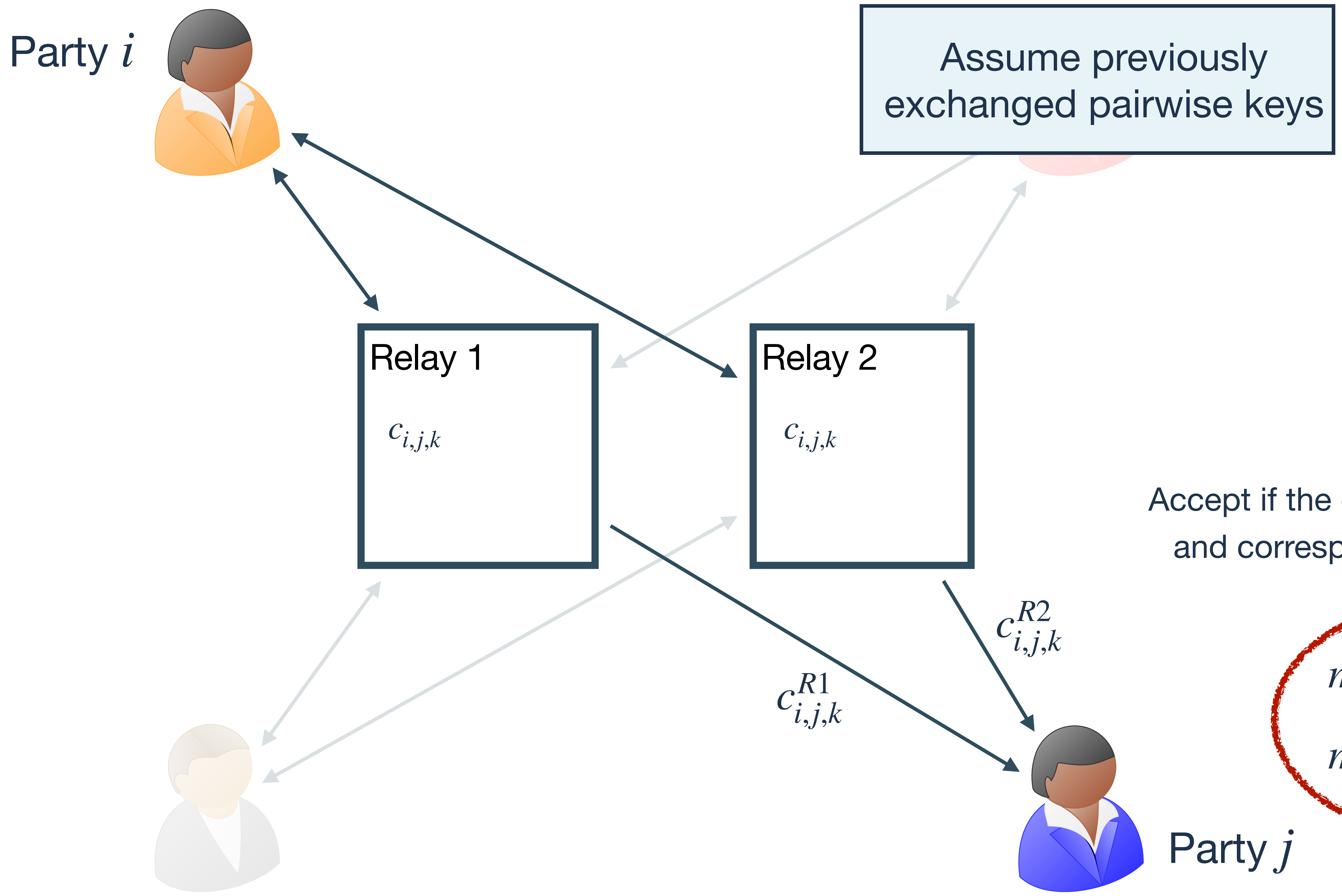
$m^{R1} = \text{Dec}(c^{R1}_{i,j,k})$

$m^{R2} = \text{Dec}(c^{R2}_{i,j,k})$

The relays: p2p messages

Message m
To party j
In round k

$c_{i,j,k} = \text{Enc}(m)$



Accept if the decryption happens correctly and corresponds to the same message

$m^{R1} = \text{Dec}(c^{R1}_{i,j,k})$

$m^{R2} = \text{Dec}(c^{R2}_{i,j,k})$

The relays

p2p messages

From party i to party j.

Commands:

Send: stores encrypted message to party j, round $k_{i,j}$

Request: retrieves message from i to j, round $k_{i,j}$

Erase: erases message from i to j, round $k_{i,j}$

Relay maintains:

- Pairwise message counter $k_{i,j}$
- Pairwise deleting counter $d_{i,j}$

Broadcast messages

The relays

p2p messages

From party i to party j.

Commands:

Send: stores encrypted message to party j, round $k_{i,j}$

Request: retrieves message from i to j, round $k_{i,j}$

Erase: erases message from i to j, round $k_{i,j}$

Relay maintains:

- Pairwise message counter $k_{i,j}$
- Pairwise deleting counter $d_{i,j}$

Broadcast messages

From party i to all other parties.

Commands:

SendToAll: stores plaintext message to all parties, round k^{all}

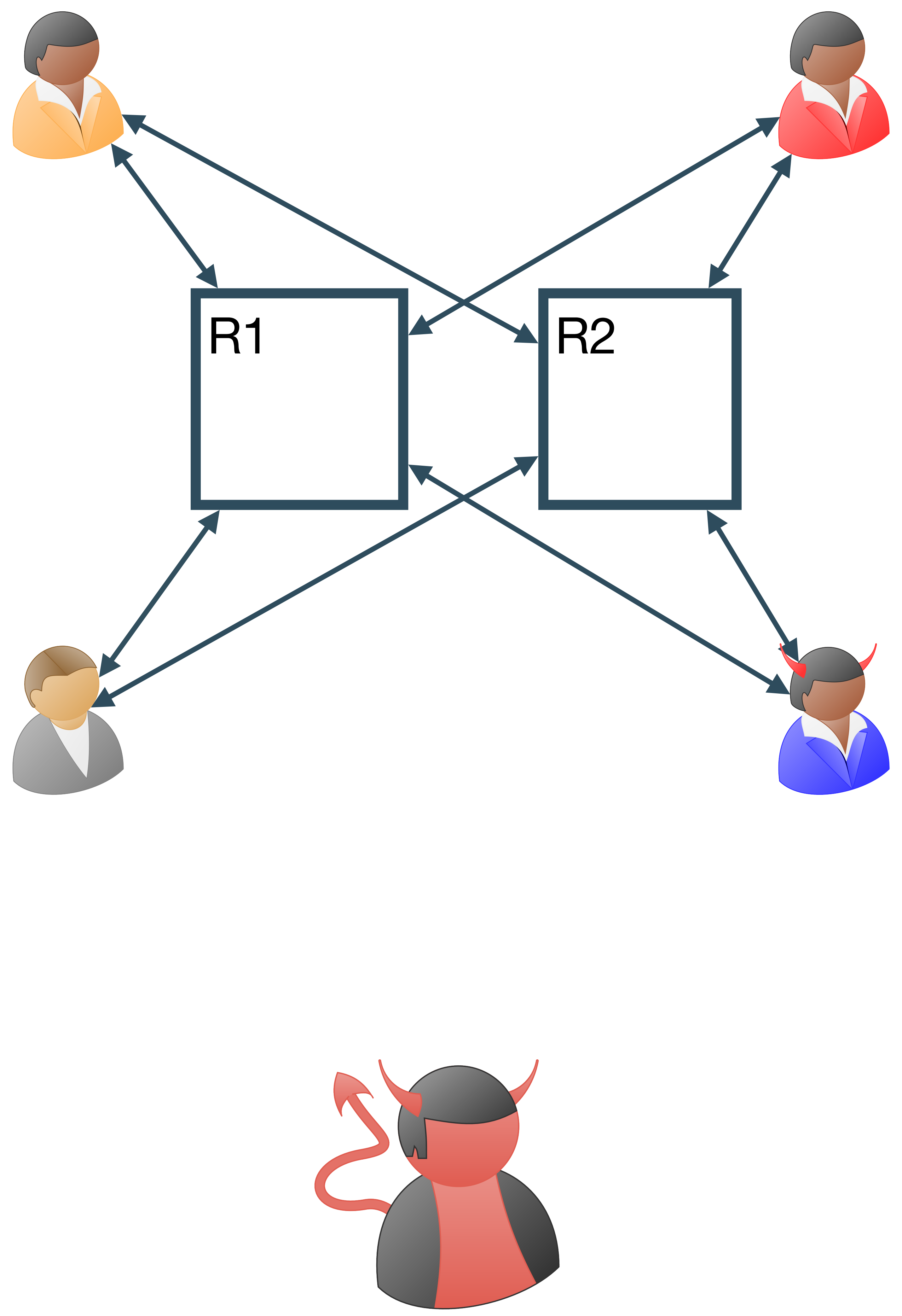
RequestFromAll: retrieves all messages for round k^{all}

EraseAll: erases all messages for round k^{all}

Relay maintains:

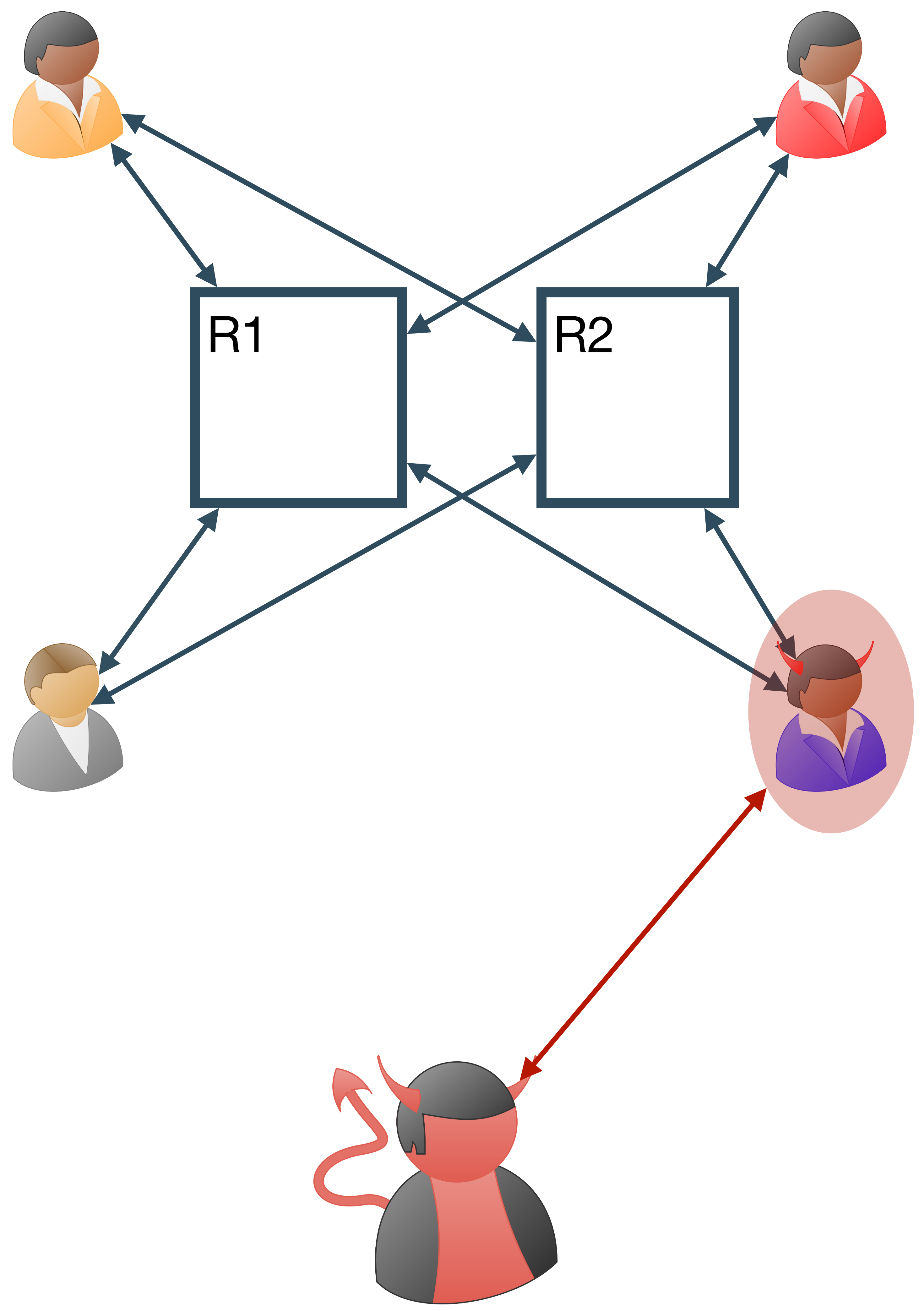
- Global message counter k^{all}
- Global deleting counter d^{all}

The adversary



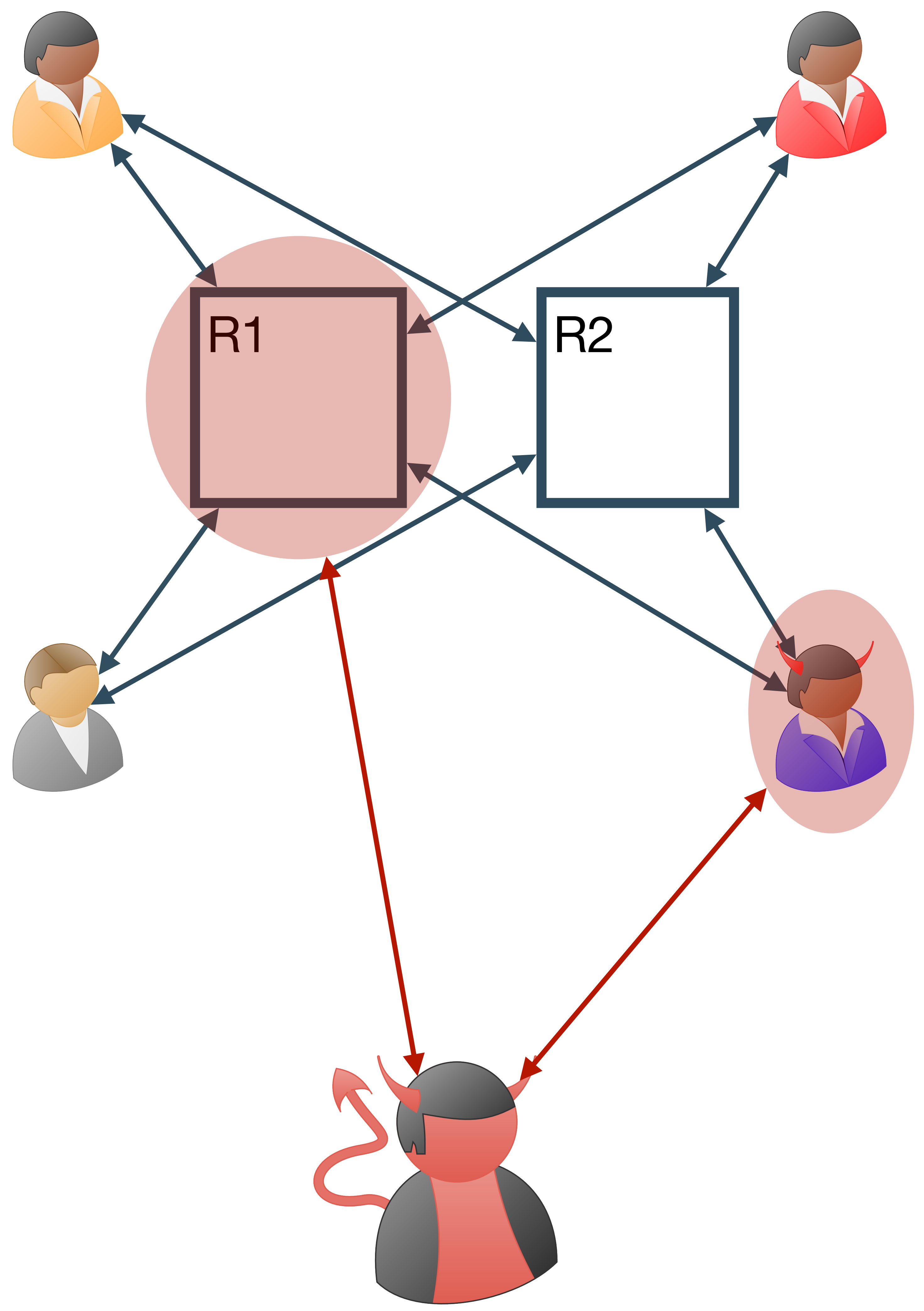
The adversary

- Can corrupt up to $t < n/2$ parties (static corruption)



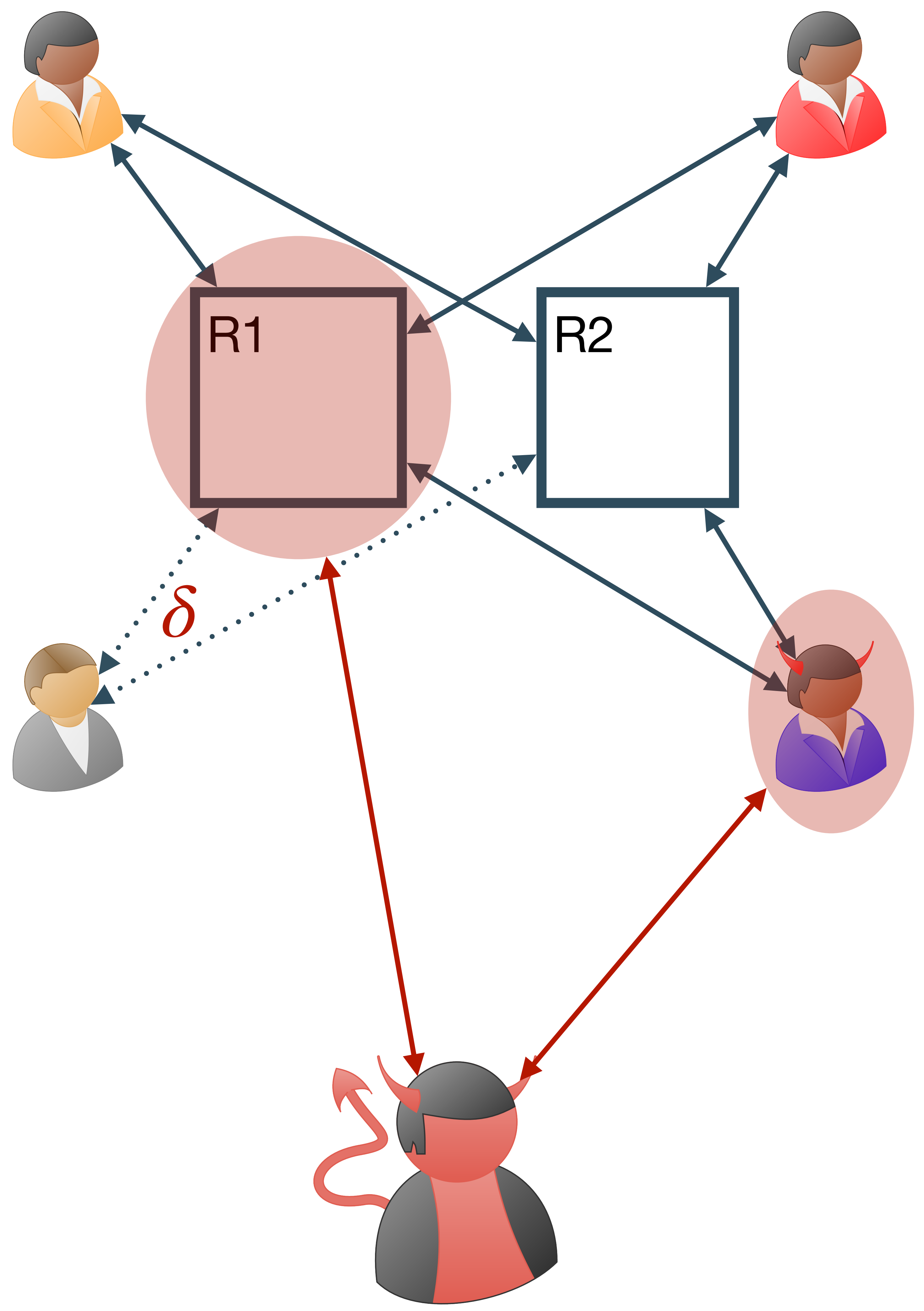
The adversary

- Can corrupt up to $t < n/2$ parties (static corruption)
- Can corrupt all but one relay



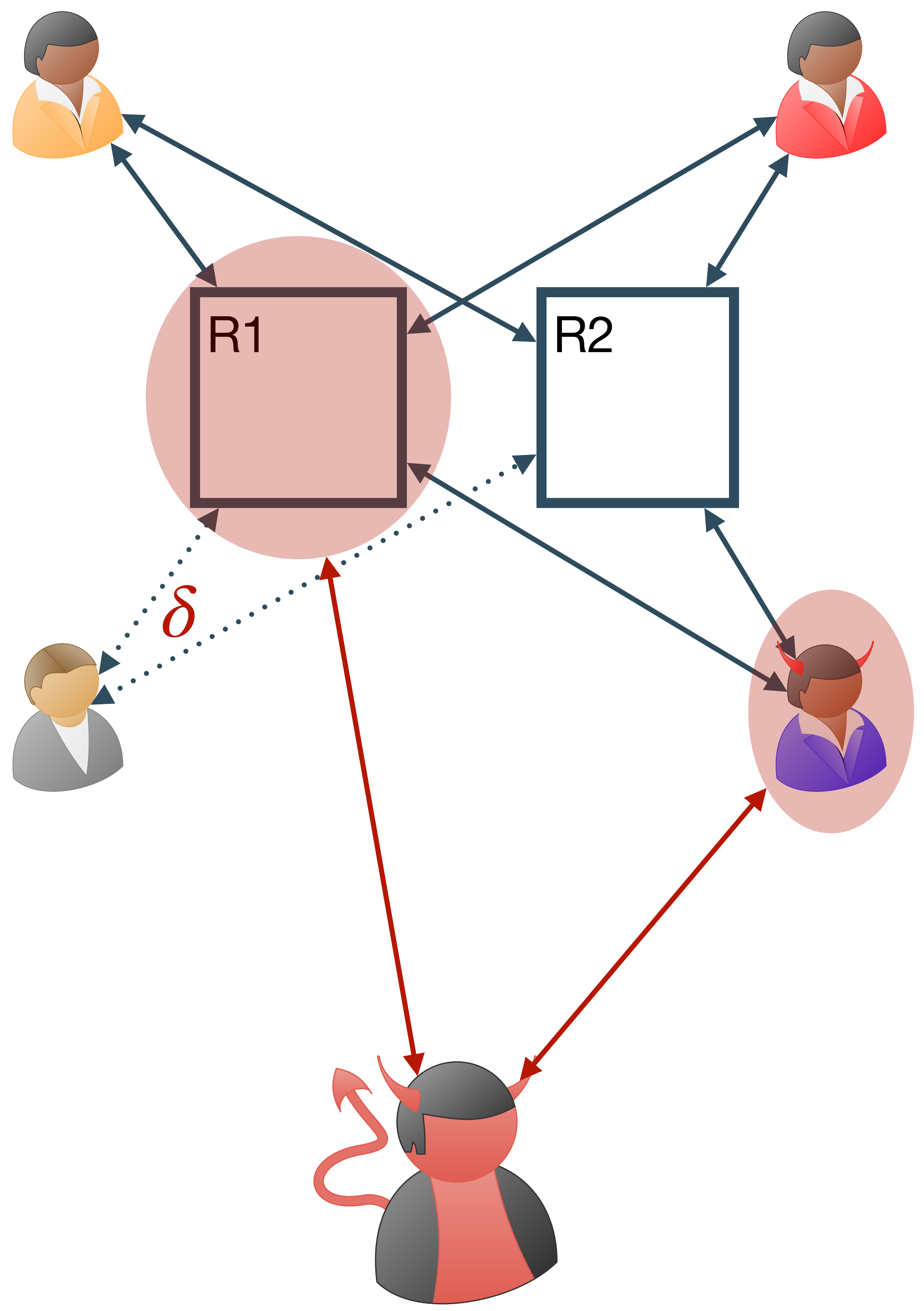
The adversary

- Can corrupt up to $t < n/2$ parties (static corruption)
- Can corrupt all but one relay
- Can delay an arbitrary number of parties for up to δ rounds



The adversary

- Can corrupt up to $t < n/2$ parties (static corruption)
- Can corrupt all but one relay
- Can delay an arbitrary number of parties for up to δ rounds



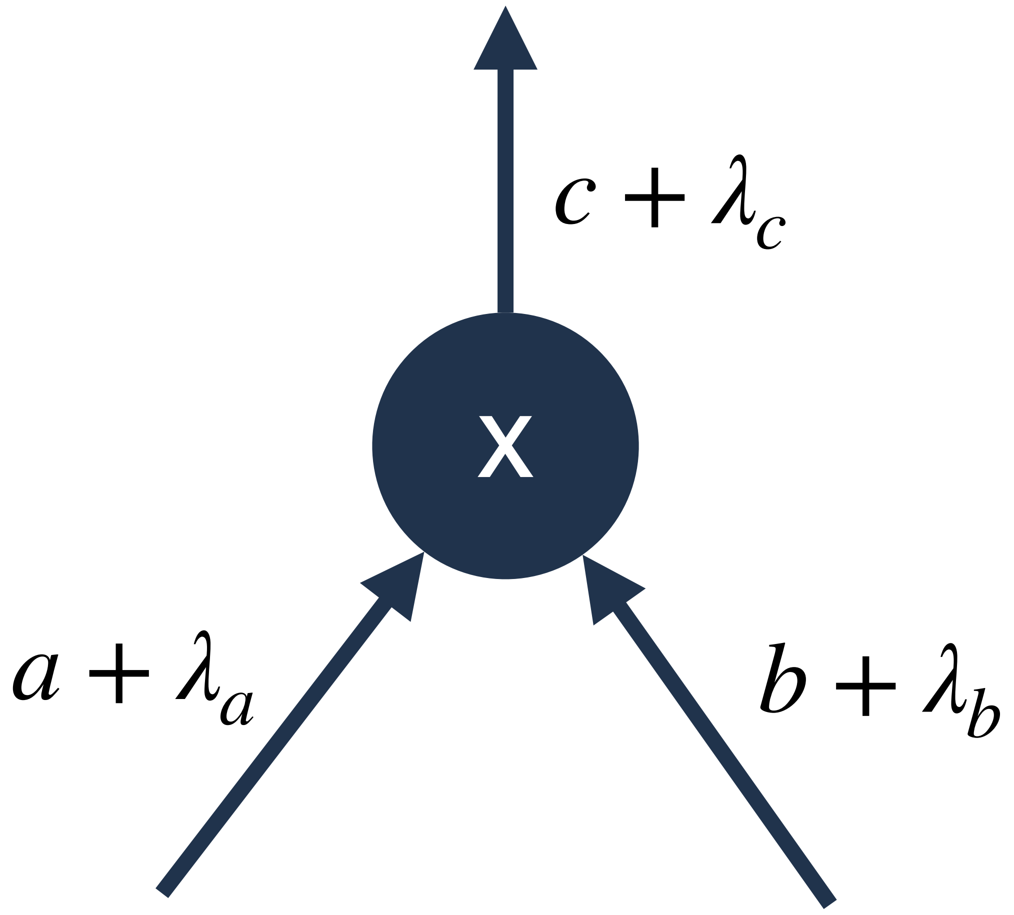
We present an MPC protocol that achieves passive security against this adversary.

Malicious security

Passive security with **additive attacks**

Circuits resilient to additive attacks with applications to secure computation.
Genkin et al. [GIP+14]

Active security with abort

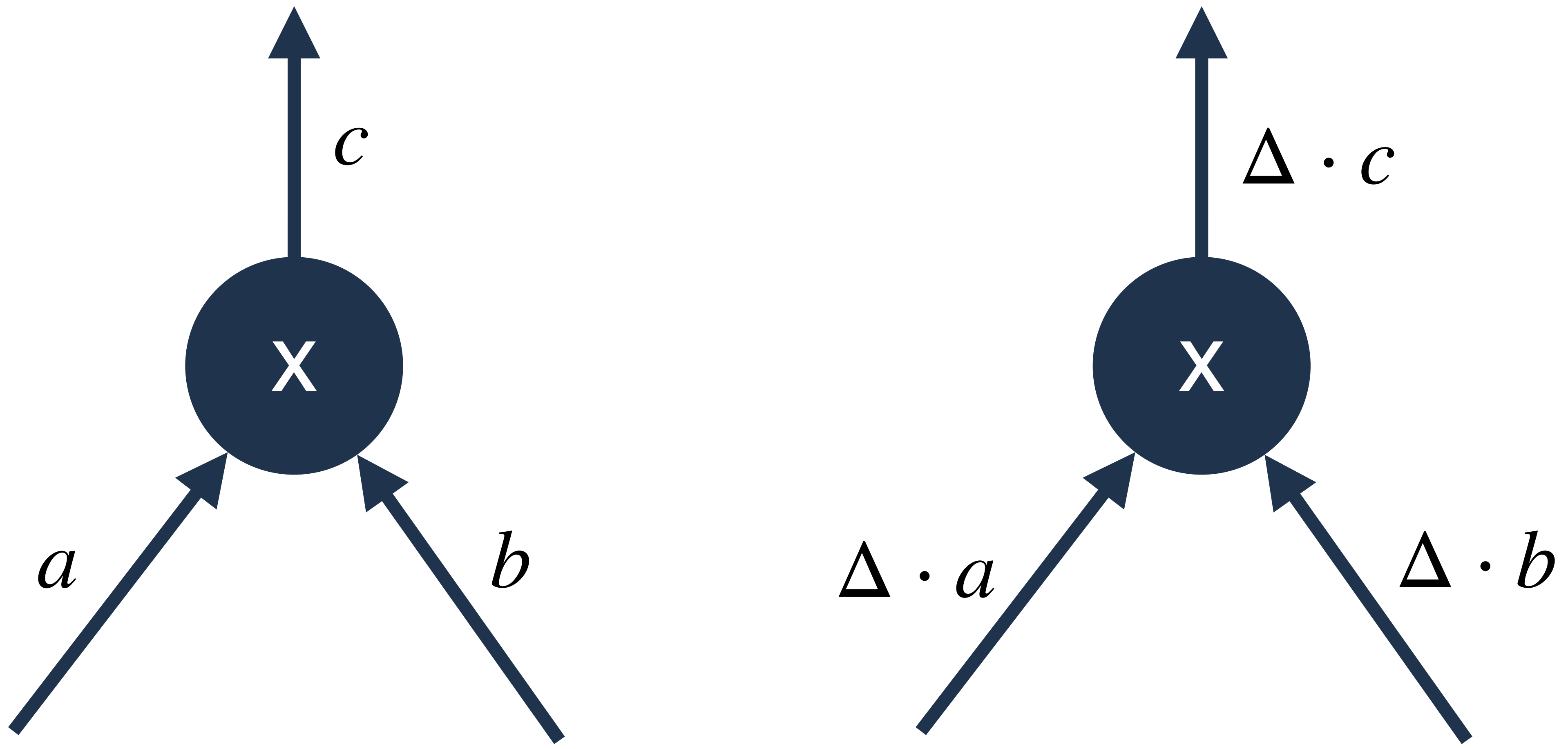


Malicious security

Passive security with **additive attacks**

[Circuits resilient to additive attacks with applications to secure computation.](#)
Genkin et al. [GIP+14]

→ Active security with abort

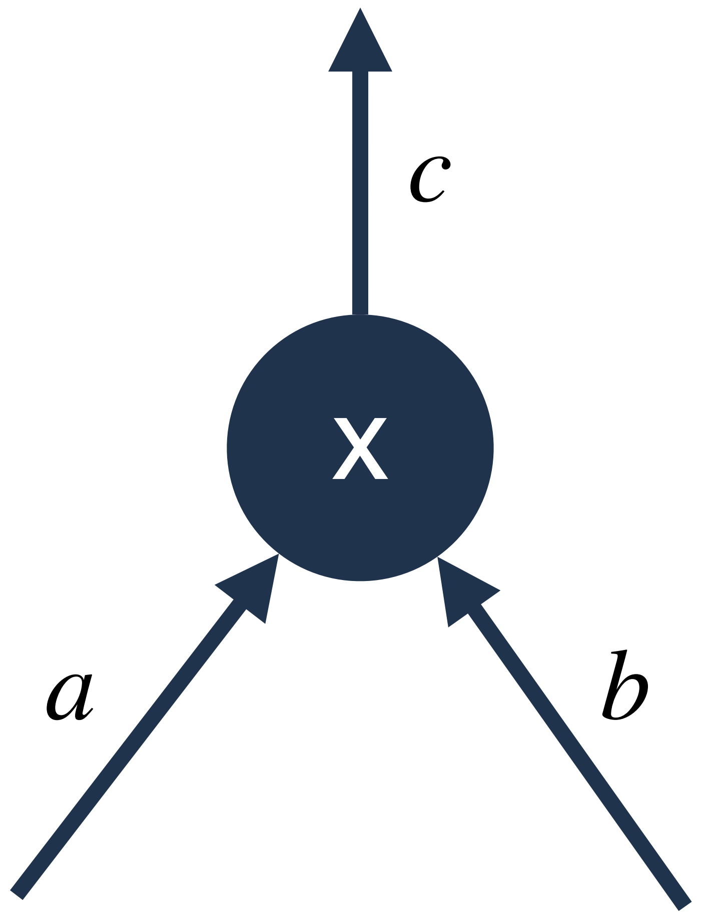


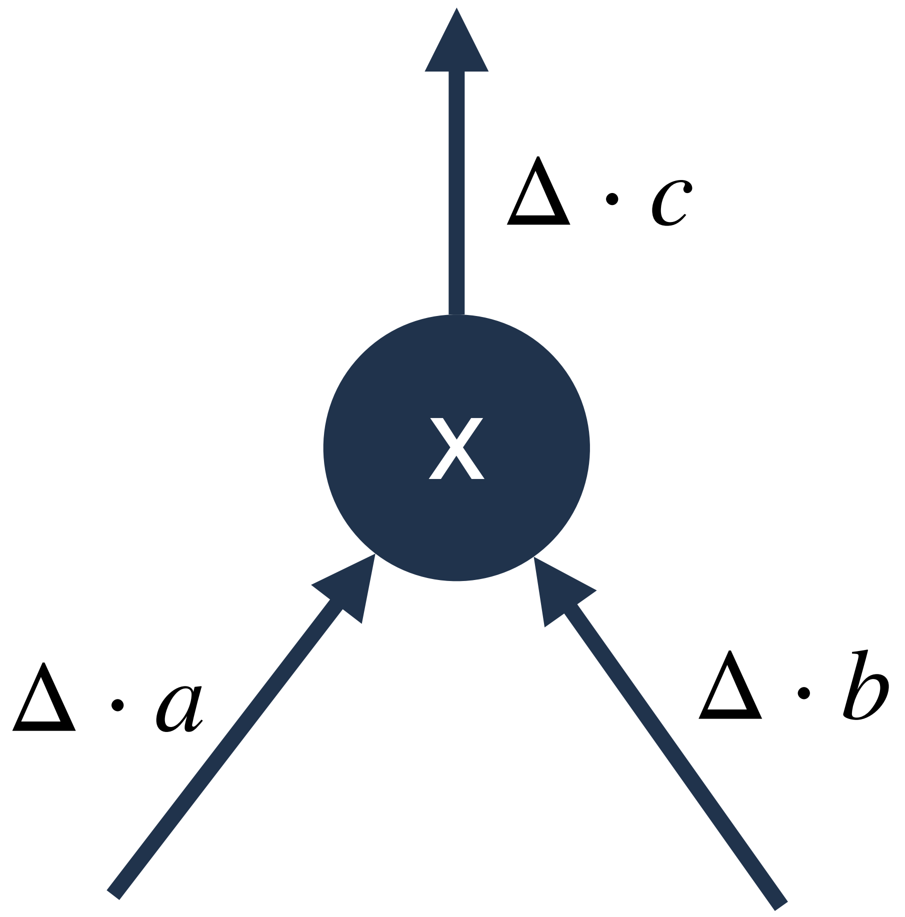
Malicious security

Passive security with **additive attacks**

[Circuits resilient to additive attacks with applications to secure computation.]
Genkin et al. [GIP+14]

Active security with abort





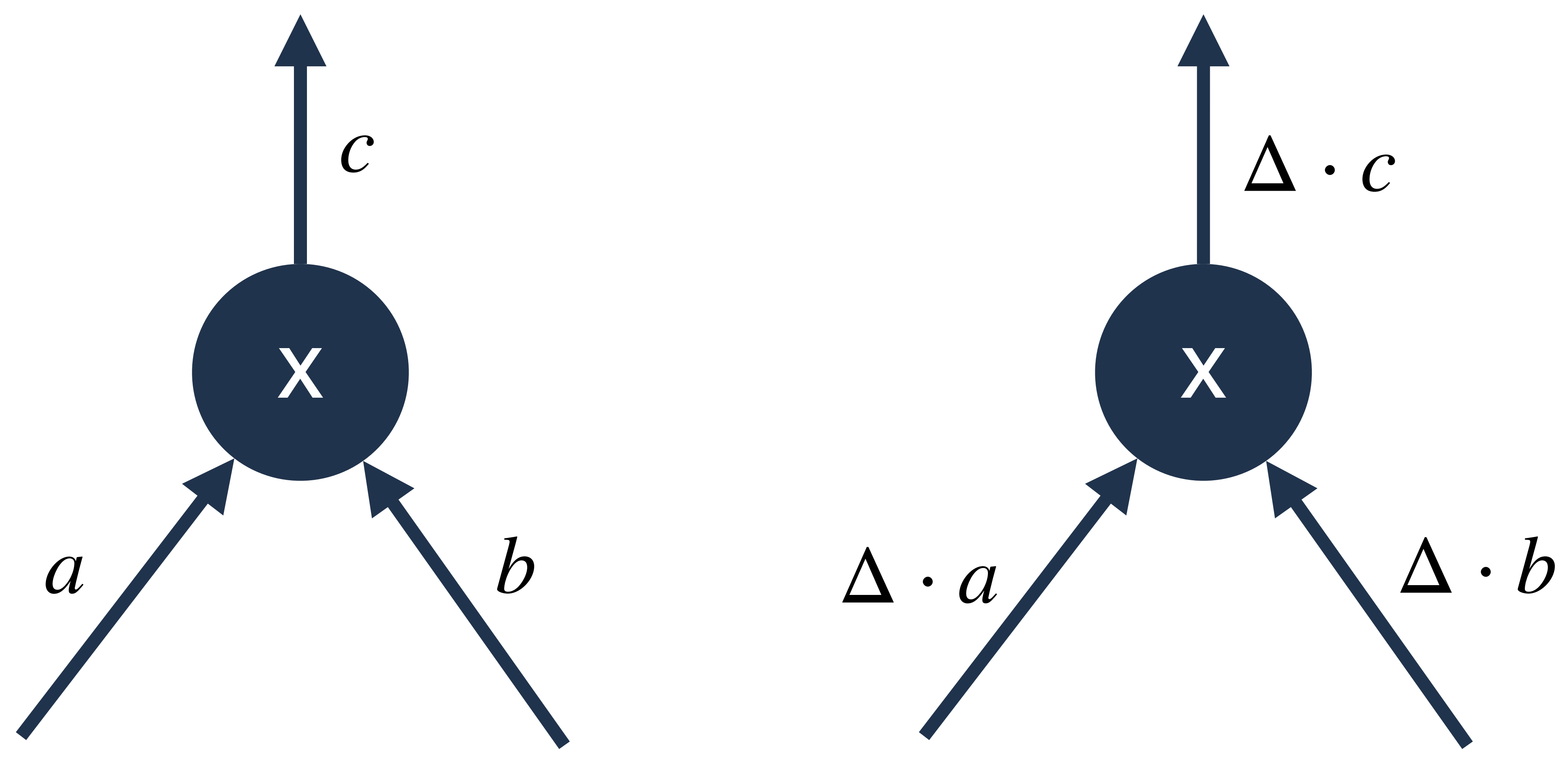
Progressively compute checking equation to avoid having to store large states (similar to FluidMPC [CGG+21])

Malicious security

Passive security with **additive attacks**

[Circuits resilient to additive attacks with applications to secure computation.](#) Genkin et al. [GIP+14]

Active security with abort



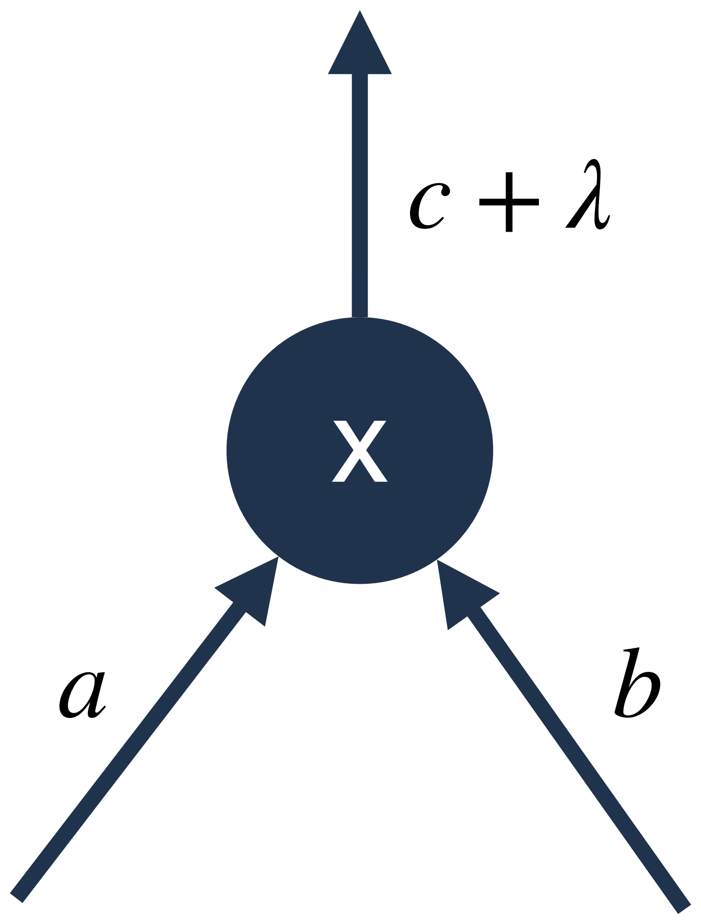
We want to obtain an MPC protocol that is **secure up to additive attacks**

Multiplication protocols

Calculating $[z] = [x] \cdot [y]$

Maurer multiplication

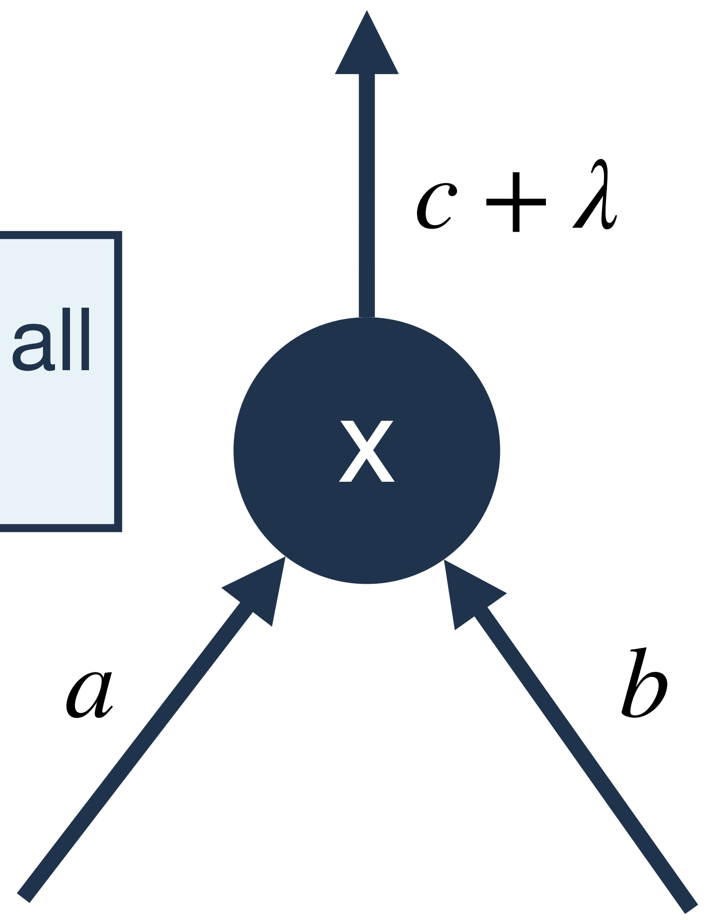
1. Parties locally multiplies their shares to obtain: $[v]_{2t} = [x \cdot y]_{2t} = [x]_t \cdot [y]_t$
2. Each party i distributes a degree t secret sharing $[v_i]_t$ of v_i among the other parties
3. Parties use their shares of the $[v_i]_t$ to calculate $[z]$.



Multiplication protocols

Calculating $[z] = [x] \cdot [y]$

λ is not uniquely determined for all parties



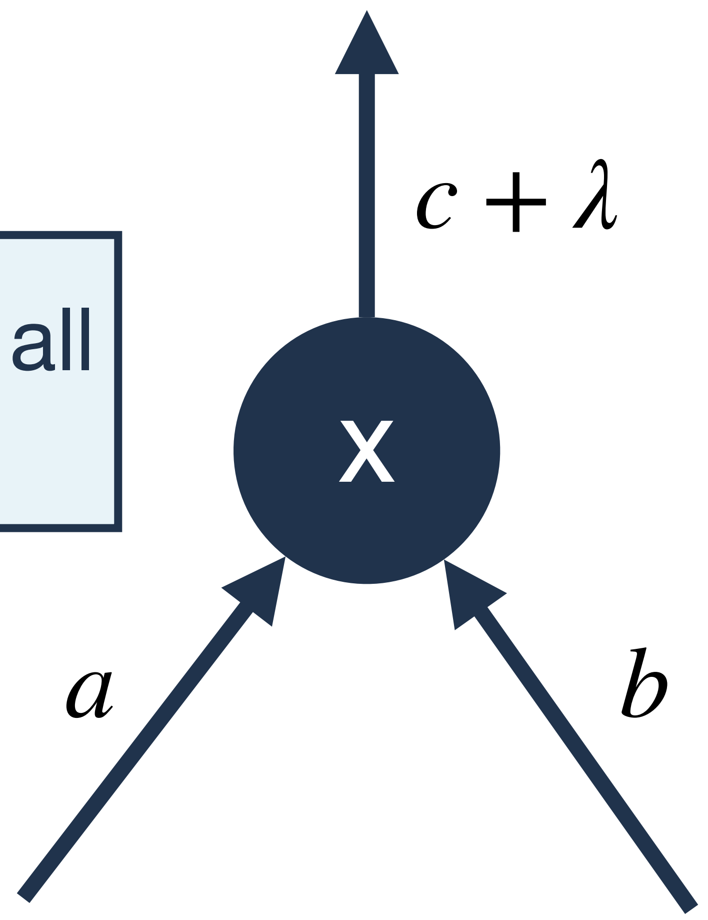
Maurer multiplication ✗

1. Parties locally multiplies their shares to obtain: $[v]_{2t} = [x \cdot y]_{2t} = [x]_t \cdot [y]_t$
2. Each party i distributes a degree t secret sharing $[v_i]_t$ of v_i among the other parties
3. Parties use their shares of the $[v_i]_t$ to calculate $[z]$.

Multiplication protocols

Calculating $[z] = [x] \cdot [y]$

λ is not uniquely determined for all parties



Maurer multiplication ✗

1. Parties locally multiplies their shares to obtain: $[v]_{2t} = [x \cdot y]_{2t} = [x]_t \cdot [y]_t$
2. Each party i distributes a degree t secret sharing $[v_i]_t$ of v_i among the other parties
3. Parties use their shares of the $[v_i]_t$ to calculate $[z]$.

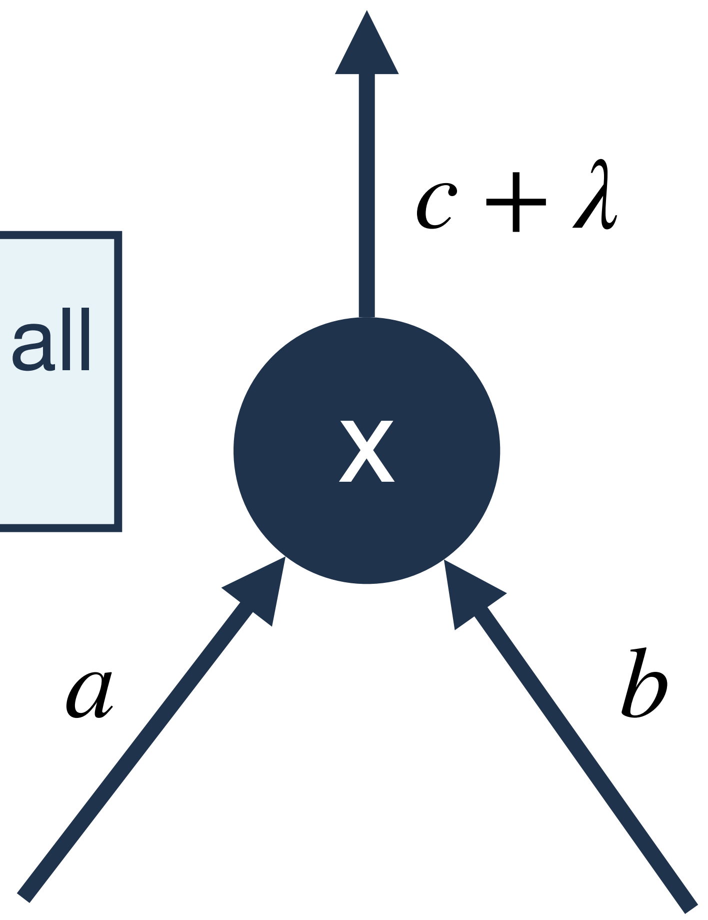
Damgård-Nielsen multiplication

1. Use a PRSS to generate a degree t and a degree $2t$ sharing of the same random value: $[r]_t, [r]_{2t}$
2. Parties locally calculate: $[v]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$
3. Each party i sends v_i to party 1. Party 1 reconstructs v and reveals it to all
4. Parties calculate $[z] = v - [r]_t$

Multiplication protocols

Calculating $[z] = [x] \cdot [y]$

λ is not uniquely determined for all parties



Maurer multiplication ✗

1. Parties locally multiplies their shares to obtain: $[v]_{2t} = [x \cdot y]_{2t} = [x]_t \cdot [y]_t$
2. Each party i distributes a degree t secret sharing $[v_i]_t$ of v_i among the other parties
3. Parties use their shares of the $[v_i]_t$ to calculate $[z]$.

Damgård-Nielsen multiplication

1. Use a PRSS to generate a degree t and a degree $2t$ sharing of the same random value: $[r]_t, [r]_{2t}$
2. Parties locally calculate: $[v]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$
3. Each party i sends v_i to party 1. Party 1 reconstructs v and reveals it to all
4. Parties calculate $[z] = v - [r]_t$

Double-dipping attack when $n > 2t + 1$

Communication-efficient unconditional MPC with guaranteed output delivery
V. Goyal, Y. Liu, Y. Song, 2019

Multiplication protocols

1-Round Damgård-Nielsen multiplication

1. Use a PRSS to generate a degree t and a degree $2t$ sharing of the same random value: $[r]_t$, $[r]_{2t}$
2. Parties locally calculate: $[v]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$
3. Each party i broadcasts v_i . All parties locally reconstruct v.
4. Parties calculate $[z] = v - [r]_t$

Multiplication protocols

1-Round Damgård-Nielsen multiplication

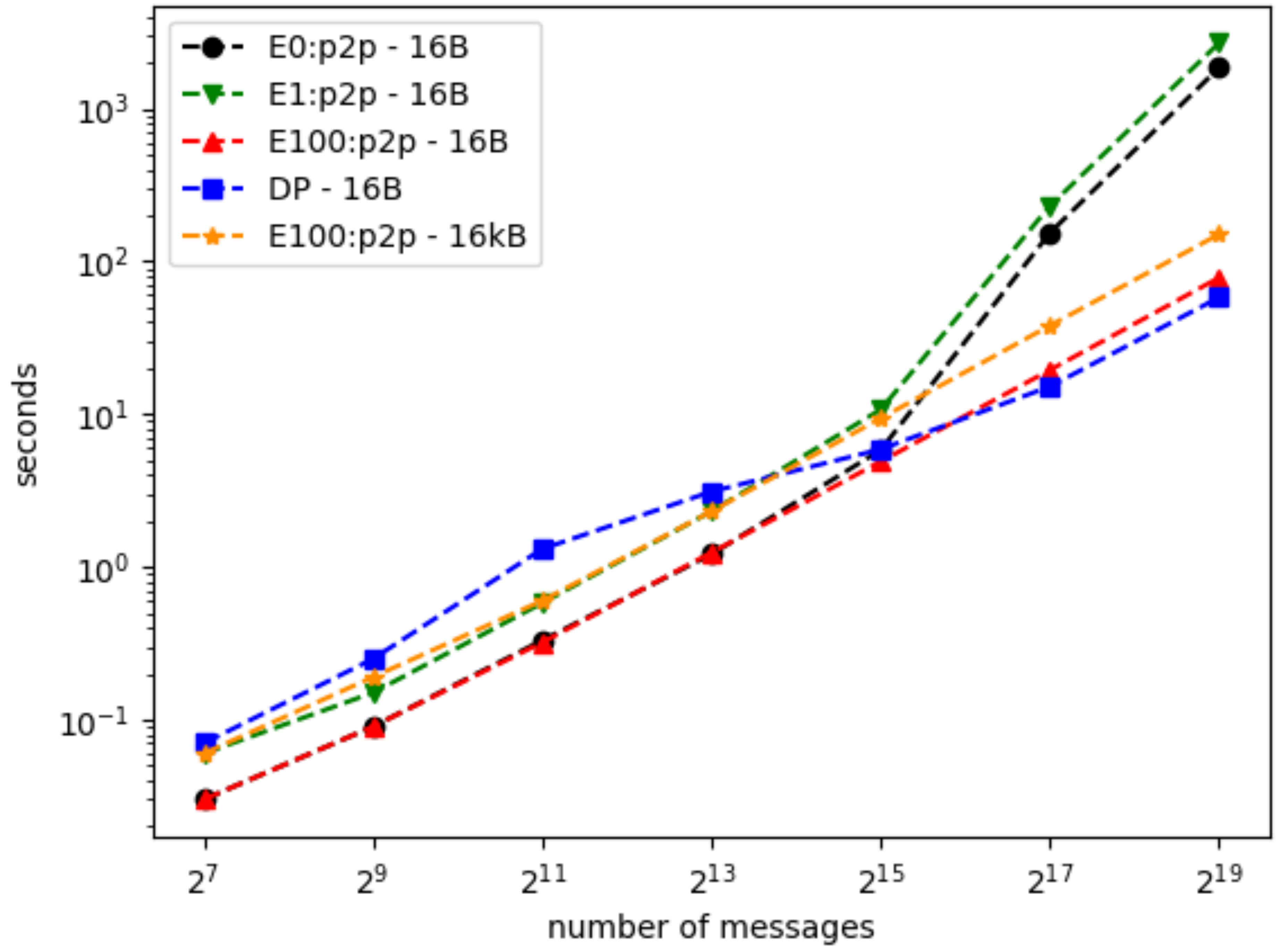
1. Use a PRSS to generate a degree t and a degree $2t$ sharing of the same random value: $[r]_t$, $[r]_{2t}$
2. Parties locally calculate: $[v]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$
3. Each party i broadcasts v_i . All parties locally reconstruct v.
4. Parties calculate $[z] = v - [r]_t$

Broadcast in a relay based network is cheap!

Experimental results

Network: relays vs direct connections

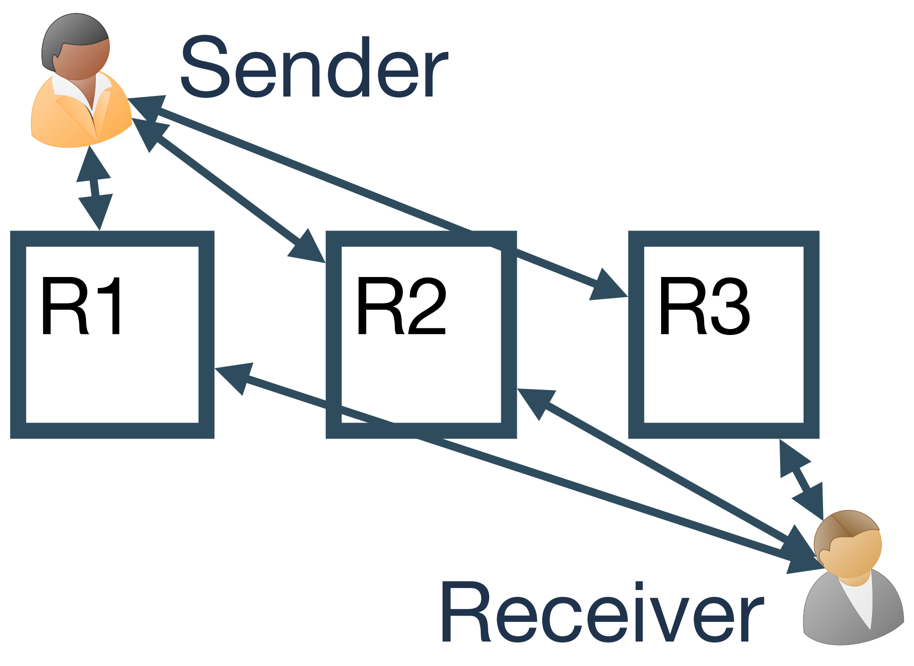
Running time for sending p2p messages.



Experiments:

- **E0**: never erase messages
- **E1**: erase every message after retrieval
- **E100**: erase in batches of 100 messages
- **DP**: communication without relays
- **E100 with large messages**

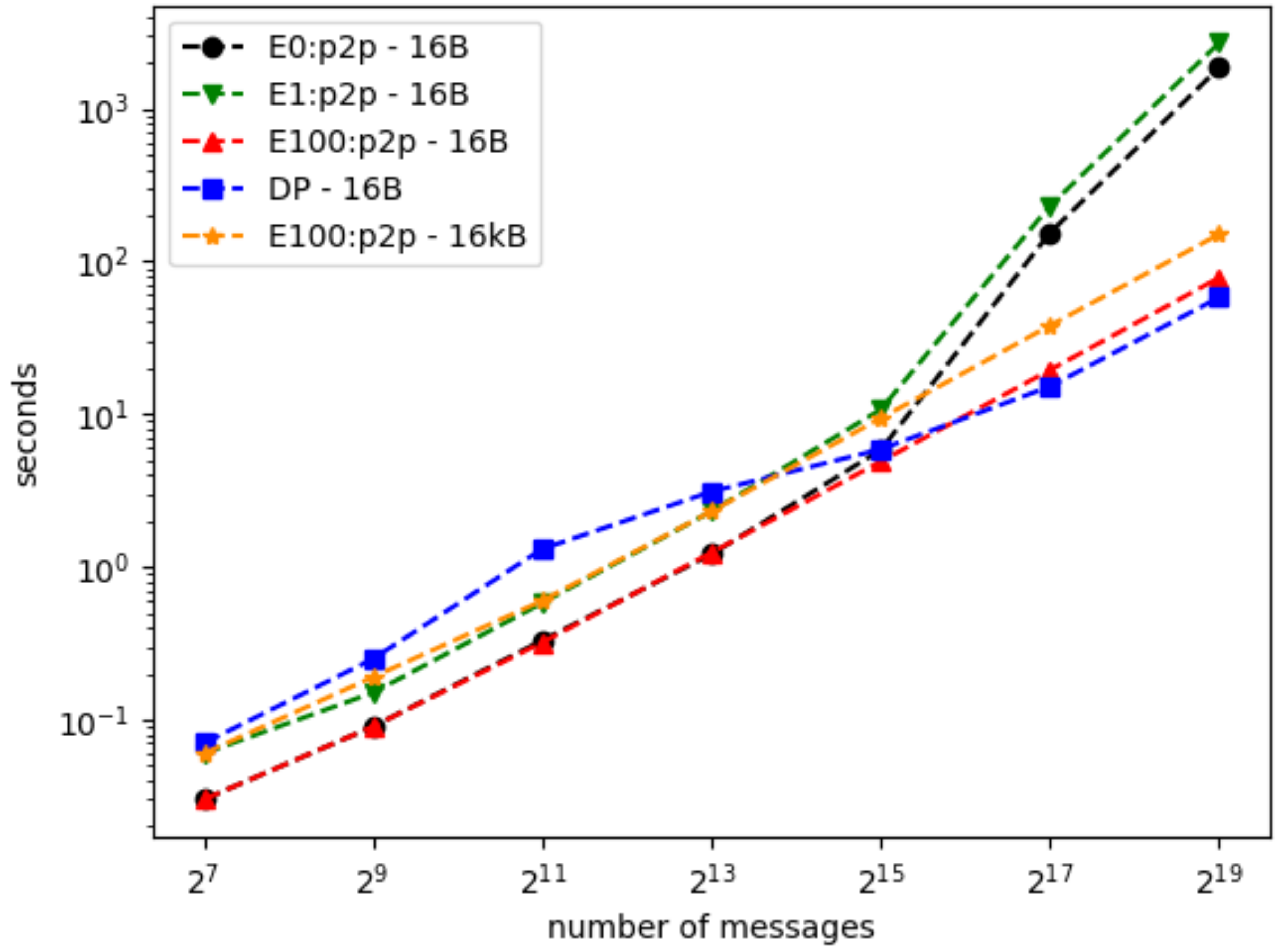
2 parties
3 relays



Experimental results

Network: relays vs direct connections

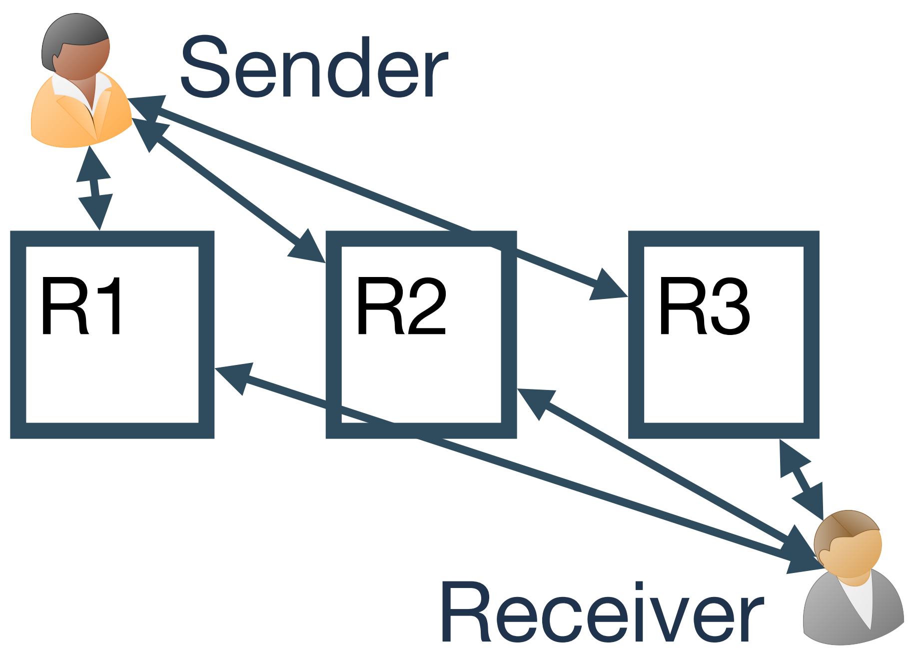
Running time for sending p2p messages.



Experiments:

- **E0**: never erase messages
- **E1**: erase every message after retrieval
- **E100**: erase in batches of 100 messages
- **DP**: communication without relays
- **E100 with large messages**

2 parties
3 relays



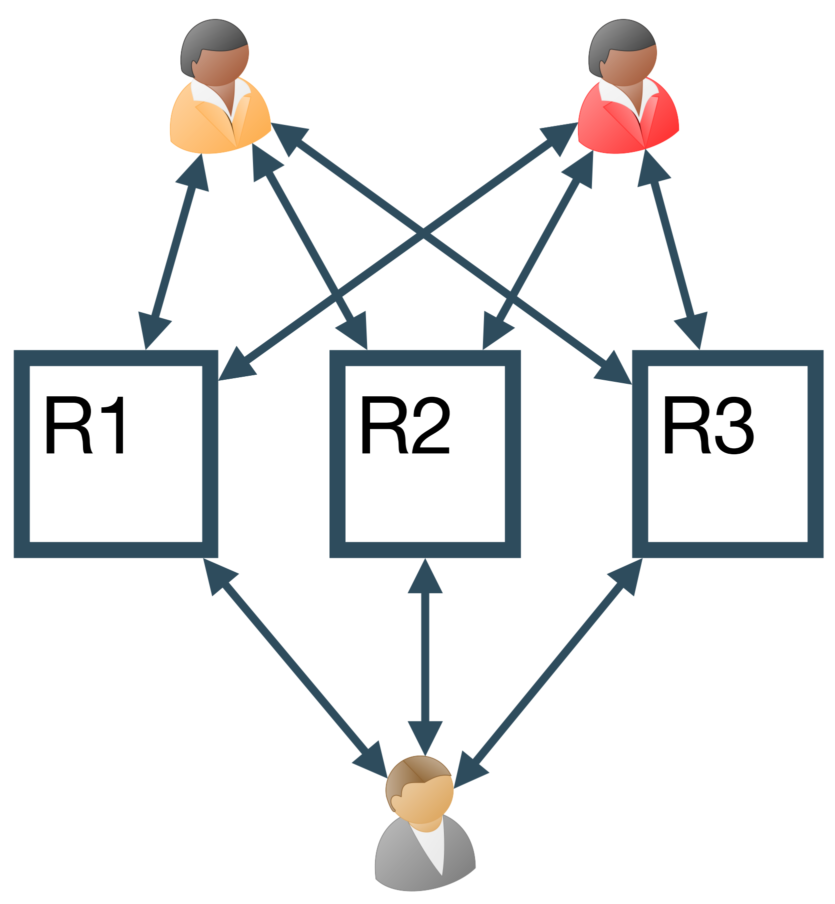
Erasing messages in **batches** ensures small overhead vs direct communication without running out of memory

Experimental results

Network: relays vs direct connections

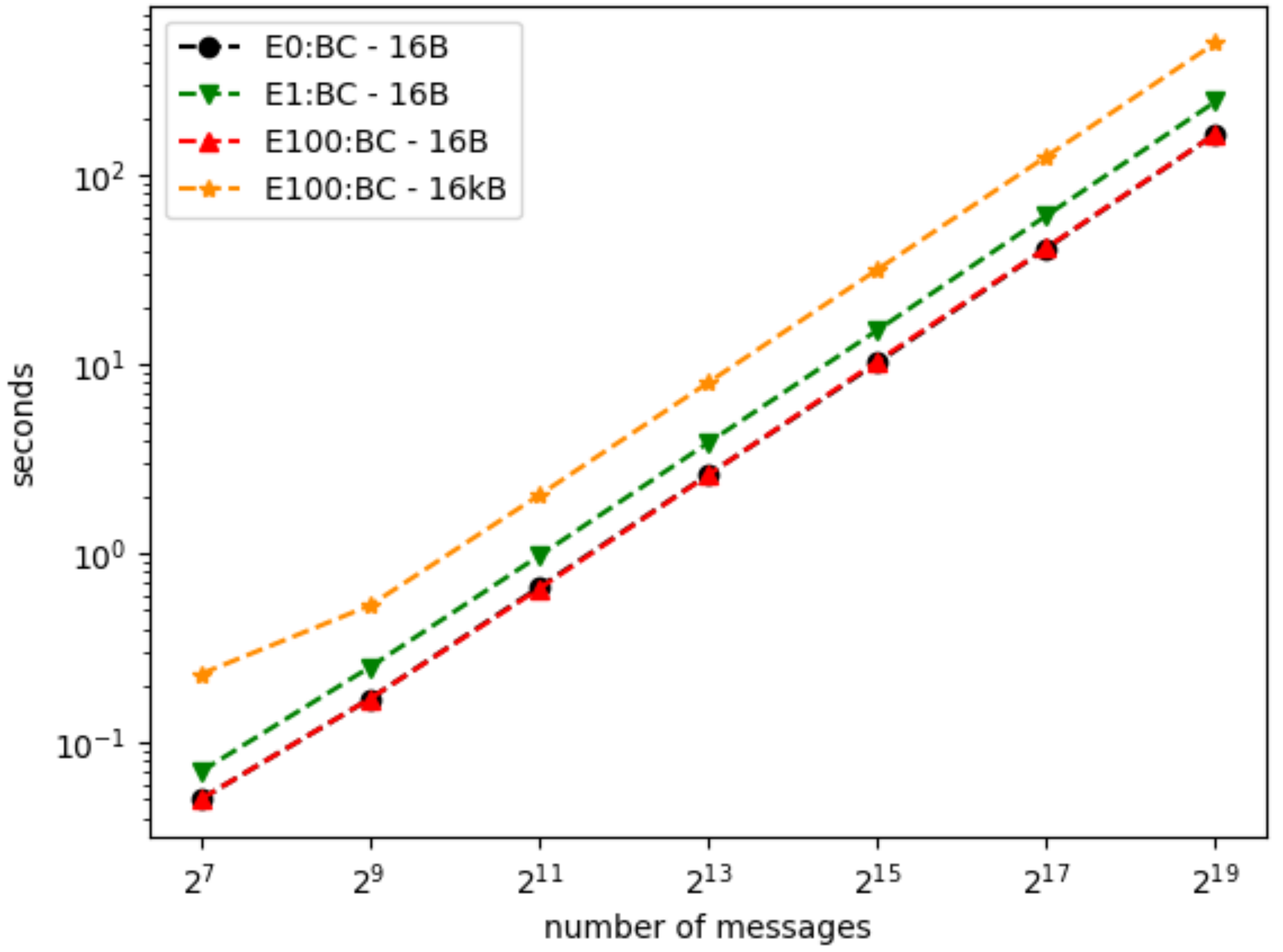
Experiments:

- **E0**: never erase messages
- **E1**: erase every message after retrieval
- **E100**: erase in batches of 100 messages
- **E100 with large messages**



3 parties
3 relays

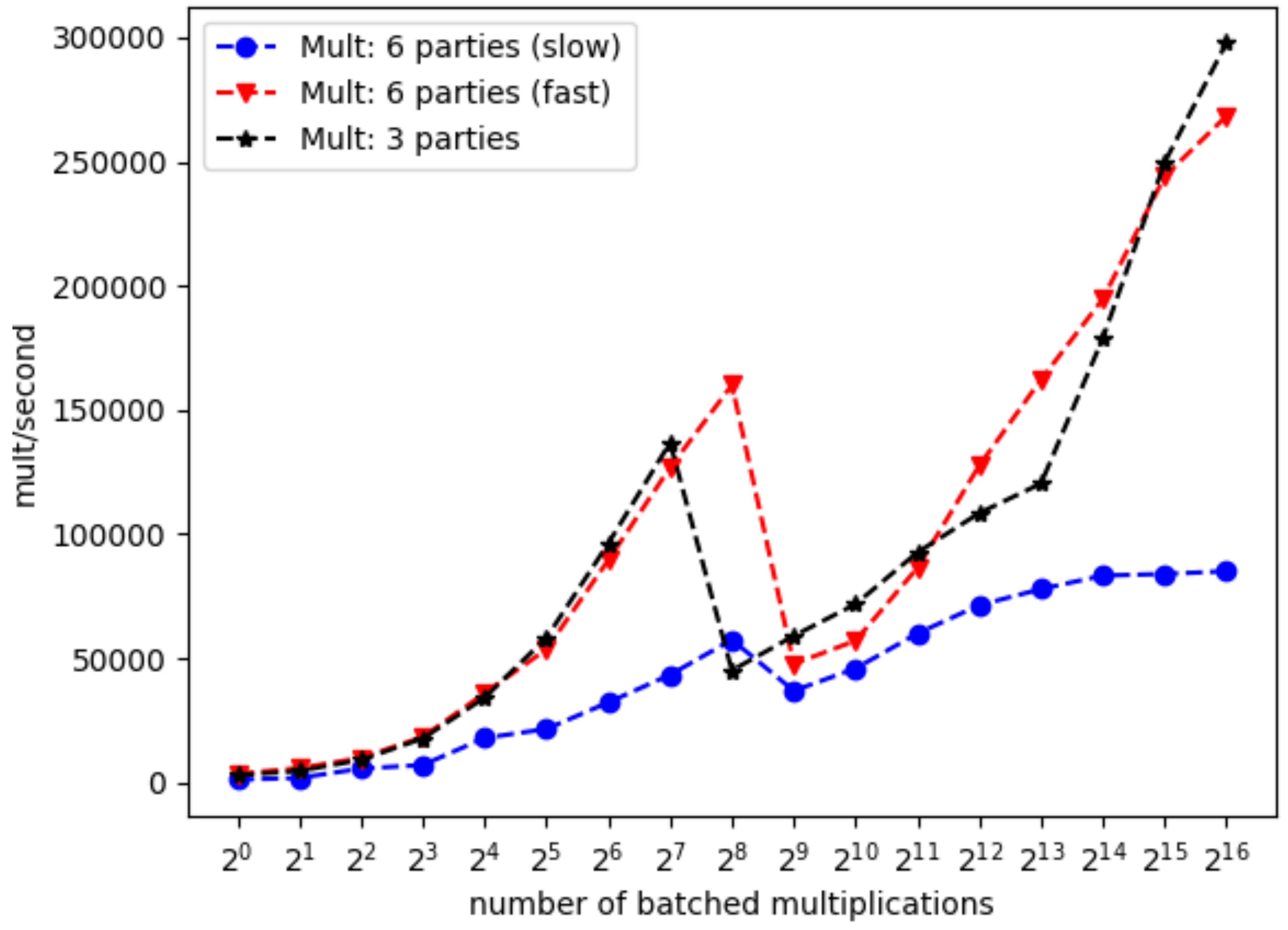
Running time for sending broadcast messages.



Experimental results

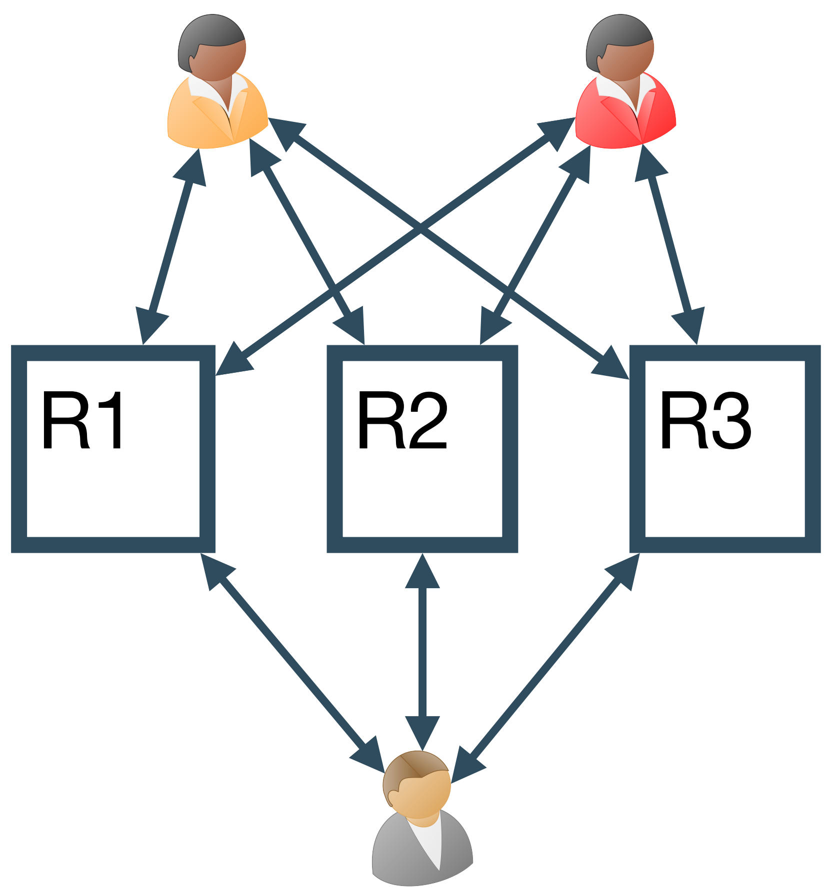
MPC multiplications

Multiplications per second for batched multiplications



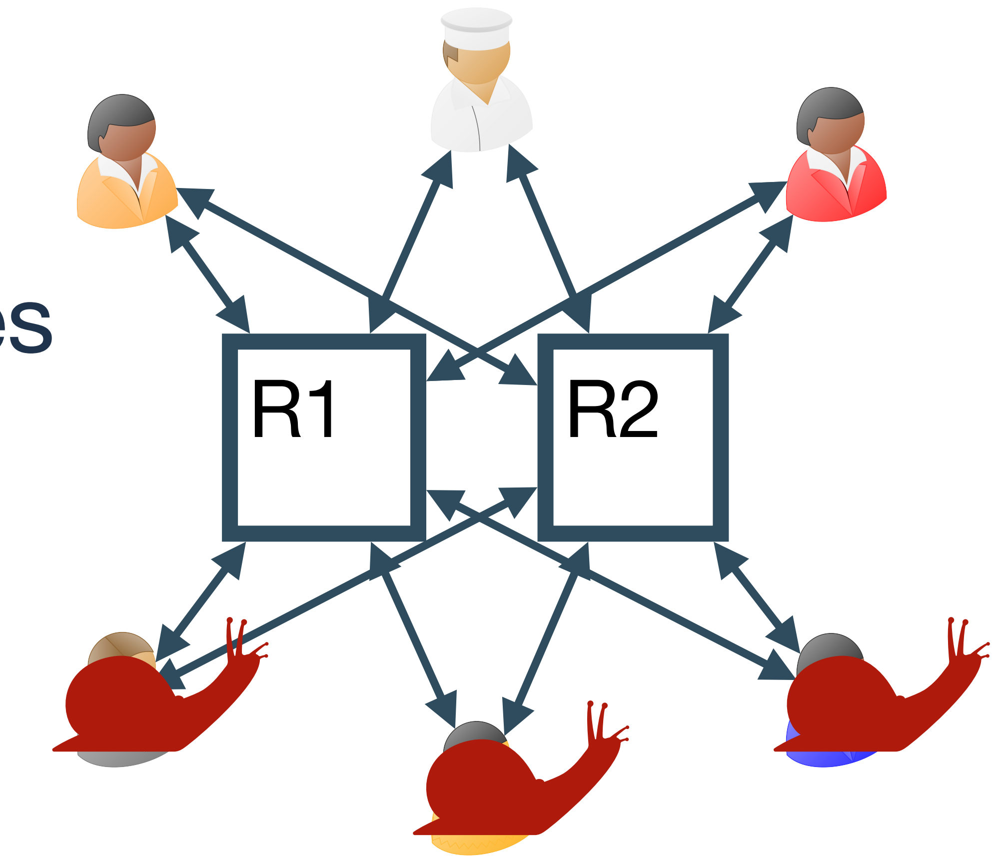
3 parties:

- At most 1 corruption



6 parties:

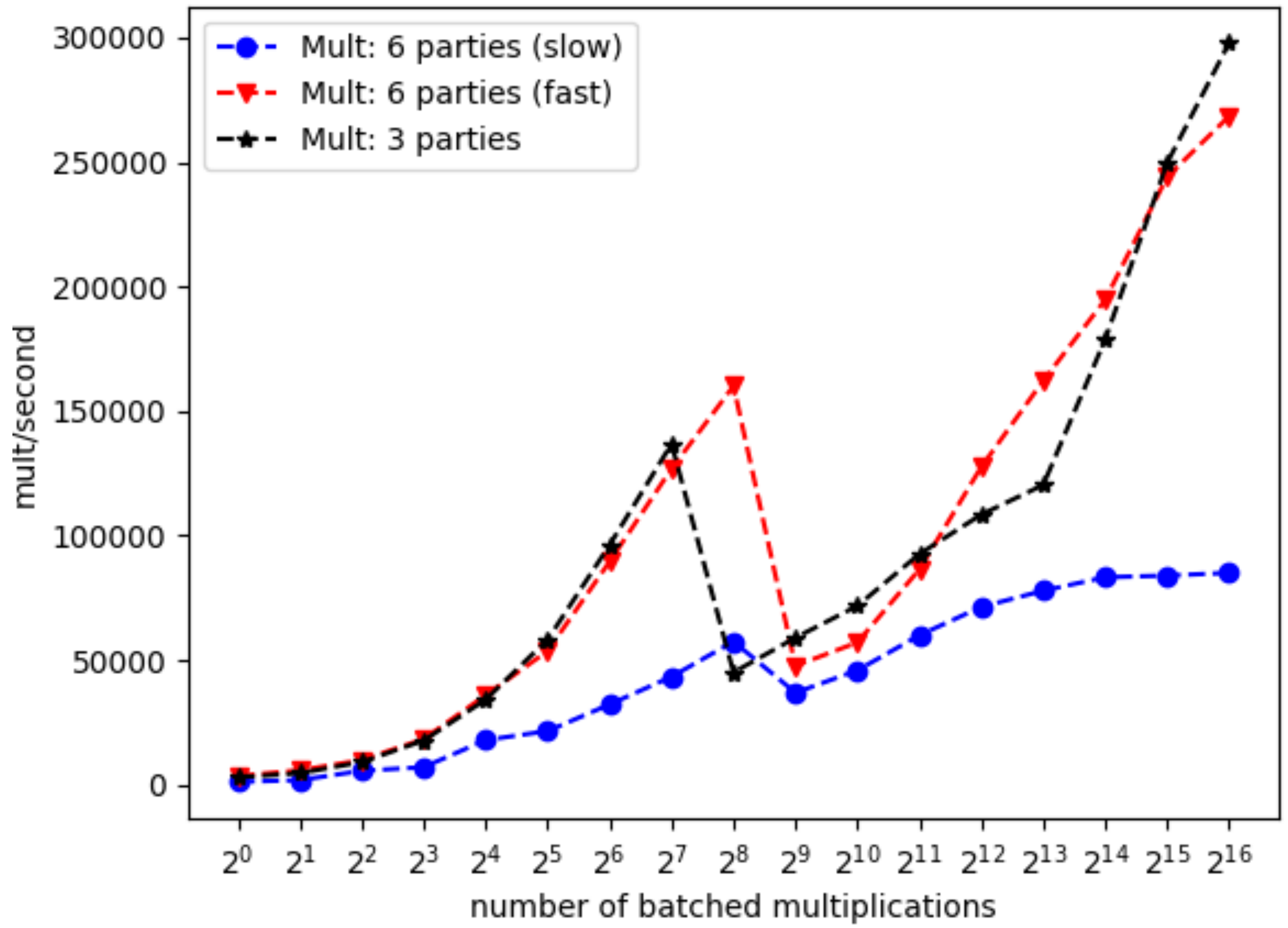
- At most 1 corruption
- 3 slow parties, 3 fast parties



Experimental results

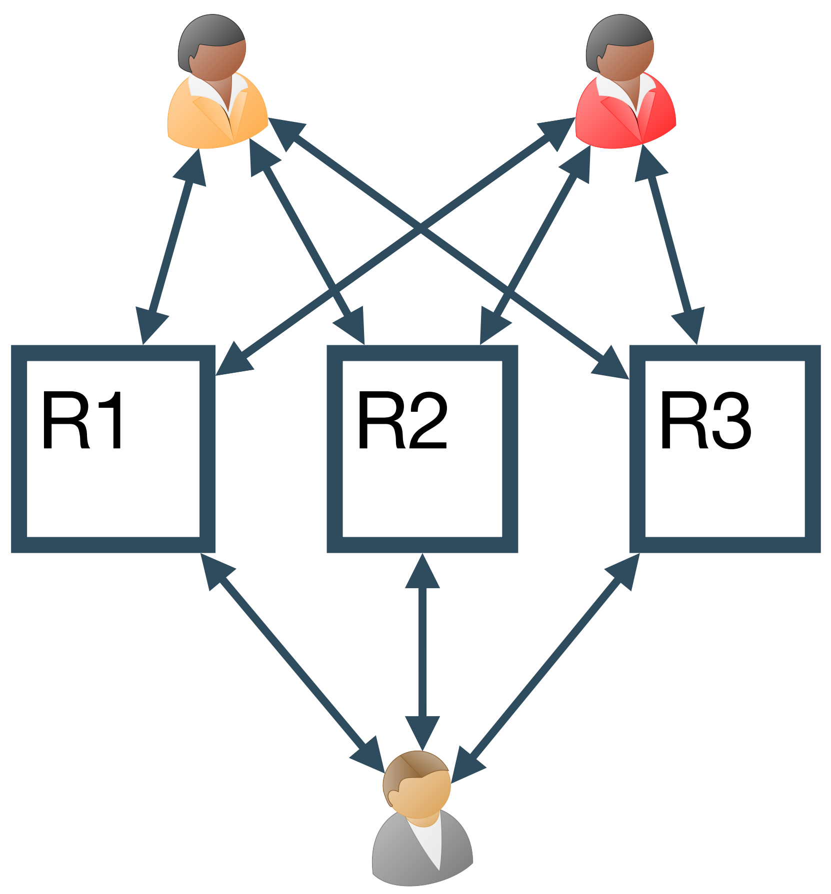
MPC multiplications

Multiplications per second for batched multiplications



3 parties:

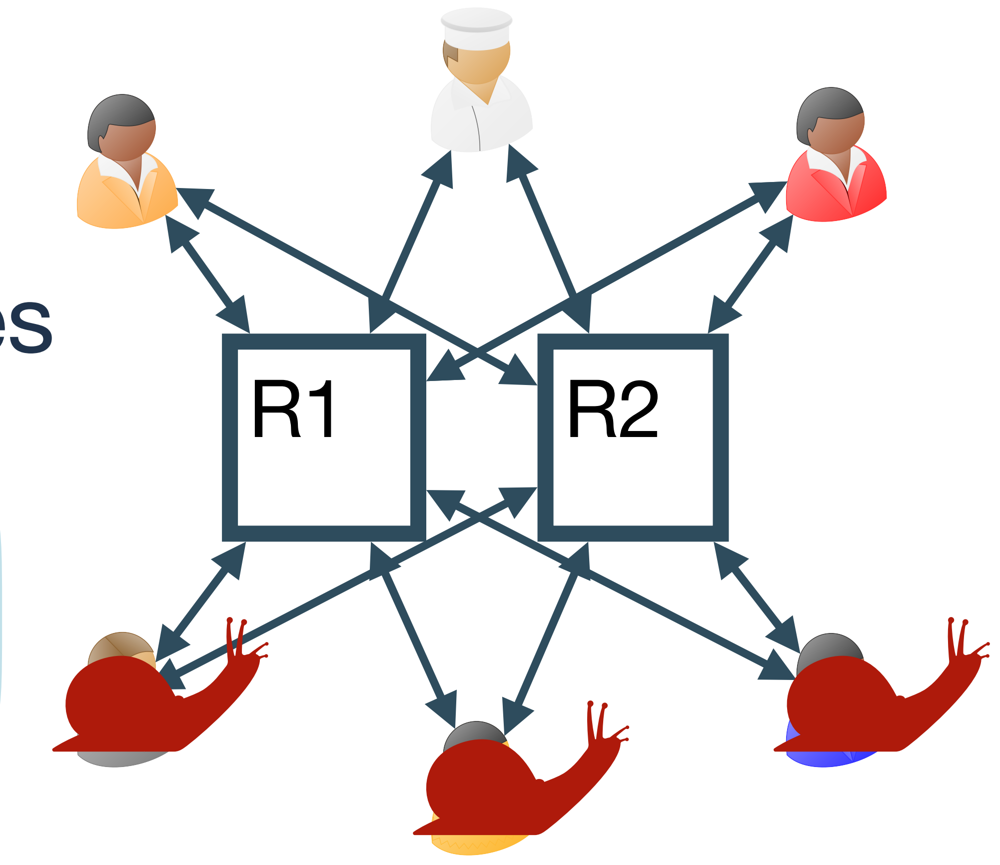
- At most 1 corruption



6 parties:

- At most 1 corruption
- 3 slow parties, 3 fast parties

Faster parties:
~ **270k multiplications/s**



Main takeaways

1. New MPC protocol addressing major constraints of deployed systems.
    - Star-like communication topology using relays
    - Secure even in the presence of delayed parties

2. Discussion on multiplication protocols with relays and delays

3. Implementation and experimental evaluation of the effect of relays.

Main takeaways

1. New MPC protocol addressing major constraints of deployed systems.
    - Star-like communication topology using relays
    - Secure even in the presence of delayed parties

2. Discussion on multiplication protocols with relays and delays

3. Implementation and experimental evaluation of the effect of relays.

Also in the paper:

- Key agreement
- Modelling the state size of relays
- Optimisation ideas for communication and round complexity