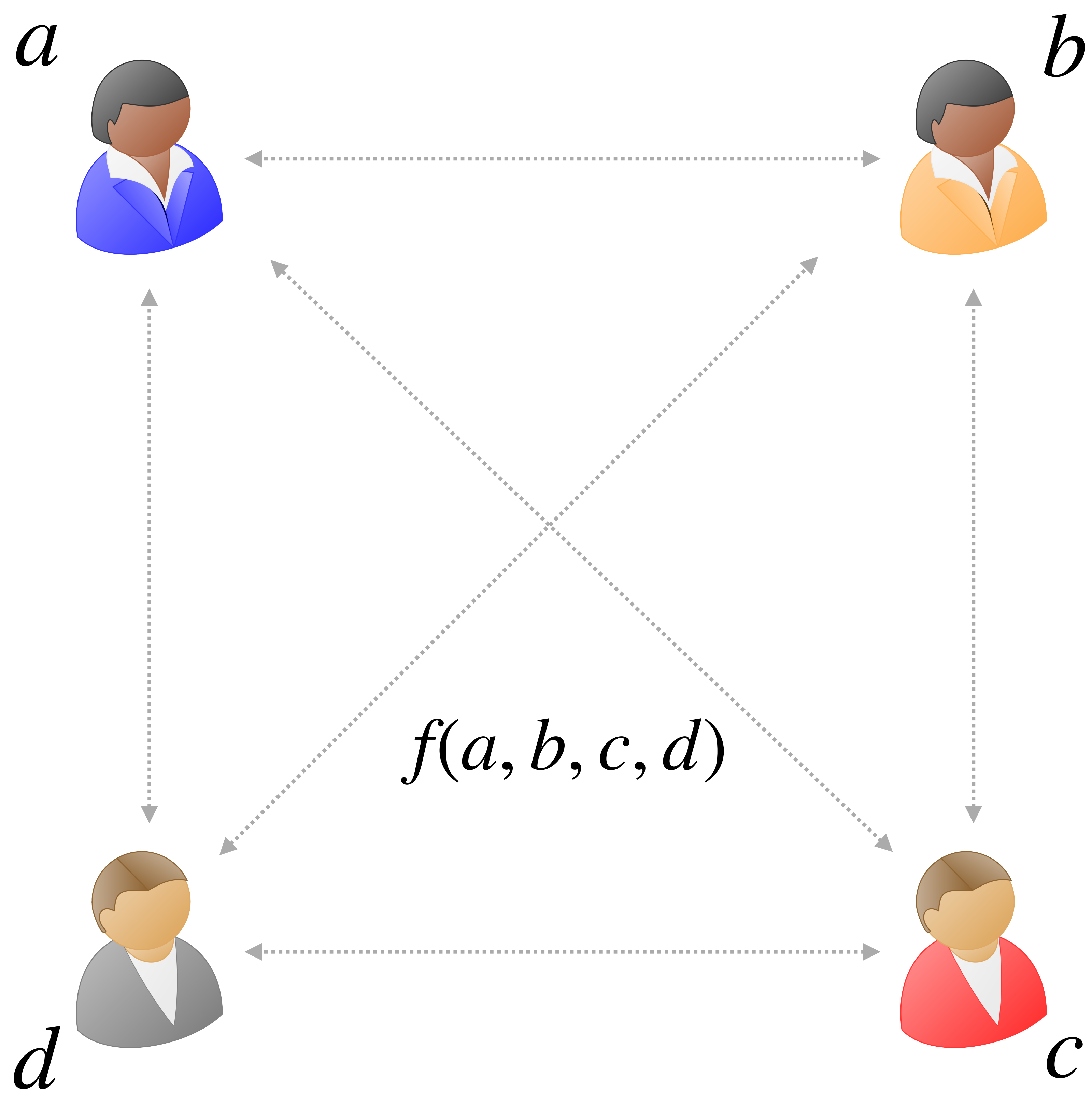
High Throughput Secure MPC Over Small Population in Hybrid Networks

Ashish Choudhury, Aditya Hegde

INDOCRYPT 2020

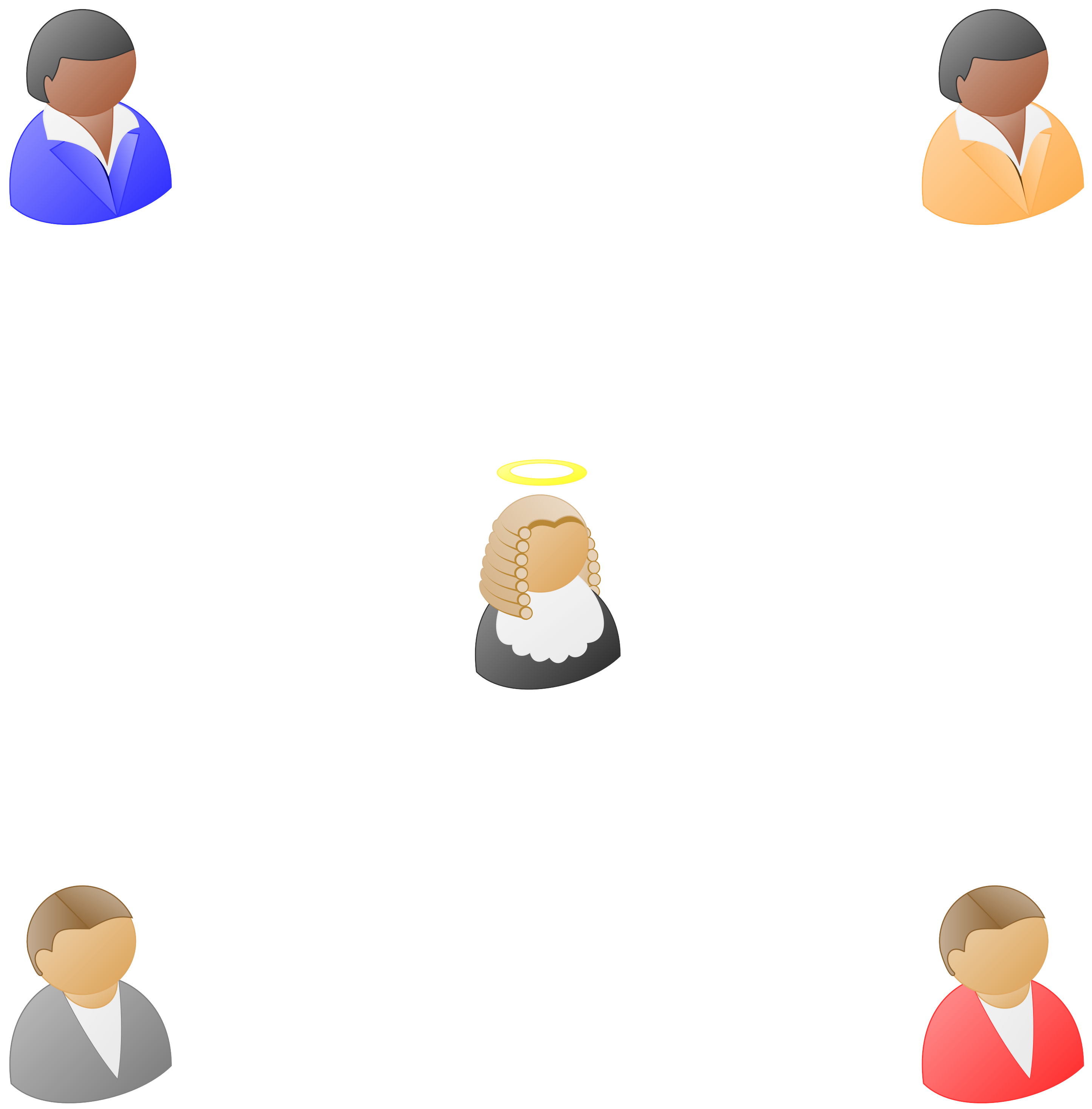
Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs



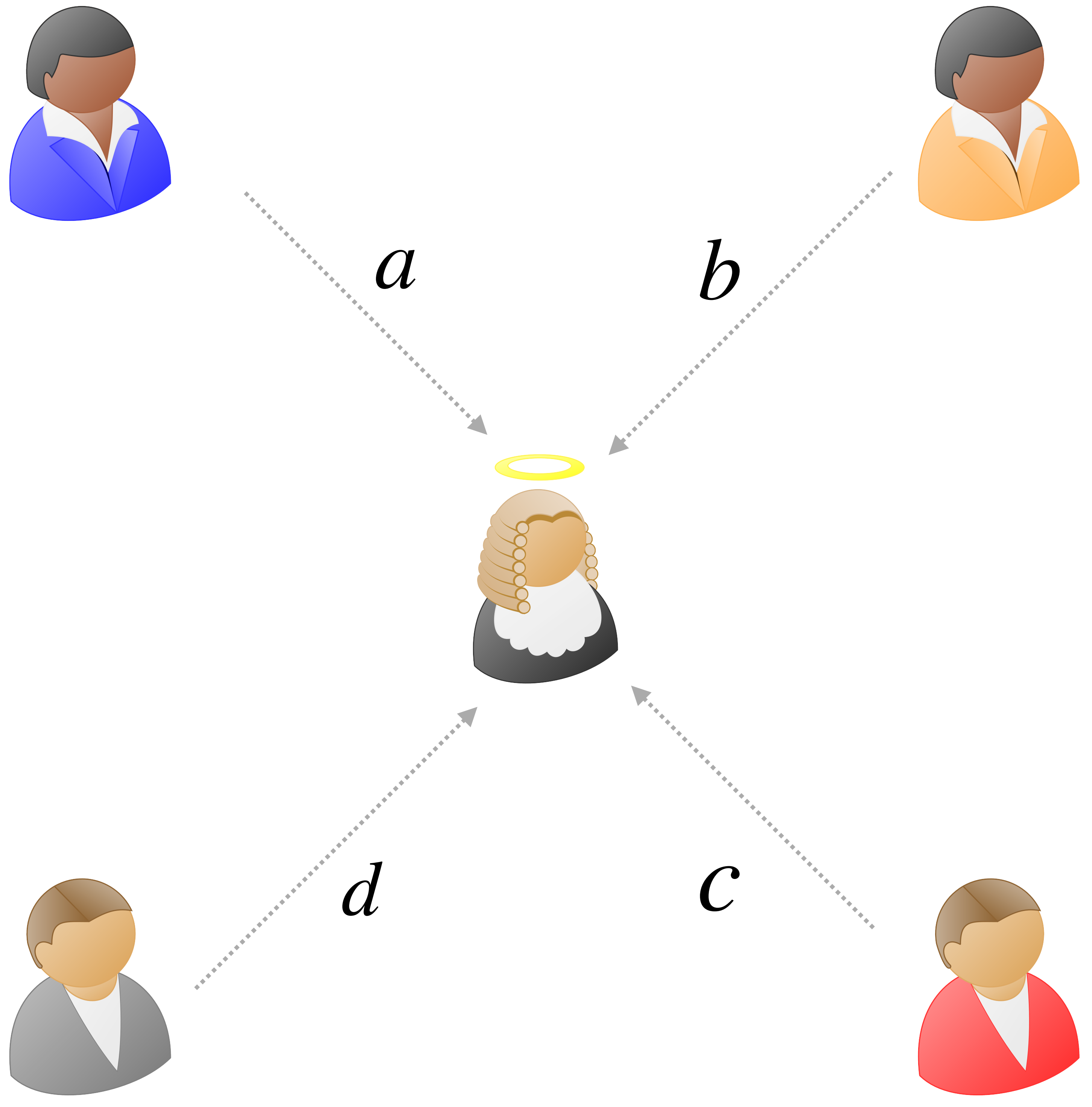
Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs
- Equivalent to interacting with a *Trusted Third Party (TTP)*



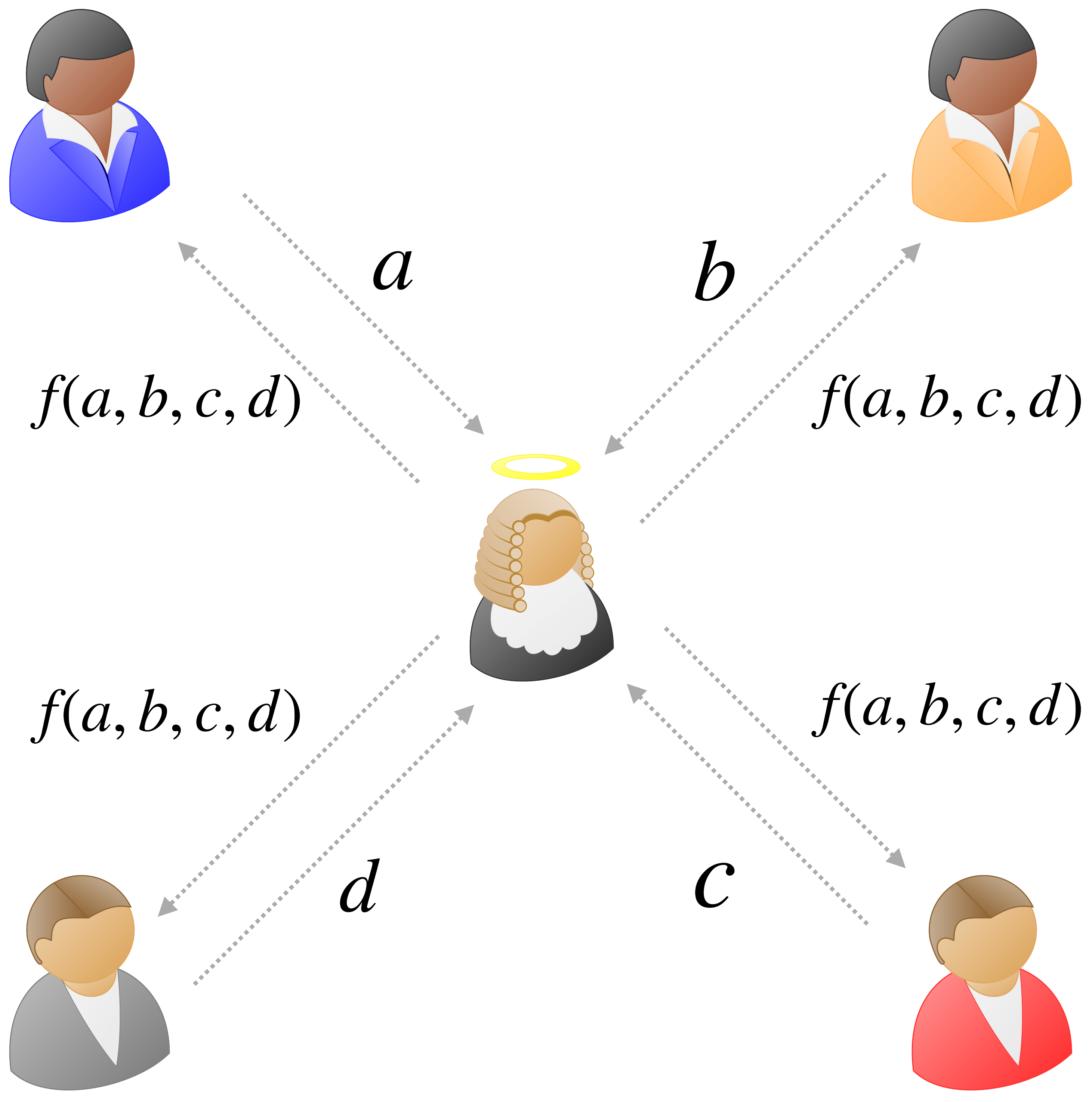
Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs
- Equivalent to interacting with a *Trusted Third Party (TTP)*



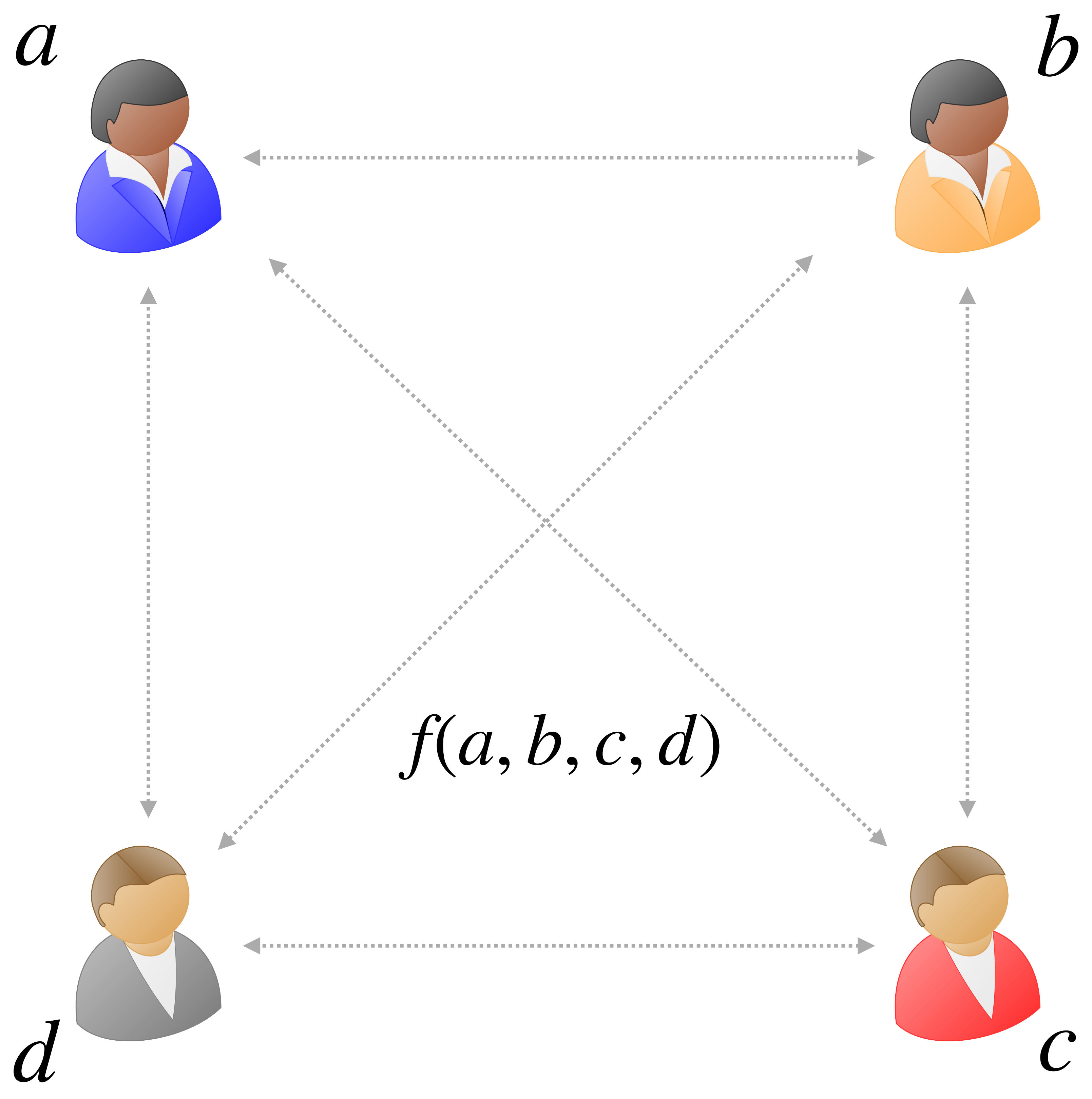
Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs
- Equivalent to interacting with a *Trusted Third Party (TTP)*



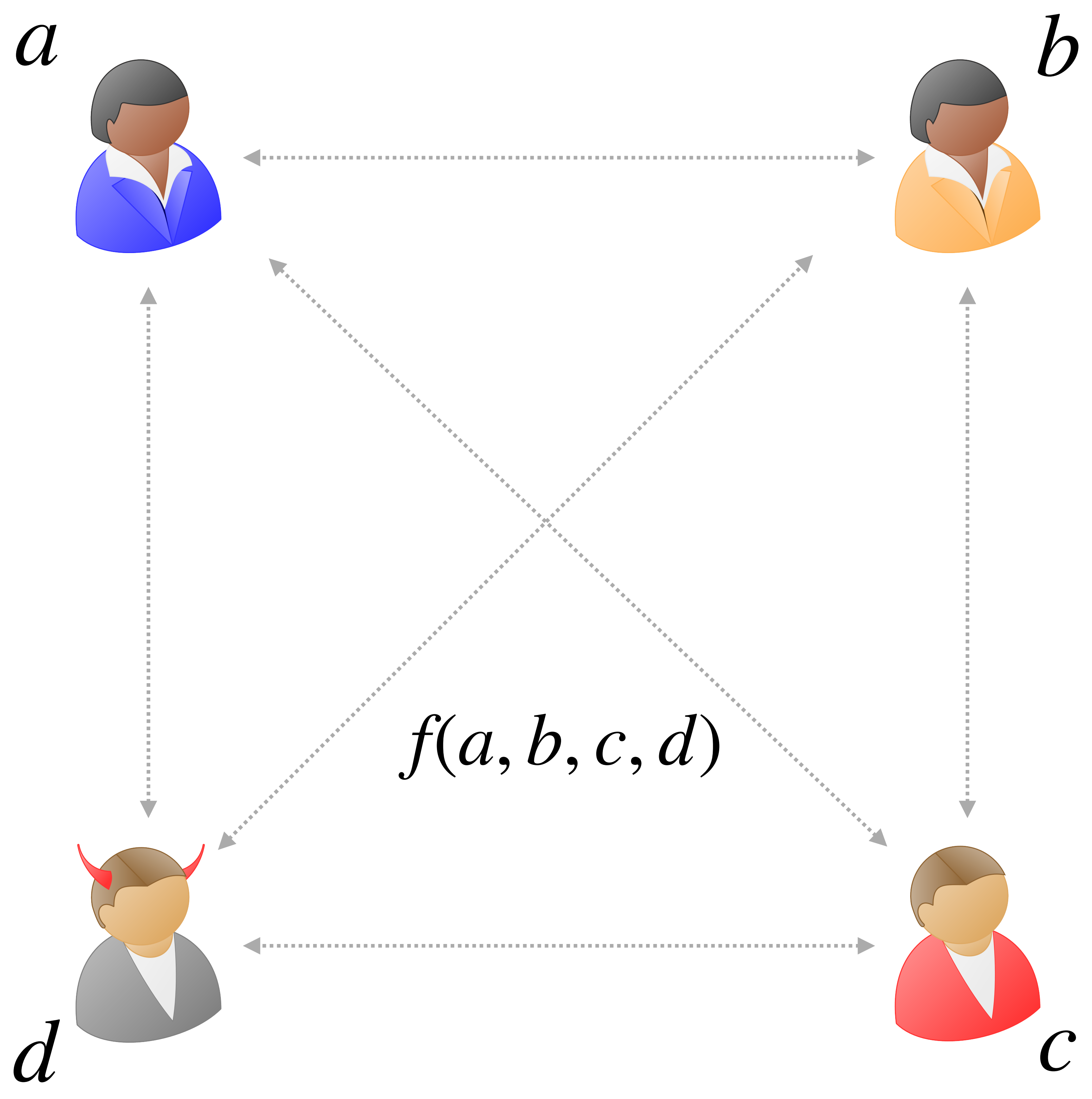
Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs
- Equivalent to interacting with a *Trusted Third Party (TTP)*
- MPC over small population
  - Simple and efficient $\Longrightarrow$ Practical applications



Secure Multi-Party Computation (MPC)

- Distrusting parties compute a function on private inputs
- Equivalent to interacting with a *Trusted Third Party (TTP)*
- MPC over small population
  - Simple and efficient $\Longrightarrow$ Practical applications
- Setting
  - $n = 4, t = 1$
  - Malicious adversary

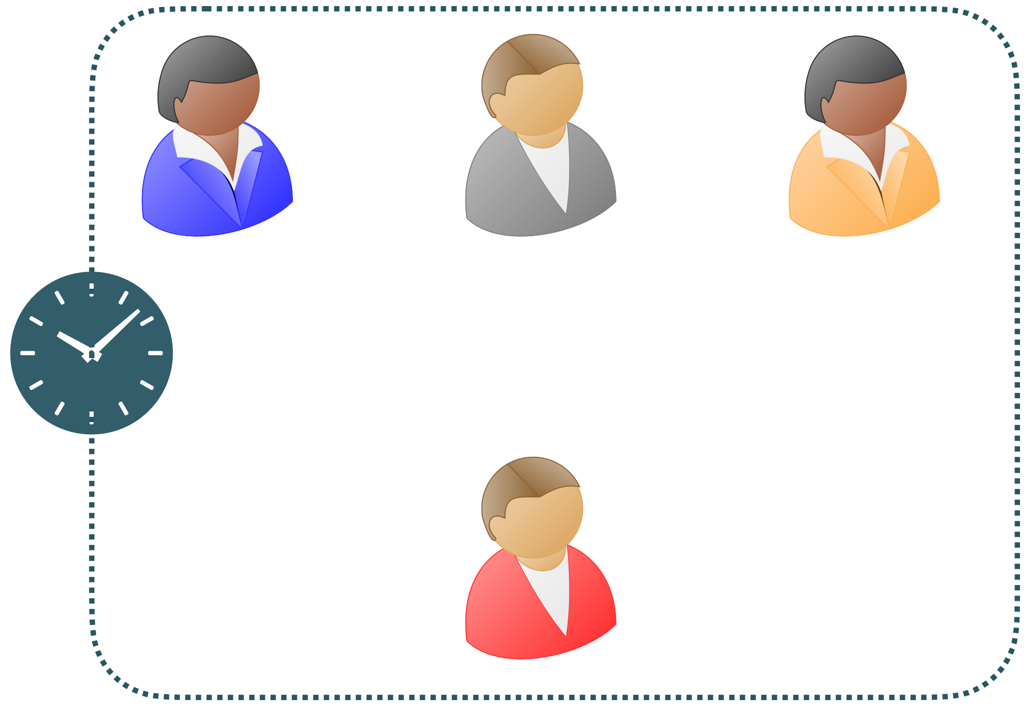


Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels

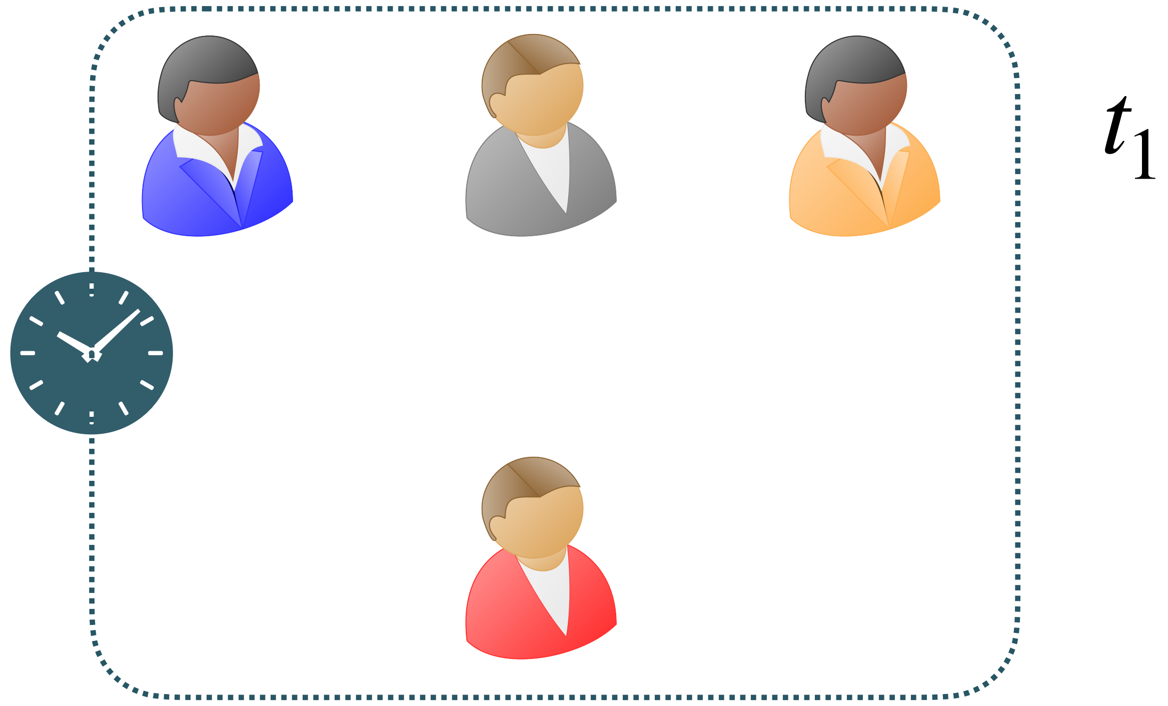
Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay



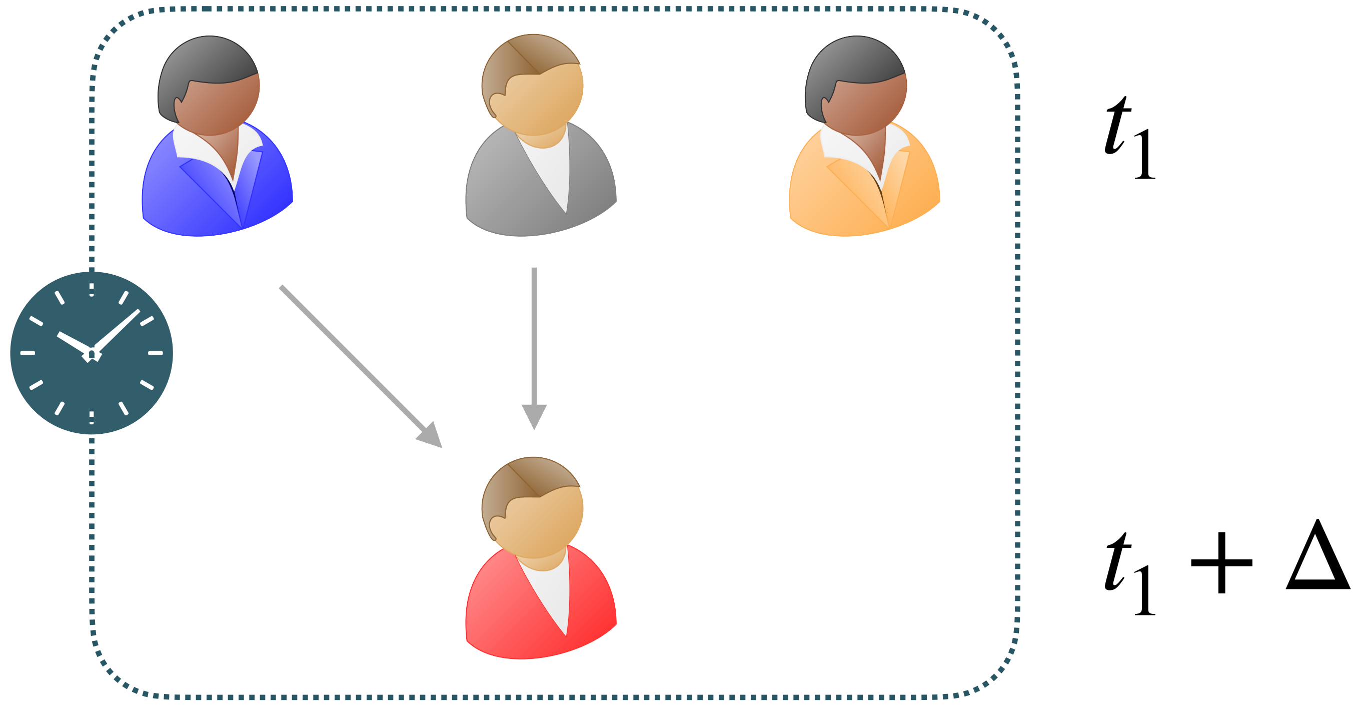
Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay



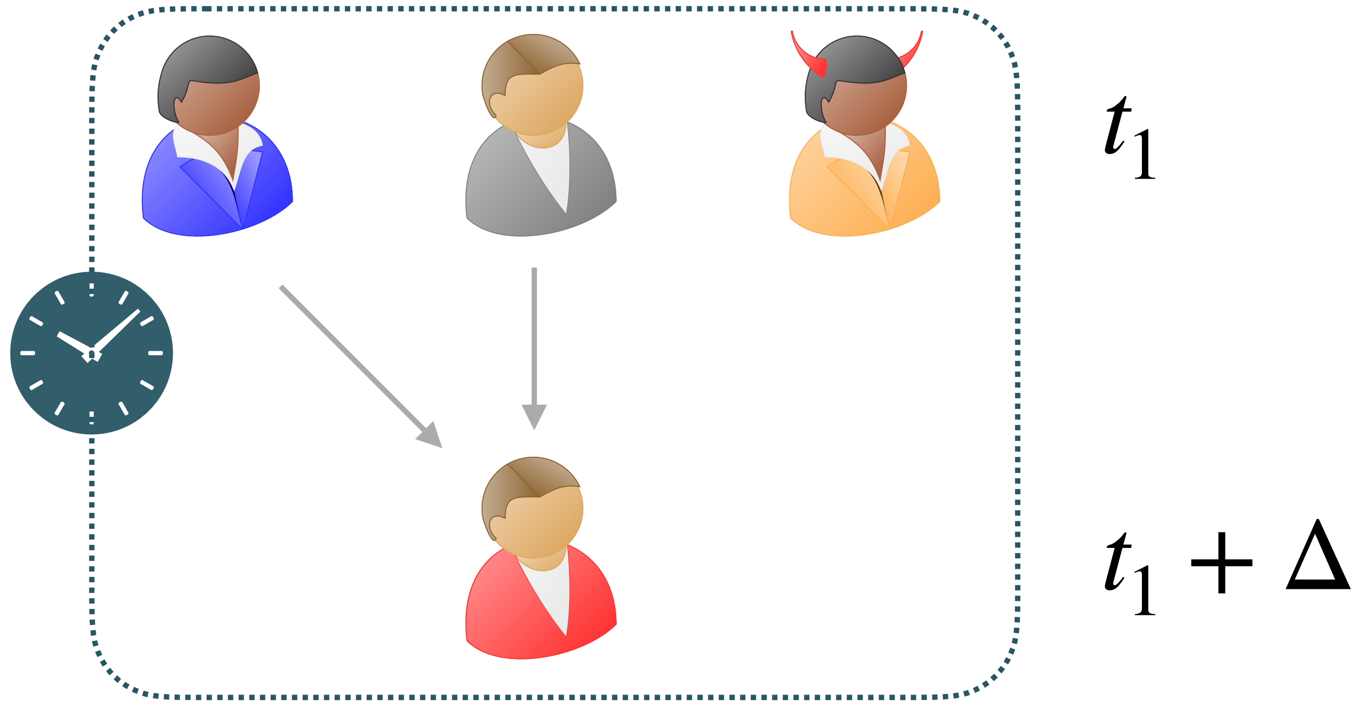
Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay



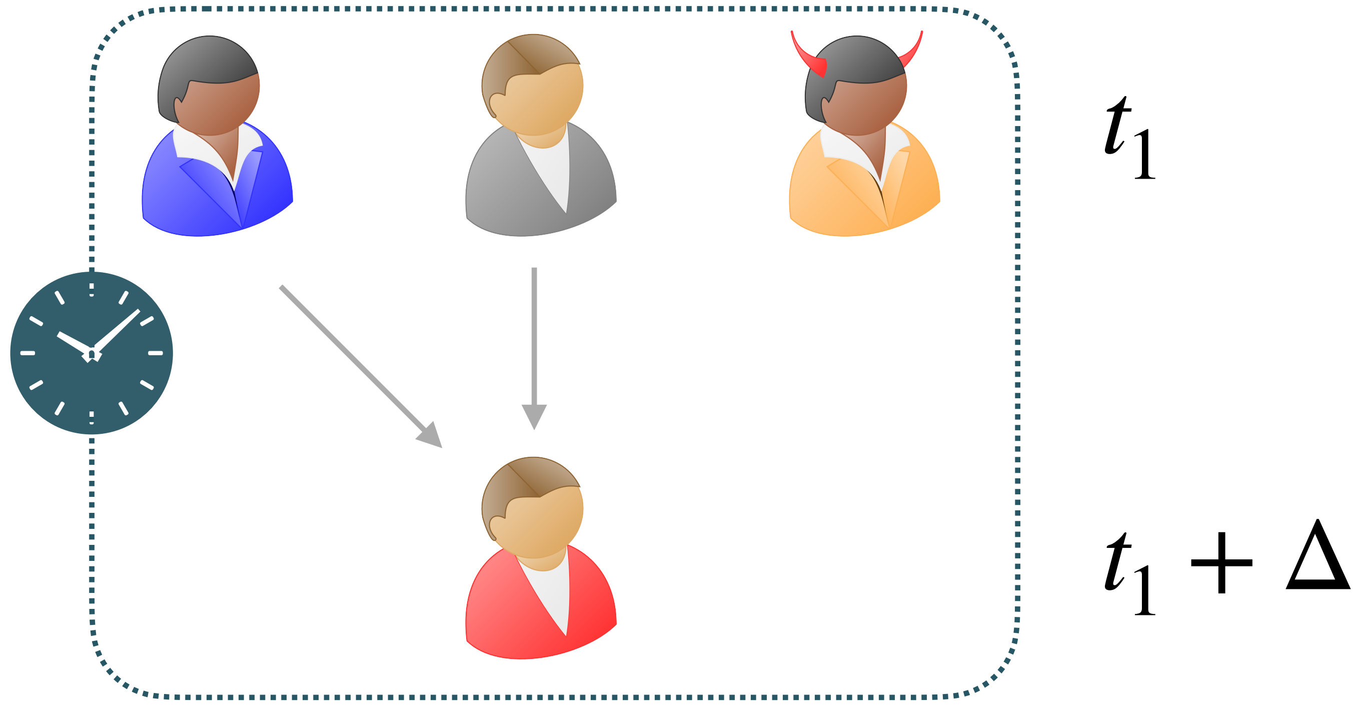
Communication Model - Synchronous and Asynchronous Networks

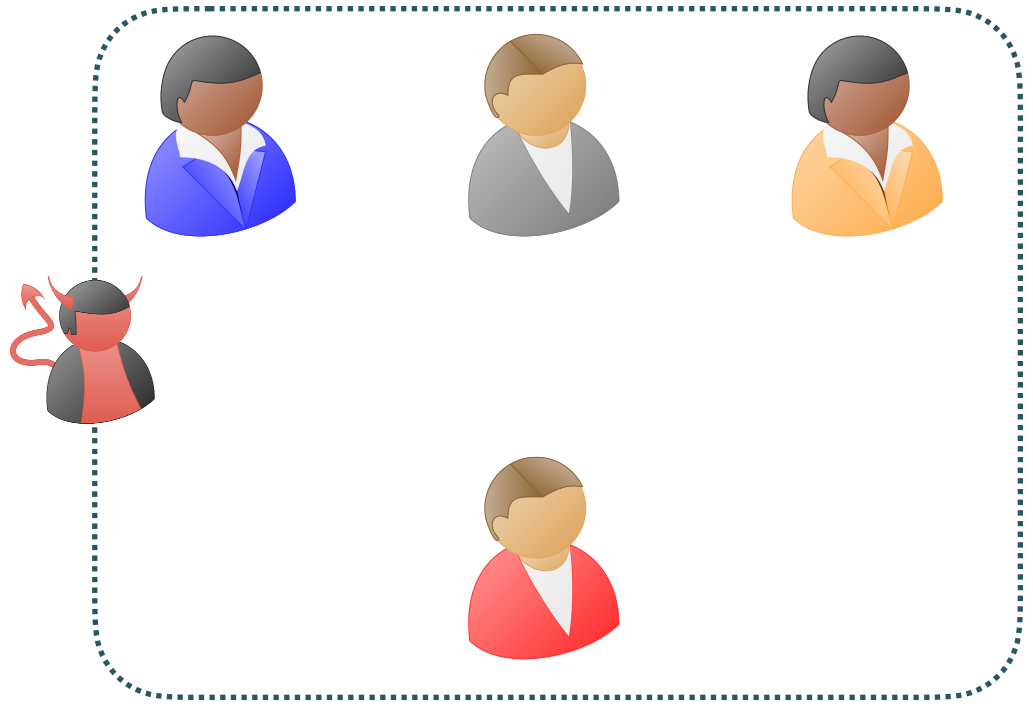
- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay



Communication Model - Synchronous and Asynchronous Networks

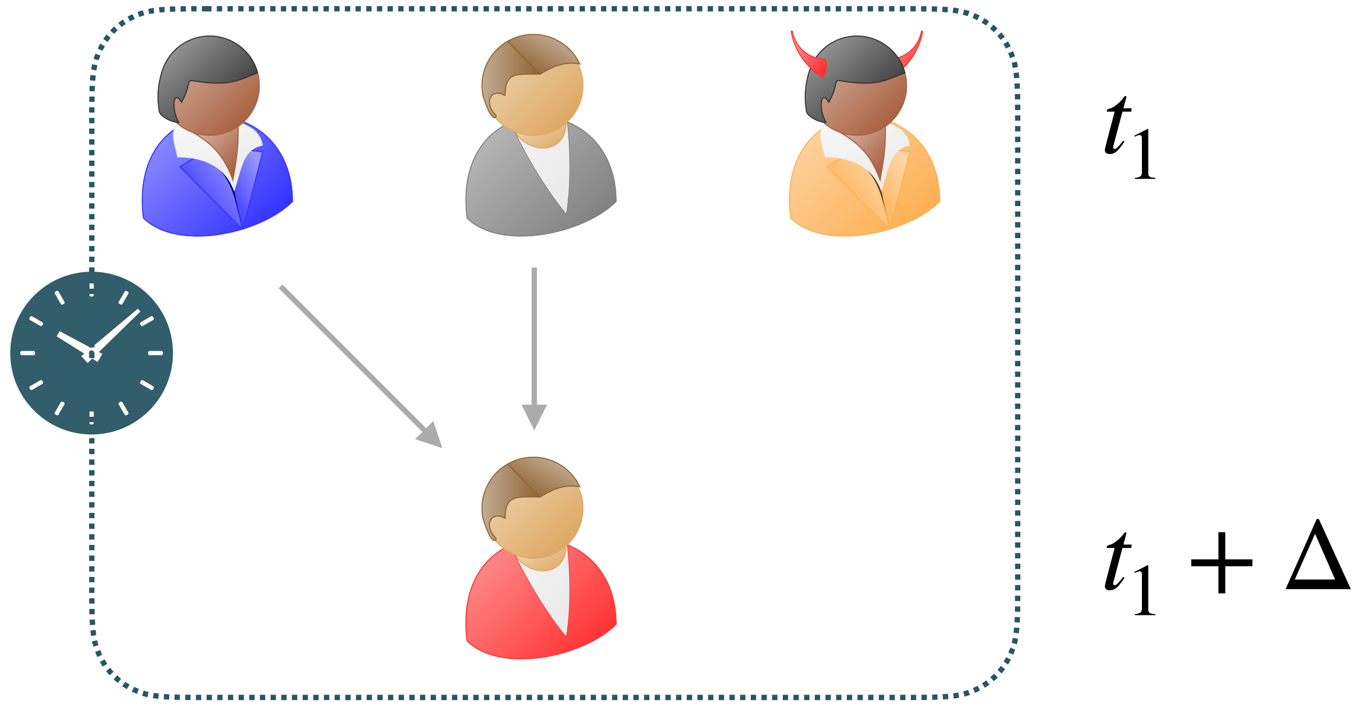
- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay
- Asynchronous networks [BCG93,Canetti95]
  - No synchronisation
  - Adversary schedules messages
  - Eventual delivery

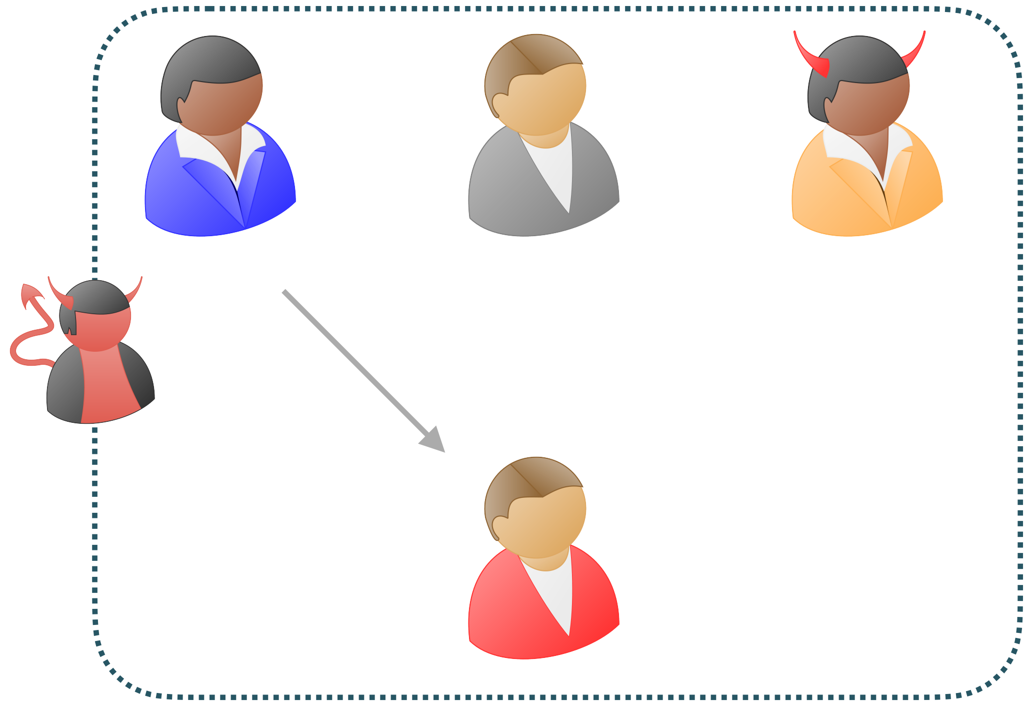




Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay
- Asynchronous networks [BCG93,Canetti95]
  - No synchronisation
  - Adversary schedules messages
  - Eventual delivery

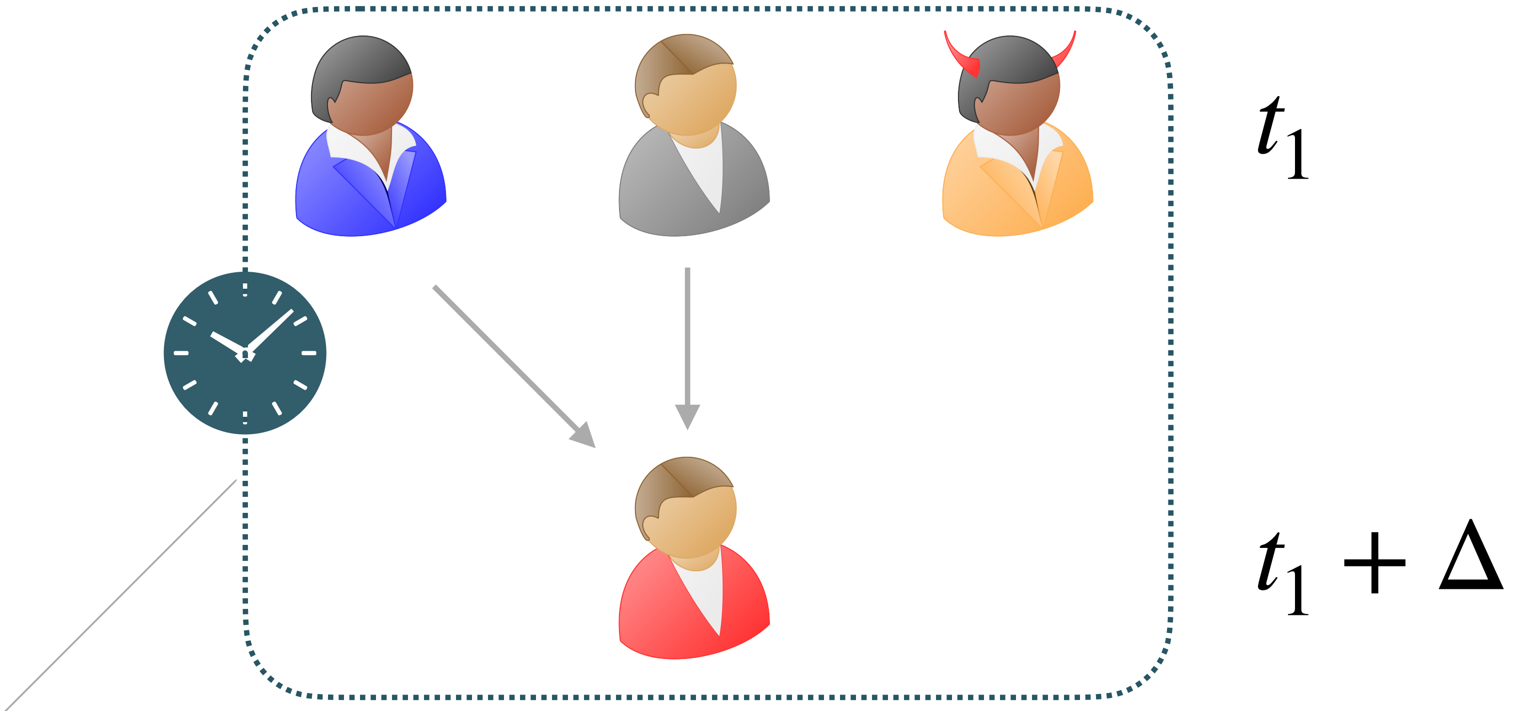


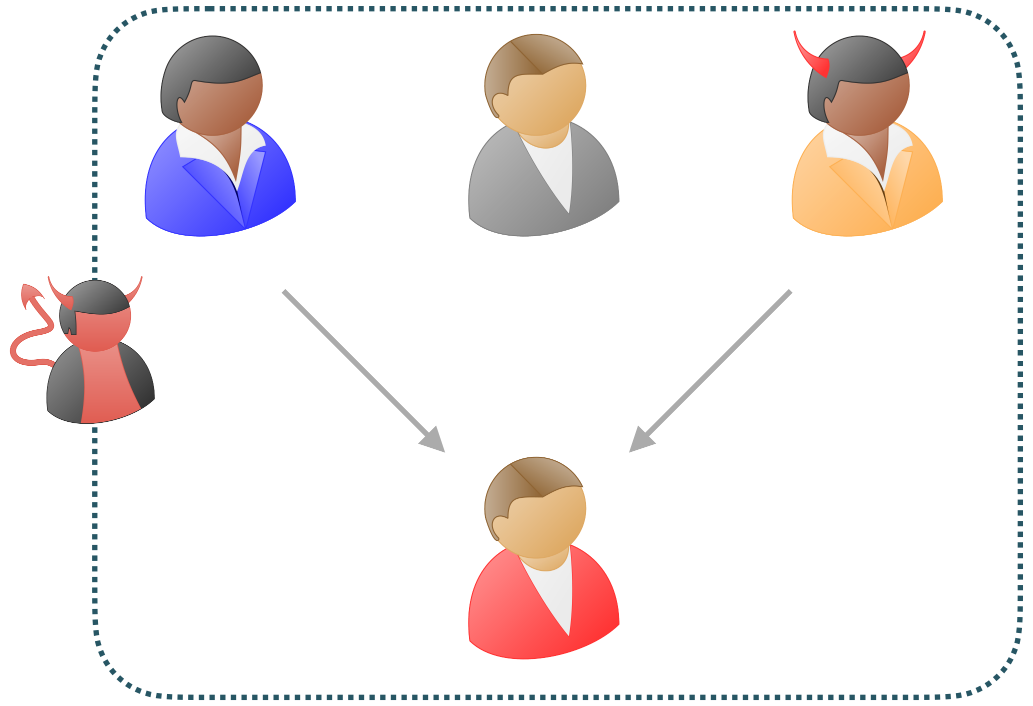


Cannot distinguish
between delayed and
unsent message

Communication Model - Synchronous and Asynchronous Networks

- Pairwise private and authentic channels
- Synchronous networks
  - Global clock
  - Publicly known upper bound on message delay
- Asynchronous networks [BCG93,Canetti95]
  - No synchronisation
  - Adversary schedules messages
  - Eventual delivery





Cannot distinguish between delayed and unsent message

Can only wait for $n - t$ messages at each step

Communication Model - Hybrid Networks

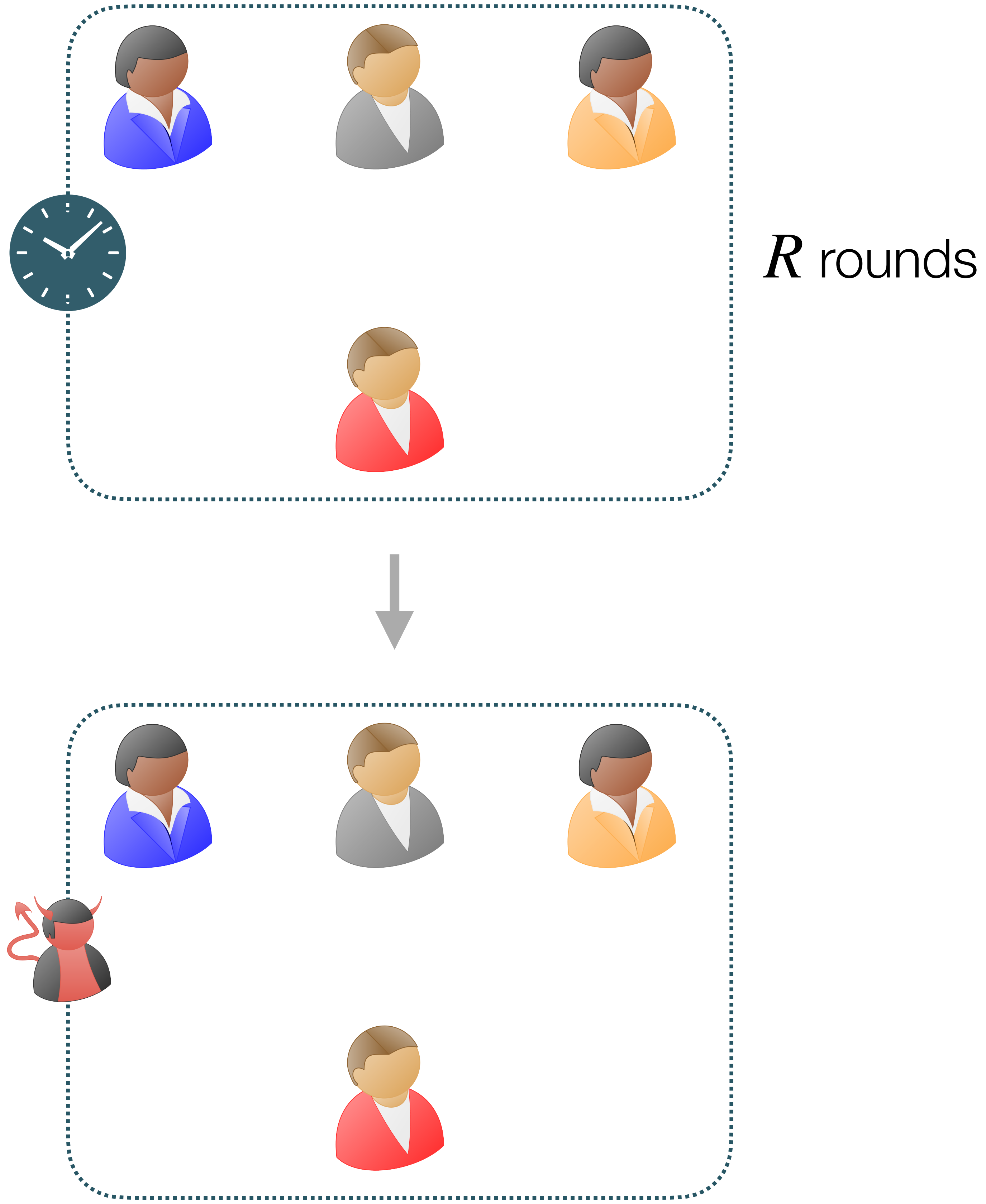
- Advantages of asynchronous networks
  - Better model for real-world networks
  - Small communication delays $\Longrightarrow$ faster than synchronous networks

Communication Model - Hybrid Networks

- Advantages of asynchronous networks
  - Better model for real-world networks
  - Small communication delays $\implies$ faster than synchronous networks
- Disadvantages of asynchronous networks
  - Lower fault tolerance
  - Input deprivation $\implies$ compute approximation of f

Communication Model - Hybrid Networks

- Advantages of asynchronous networks
  - Better model for real-world networks
  - Small communication delays $\Longrightarrow$ faster than synchronous networks
- Disadvantages of asynchronous networks
  - Lower fault tolerance
  - Input deprivation $\Longrightarrow$ compute approximation of f
- **Hybrid networks**: R initial synchronous rounds followed by asynchronous computation [BHN10,CHP13,PR18]
  - Assume synchronous broadcast channel in first R rounds



Our Contributions

- Perfectly secure MPC protocol over hybrid network with $R = 2$
  - First protocol in this setting
- Cryptographically secure MPC protocol over hybrid network with $R = 1$
- Cryptographically secure MPC protocol over asynchronous network

Our Contributions

- Perfectly secure MPC protocol over hybrid network with $R = 2$
  - First protocol in this setting
- Cryptographically secure MPC protocol over hybrid network with $R = 1$

Optimal number of synchronous rounds

- Cryptographically secure MPC protocol over asynchronous network

Our Contributions

- Perfectly secure MPC protocol over hybrid network with $R = 2$
  - First protocol in this setting

- Cryptographically secure MPC protocol over hybrid network with $R = 1$

- Cryptographically secure MPC protocol over asynchronous network

Optimal number of synchronous rounds

Rely only on symmetric key primitives

Our Contributions

- Perfectly secure MPC protocol over hybrid network with $R = 2$
  - First protocol in this setting
- Cryptographically secure MPC protocol over hybrid network with $R = 1$
- Cryptographically secure MPC protocol over asynchronous network

Optimal number of synchronous rounds

Optimal resilience and Guaranteed Output Delivery

Rely only on symmetric key primitives

Our Contributions

- Perfectly secure MPC protocol over hybrid network with $R = 2$
  - First protocol in this setting
- Cryptographically secure MPC protocol over hybrid network with $R = 1$
- Cryptographically secure MPC protocol over asynchronous network
  - Implementation and benchmarks

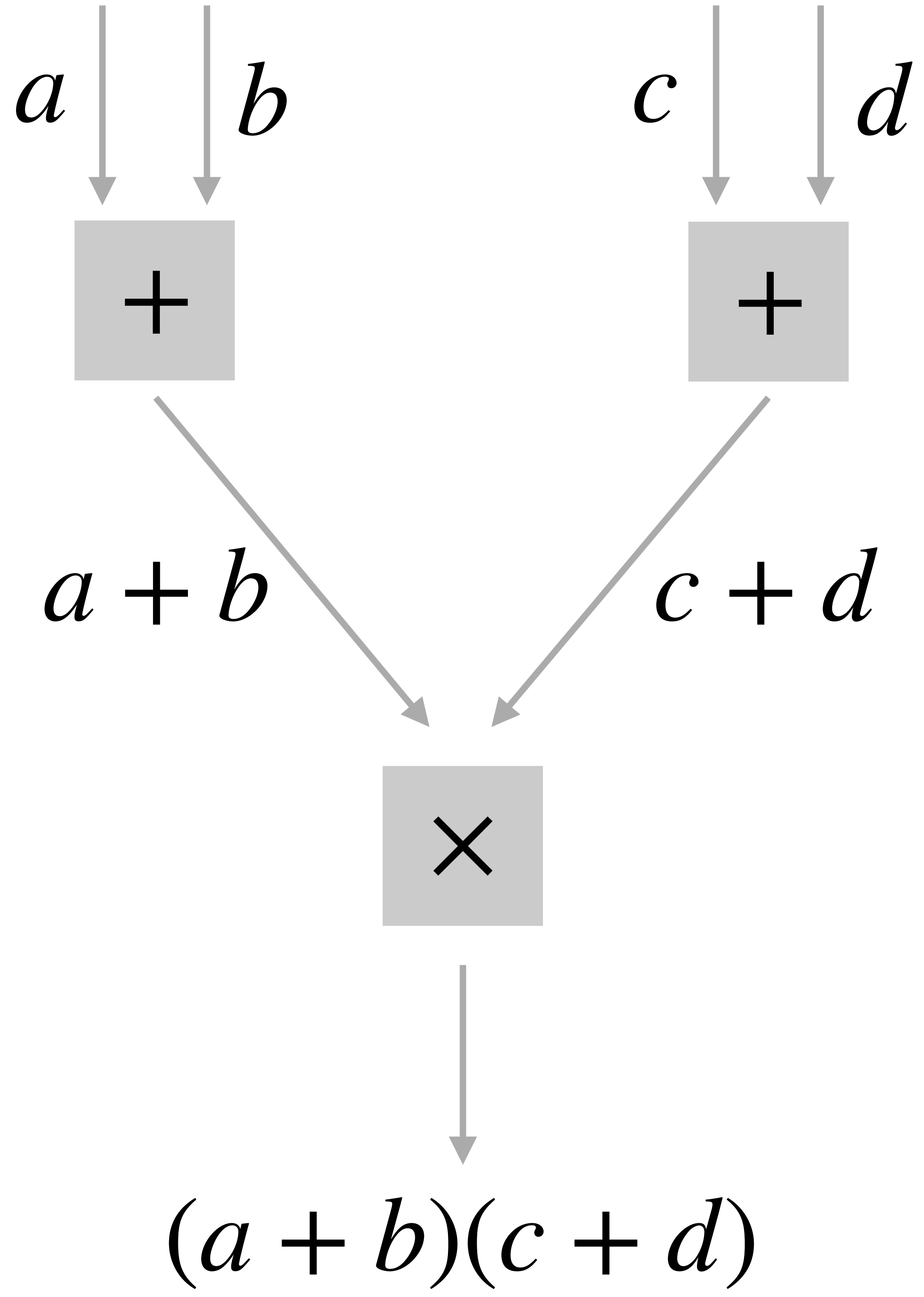
Optimal number of synchronous rounds

Optimal resilience and Guaranteed Output Delivery

Rely only on symmetric key primitives

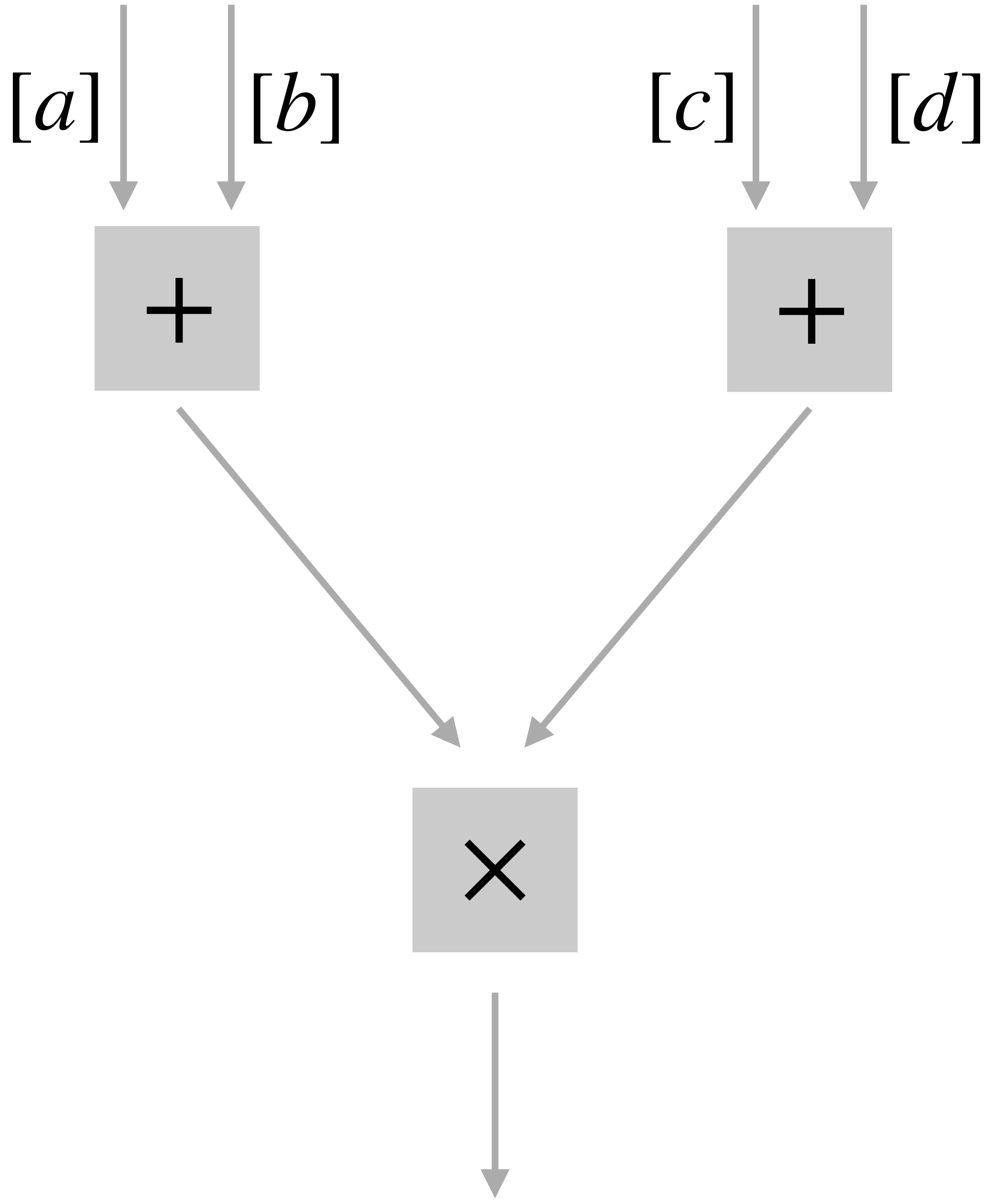
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field



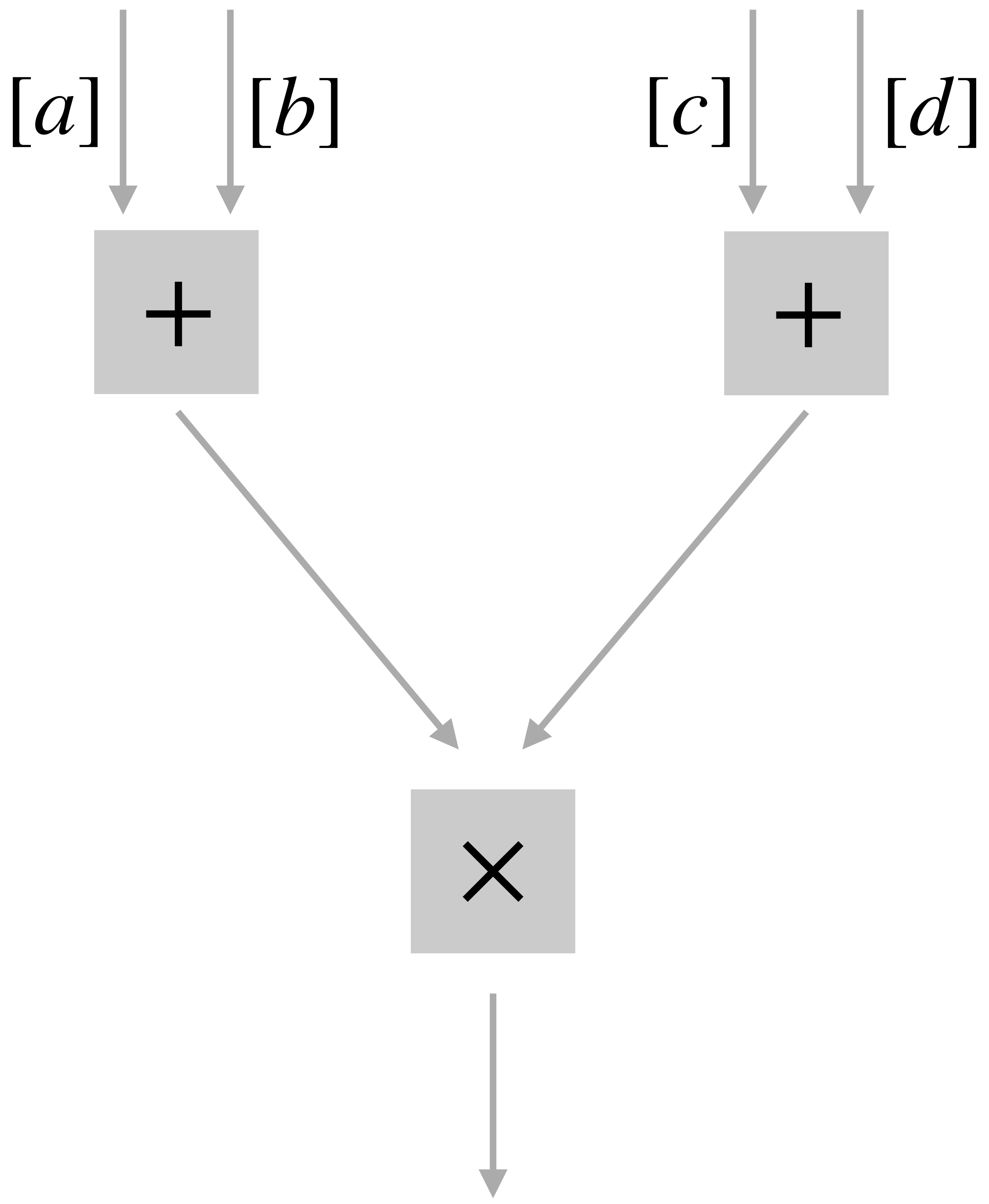
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput



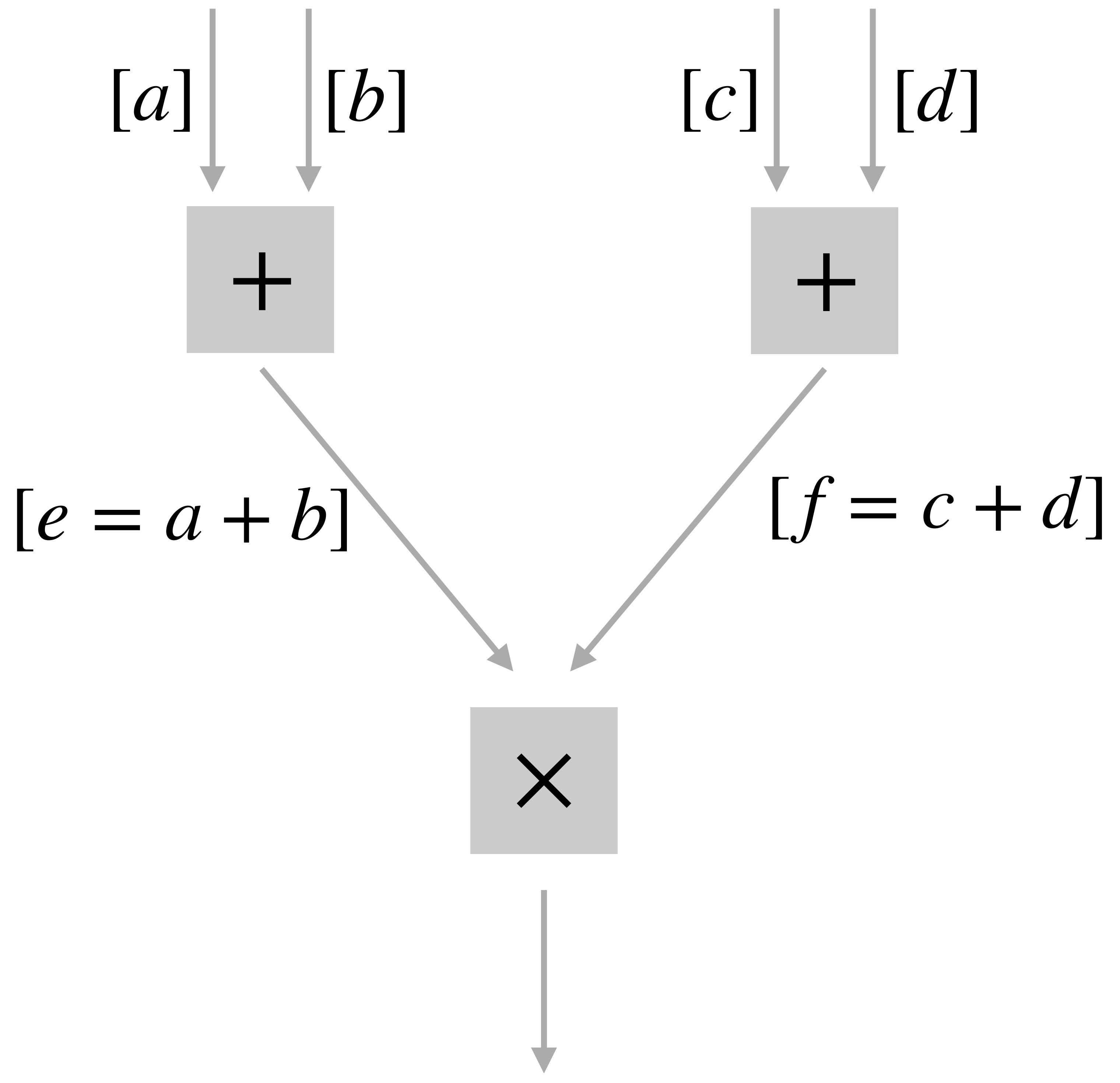
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$



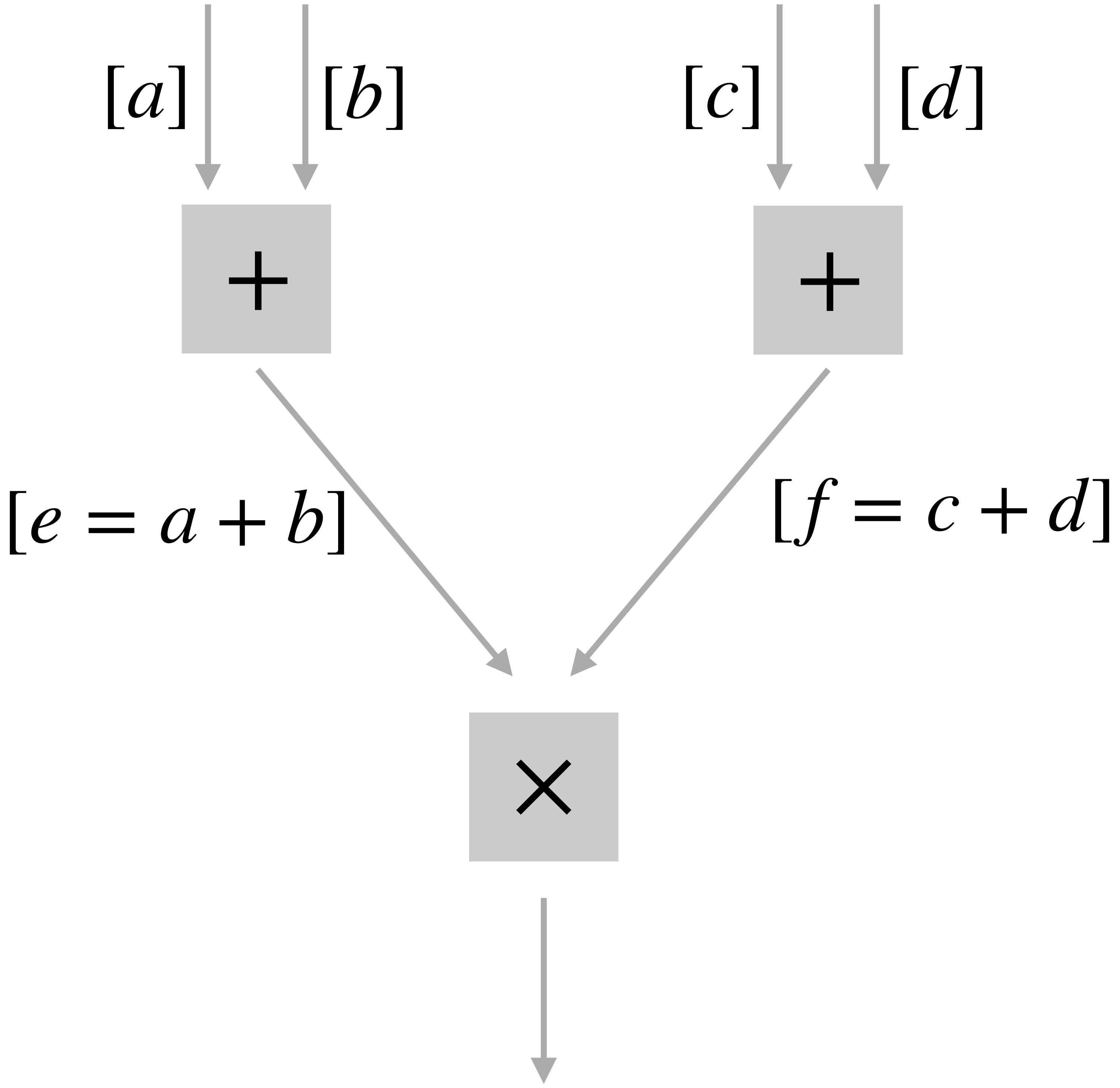
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation



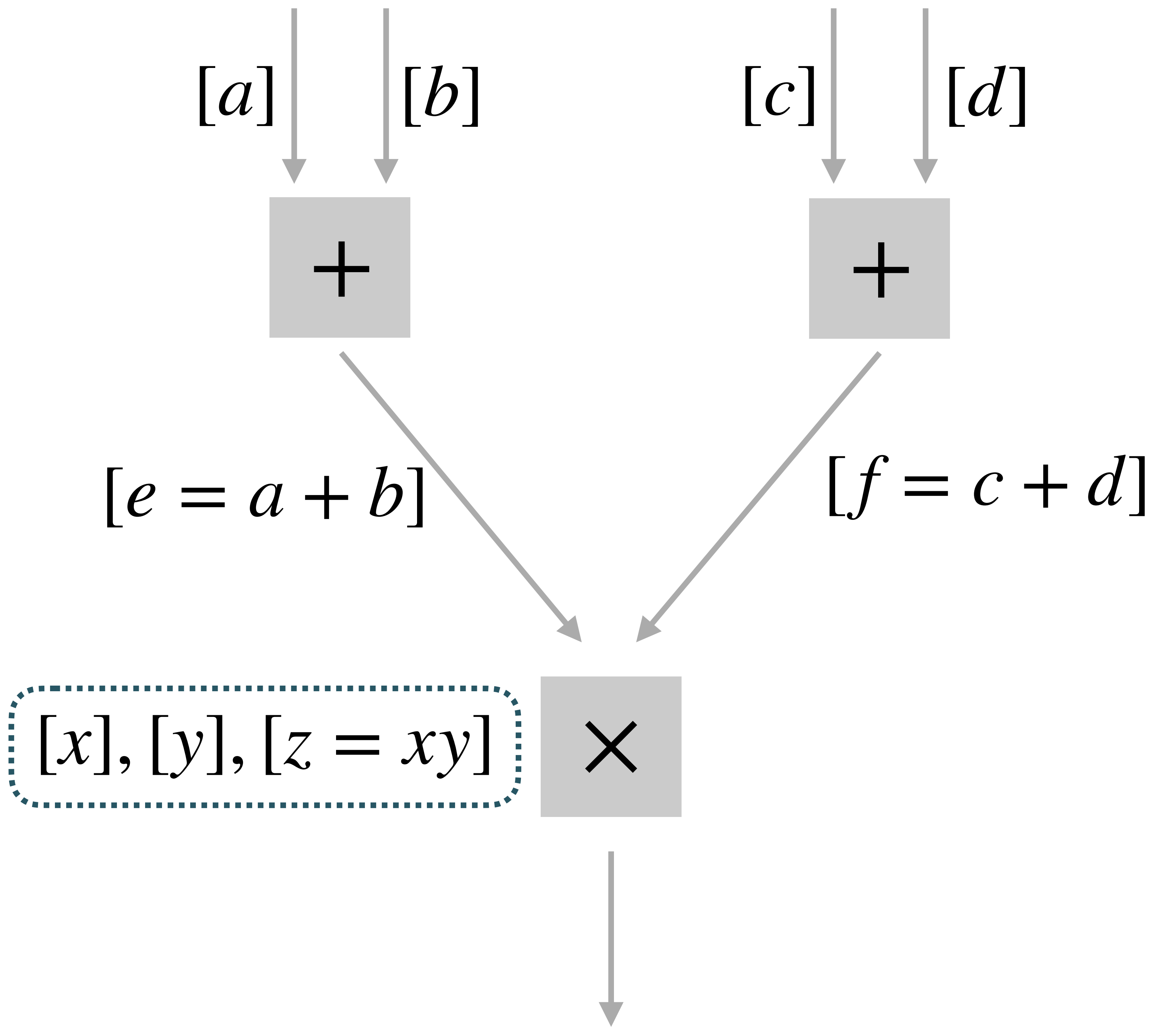
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation
  - Addition gates: Locally evaluated



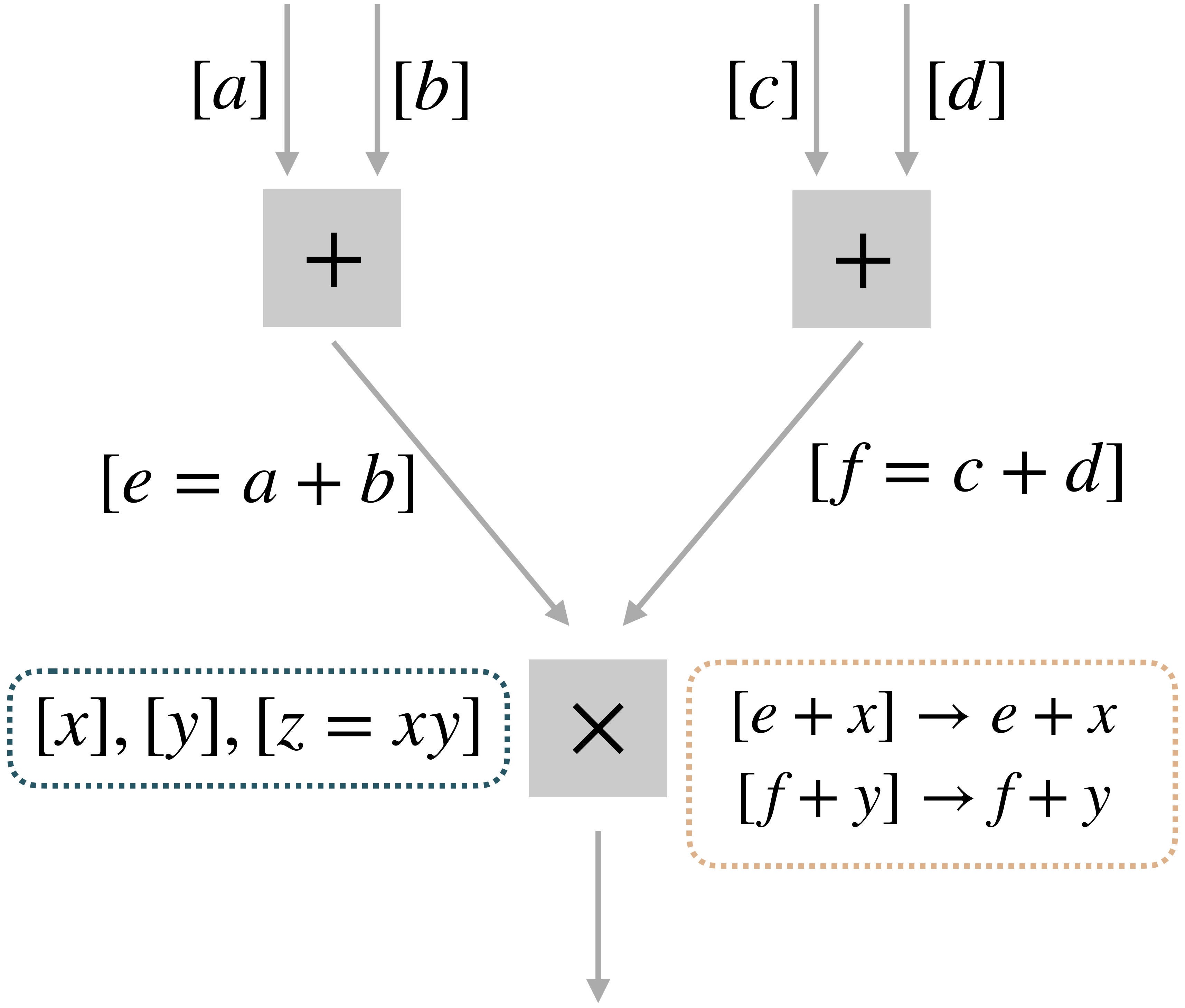
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation
  - Addition gates: Locally evaluated
  - Multiplication gates: Random and private multiplication triples [Beaver91a]



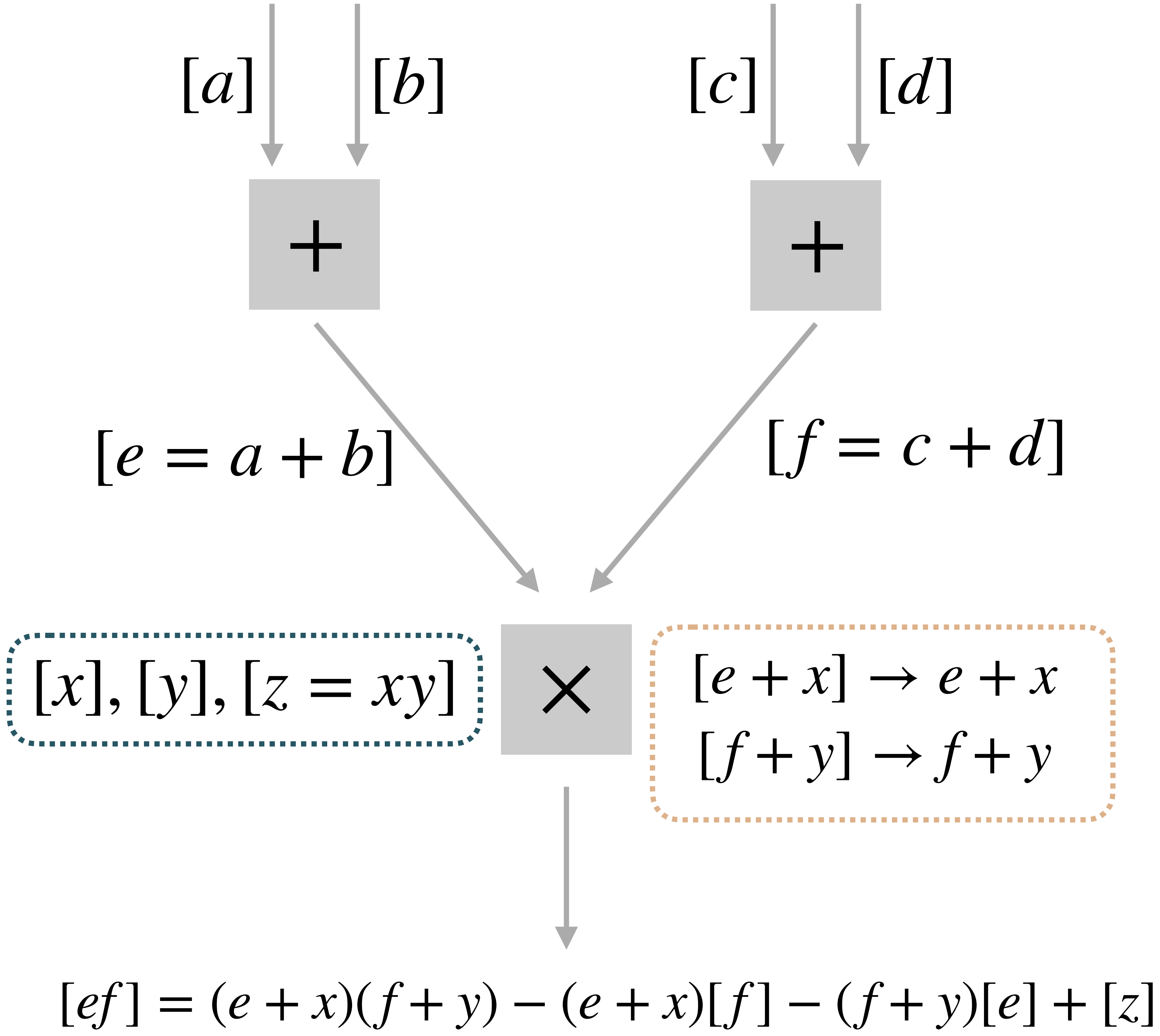
Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation
  - Addition gates: Locally evaluated
  - Multiplication gates: Random and private multiplication triples [Beaver91a]



Overview - Circuit Evaluation

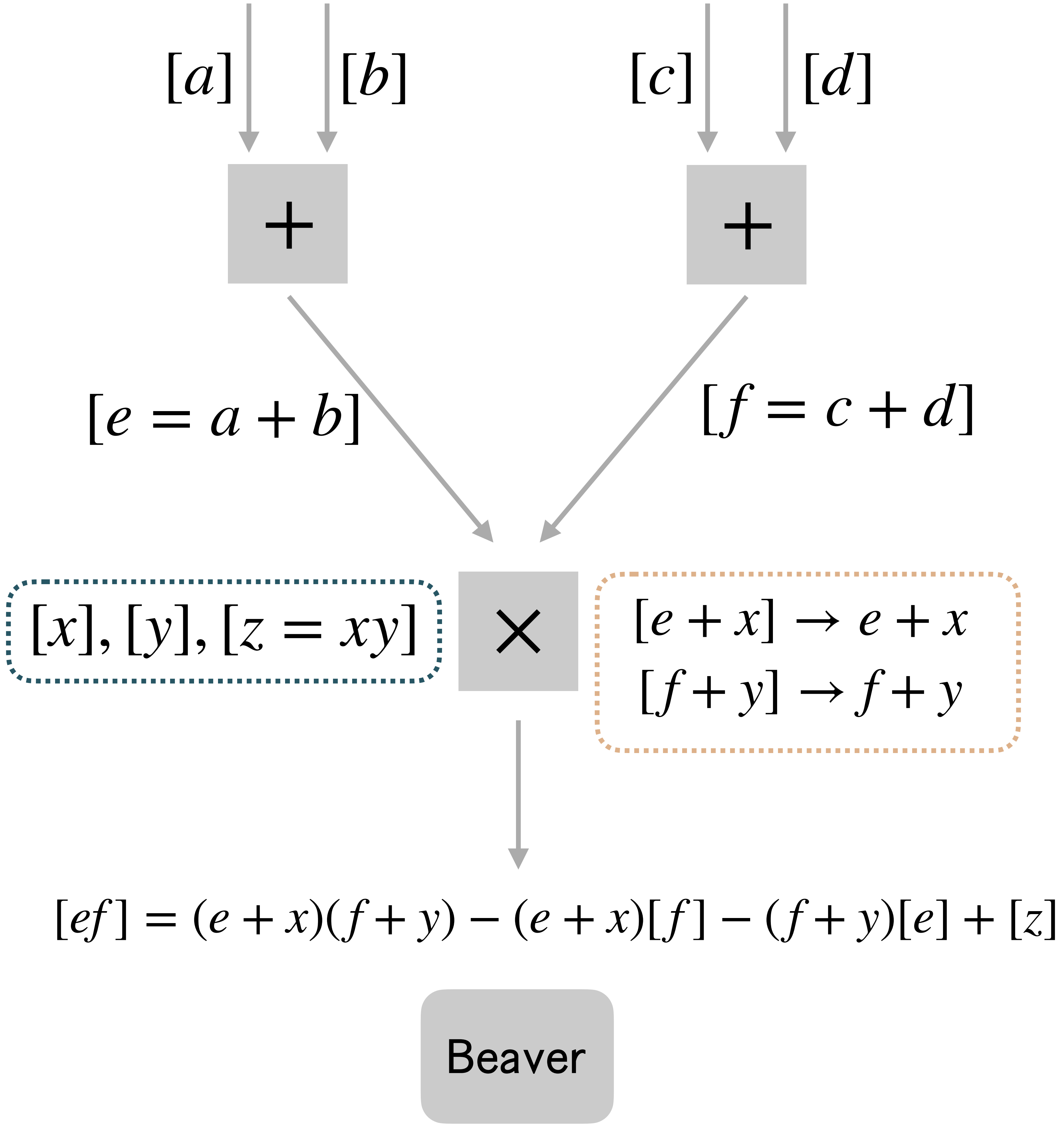
- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation
  - Addition gates: Locally evaluated
  - Multiplication gates: Random and private multiplication triples [Beaver91a]



$[a]$
$[b]$
$[c]$
$[d]$
$+$
$+$
$[e = a + b]$
$[f = c + d]$
$[x], [y], [z = xy]$
$\times$
$[e + x] \rightarrow e + x$
$[f + y] \rightarrow f + y$
$[ef] = (e + x)(f + y) - (e + x)[f] - (f + y)[e] + [z]$

Overview - Circuit Evaluation

- f represented as arithmetic circuit over finite field
- Shared circuit evaluation using a LSS scheme
  - High throughput
  - Linearity: $p[x] + q[y] = [px + qy]$
- Evaluation
  - Addition gates: Locally evaluated
  - Multiplication gates: Random and private multiplication triples [Beaver91a]



$[a]$
$[b]$
$[c]$
$[d]$
$+$
$+$
$[e = a + b]$
$[f = c + d]$
$[x], [y], [z = xy]$
$\times$
$[e + x] \rightarrow e + x$
$[f + y] \rightarrow f + y$
$[ef] = (e + x)(f + y) - (e + x)[f] - (f + y)[e] + [z]$
Beaver

Overview - Generating Multiplication Triples

- Triple generation framework of [CP17]

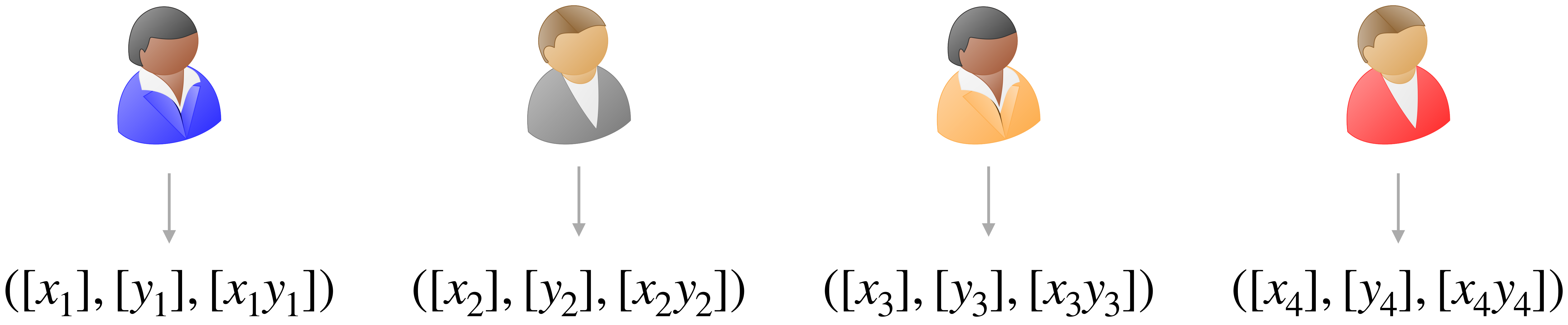
Overview - Generating Multiplication Triples

- Triple generation framework of [CP17]
    - Triple sharing protocol

$([x_1], [y_1], [x_1y_1])$ $([x_2], [y_2], [x_2y_2])$ $([x_3], [y_3], [x_3y_3])$ $([x_4], [y_4], [x_4y_4])$

Overview - Generating Multiplication Triples

- Triple generation framework of [CP17]
    - Triple sharing protocol

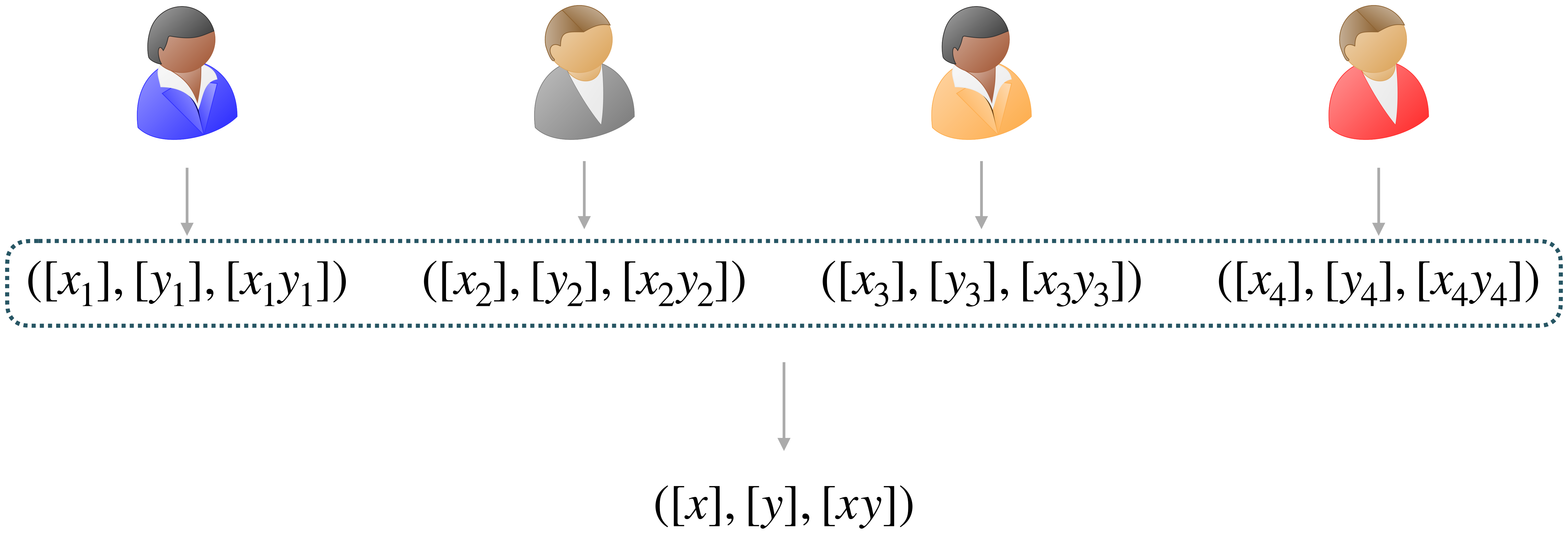


$([x_1], [y_1], [x_1 y_1])$ $([x_2], [y_2], [x_2 y_2])$ $([x_3], [y_3], [x_3 y_3])$ $([x_4], [y_4], [x_4 y_4])$

- Multiplication triple ✓
- Triple known to party ✗

Overview - Generating Multiplication Triples

- Triple generation framework of [CP17]
  - Triple sharing protocol
  - Triple extraction protocol



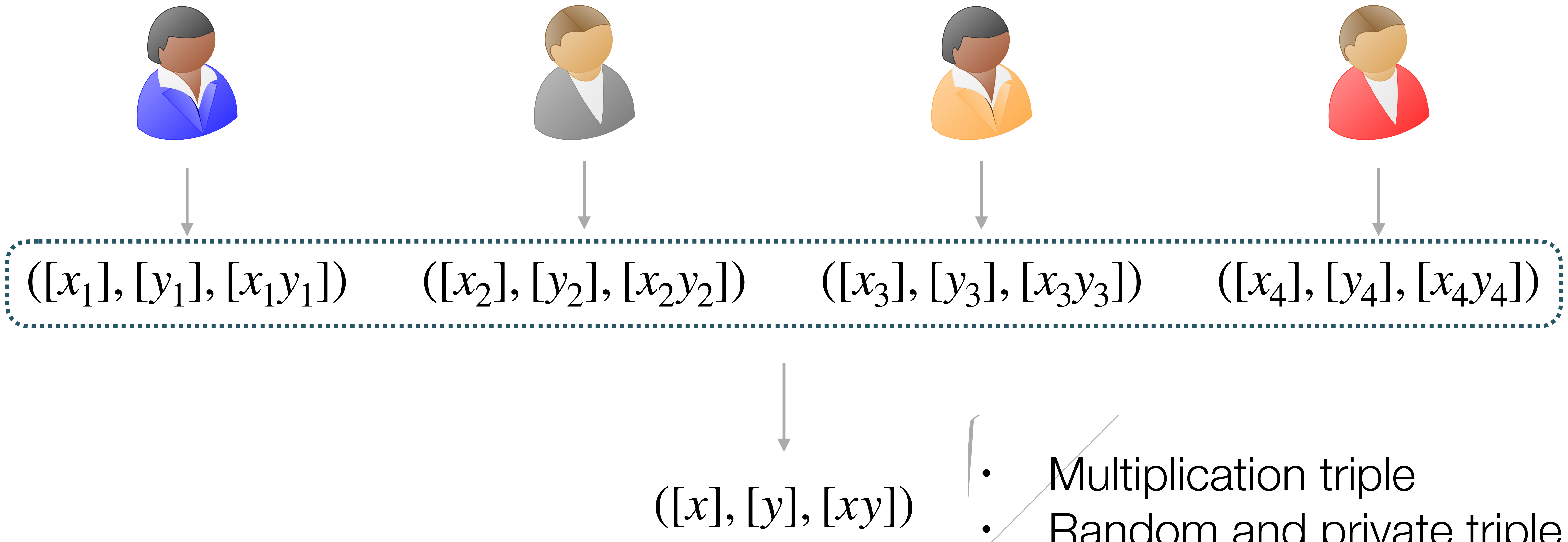
$([x_1], [y_1], [x_1 y_1])$ $([x_2], [y_2], [x_2 y_2])$ $([x_3], [y_3], [x_3 y_3])$ $([x_4], [y_4], [x_4 y_4])$

$([x], [y], [xy])$

- Multiplication triple ✓
- Triple known to party ✗

Overview - Generating Multiplication Triples

- Triple generation framework of [CP17]
  - Triple sharing protocol
  - Triple extraction protocol



$([x_1], [y_1], [x_1y_1])$ $([x_2], [y_2], [x_2y_2])$ $([x_3], [y_3], [x_3y_3])$ $([x_4], [y_4], [x_4y_4])$

- Multiplication triple ✓
- Triple known to party ✗

$([x], [y], [xy])$

- Multiplication triple ✓
- Random and private triple ✓

Perfect HMPC

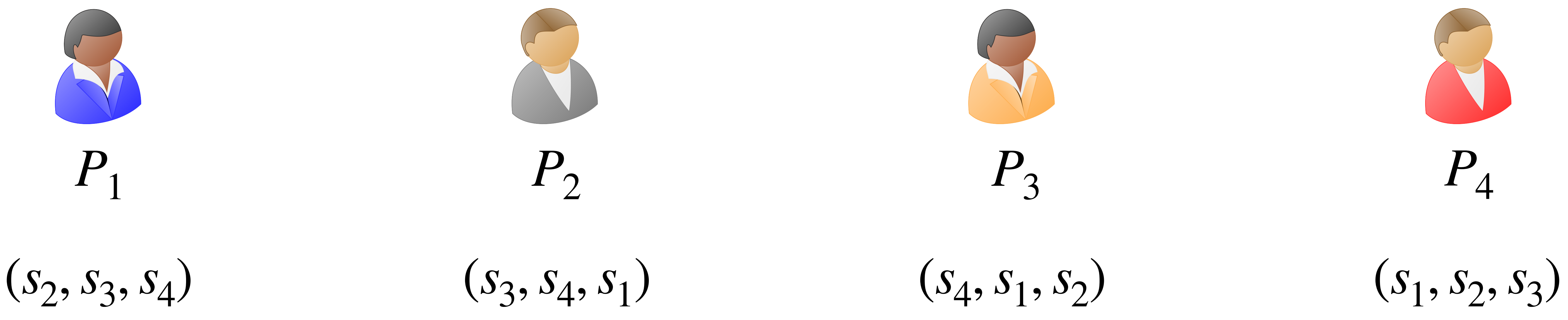
- Open Problem [PR18]: Perfectly secure MPC protocol over hybrid network
    - Two synchronous rounds
    - Tolerating $t < n/3$ corruptions
    - With synchronous broadcast channel
    - Guaranteed output delivery
- Input provision impossible in this setting [PR18]

Perfect HMPC - Linear Secret Sharing Scheme

- Replicated Secret Sharing [ISN89]
  - $[s] = (s_1, s_2, s_3, s_4)$
  - $s = s_1 + s_2 + s_3 + s_4$

Perfect HMPC - Linear Secret Sharing Scheme

- Replicated Secret Sharing [ISN89]
  - $[s] = (s_1, s_2, s_3, s_4)$
  - $s = s_1 + s_2 + s_3 + s_4$
- P_i does not have s_i



P_1 (s_2, s_3, s_4)

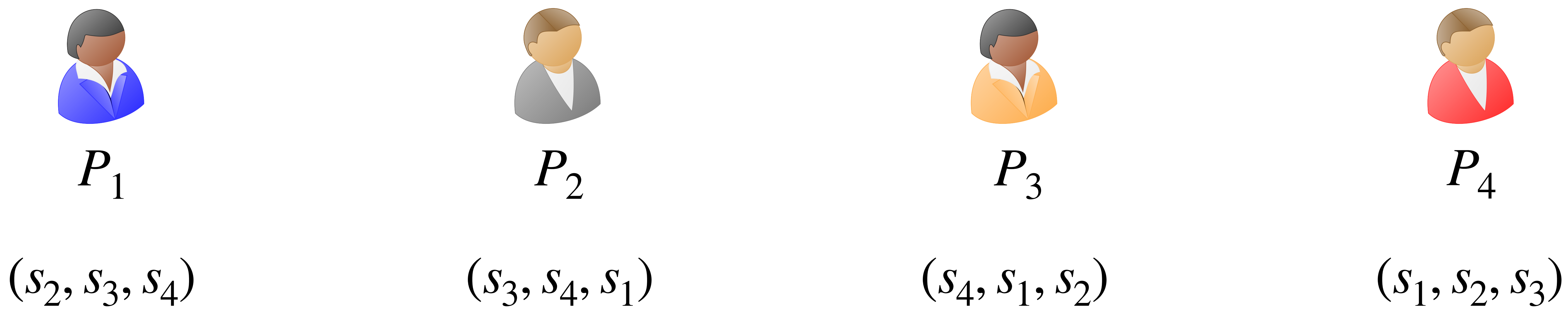
P_2 (s_3, s_4, s_1)

P_3 (s_4, s_1, s_2)

P_4 (s_1, s_2, s_3)

Perfect HMPC - Linear Secret Sharing Scheme

- Replicated Secret Sharing [ISN89]
  - $[s] = (s_1, s_2, s_3, s_4)$
  - $s = s_1 + s_2 + s_3 + s_4$
- P_i does not have s_i
  - All other parties except P_i have s_i



P_1 (s_2, s_3, s_4)

P_2 (s_3, s_4, s_1)

P_3 (s_4, s_1, s_2)

P_4 (s_1, s_2, s_3)

Perfect HMPC - Linear Secret Sharing Scheme

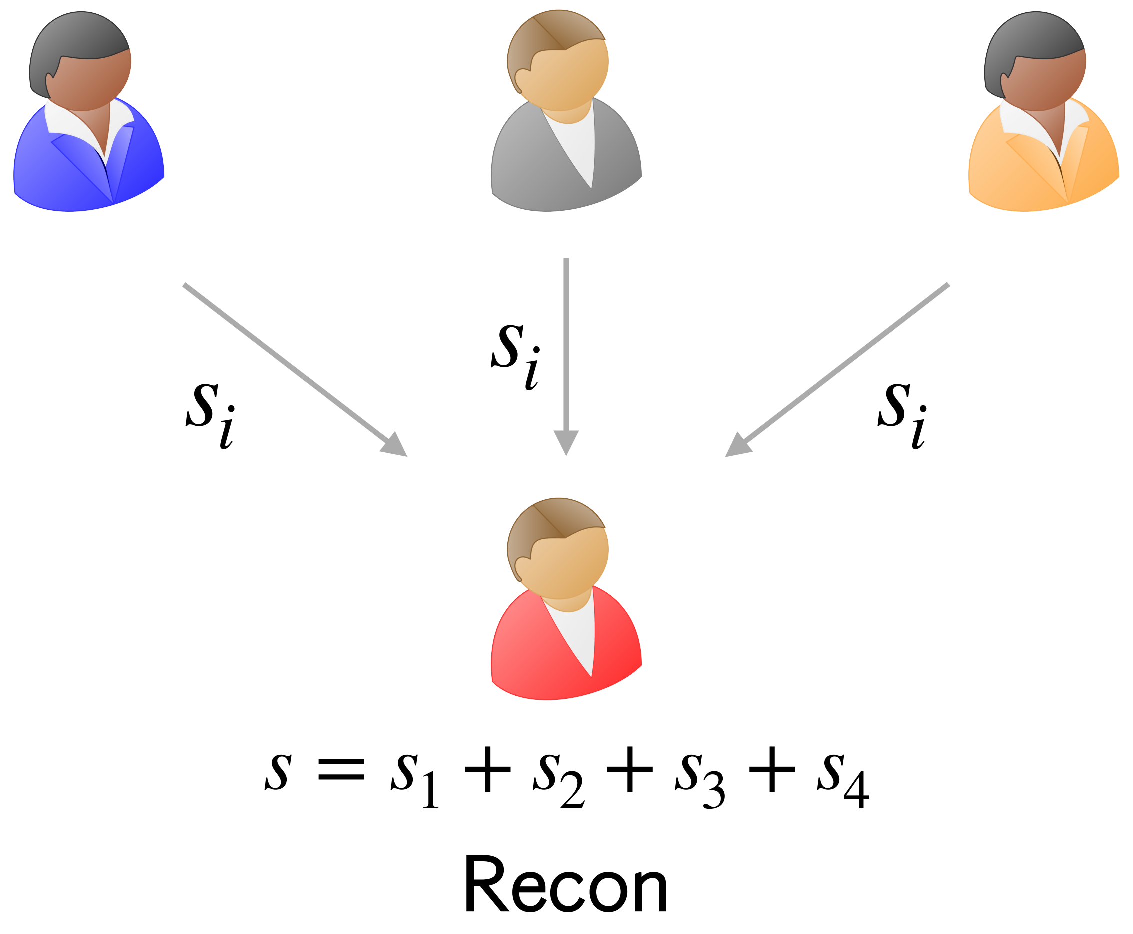
- Replicated Secret Sharing [ISN89]
  - $[s] = (s_1, s_2, s_3, s_4)$
  - $s = s_1 + s_2 + s_3 + s_4$
- P_i does not have s_i
  - All other parties except P_i have s_i

P_1	P_2	P_3	P_4
(s_2, s_3, s_4)	(s_3, s_4, s_1)	(s_4, s_1, s_2)	(s_1, s_2, s_3)

$$[cs] = (cs_1, cs_2, cs_3, cs_4)$$
$$[s + s'] = (s_1 + s_1', s_2 + s_2', s_3 + s_3', s_4 + s_4')$$

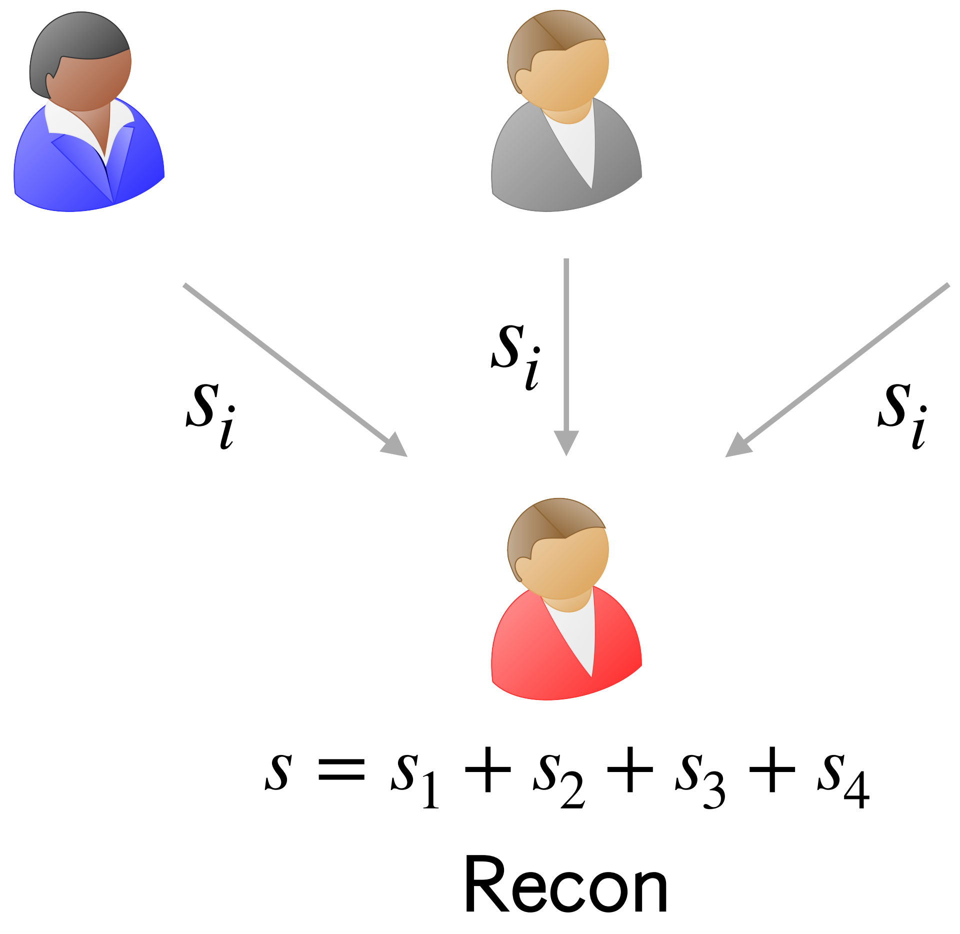
Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i



Perfect HMPC - Linear Secret Sharing Scheme

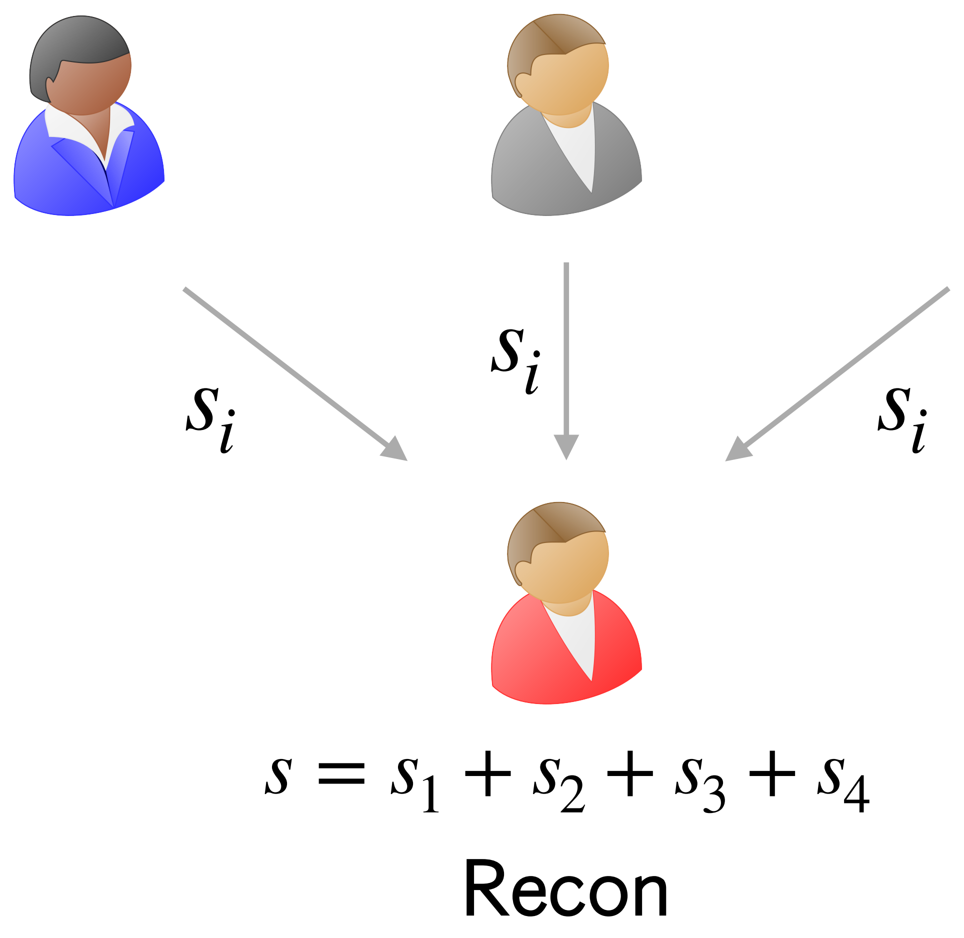
- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i



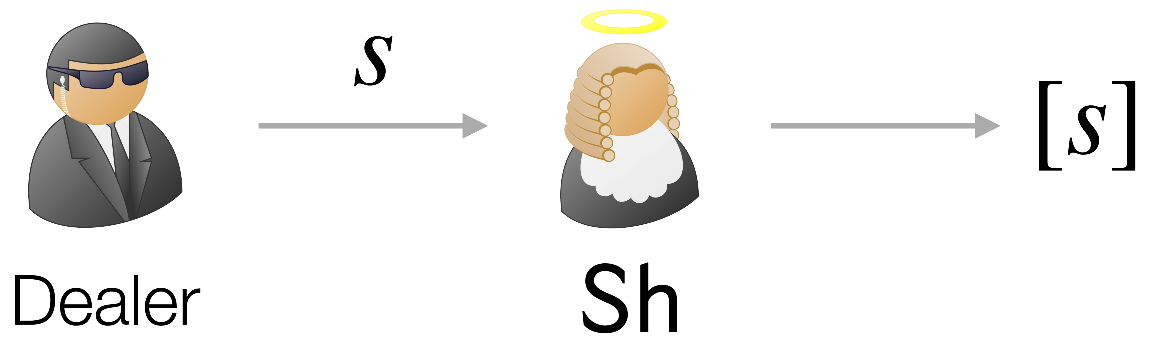
Completely Asynchronous

Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set

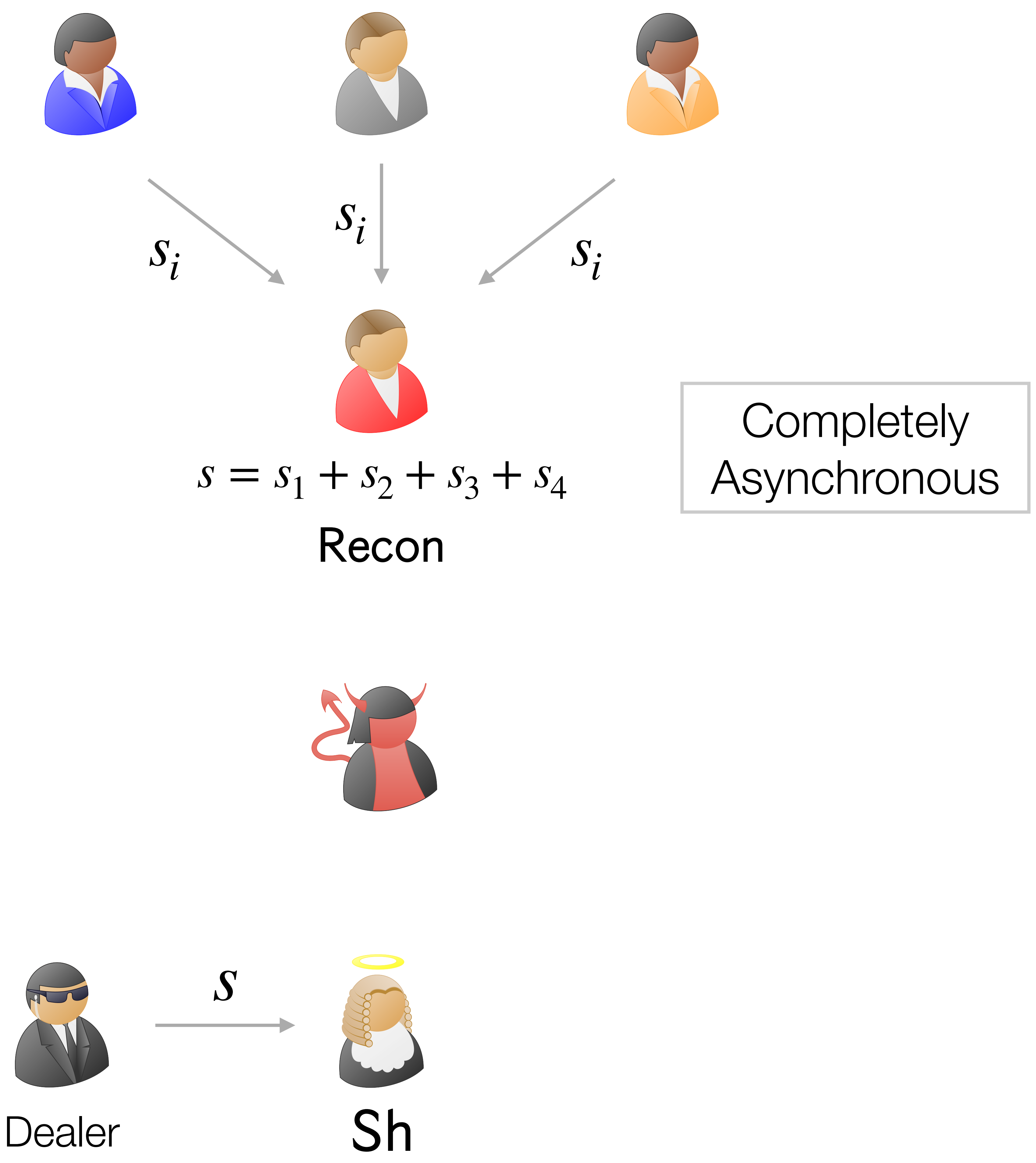


Completely Asynchronous



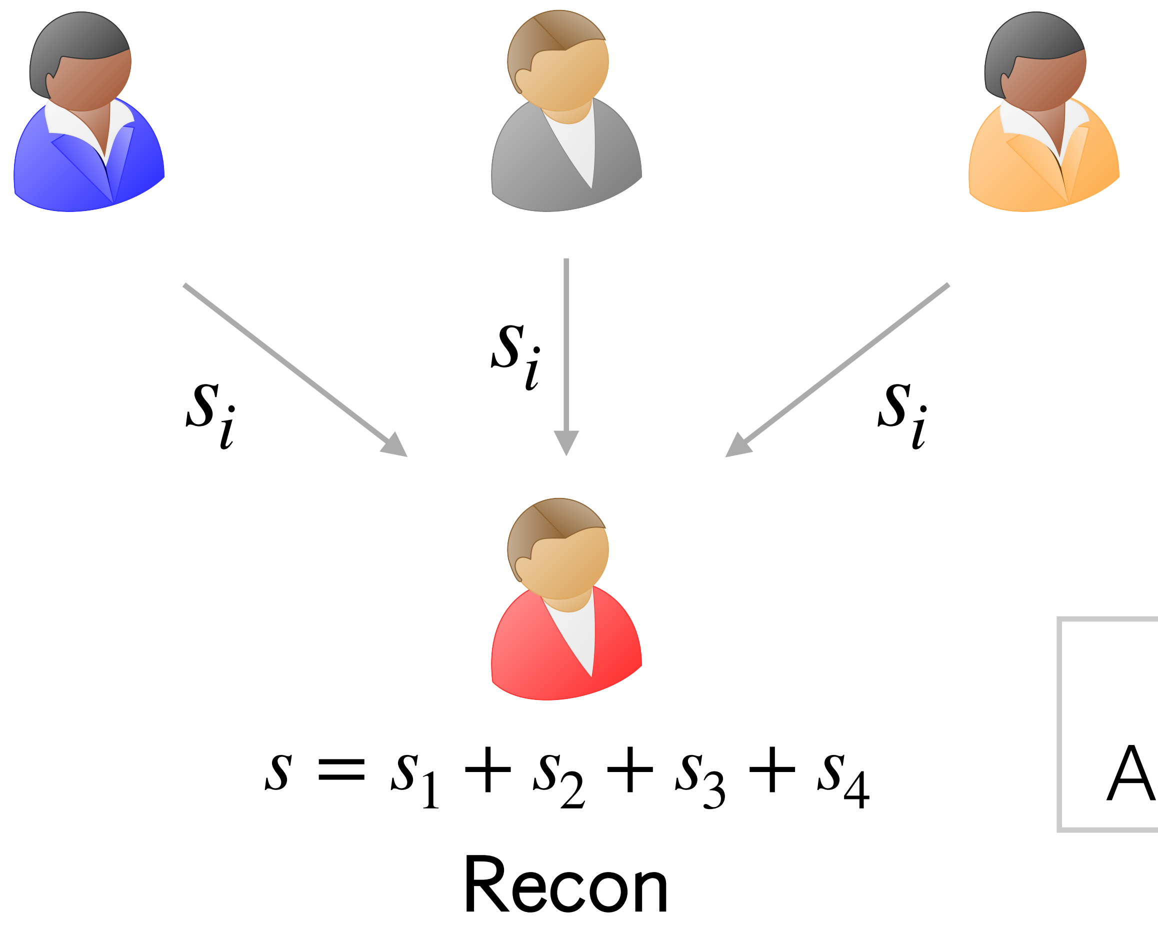
Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set

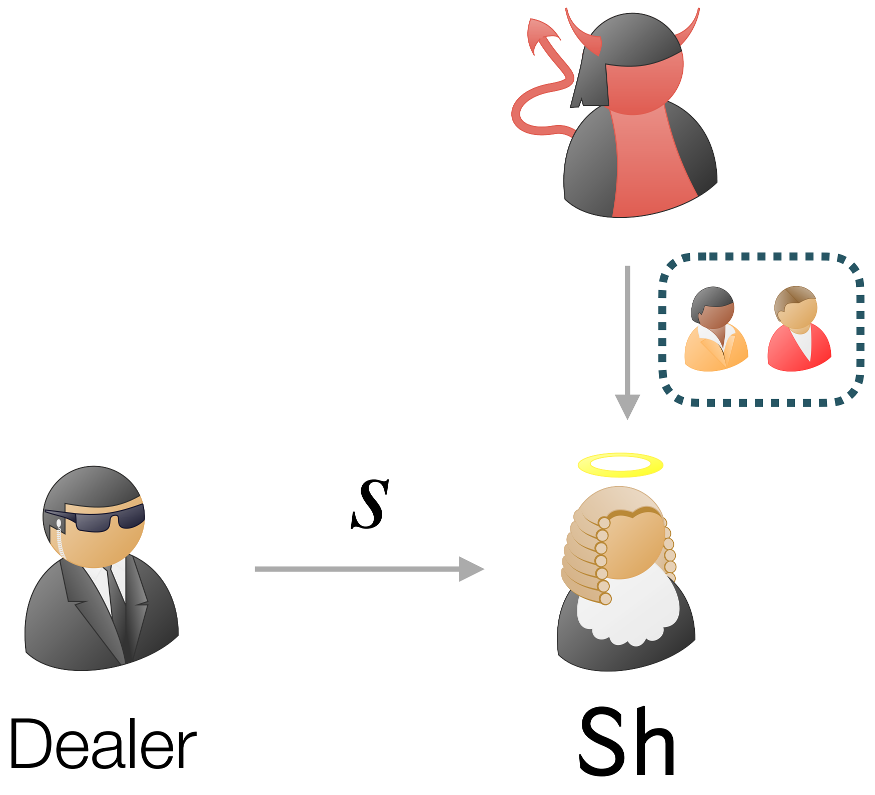


Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set

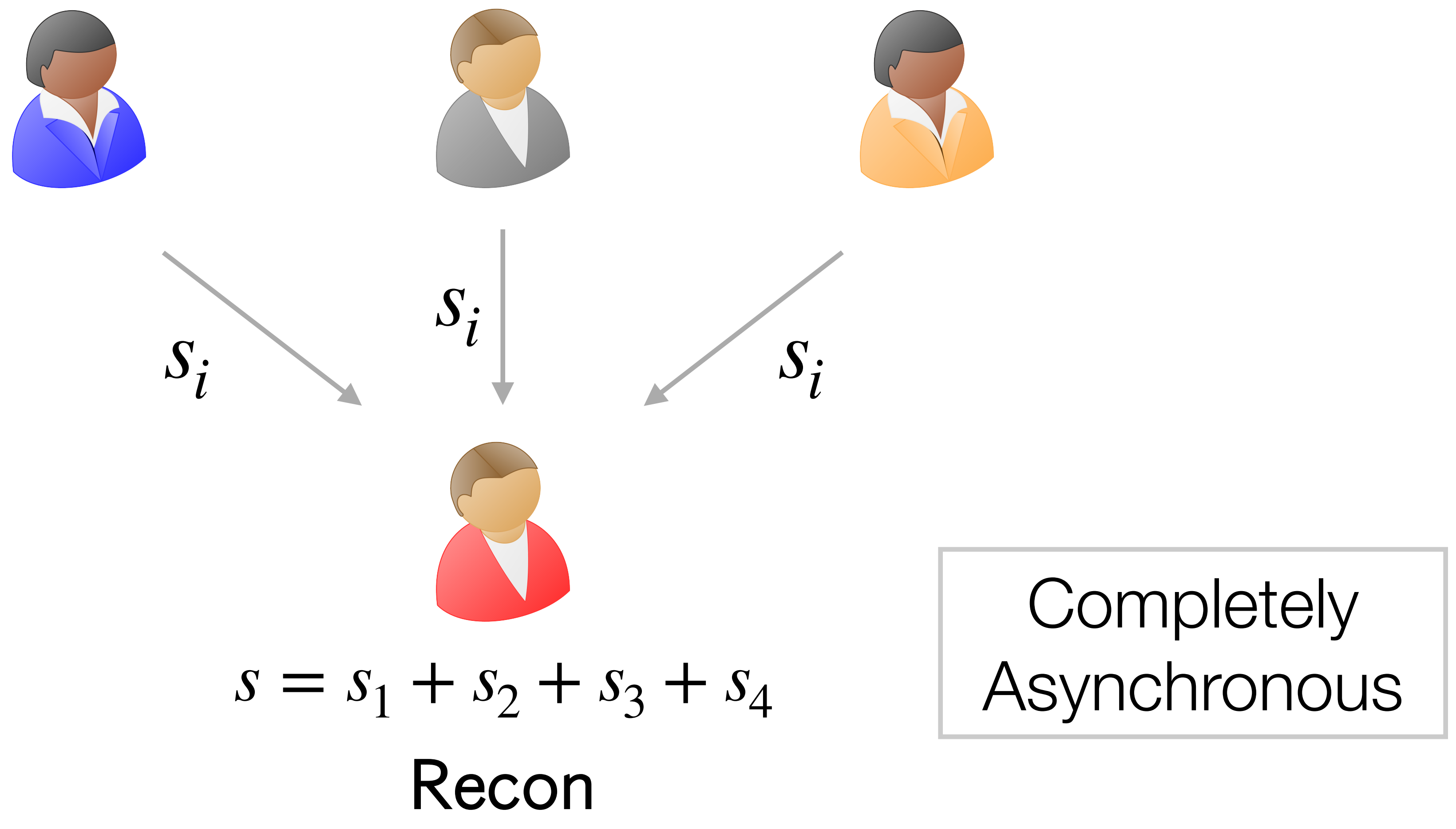


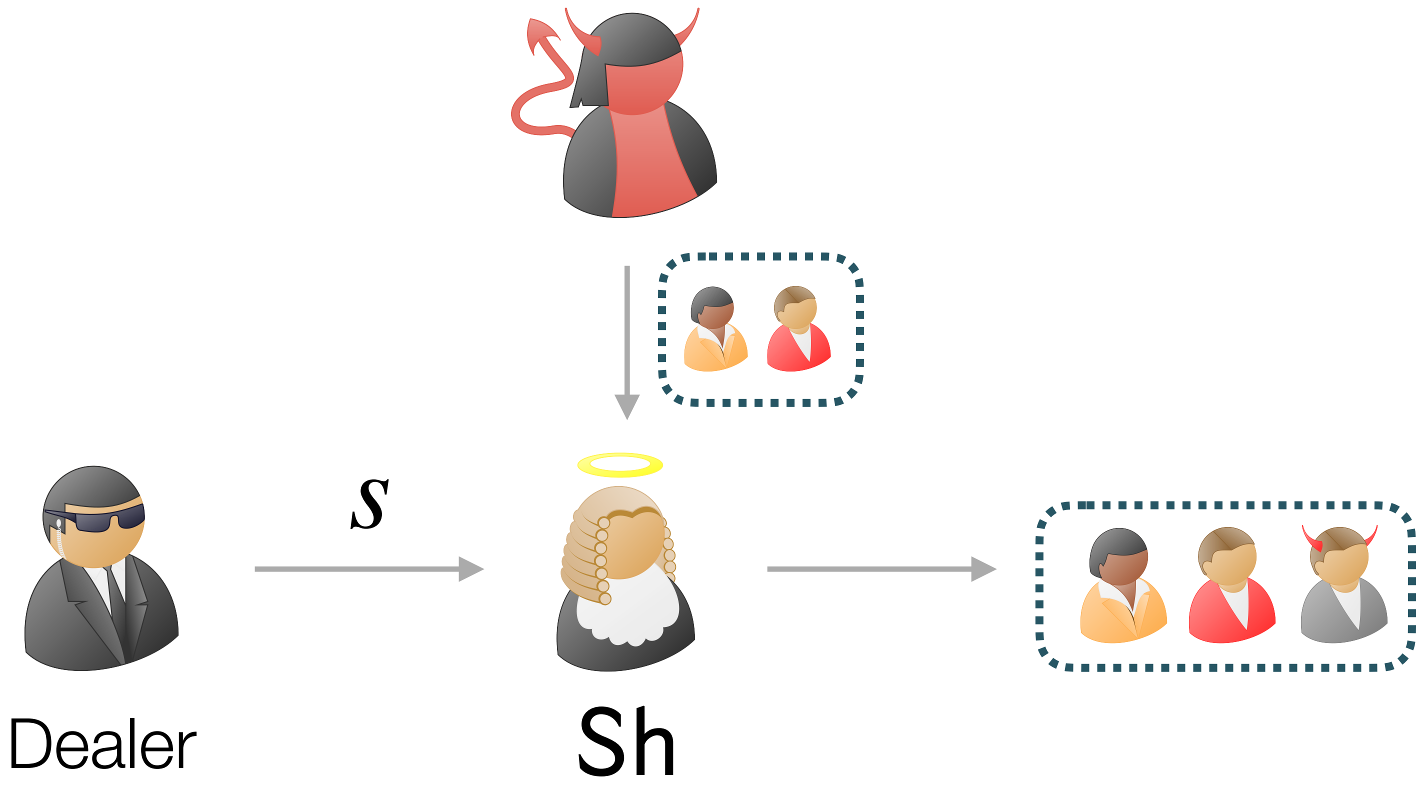
Completely Asynchronous



Perfect HMPC - Linear Secret Sharing Scheme

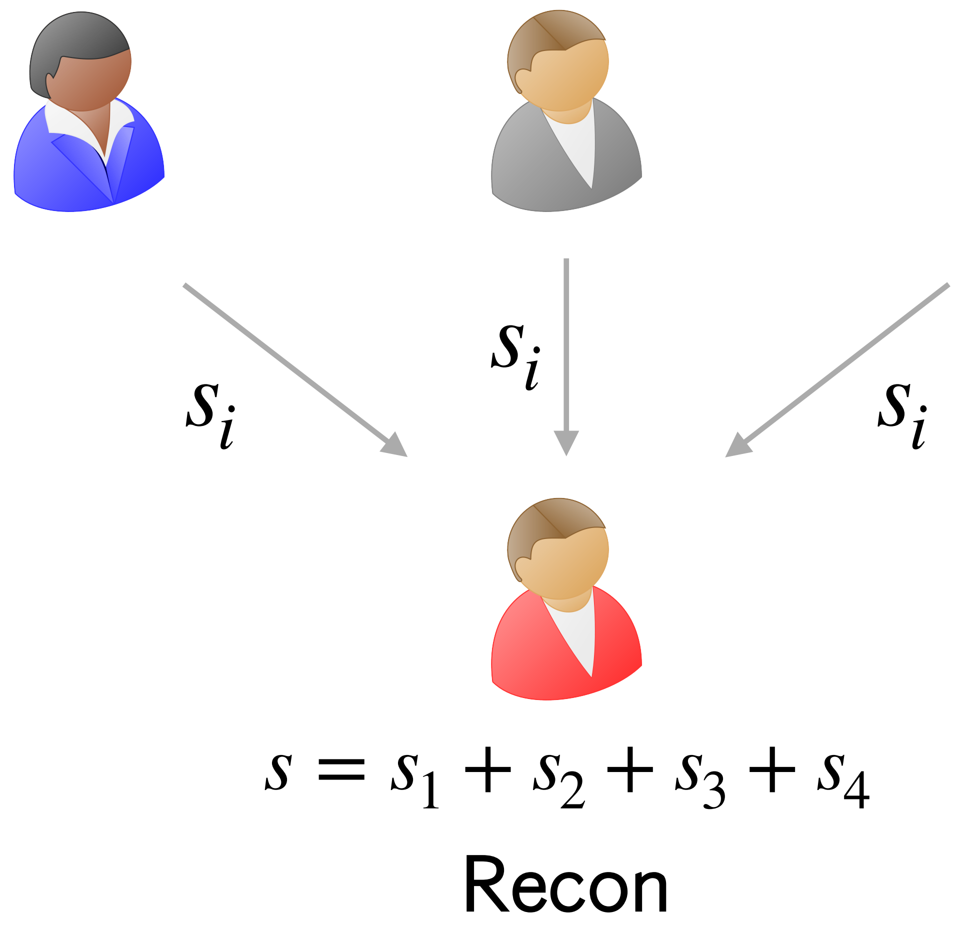
- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set



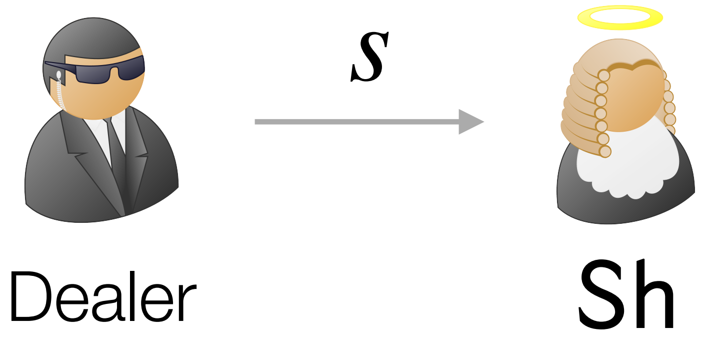


Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set

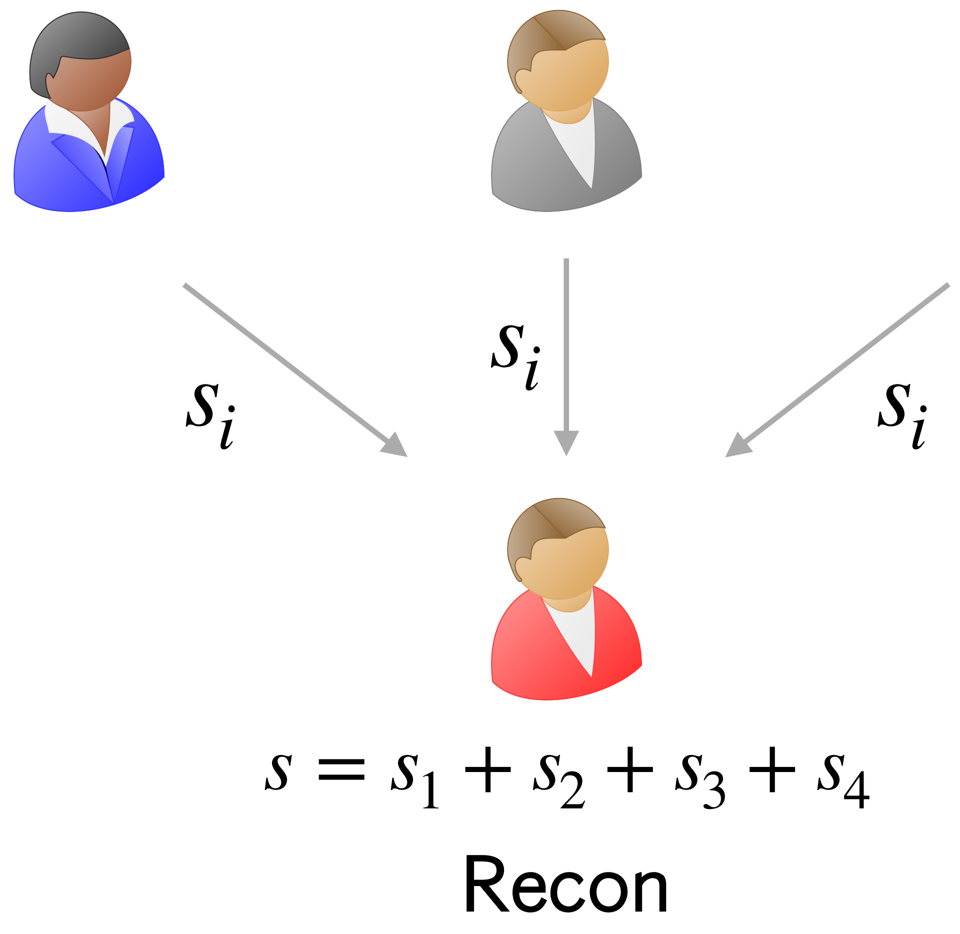


Completely Asynchronous

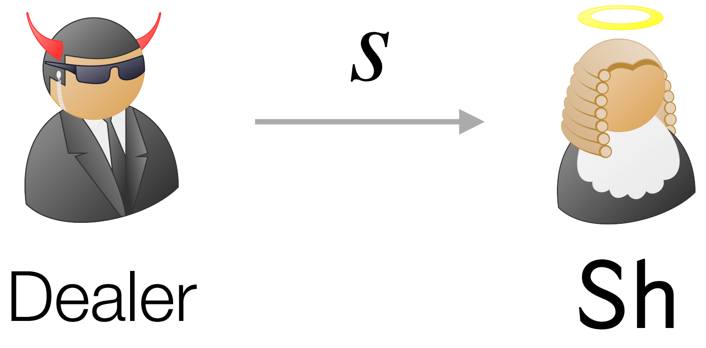


Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set

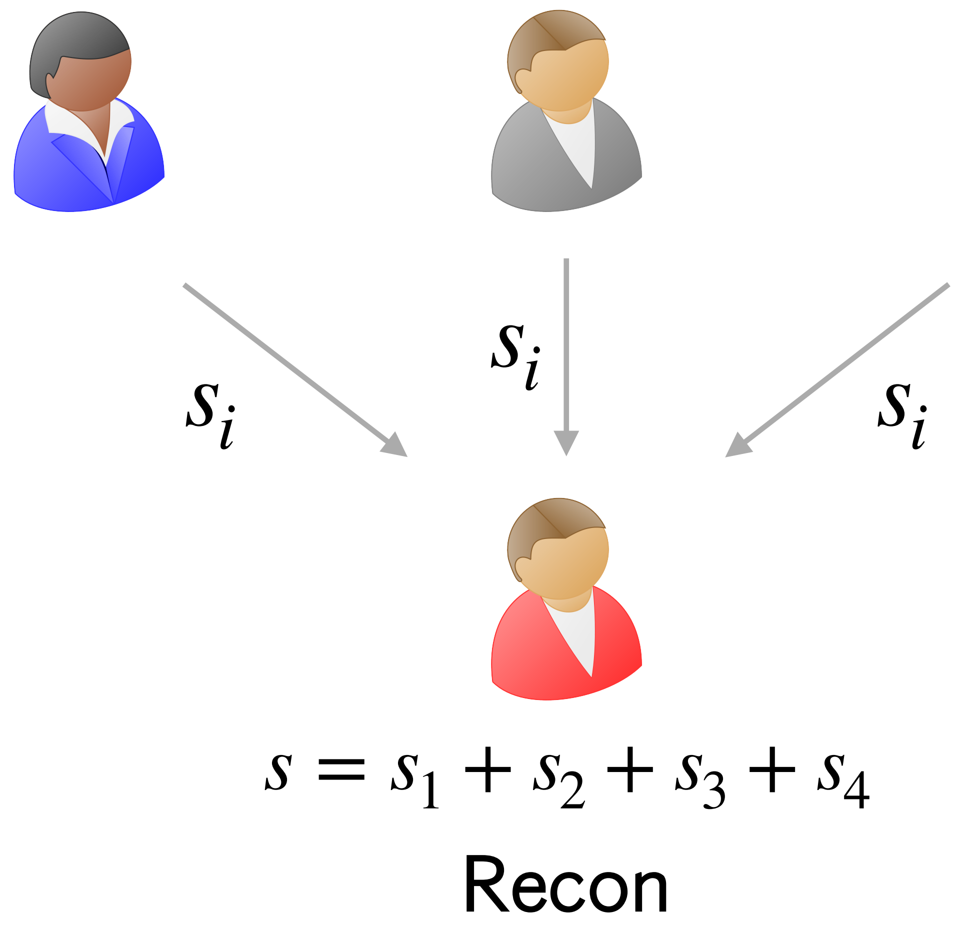


Completely Asynchronous

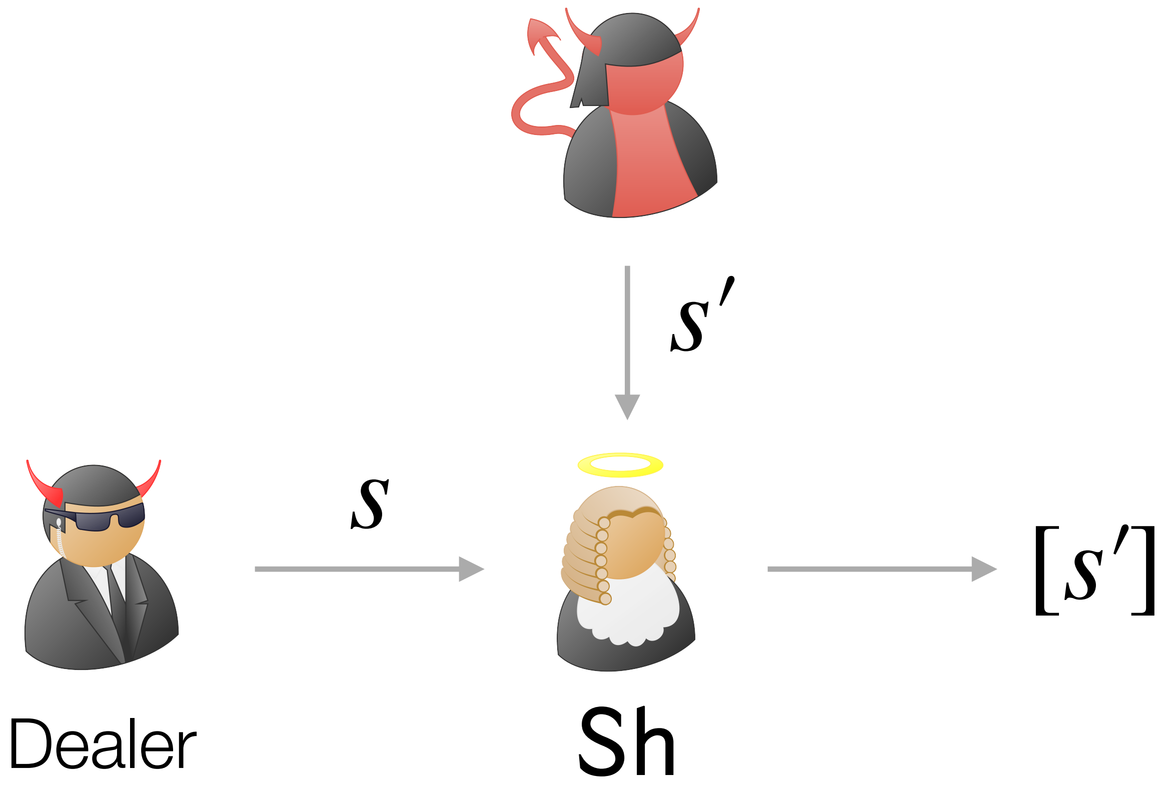


Perfect HMPC - Linear Secret Sharing Scheme

- Reconstruction
  - P_j sends s_i to P_i
  - P_i waits to receive 2 identical copies of s_i
- VSS with Party Elimination
  - Secret sharing or dispute set



Completely Asynchronous

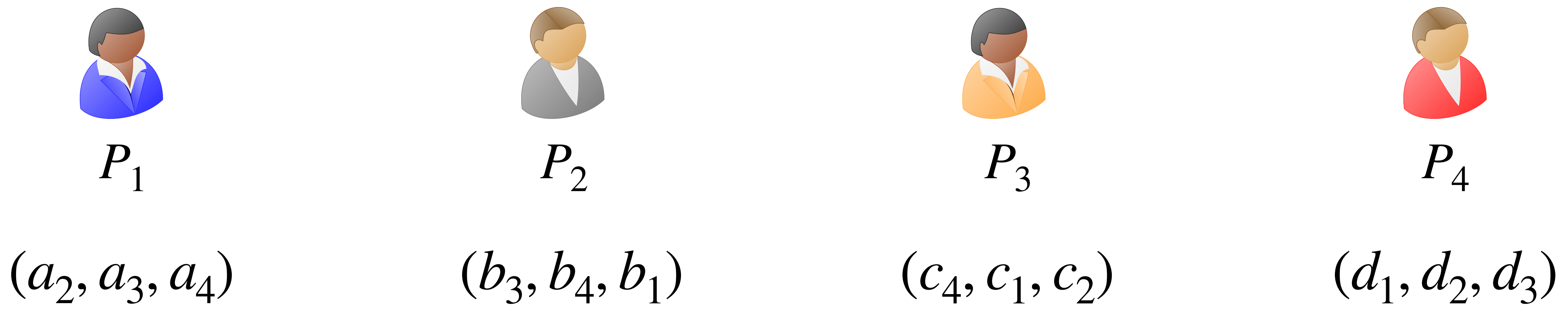


Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share

Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share



P_1 (a_2, a_3, a_4)

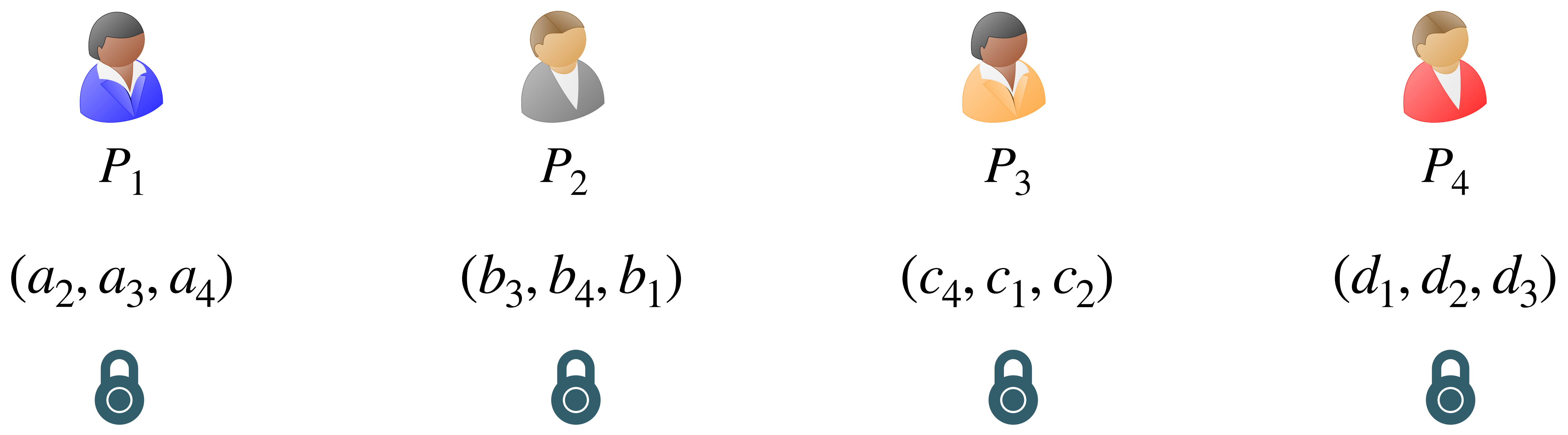
P_2 (b_3, b_4, b_1)

P_3 (c_4, c_1, c_2)

P_4 (d_1, d_2, d_3)

Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share



P_1 (a_2, a_3, a_4)

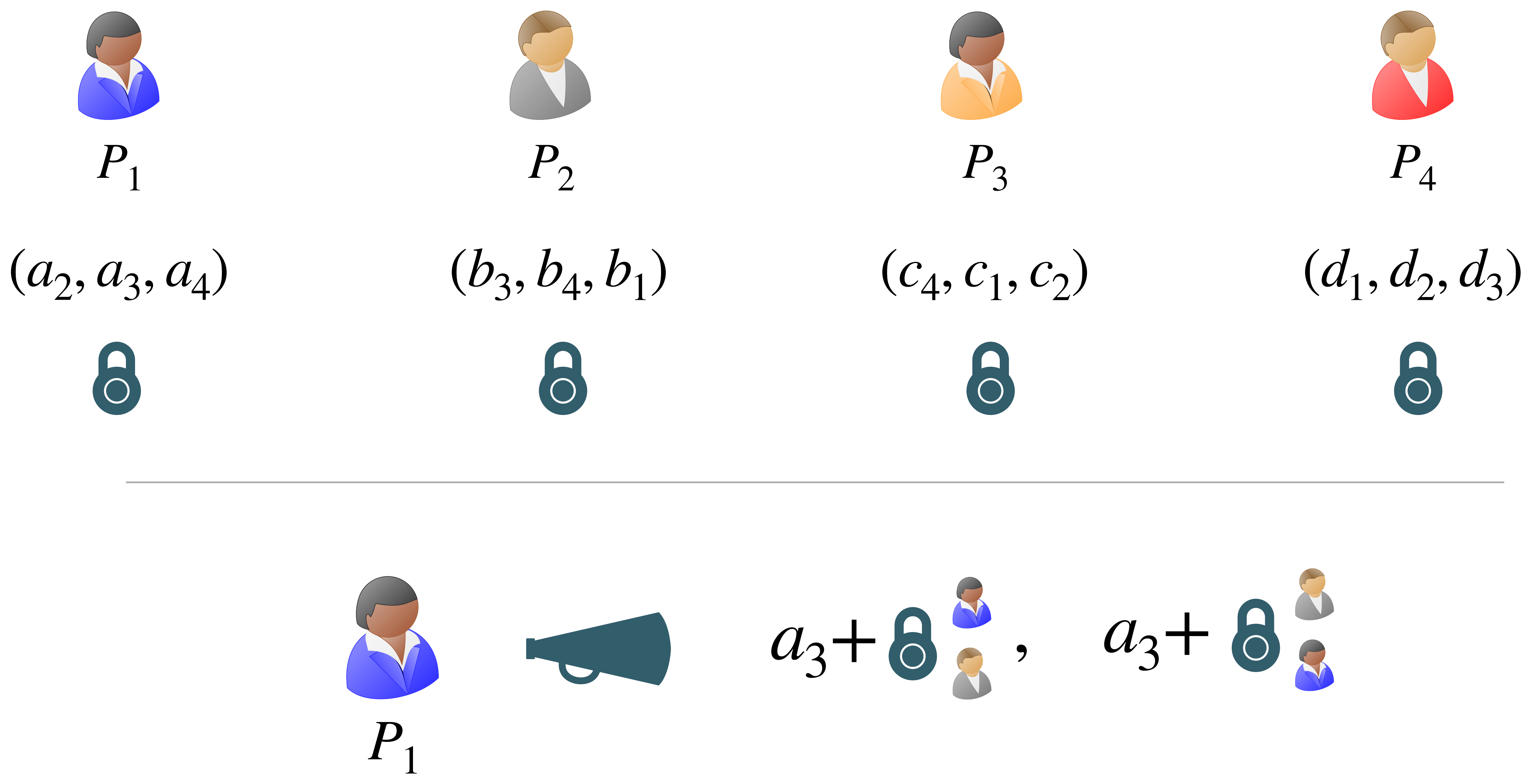
P_2 (b_3, b_4, b_1)

P_3 (c_4, c_1, c_2)

P_4 (d_1, d_2, d_3)

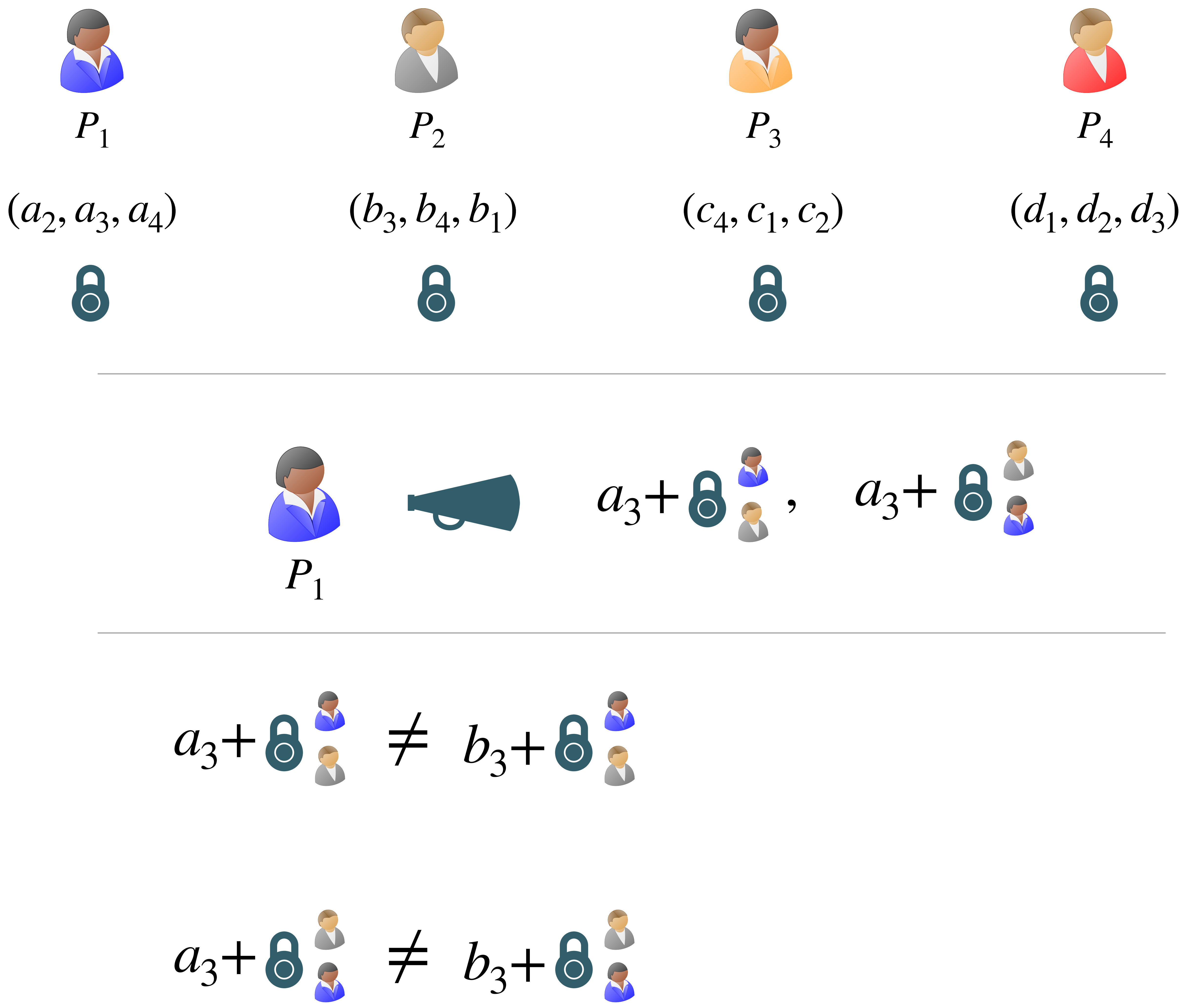
Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share
- Round 2
  - Parties broadcast masked shares



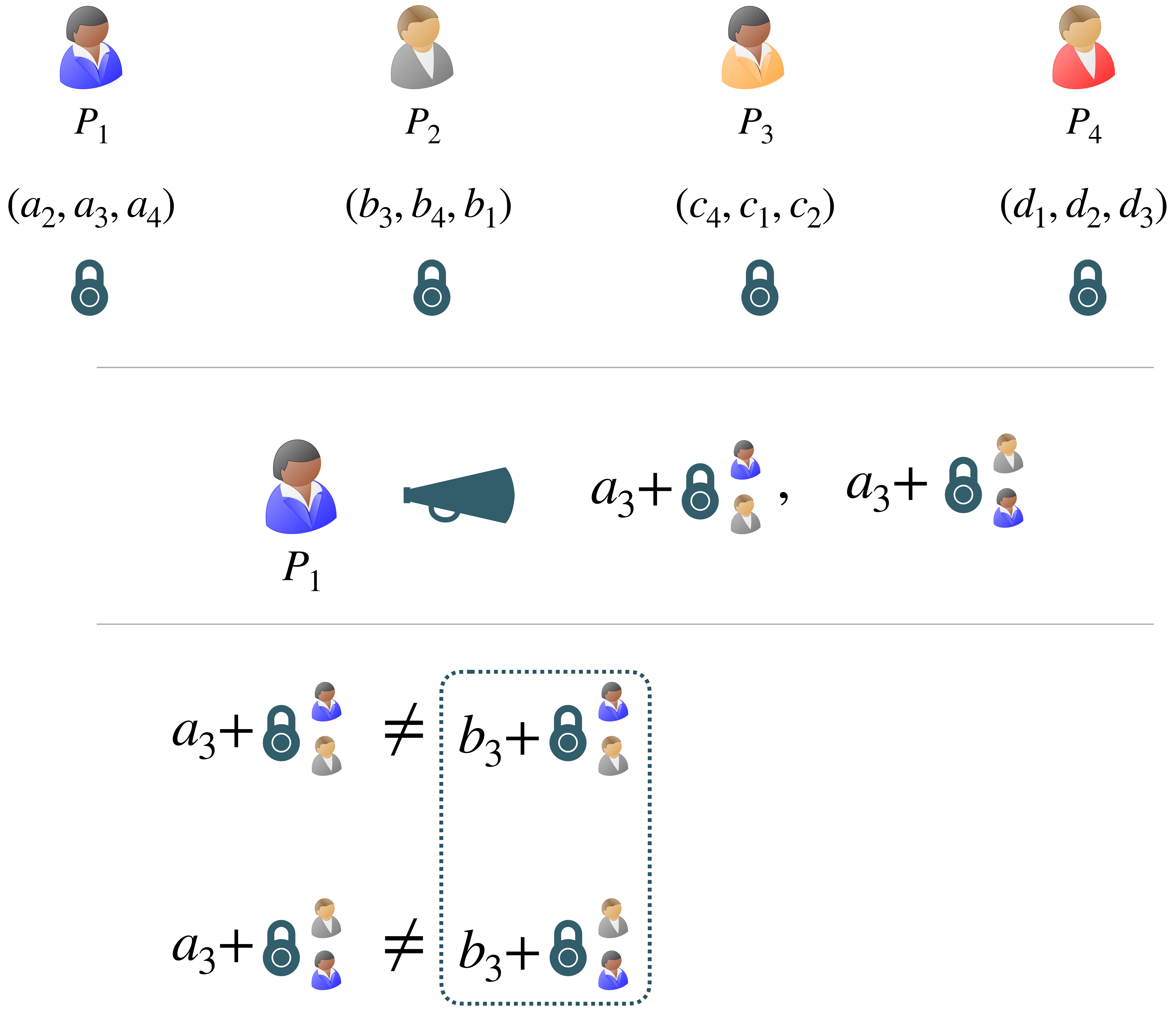
Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share
- Round 2
  - Parties broadcast masked shares
- Local computation
  - If shares inconsistent, output dispute set
  - Else output with secret shares



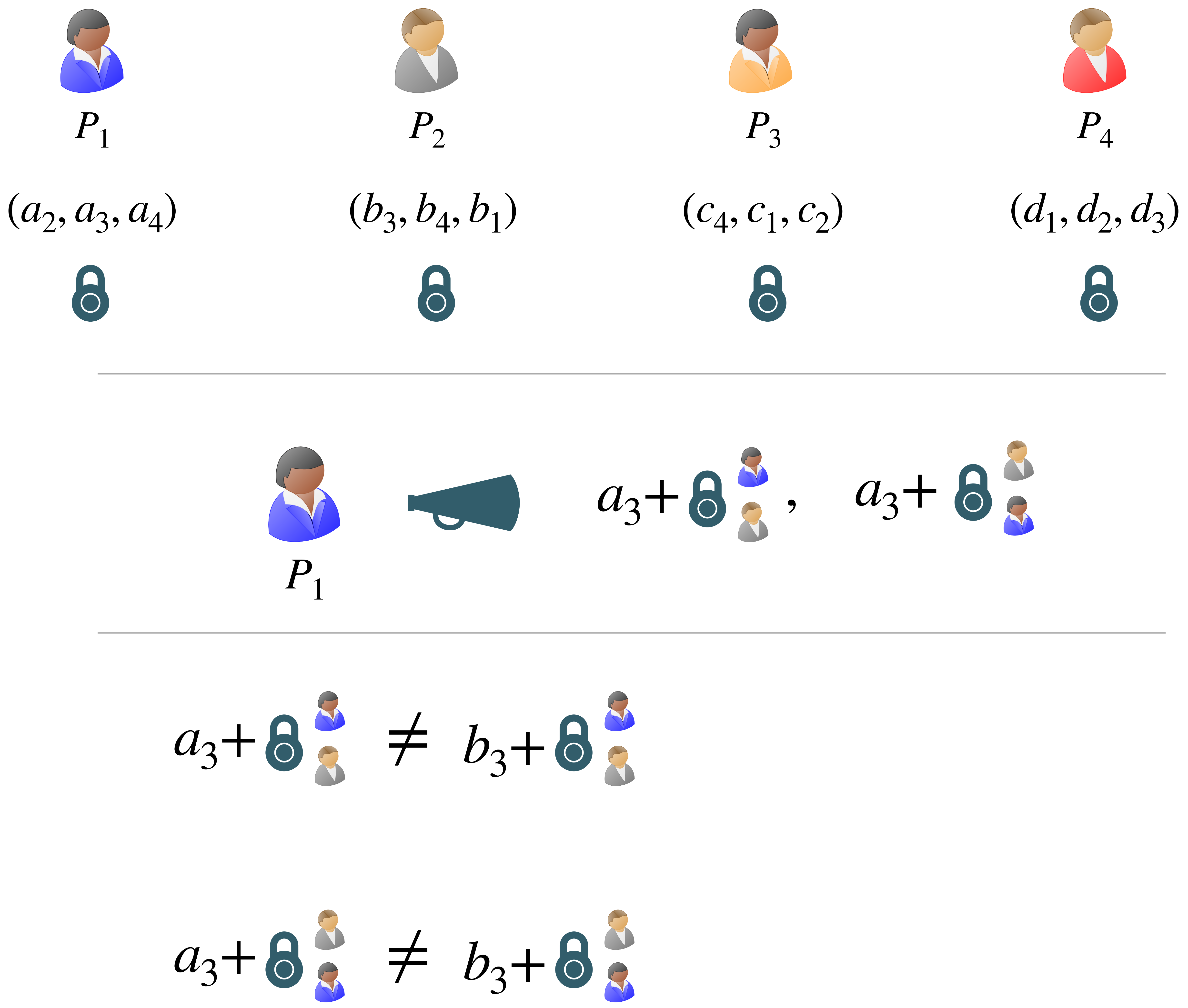
Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share
- Round 2
  - Parties broadcast masked shares
- Local computation
  - If shares inconsistent, output dispute set
  - Else output with secret shares



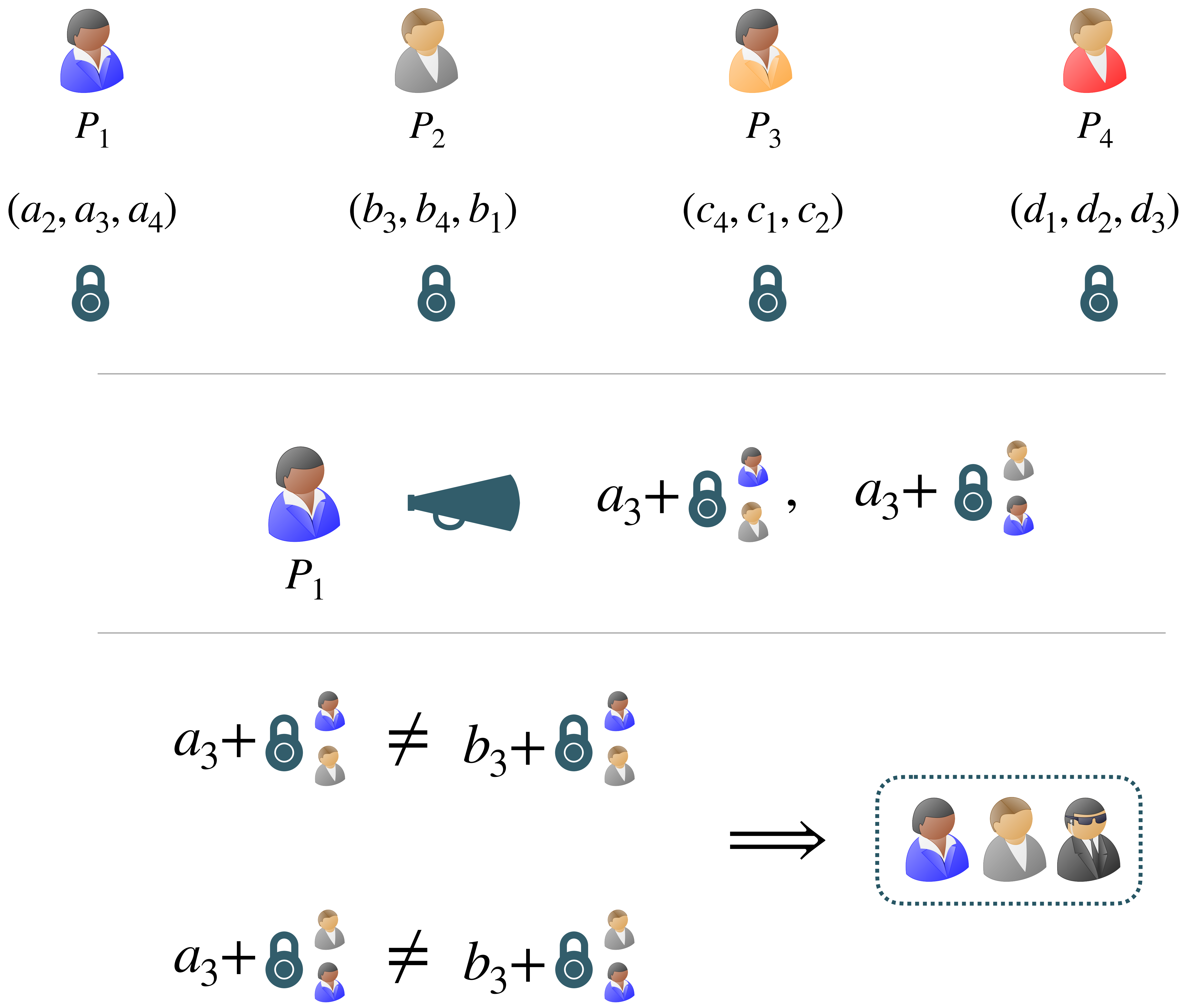
Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share
- Round 2
  - Parties broadcast masked shares
- Local computation
  - If shares inconsistent, output dispute set
  - Else output with secret shares

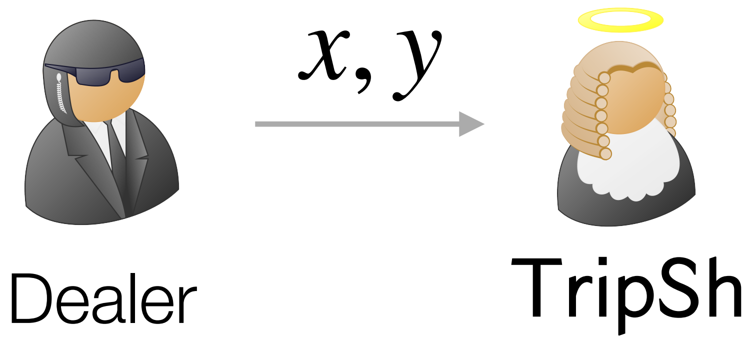


Perfect HMPC - VSS with Party Elimination

- Round 1
  - D sends share to each party
  - Parties exchange random pad for each common element in share
- Round 2
  - Parties broadcast masked shares
- Local computation
  - If shares inconsistent, output dispute set
  - Else output with secret shares

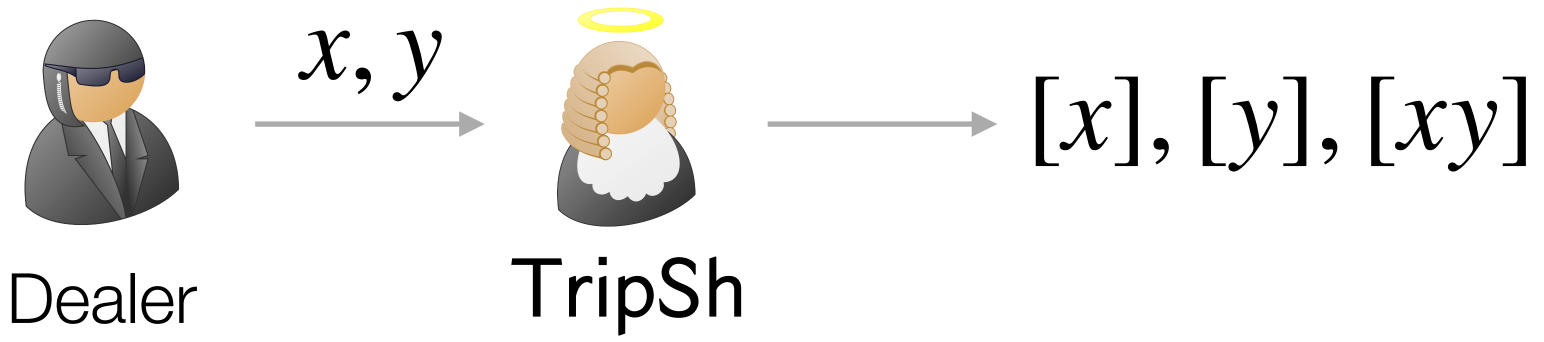


Perfect HMPC - Triple Sharing with Party Elimination Functionality



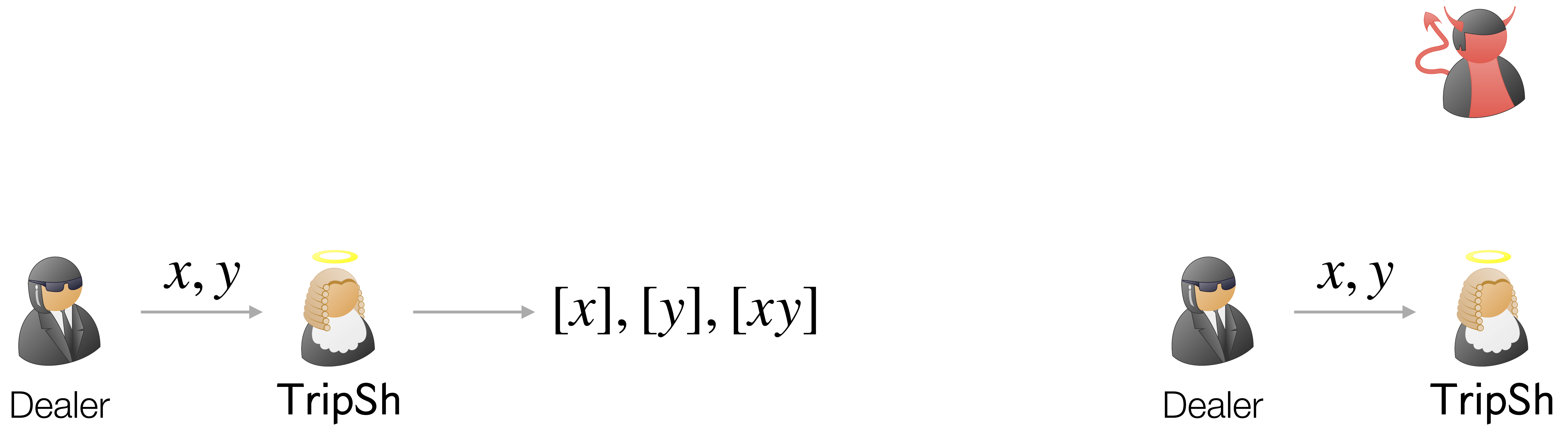
- Triple sharing with Party Elimination
  - Verified multiplication triple or dispute set

Perfect HMPC - Triple Sharing with Party Elimination Functionality



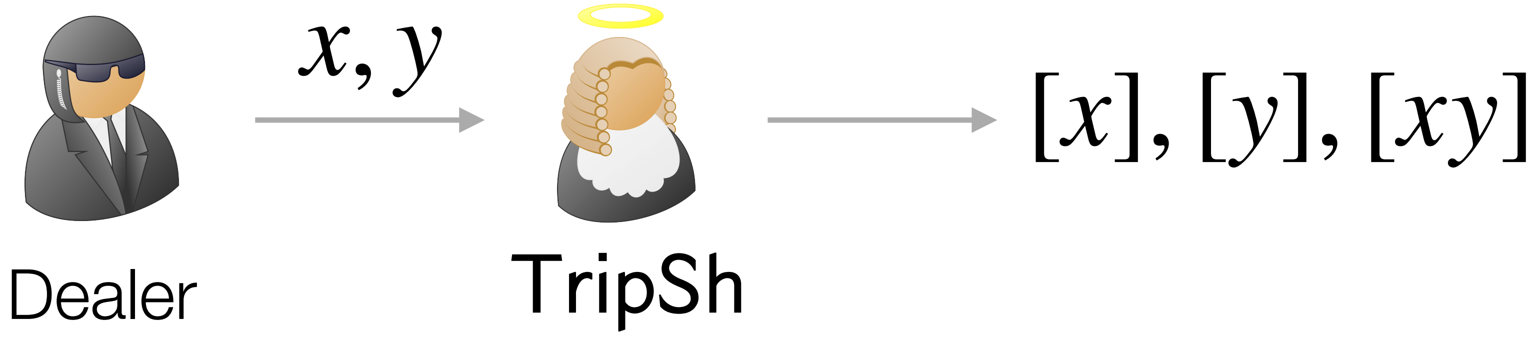
- Triple sharing with Party Elimination
  - Verified multiplication triple or dispute set

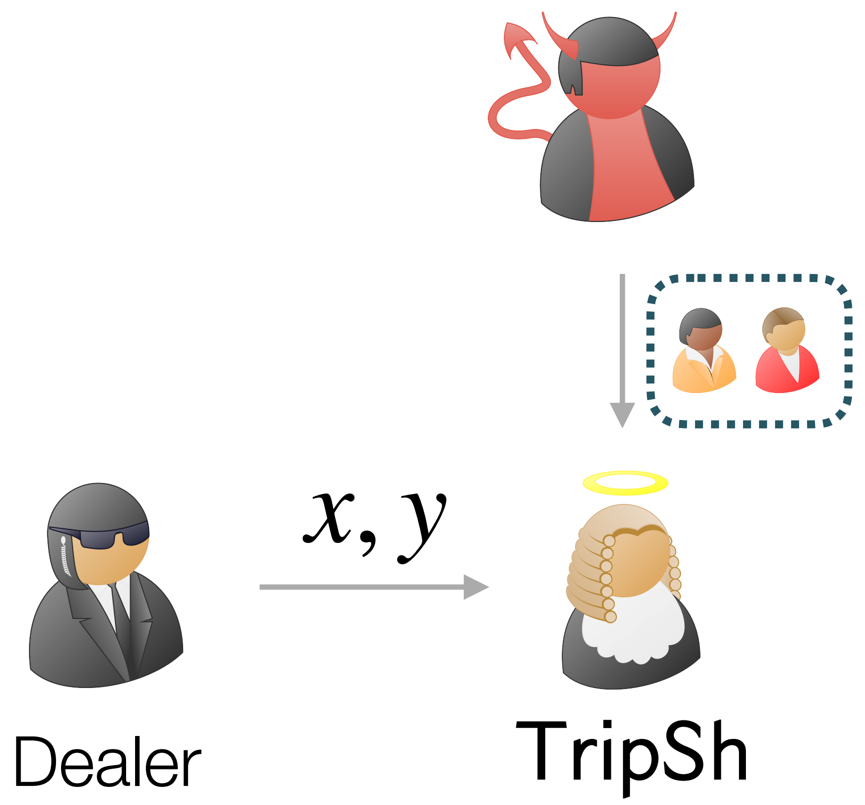
Perfect HMPC - Triple Sharing with Party Elimination Functionality



- Triple sharing with Party Elimination
  - Verified multiplication triple or dispute set

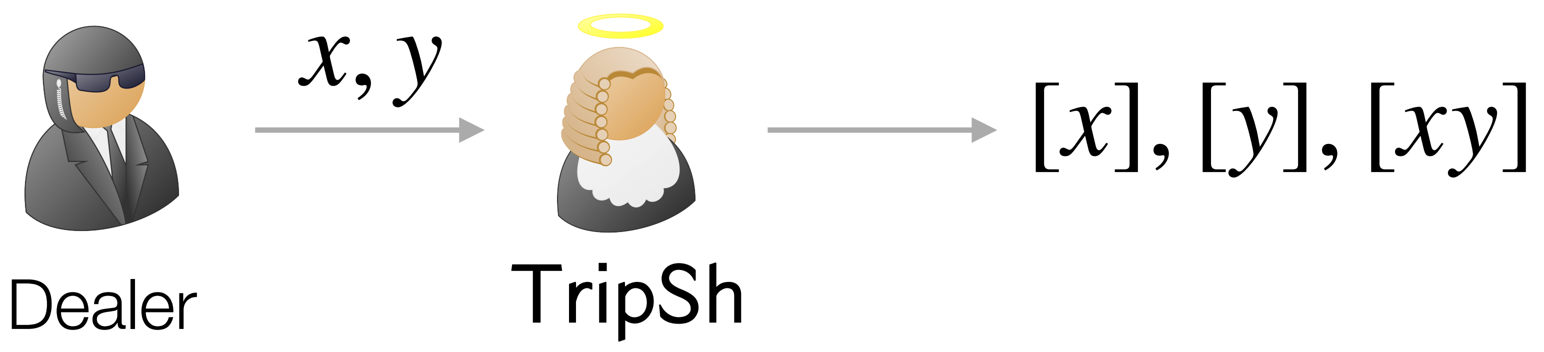
Perfect HMPC - Triple Sharing with Party Elimination Functionality



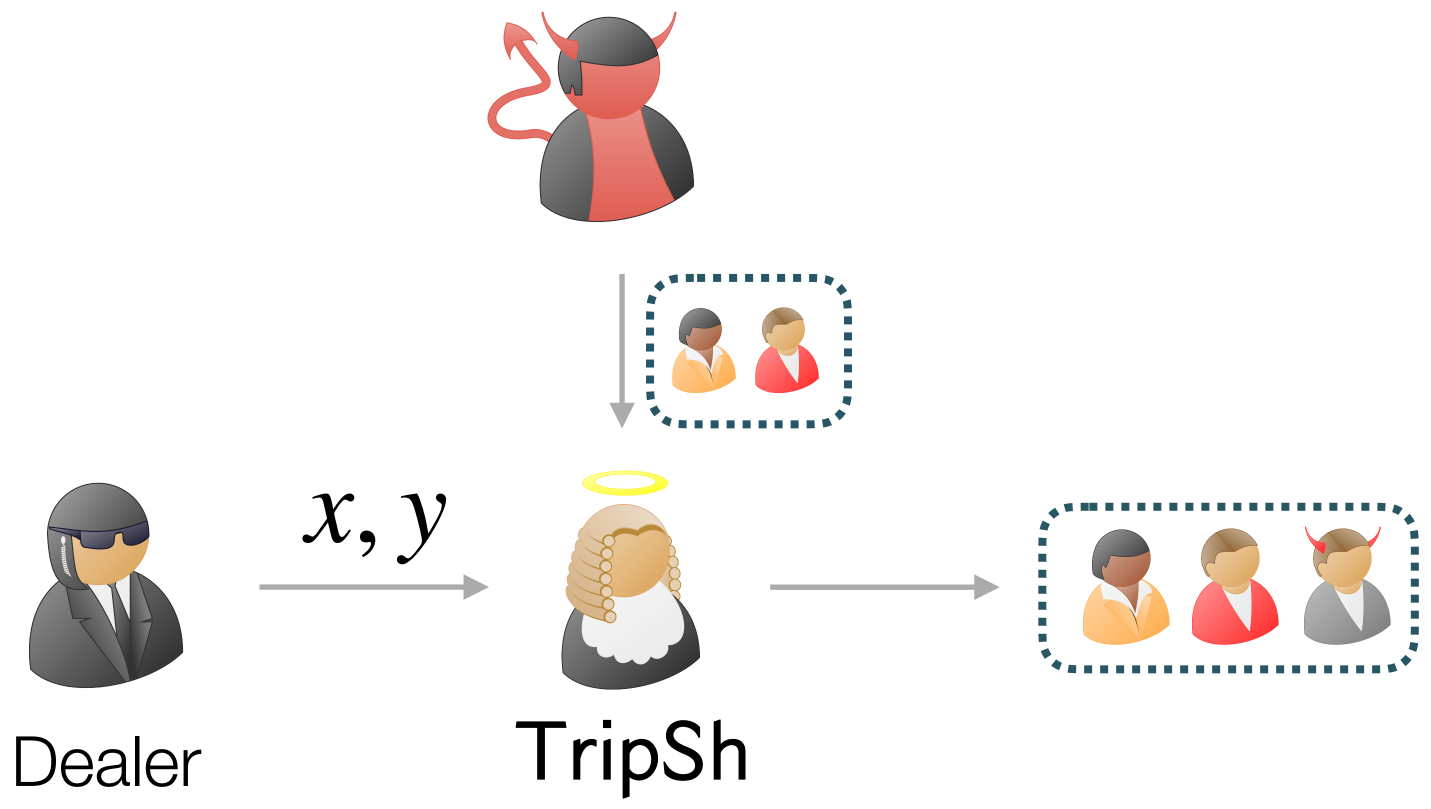


- Triple sharing with Party Elimination
  - Verified multiplication triple or dispute set

Perfect HMPC - Triple Sharing with Party Elimination Functionality



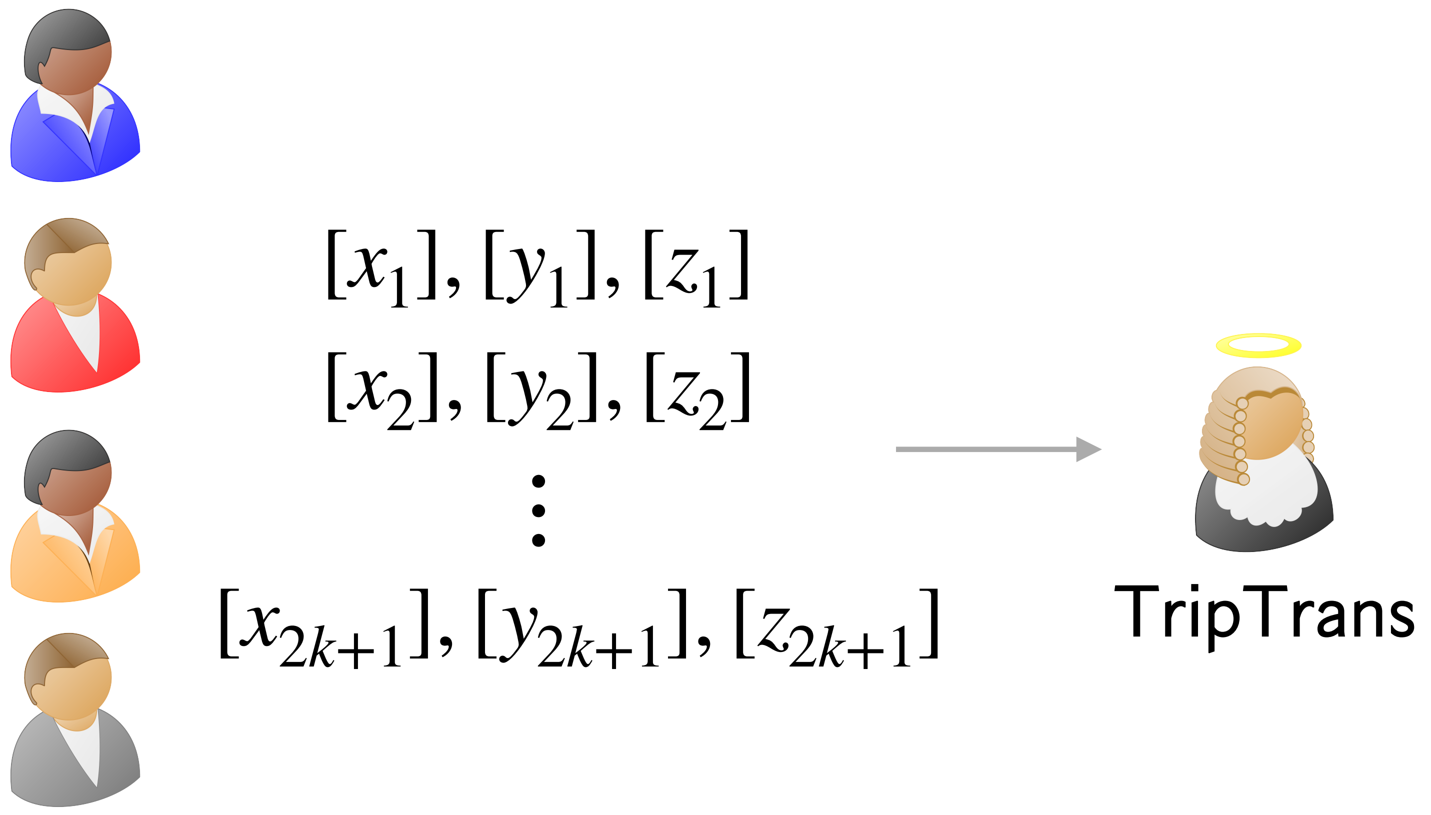
Dealer $\xrightarrow{x, y}$ TripSh $\longrightarrow$ $[x], [y], [xy]$



Dealer $\xrightarrow{x, y}$ TripSh $\longrightarrow$

- Triple sharing with Party Elimination
  - Verified multiplication triple or dispute set

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

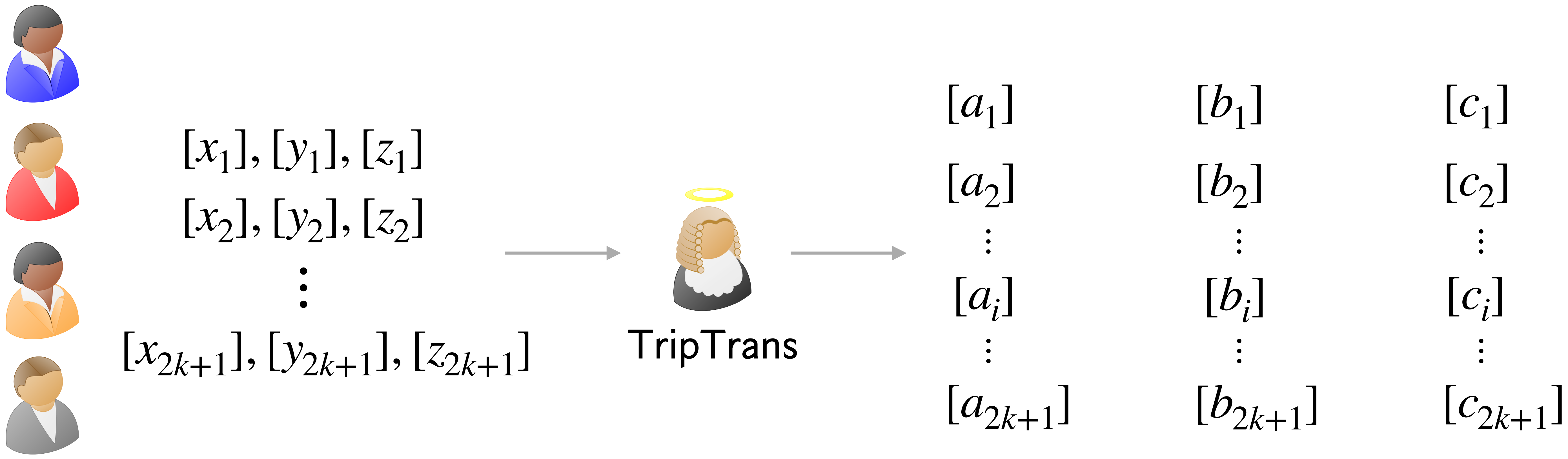
$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$\vdots$ $\vdots$ $\vdots$

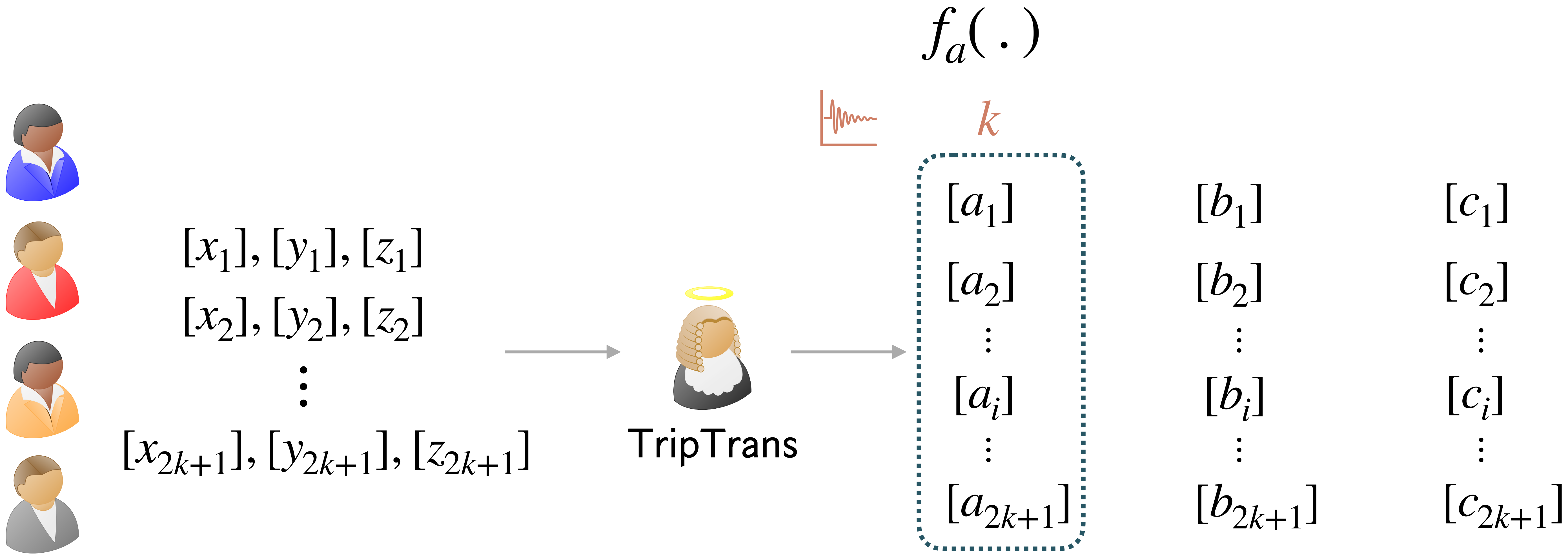
$[a_i]$ $[b_i]$ $[c_i]$

$\vdots$ $\vdots$ $\vdots$

$[a_{2k+1}]$ $[b_{2k+1}]$ $[c_{2k+1}]$

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

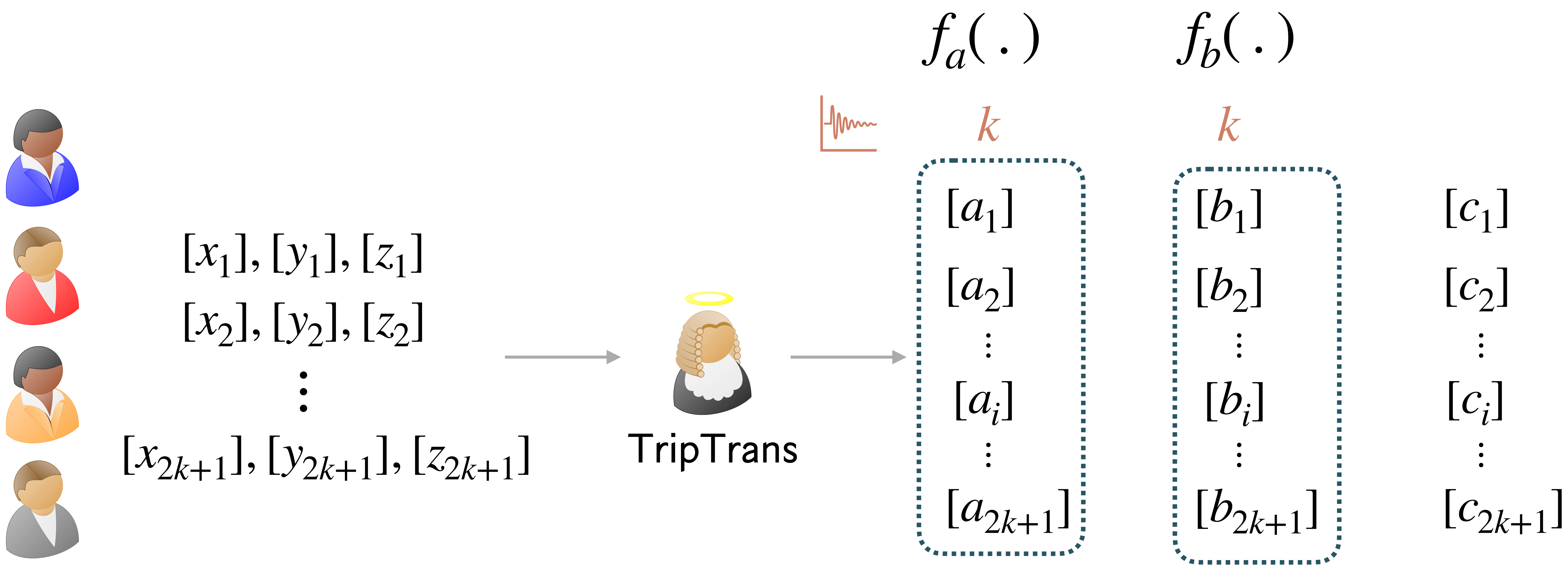
$f_a(\,.\,)$

k

$$\begin{array}{ccc} [a_1] & [b_1] & [c_1] \\ [a_2] & [b_2] & [c_2] \\ \vdots & \vdots & \vdots \\ [a_i] & [b_i] & [c_i] \\ \vdots & \vdots & \vdots \\ [a_{2k+1}] & [b_{2k+1}] & [c_{2k+1}] \end{array}$$

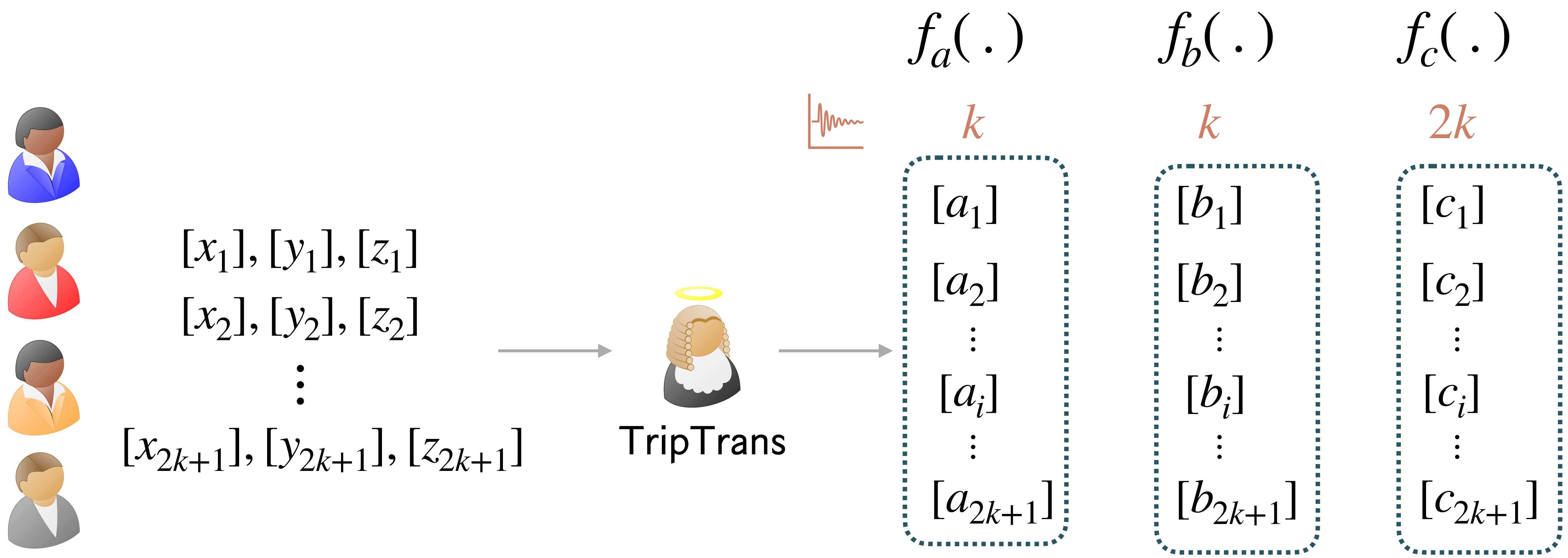
- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

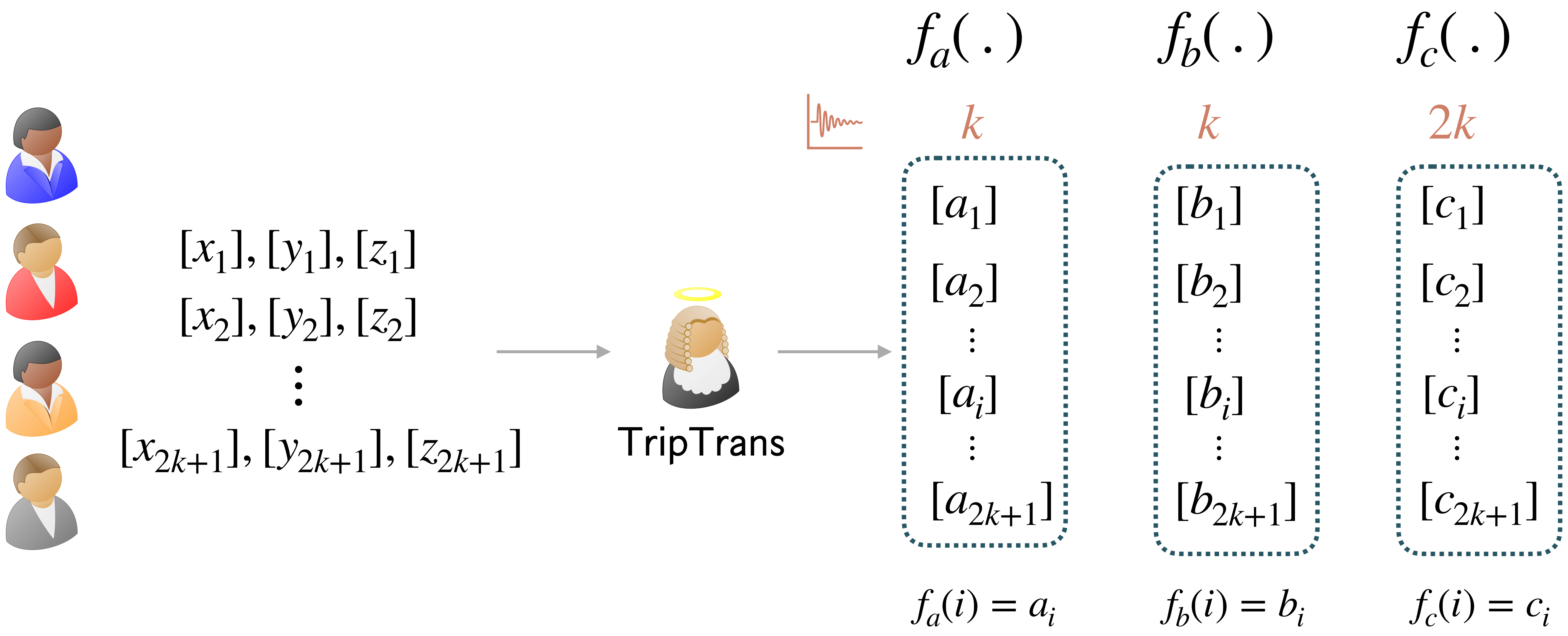
$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$	$f_b(\,.\,)$	$f_c(\,.\,)$
k	k	$2k$
$[a_1]$	$[b_1]$	$[c_1]$
$[a_2]$	$[b_2]$	$[c_2]$
$\vdots$	$\vdots$	$\vdots$
$[a_i]$	$[b_i]$	$[c_i]$
$\vdots$	$\vdots$	$\vdots$
$[a_{2k+1}]$	$[b_{2k+1}]$	$[c_{2k+1}]$

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$ $f_b(\,.\,)$ $f_c(\,.\,)$

k k $2k$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$\vdots$ $\vdots$ $\vdots$

$[a_i]$ $[b_i]$ $[c_i]$

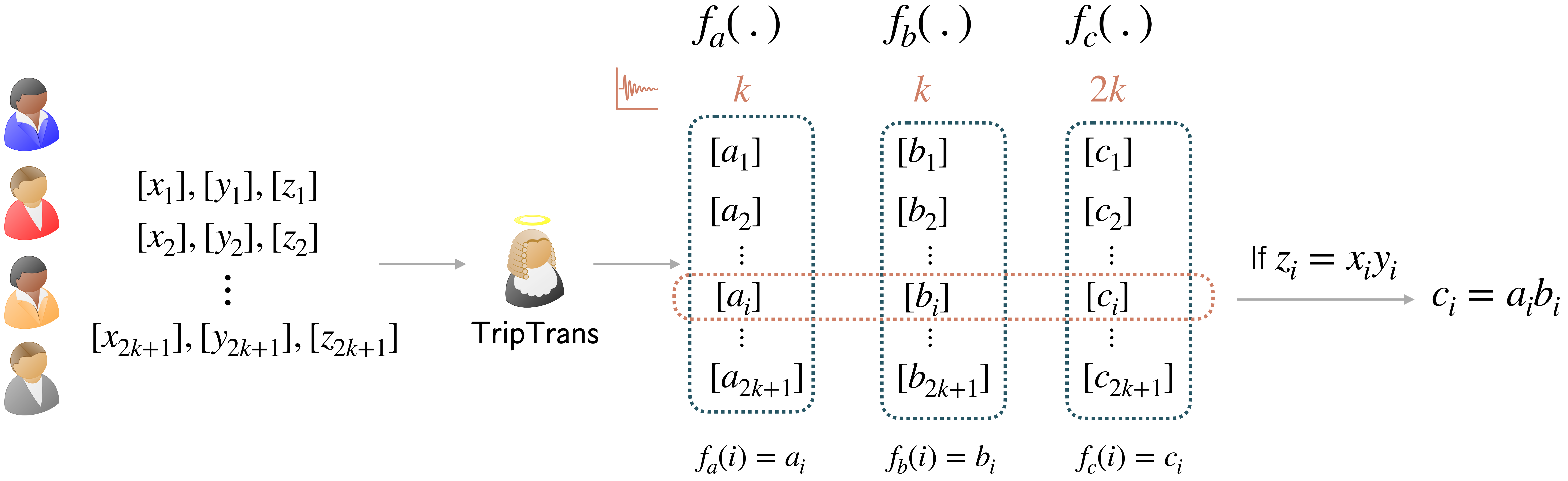
$\vdots$ $\vdots$ $\vdots$

$[a_{2k+1}]$ $[b_{2k+1}]$ $[c_{2k+1}]$

$f_a(i) = a_i$ $f_b(i) = b_i$ $f_c(i) = c_i$

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$ $f_b(\,.\,)$ $f_c(\,.\,)$

k k $2k$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$\vdots$ $\vdots$ $\vdots$

$[a_i]$ $[b_i]$ $[c_i]$

$\vdots$ $\vdots$ $\vdots$

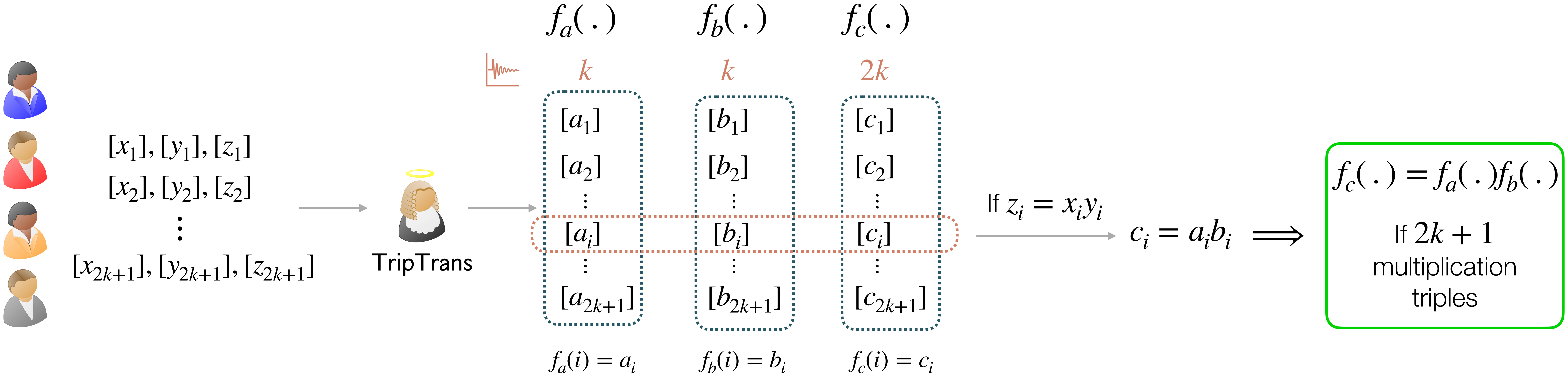
$[a_{2k+1}]$ $[b_{2k+1}]$ $[c_{2k+1}]$

$f_a(i) = a_i$ $f_b(i) = b_i$ $f_c(i) = c_i$

If $z_i = x_i y_i$ $\longrightarrow$ $c_i = a_i b_i$

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$ $f_b(\,.\,)$ $f_c(\,.\,)$

k k $2k$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$\vdots$ $\vdots$ $\vdots$

$[a_i]$ $[b_i]$ $[c_i]$

$\vdots$ $\vdots$ $\vdots$

$[a_{2k+1}]$ $[b_{2k+1}]$ $[c_{2k+1}]$

$f_a(i) = a_i$ $f_b(i) = b_i$ $f_c(i) = c_i$

If $z_i = x_i y_i$

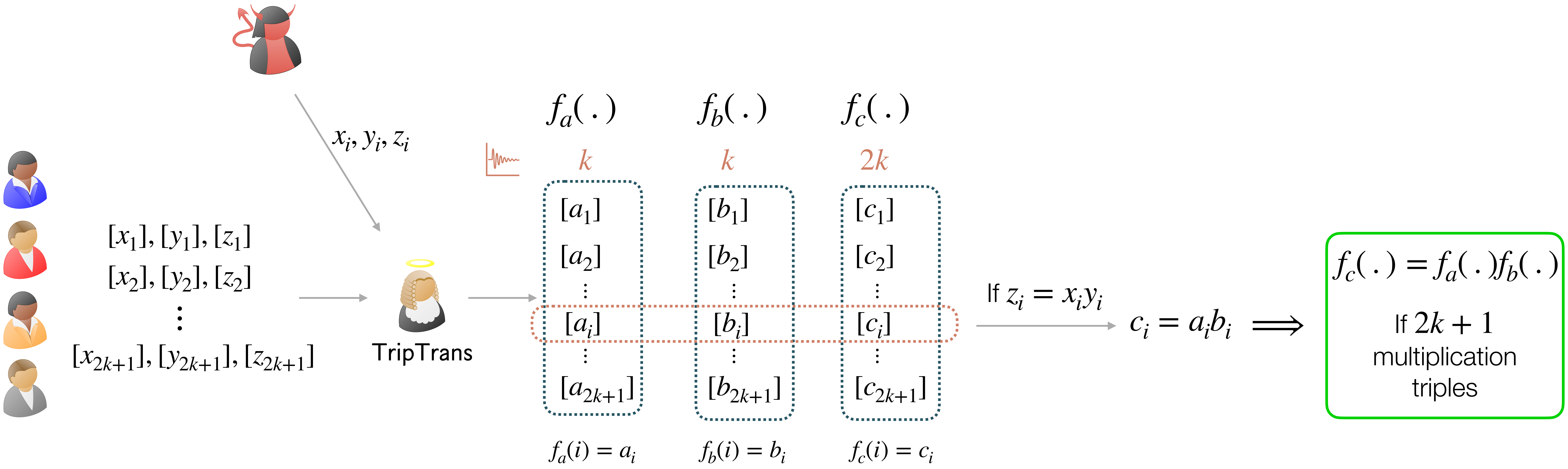
$c_i = a_i b_i \implies$

$f_c(\,.\,) = f_a(\,.\,) f_b(\,.\,)$

If $2k + 1$ multiplication triples

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



x_i, y_i, z_i

$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$	$f_b(\,.\,)$	$f_c(\,.\,)$
k	k	$2k$
$[a_1]$	$[b_1]$	$[c_1]$
$[a_2]$	$[b_2]$	$[c_2]$
$\vdots$	$\vdots$	$\vdots$
$[a_i]$	$[b_i]$	$[c_i]$
$\vdots$	$\vdots$	$\vdots$
$[a_{2k+1}]$	$[b_{2k+1}]$	$[c_{2k+1}]$
$f_a(i) = a_i$	$f_b(i) = b_i$	$f_c(i) = c_i$

If $z_i = x_i y_i$

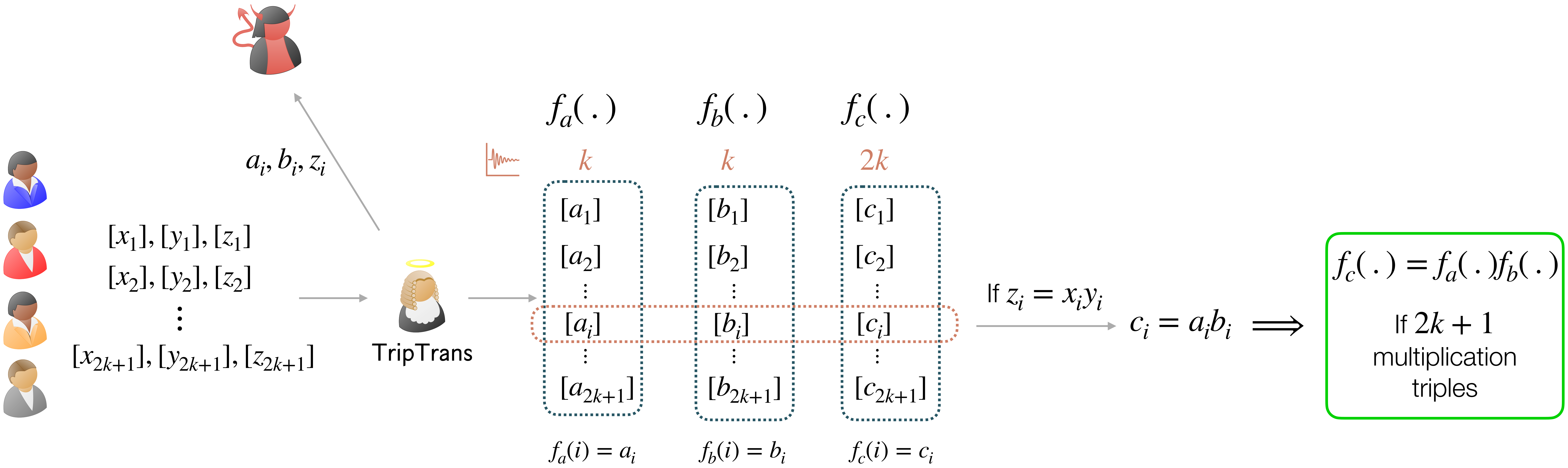
$c_i = a_i b_i \implies$

$f_c(\,.\,) = f_a(\,.\,) f_b(\,.\,)$

If $2k+1$ multiplication triples

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



a_i, b_i, z_i

$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2k+1}], [y_{2k+1}], [z_{2k+1}]$

TripTrans

$f_a(\,.\,)$	$f_b(\,.\,)$	$f_c(\,.\,)$
k	k	$2k$
$[a_1]$	$[b_1]$	$[c_1]$
$[a_2]$	$[b_2]$	$[c_2]$
$\vdots$	$\vdots$	$\vdots$
$[a_i]$	$[b_i]$	$[c_i]$
$\vdots$	$\vdots$	$\vdots$
$[a_{2k+1}]$	$[b_{2k+1}]$	$[c_{2k+1}]$
$f_a(i) = a_i$	$f_b(i) = b_i$	$f_c(i) = c_i$

If $z_i = x_i y_i$

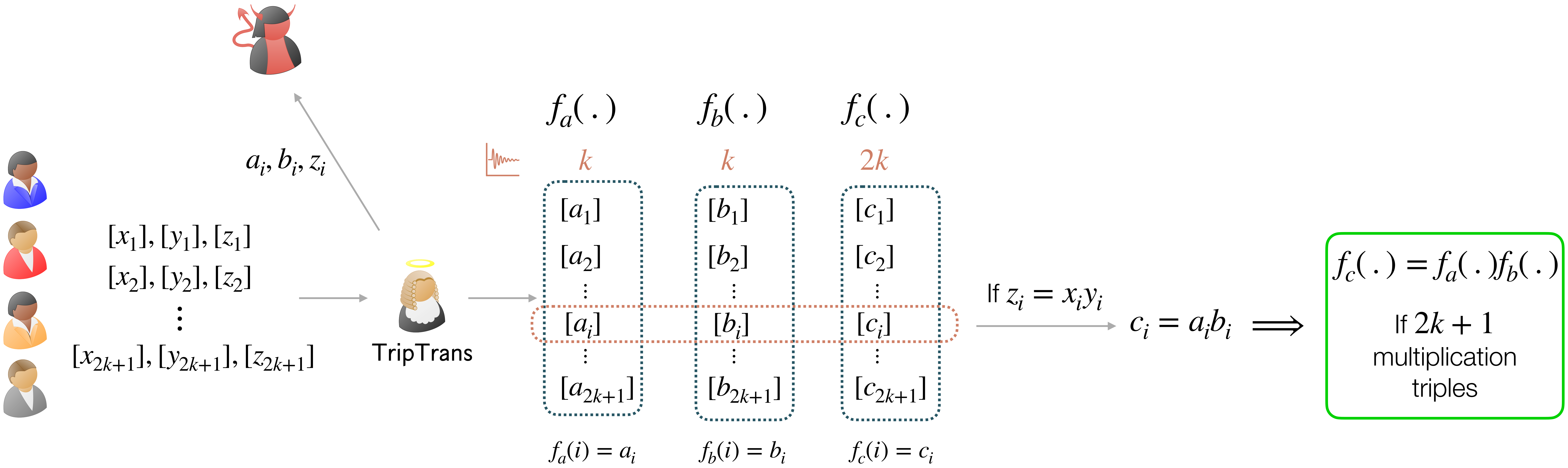
$c_i = a_i b_i \implies$

$f_c(\,.\,) = f_a(\,.\,)f_b(\,.\,)$

If $2k+1$ multiplication triples

- Random triples → correlated random triples

Perfect HMPC - Triple Transform Functionality



- Random triples → correlated random triples
- Completely asynchronous instantiation in [CP17]

Perfect HMPC - Triple Sharing with Party Elimination Protocol

Perfect HMPC - Triple Sharing with Party Elimination Protocol

$x_1 \quad y_1 \quad x_1y_1$

$x_2 \quad y_2 \quad x_2y_2$

$x_3 \quad y_3 \quad x_3y_3$

Perfect HMPC - Triple Sharing with Party Elimination Protocol

x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

Perfect HMPC - Triple Sharing with Party Elimination Protocol

x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

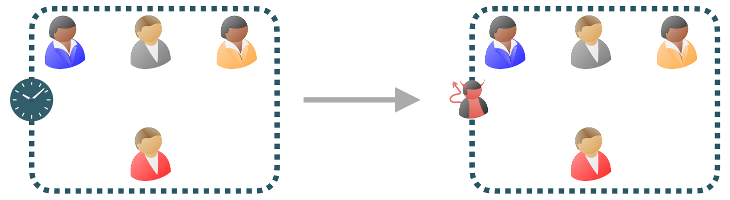
u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

Sh



$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

$[u_1]$ $[v_1]$ $[w_1]$

$[u_2]$ $[v_2]$ $[w_2]$

$[u_3]$ $[v_3]$ $[w_3]$

$[u_4]$ $[v_4]$ $[w_4]$

Perfect HMPC - Triple Sharing with Party Elimination Protocol

$x_1 \quad y_1 \quad x_1y_1$

$x_2 \quad y_2 \quad x_2y_2$

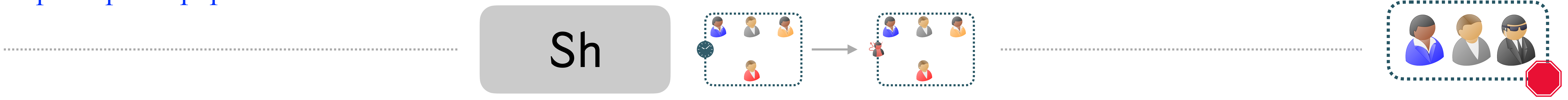
$x_3 \quad y_3 \quad x_3y_3$

$u_1 \quad v_1 \quad u_1v_1$

$u_2 \quad v_2 \quad u_2v_2$

$u_3 \quad v_3 \quad u_3v_3$

$u_4 \quad v_4 \quad u_4v_4$



$[x_1] \; [y_1] \; [z_1]$

$[x_2] \; [y_2] \; [z_2]$

$[x_3] \; [y_3] \; [z_3]$

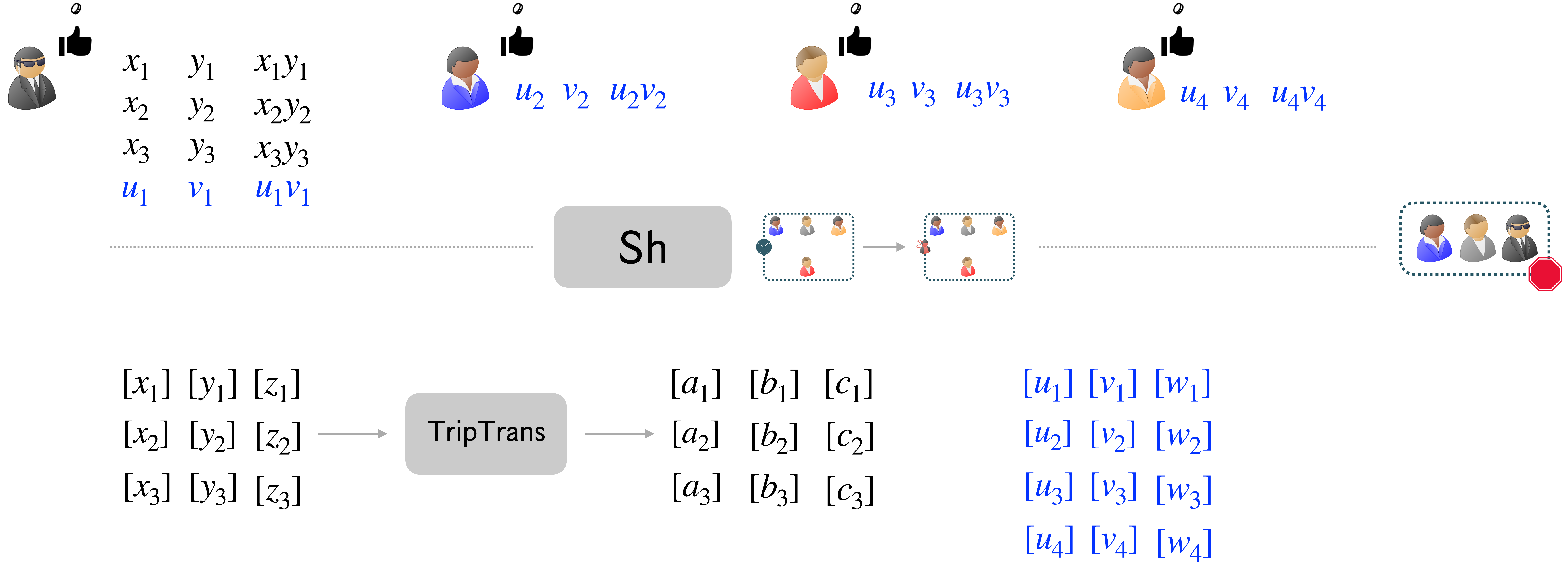
$[u_1] \; [v_1] \; [w_1]$

$[u_2] \; [v_2] \; [w_2]$

$[u_3] \; [v_3] \; [w_3]$

$[u_4] \; [v_4] \; [w_4]$

Perfect HMPC - Triple Sharing with Party Elimination Protocol



x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$[a_3]$ $[b_3]$ $[c_3]$

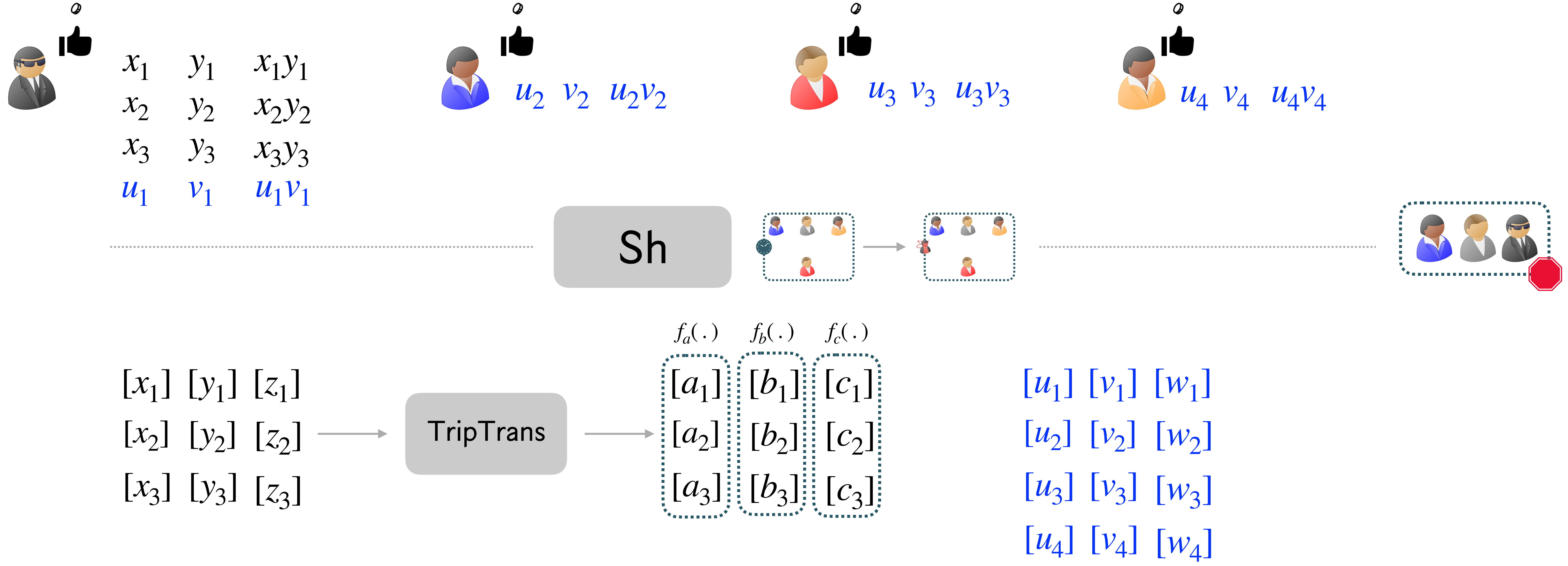
$[u_1]$ $[v_1]$ $[w_1]$

$[u_2]$ $[v_2]$ $[w_2]$

$[u_3]$ $[v_3]$ $[w_3]$

$[u_4]$ $[v_4]$ $[w_4]$

Perfect HMPC - Triple Sharing with Party Elimination Protocol



$x_1 \quad y_1 \quad x_1y_1$

$x_2 \quad y_2 \quad x_2y_2$

$x_3 \quad y_3 \quad x_3y_3$

$u_1 \quad v_1 \quad u_1v_1$

$u_2 \quad v_2 \quad u_2v_2$

$u_3 \quad v_3 \quad u_3v_3$

$u_4 \quad v_4 \quad u_4v_4$

Sh

$[x_1] \; [y_1] \; [z_1]$

$[x_2] \; [y_2] \; [z_2]$

$[x_3] \; [y_3] \; [z_3]$

TripTrans

$f_a(.) \quad f_b(.) \quad f_c(.)$

$[a_1] \; [b_1] \; [c_1]$

$[a_2] \; [b_2] \; [c_2]$

$[a_3] \; [b_3] \; [c_3]$

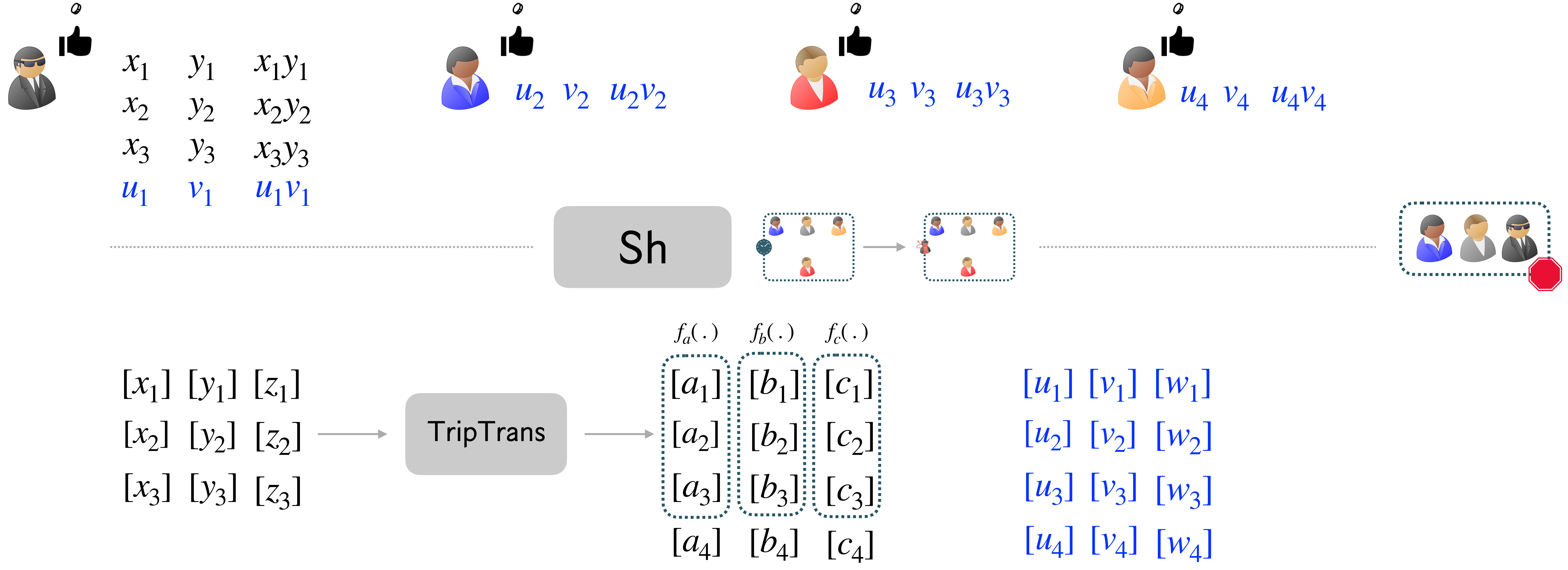
$[u_1] \; [v_1] \; [w_1]$

$[u_2] \; [v_2] \; [w_2]$

$[u_3] \; [v_3] \; [w_3]$

$[u_4] \; [v_4] \; [w_4]$

Perfect HMPC - Triple Sharing with Party Elimination Protocol



x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

Sh

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

TripTrans

$f_a(.)$ $f_b(.)$ $f_c(.)$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$[a_3]$ $[b_3]$ $[c_3]$

$[a_4]$ $[b_4]$ $[c_4]$

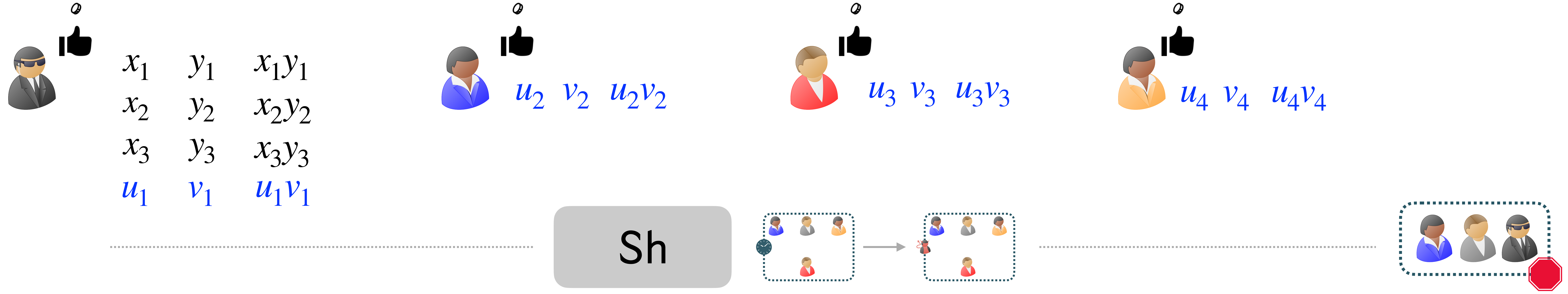
$[u_1]$ $[v_1]$ $[w_1]$

$[u_2]$ $[v_2]$ $[w_2]$

$[u_3]$ $[v_3]$ $[w_3]$

$[u_4]$ $[v_4]$ $[w_4]$

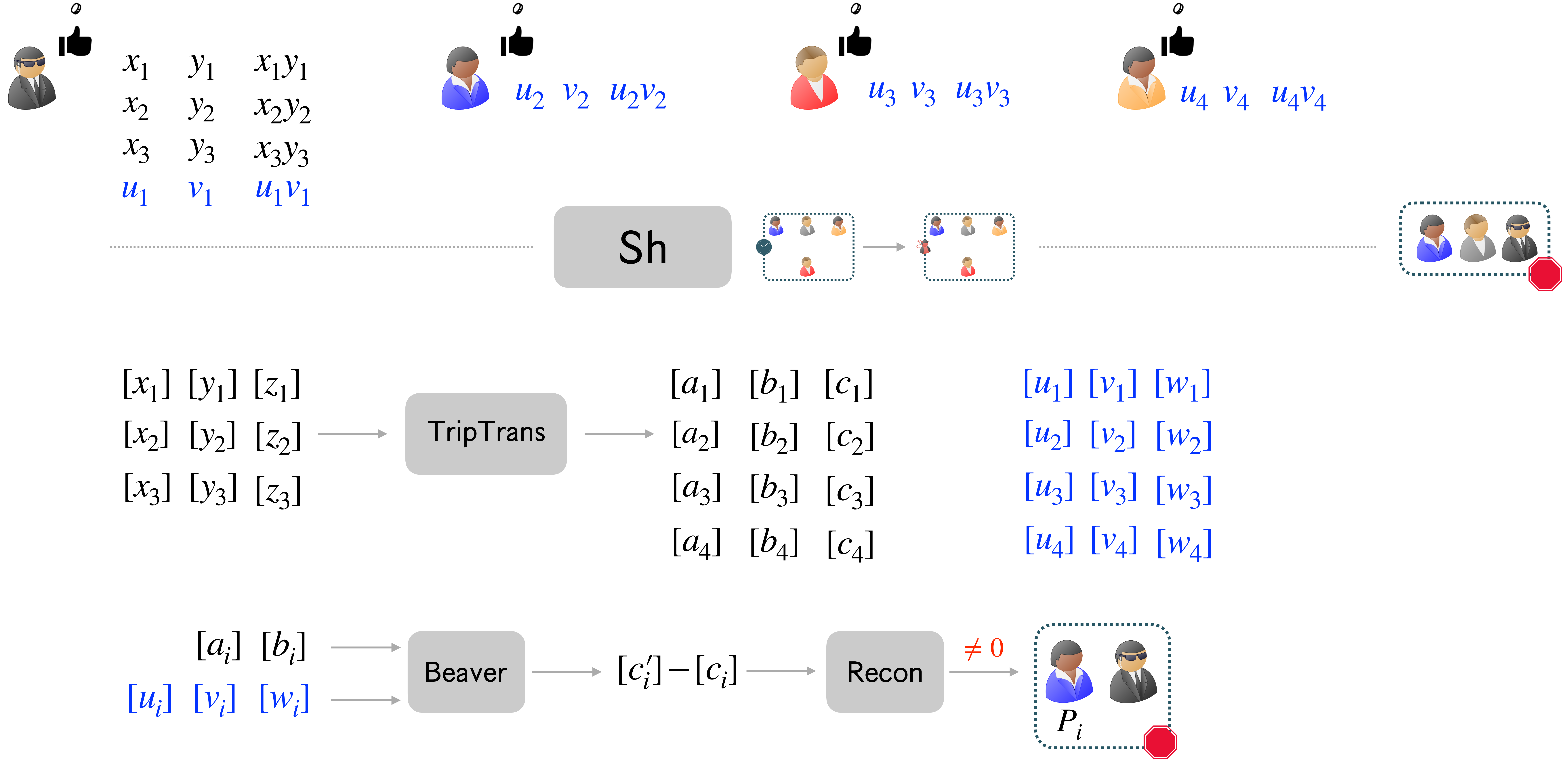
Perfect HMPC - Triple Sharing with Party Elimination Protocol



$$\begin{matrix}[x_1] & [y_1] & [z_1]\\ [x_2] & [y_2] & [z_2]\\ [x_3] & [y_3] & [z_3]\end{matrix} \longrightarrow \boxed{\text{TripTrans}} \longrightarrow \begin{matrix}[a_1] & [b_1] & [c_1]\\ [a_2] & [b_2] & [c_2]\\ [a_3] & [b_3] & [c_3]\\ [a_4] & [b_4] & [c_4]\end{matrix} \qquad \begin{matrix}[u_1] & [v_1] & [w_1]\\ [u_2] & [v_2] & [w_2]\\ [u_3] & [v_3] & [w_3]\\ [u_4] & [v_4] & [w_4]\end{matrix}$$

$$\begin{matrix}[a_i] \ [b_i] \longrightarrow\\ [u_i] \ [v_i] \ [w_i] \longrightarrow\end{matrix} \boxed{\text{Beaver}} \longrightarrow [c_i']$$

Perfect HMPC - Triple Sharing with Party Elimination Protocol



x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

$[a_1]$ $[b_1]$ $[c_1]$ $[u_1]$ $[v_1]$ $[w_1]$

$[a_2]$ $[b_2]$ $[c_2]$ $[u_2]$ $[v_2]$ $[w_2]$

$[a_3]$ $[b_3]$ $[c_3]$ $[u_3]$ $[v_3]$ $[w_3]$

$[a_4]$ $[b_4]$ $[c_4]$ $[u_4]$ $[v_4]$ $[w_4]$

$[a_i]$ $[b_i]$

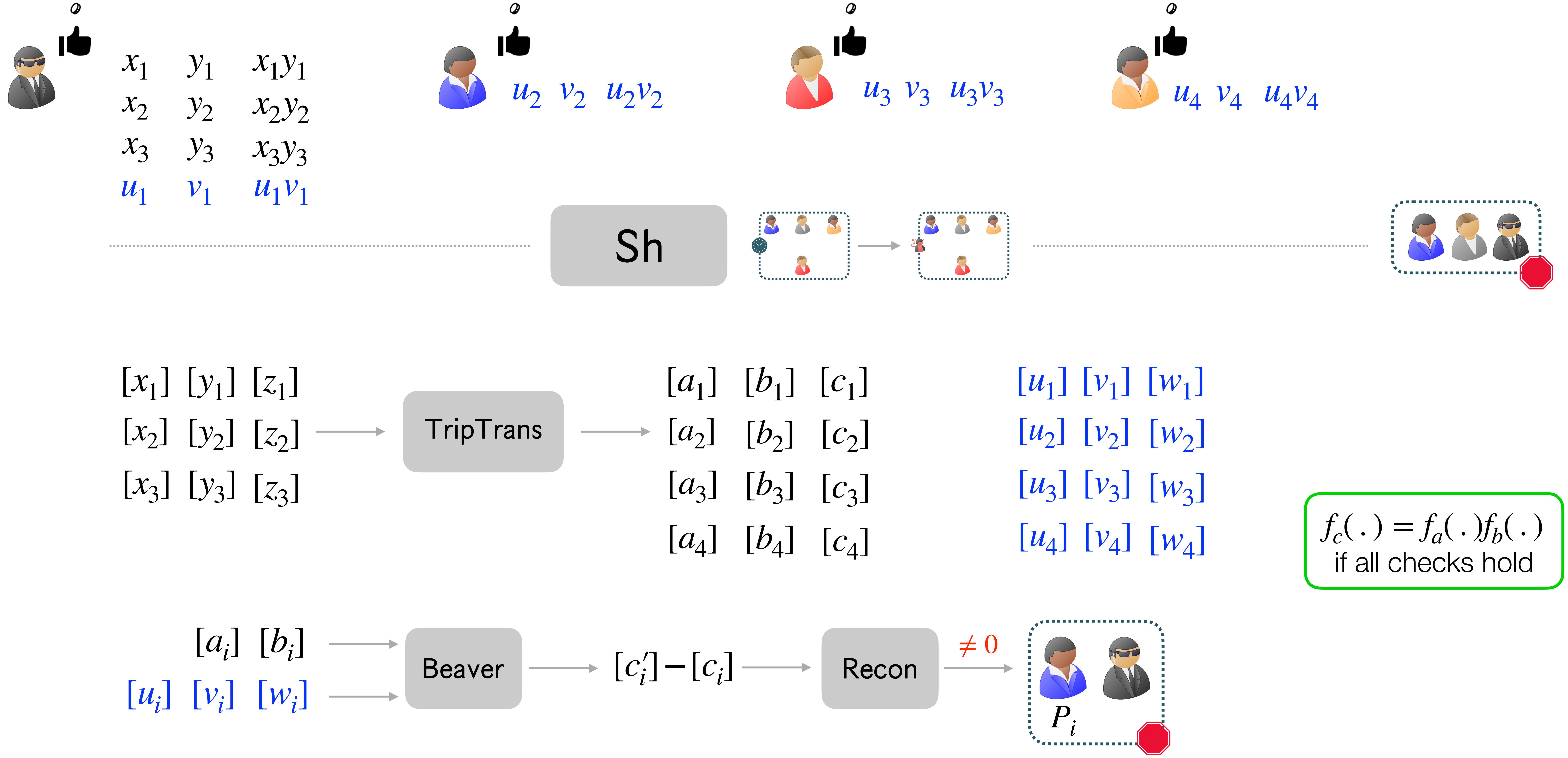
$[u_i]$ $[v_i]$ $[w_i]$

$[c_i'] - [c_i]$

$\neq 0$

P_i

Perfect HMPC - Triple Sharing with Party Elimination Protocol



$x_1 \quad y_1 \quad x_1y_1$

$x_2 \quad y_2 \quad x_2y_2$

$x_3 \quad y_3 \quad x_3y_3$

$u_1 \quad v_1 \quad u_1v_1$

$u_2 \quad v_2 \quad u_2v_2$

$u_3 \quad v_3 \quad u_3v_3$

$u_4 \quad v_4 \quad u_4v_4$

Sh

$[x_1]\ [y_1]\ [z_1]$

$[x_2]\ [y_2]\ [z_2]$

$[x_3]\ [y_3]\ [z_3]$

TripTrans

$[a_1]\ [b_1]\ [c_1] \qquad [u_1]\ [v_1]\ [w_1]$

$[a_2]\ [b_2]\ [c_2] \qquad [u_2]\ [v_2]\ [w_2]$

$[a_3]\ [b_3]\ [c_3] \qquad [u_3]\ [v_3]\ [w_3]$

$[a_4]\ [b_4]\ [c_4] \qquad [u_4]\ [v_4]\ [w_4]$

$f_c(\,.\,) = f_a(\,.\,)f_b(\,.\,)$
if all checks hold

$[a_i]\ [b_i]$

$[u_i]\ [v_i]\ [w_i]$

Beaver

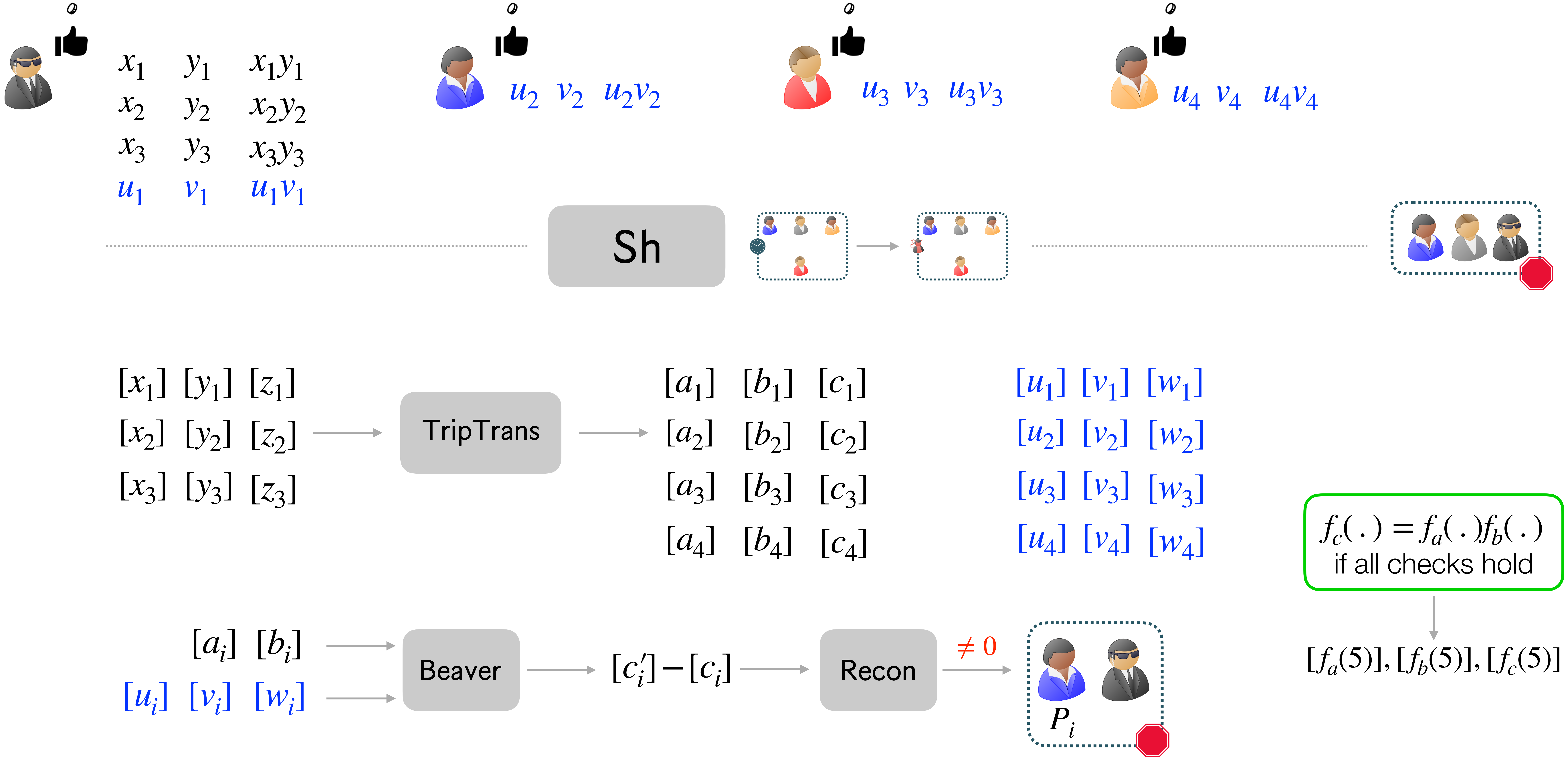
$[c_i'] - [c_i]$

Recon

$\neq 0$

P_i

Perfect HMPC - Triple Sharing with Party Elimination Protocol



x_1 y_1 x_1y_1

x_2 y_2 x_2y_2

x_3 y_3 x_3y_3

u_1 v_1 u_1v_1

u_2 v_2 u_2v_2

u_3 v_3 u_3v_3

u_4 v_4 u_4v_4

Sh

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

TripTrans

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$[a_3]$ $[b_3]$ $[c_3]$

$[a_4]$ $[b_4]$ $[c_4]$

$[u_1]$ $[v_1]$ $[w_1]$

$[u_2]$ $[v_2]$ $[w_2]$

$[u_3]$ $[v_3]$ $[w_3]$

$[u_4]$ $[v_4]$ $[w_4]$

$f_c(\,.\,) = f_a(\,.\,)f_b(\,.\,)$
if all checks hold

$[f_a(5)], [f_b(5)], [f_c(5)]$

$[a_i]$ $[b_i]$

$[u_i]$ $[v_i]$ $[w_i]$

Beaver

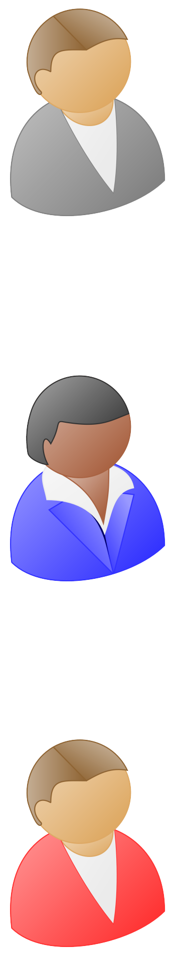
$[c_i'] - [c_i]$

Recon

$\neq 0$

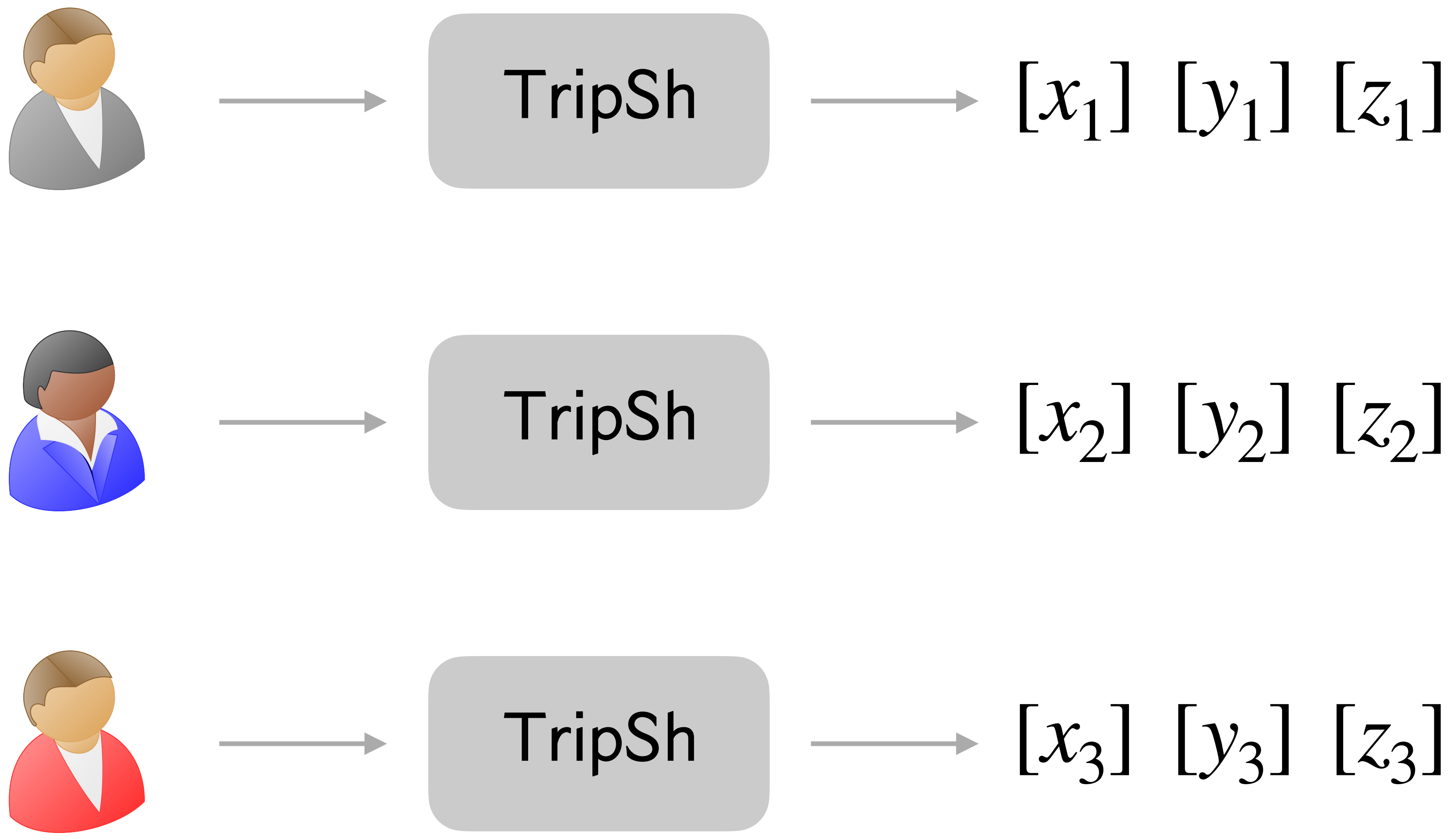
P_i

Perfect HMPC - Triple Generation with Party Elimination



TripGen

Perfect HMPC - Triple Generation with Party Elimination



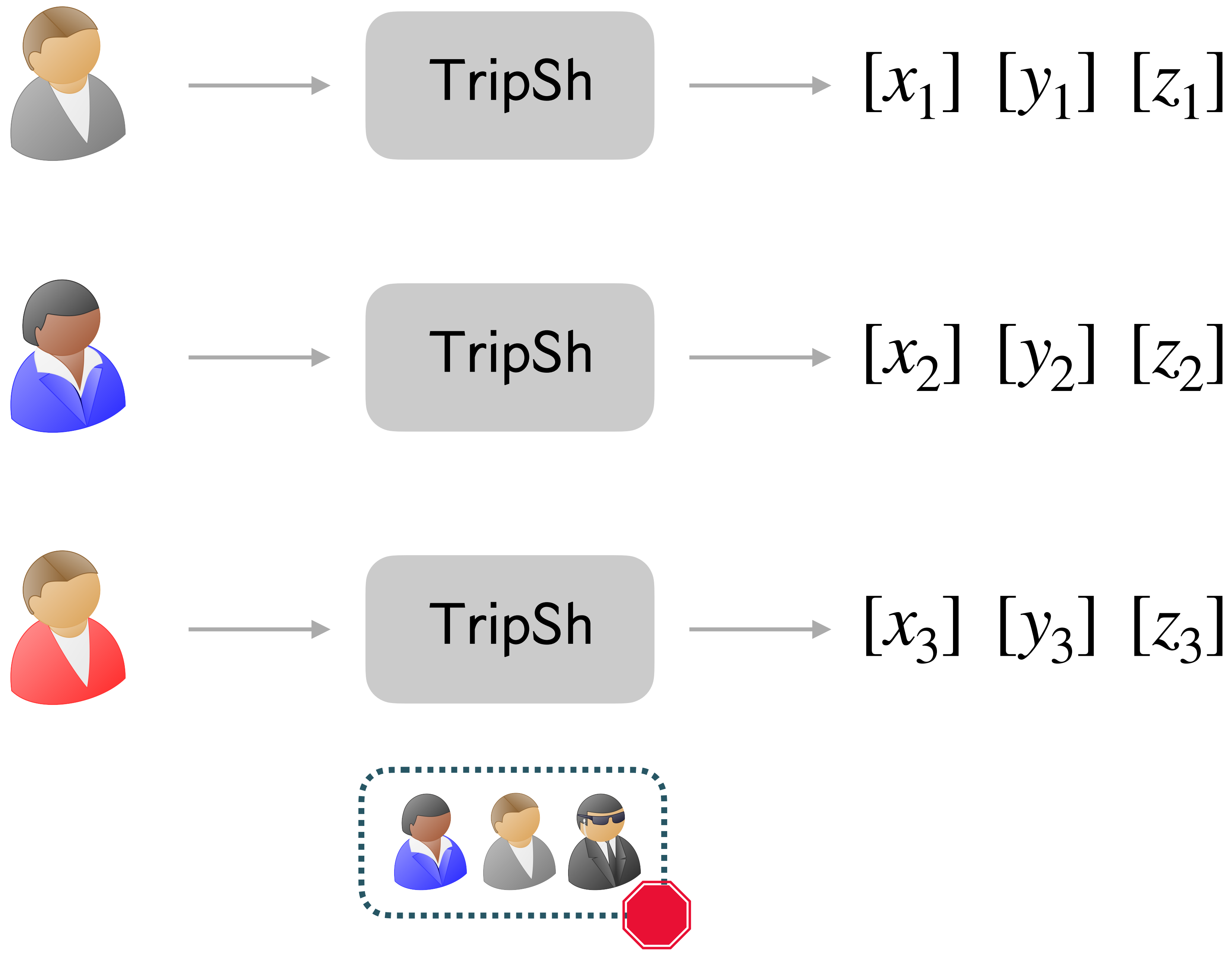
TripSh → $[x_1]$ $[y_1]$ $[z_1]$

TripSh → $[x_2]$ $[y_2]$ $[z_2]$

TripSh → $[x_3]$ $[y_3]$ $[z_3]$

TripGen

Perfect HMPC - Triple Generation with Party Elimination



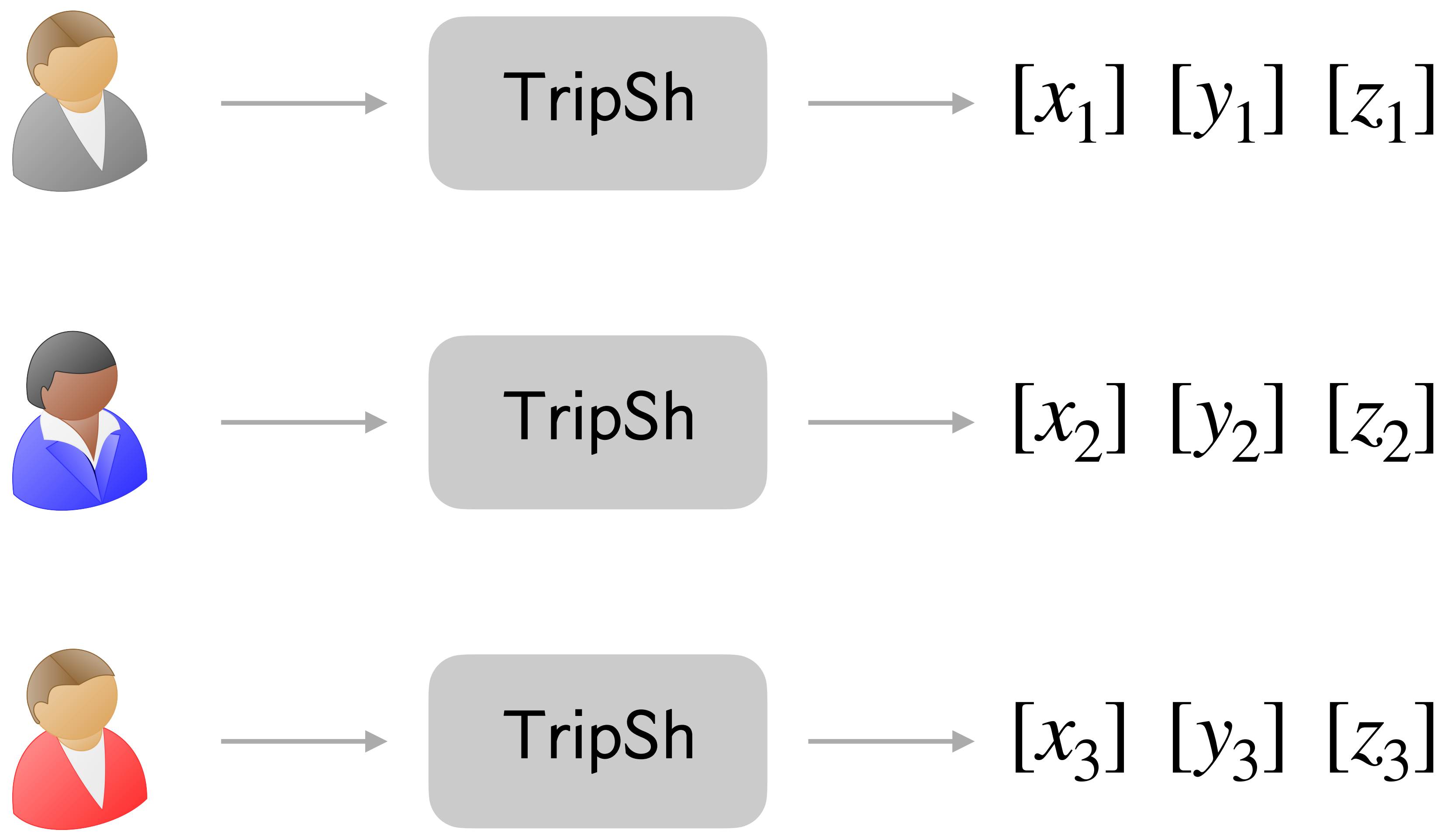
TripSh → $[x_1]$ $[y_1]$ $[z_1]$

TripSh → $[x_2]$ $[y_2]$ $[z_2]$

TripSh → $[x_3]$ $[y_3]$ $[z_3]$

TripGen

Perfect HMPC - Triple Generation with Party Elimination



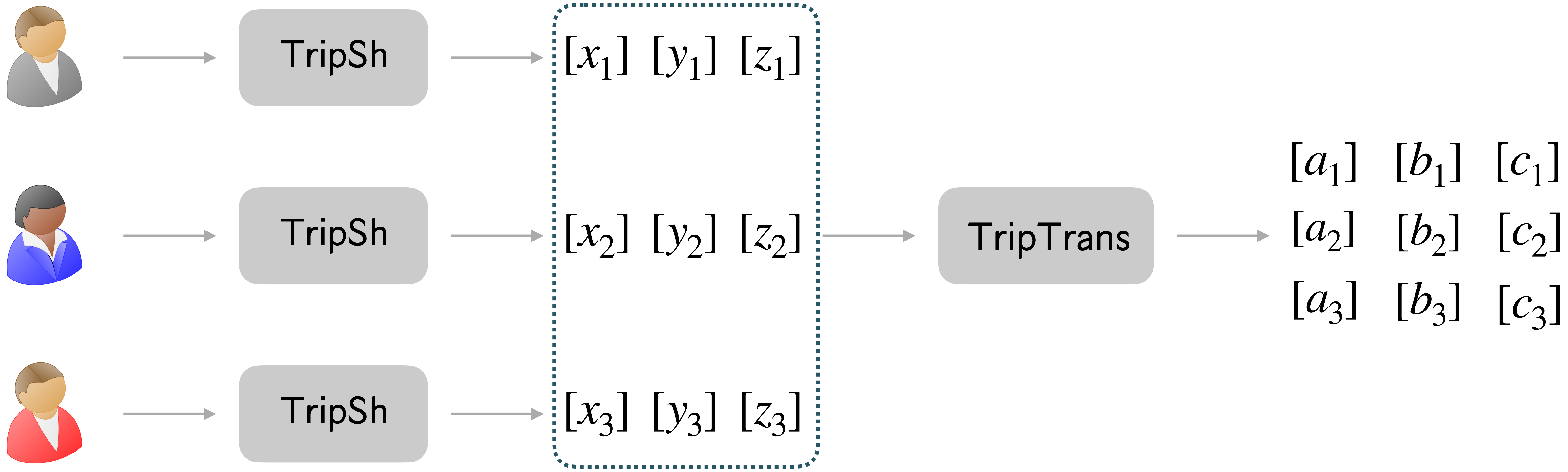
TripSh $\rightarrow [x_1]\ [y_1]\ [z_1]$

TripSh $\rightarrow [x_2]\ [y_2]\ [z_2]$

TripSh $\rightarrow [x_3]\ [y_3]\ [z_3]$

TripGen

Perfect HMPC - Triple Generation with Party Elimination



TripSh

TripSh

TripSh

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

TripTrans

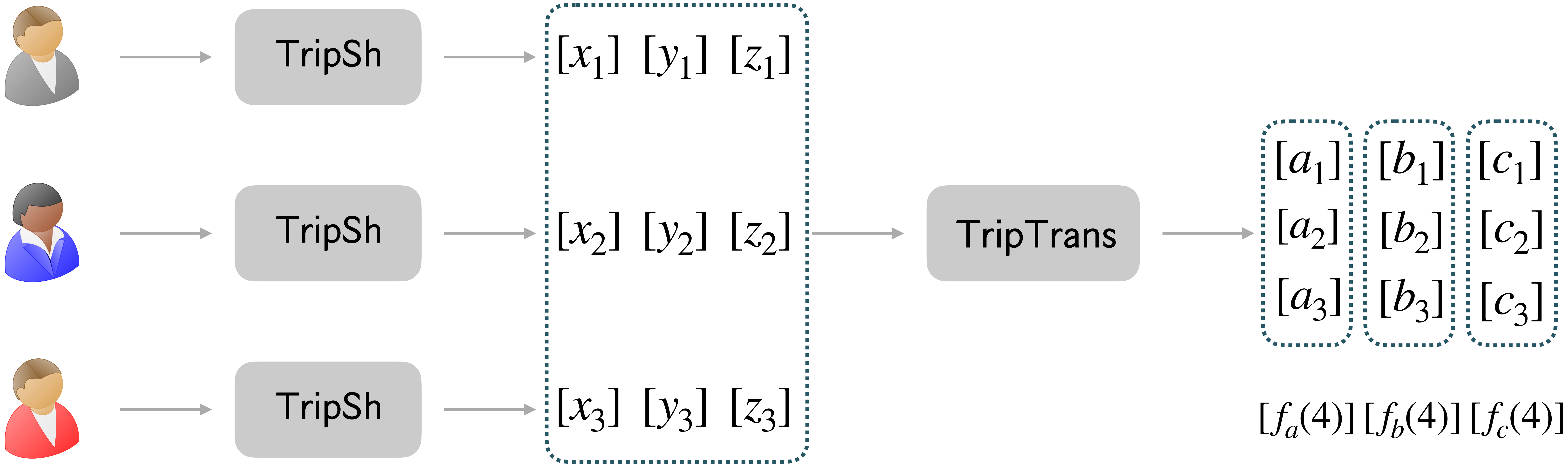
$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$[a_3]$ $[b_3]$ $[c_3]$

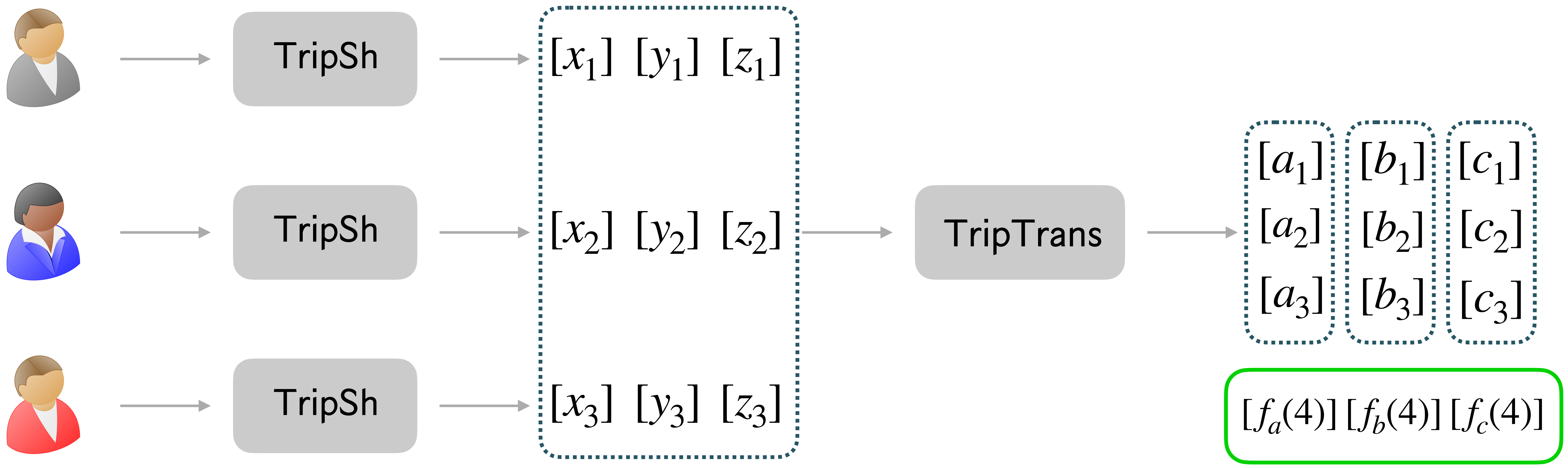
TripGen

Perfect HMPC - Triple Generation with Party Elimination



TripGen

Perfect HMPC - Triple Generation with Party Elimination



TripSh

TripSh

TripSh

$[x_1]$ $[y_1]$ $[z_1]$

$[x_2]$ $[y_2]$ $[z_2]$

$[x_3]$ $[y_3]$ $[z_3]$

TripTrans

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$[a_3]$ $[b_3]$ $[c_3]$

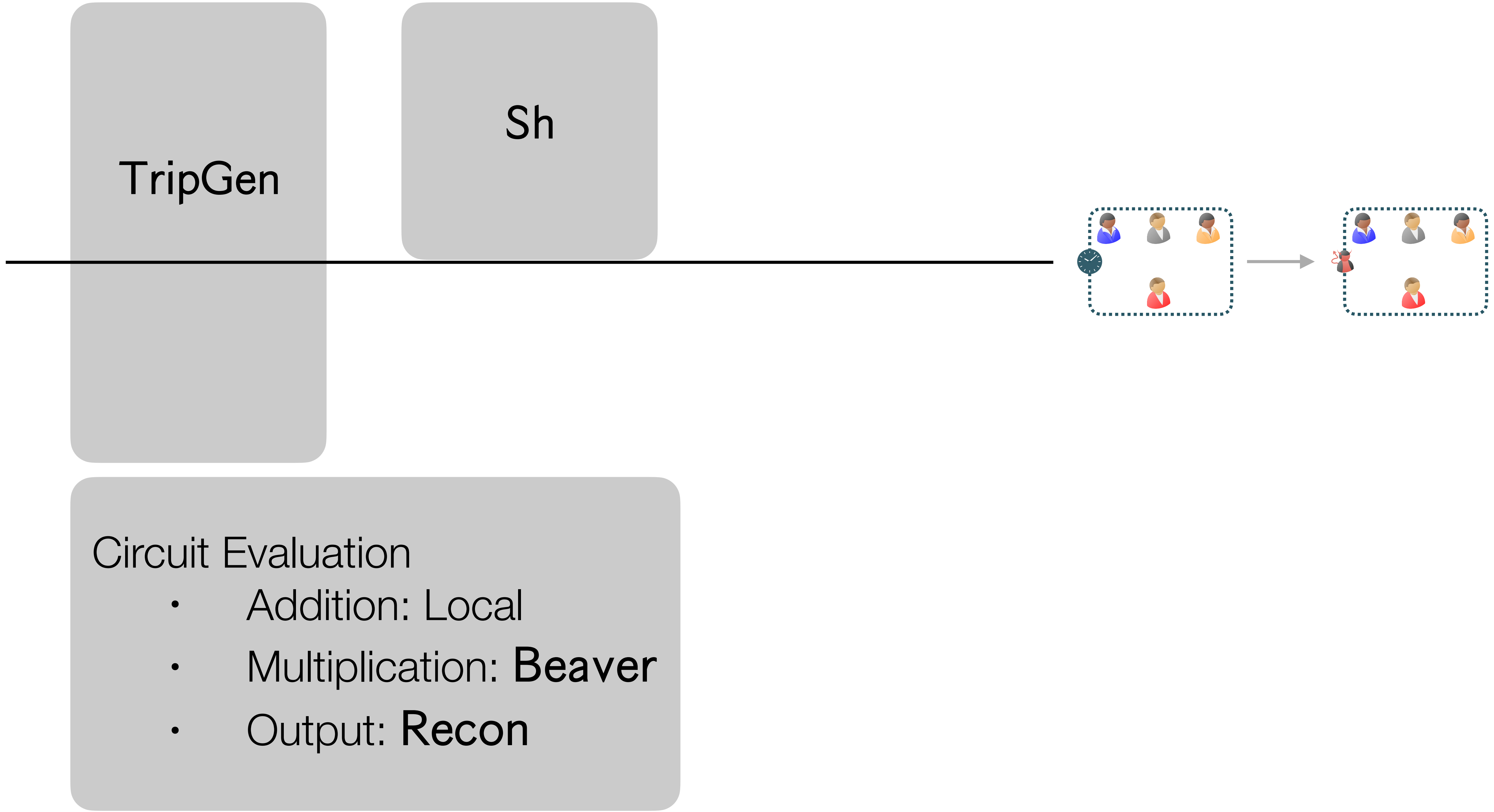
$[f_a(4)]$ $[f_b(4)]$ $[f_c(4)]$

Random and private
multiplication triple.

TripGen

Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase

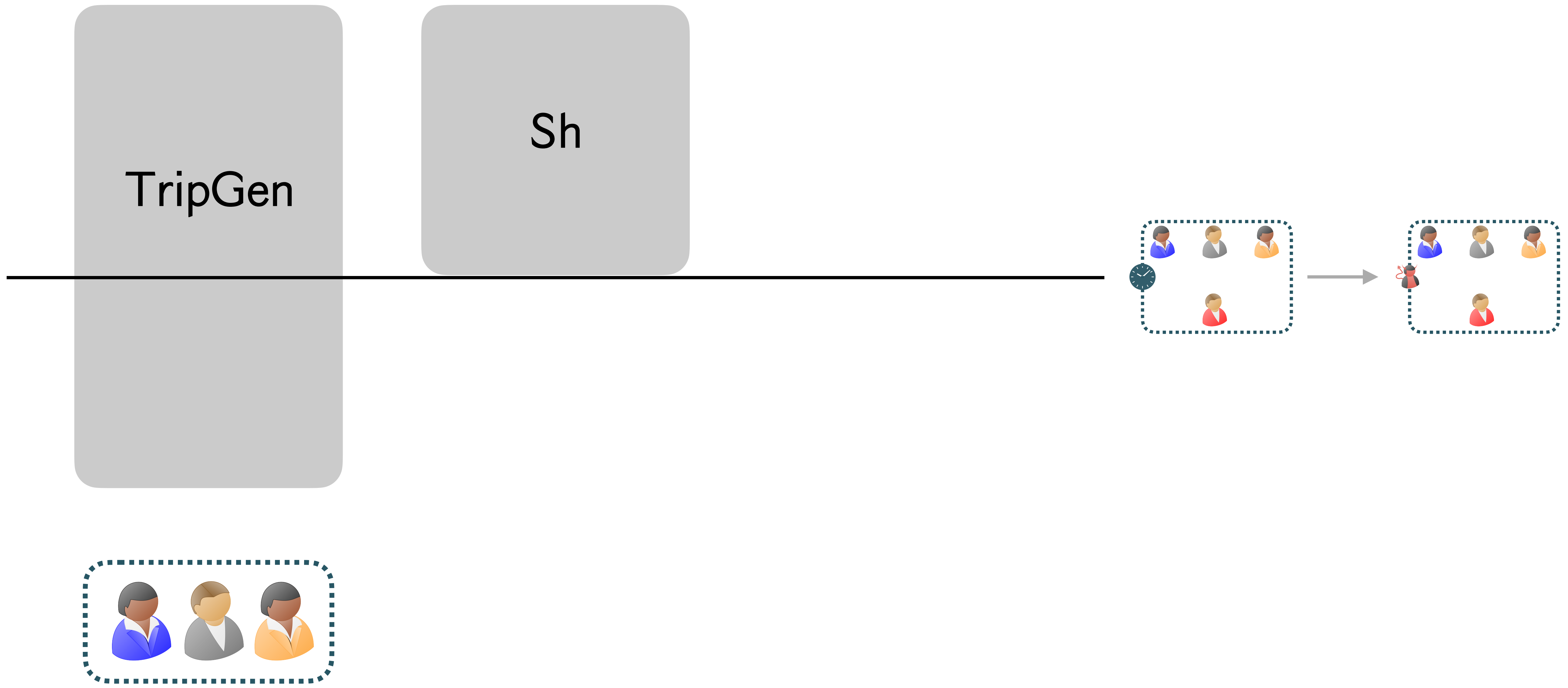


Circuit Evaluation

- Addition: Local
- Multiplication: **Beaver**
- Output: **Recon**

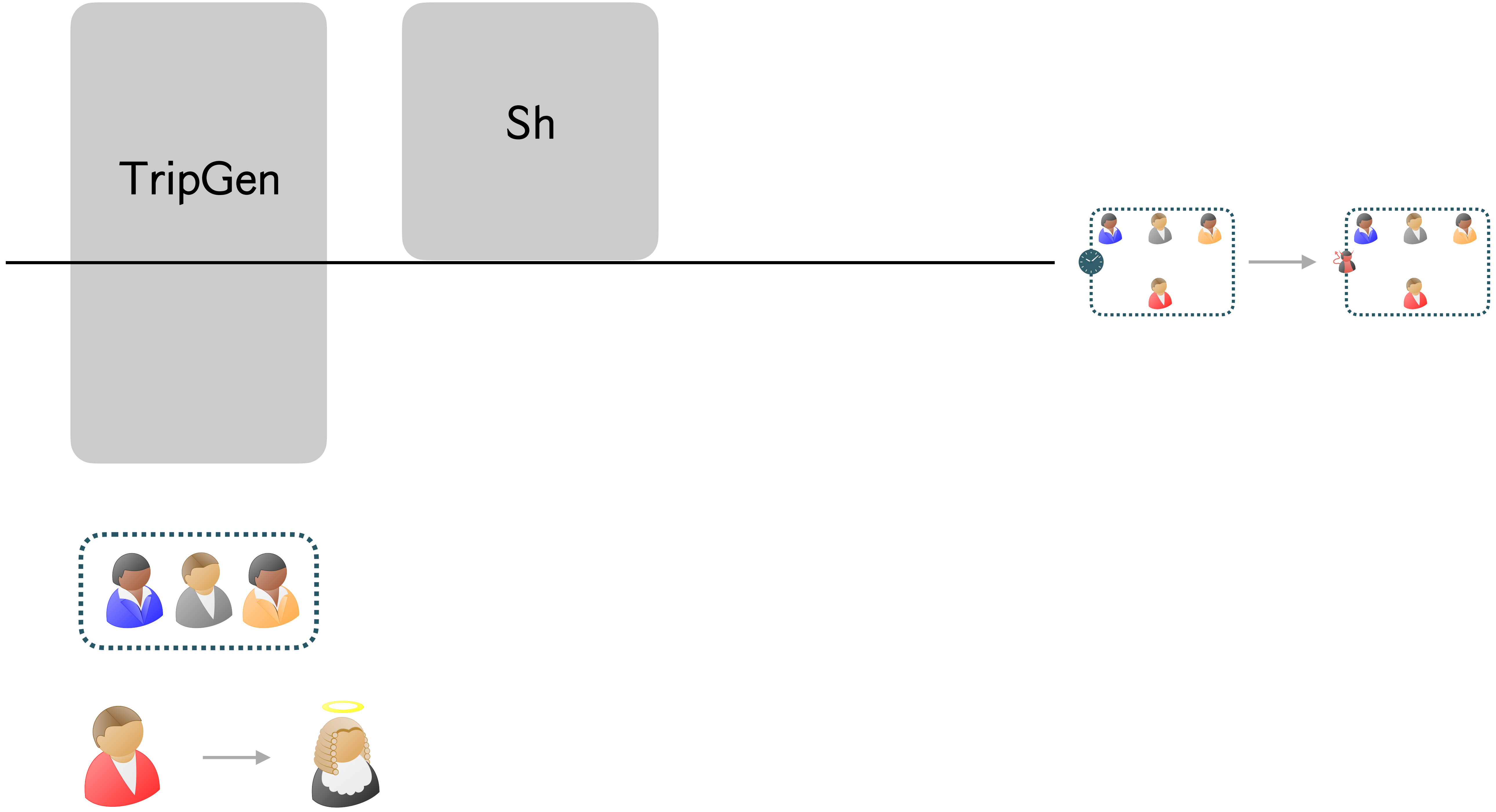
Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase



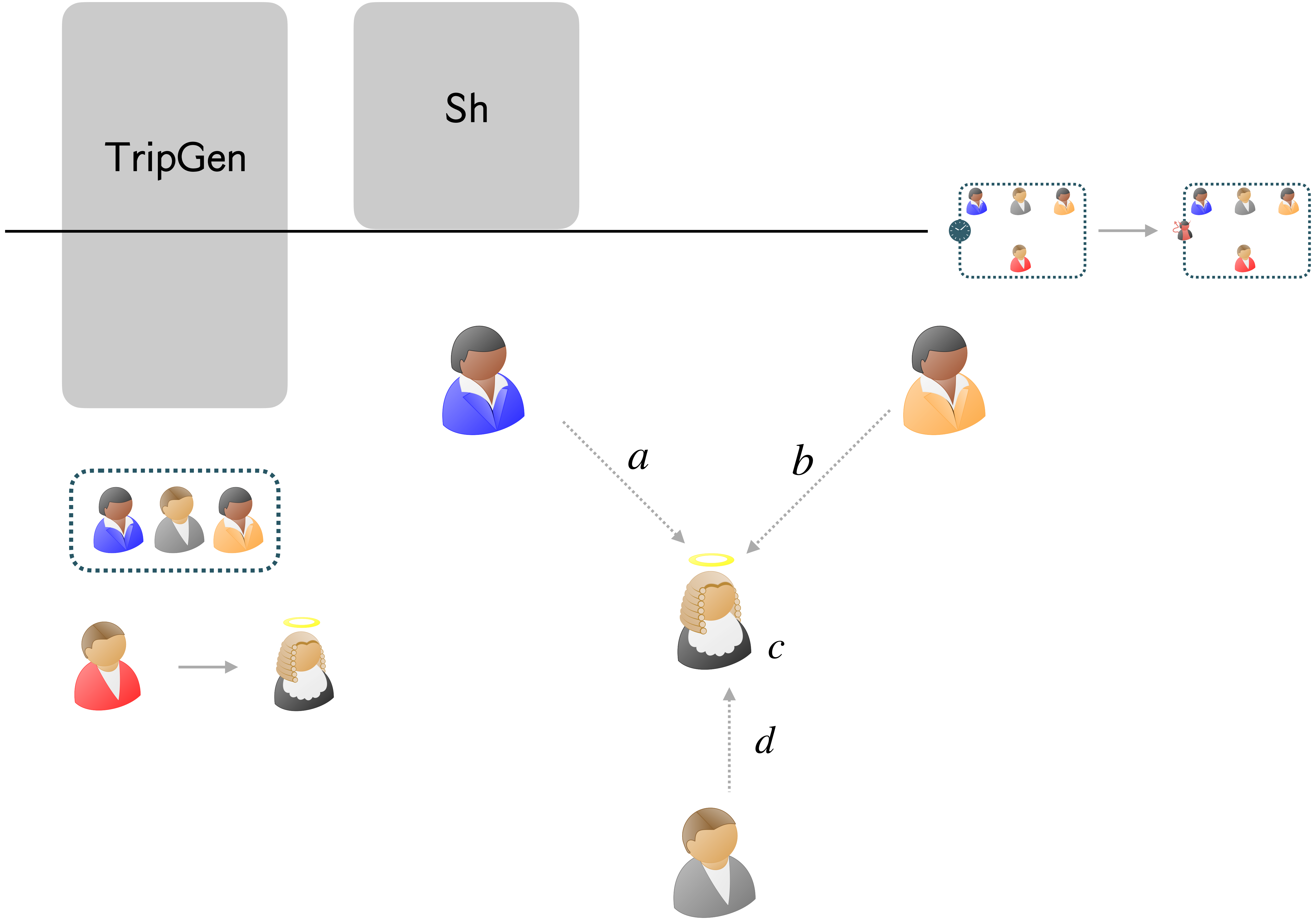
Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase



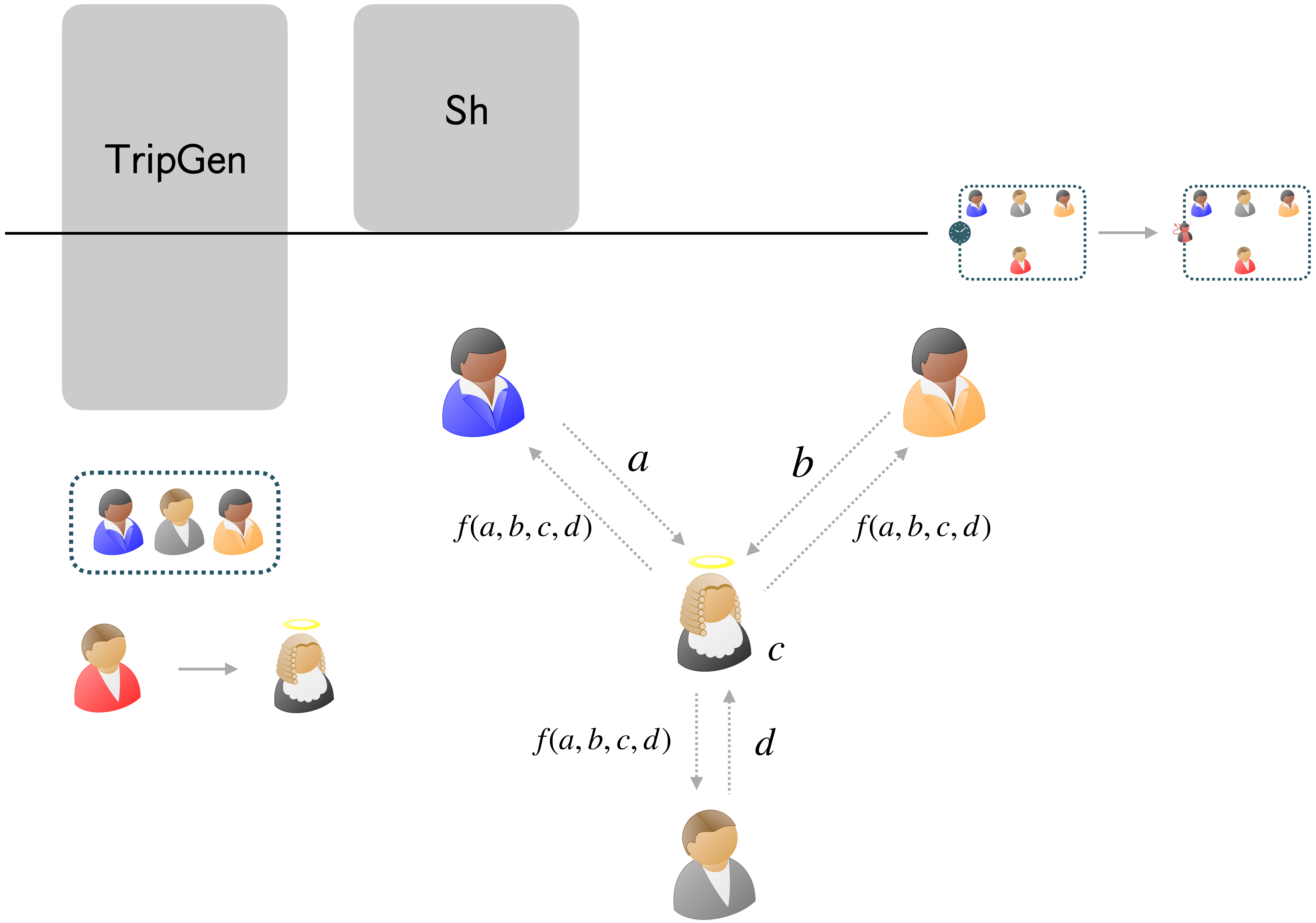
Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase



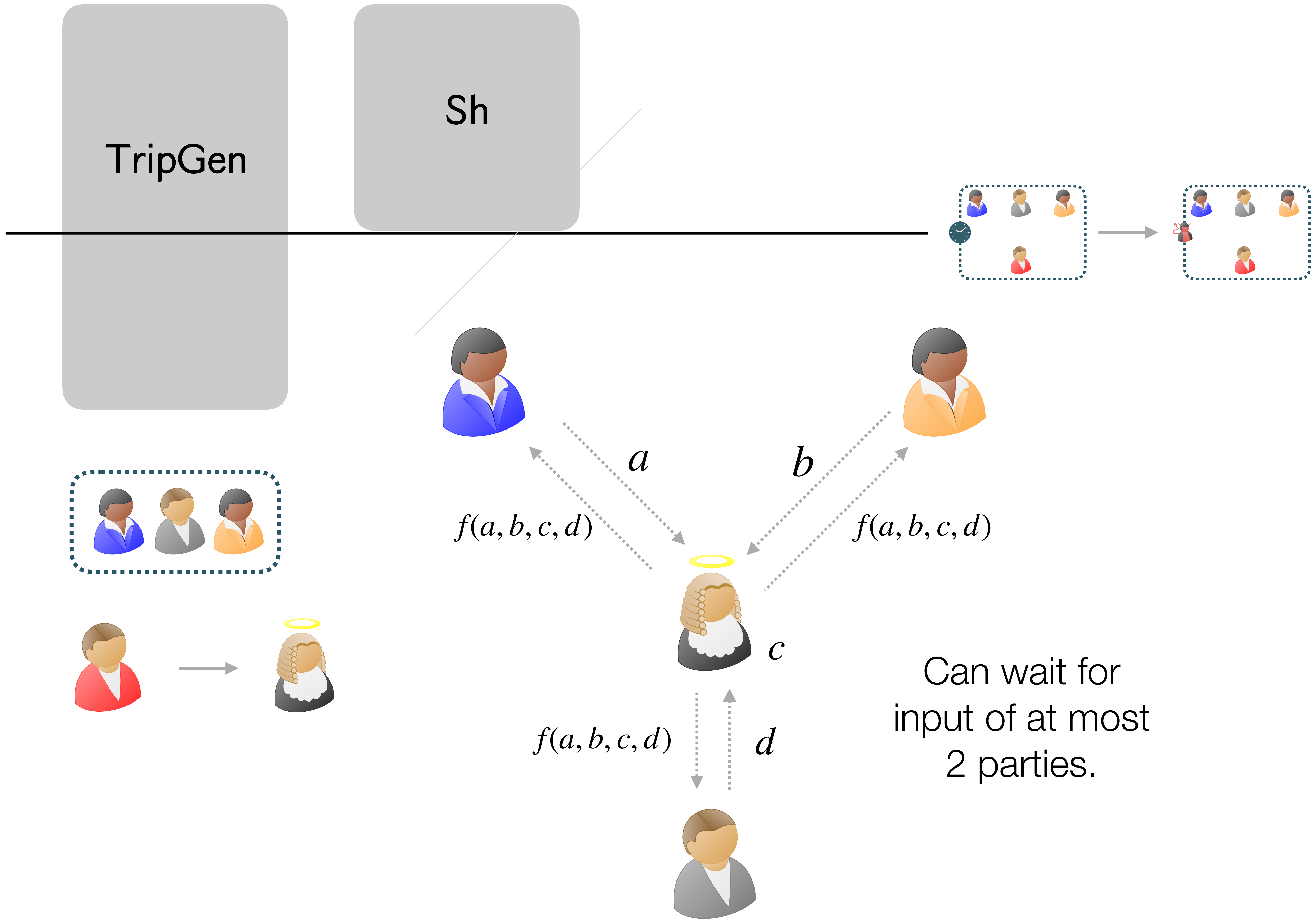
Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase



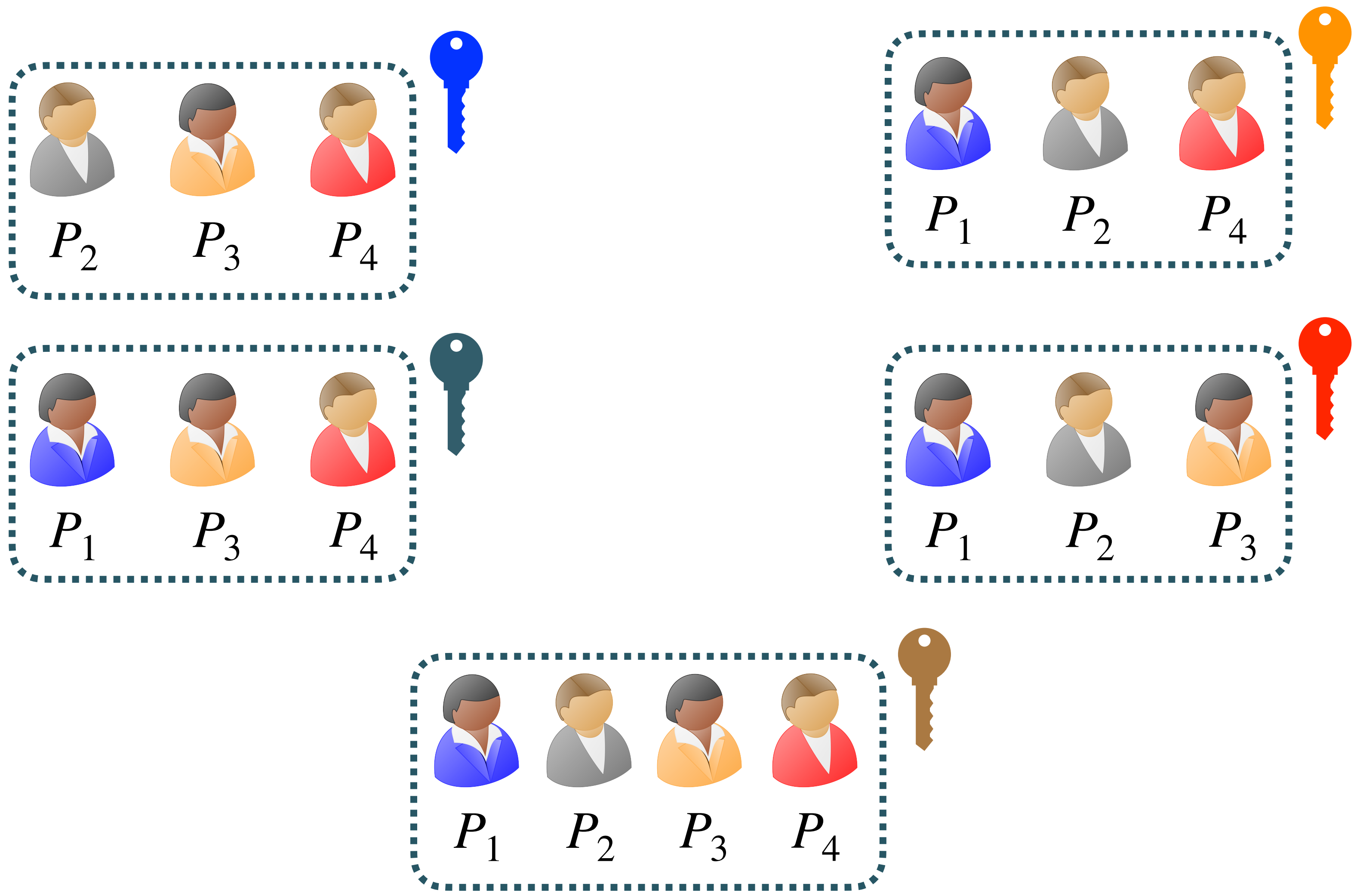
Perfect HMPC

- 3 phases
  - Triple generation phase
  - Input phase
  - Circuit evaluation and output phase



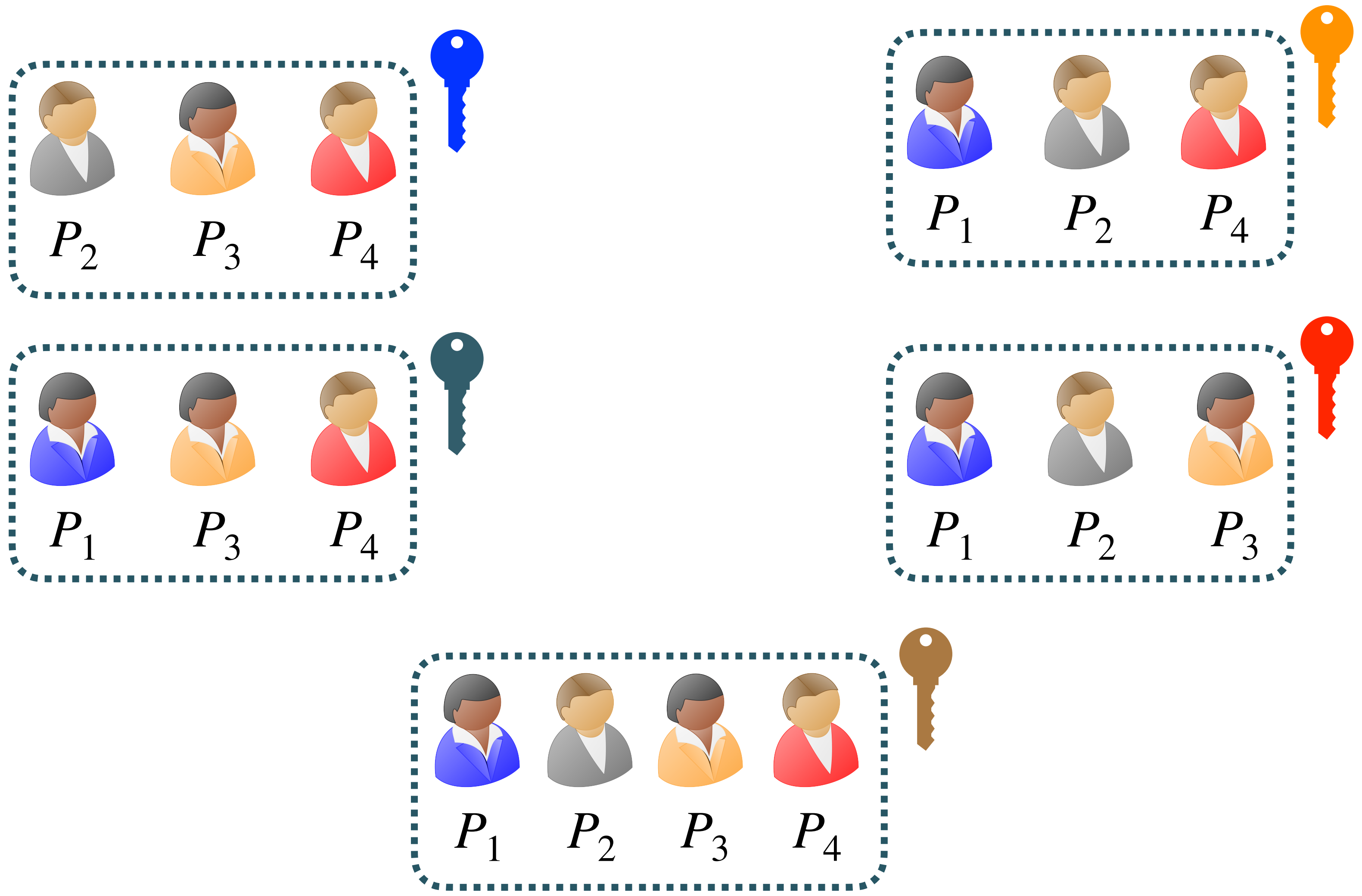
Can wait for input of at most 2 parties.

Cryptographically Secure HMPC - Secret Sharing and Reconstruction



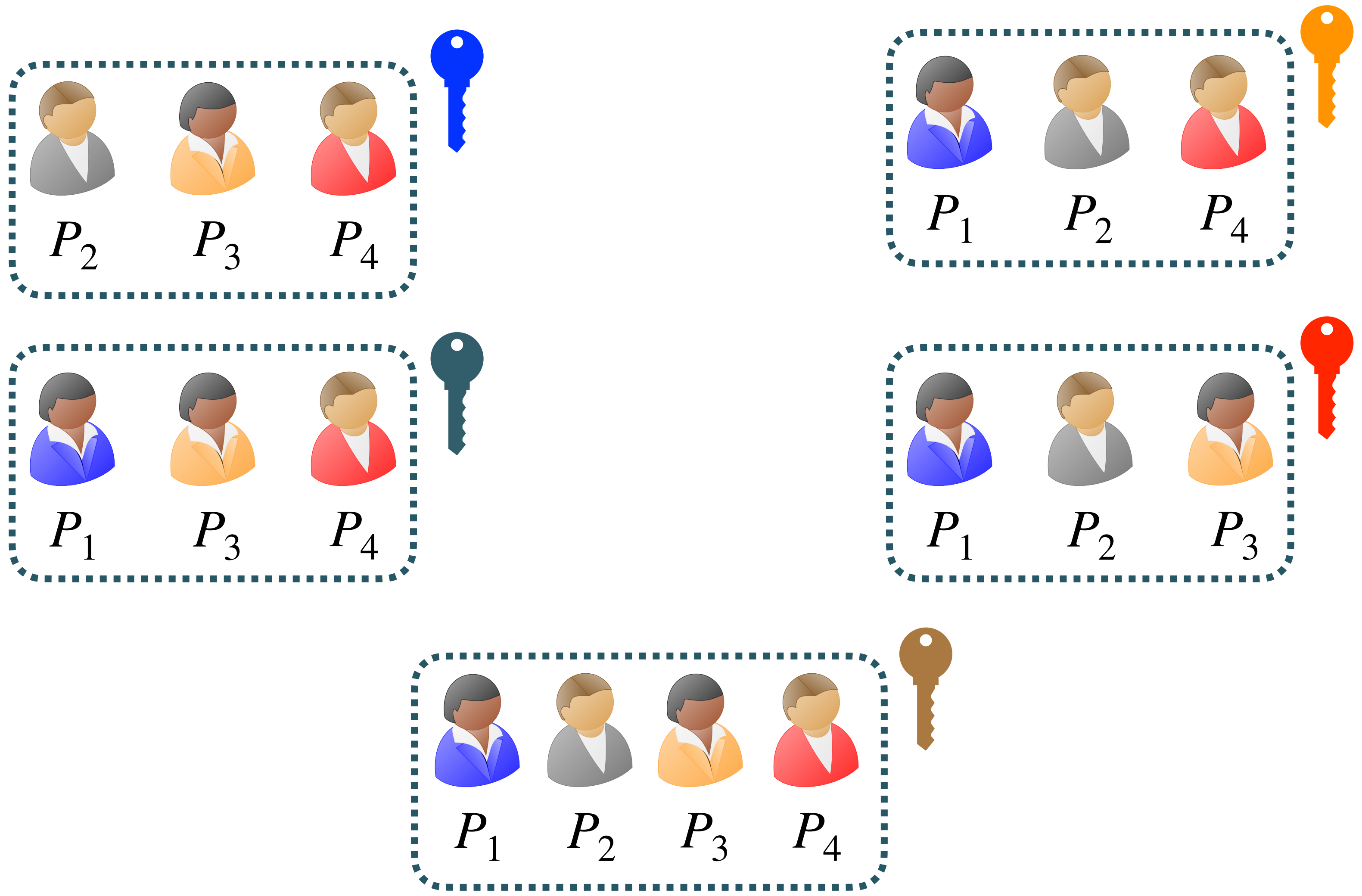
- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]

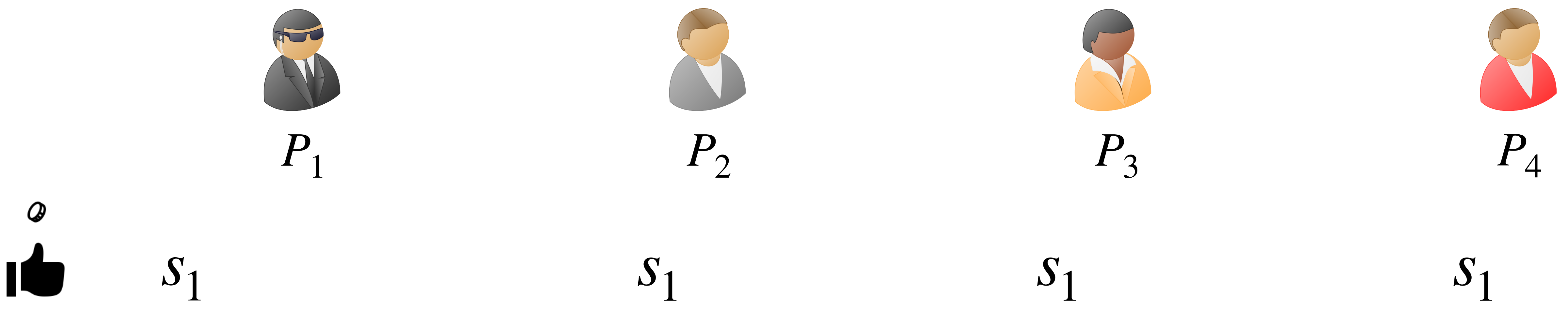
Cryptographically Secure HMPC - Secret Sharing and Reconstruction



- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

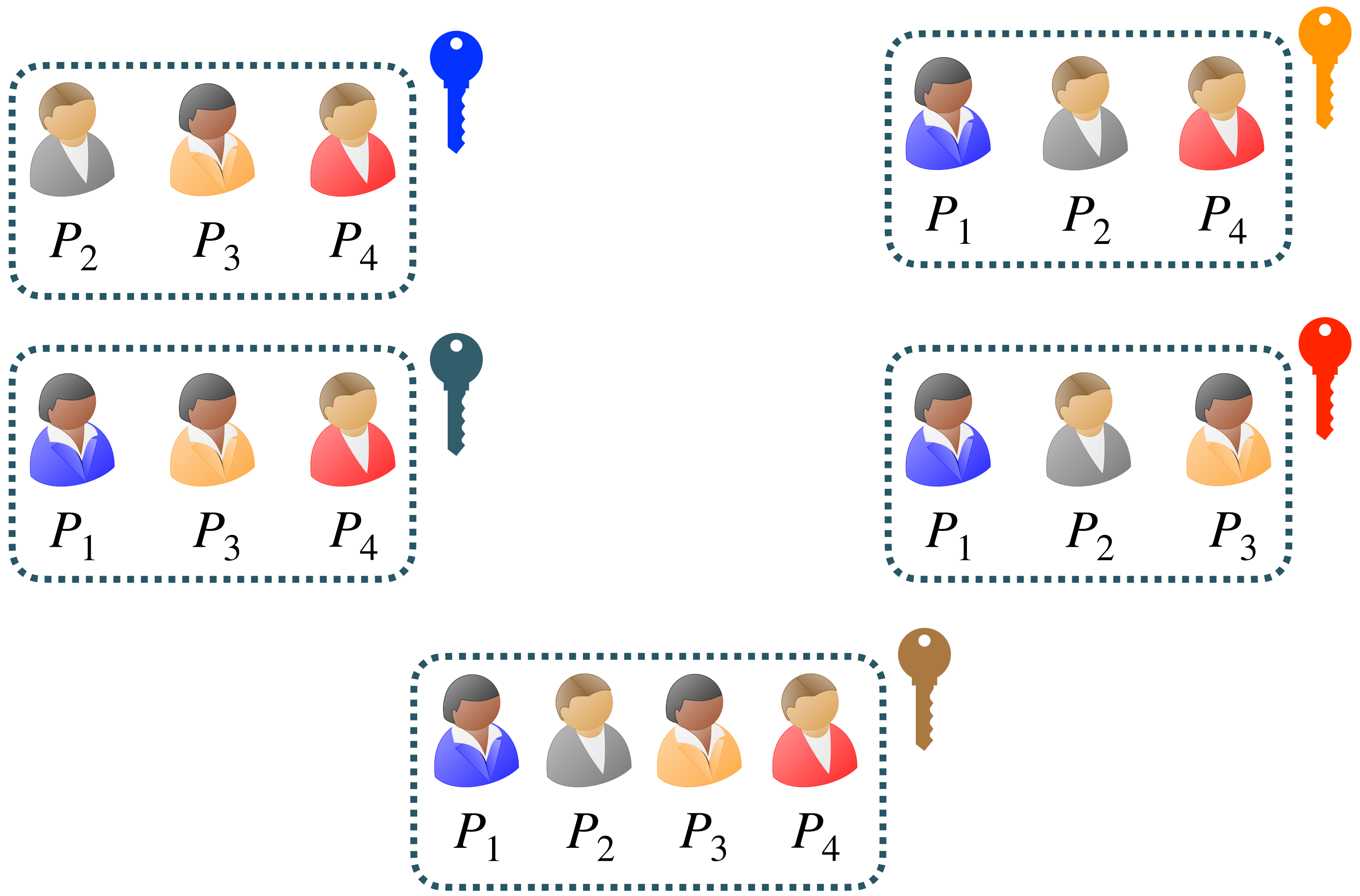
Cryptographically Secure HMPC - Secret Sharing and Reconstruction

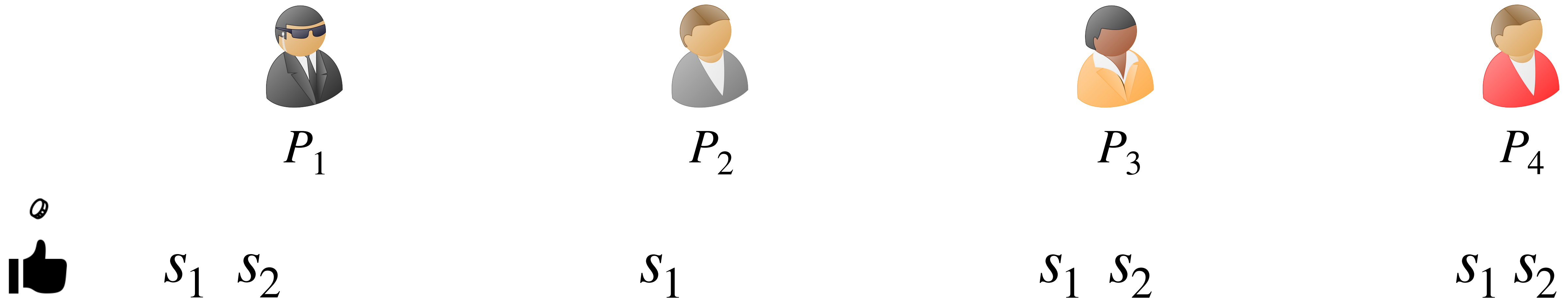




- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

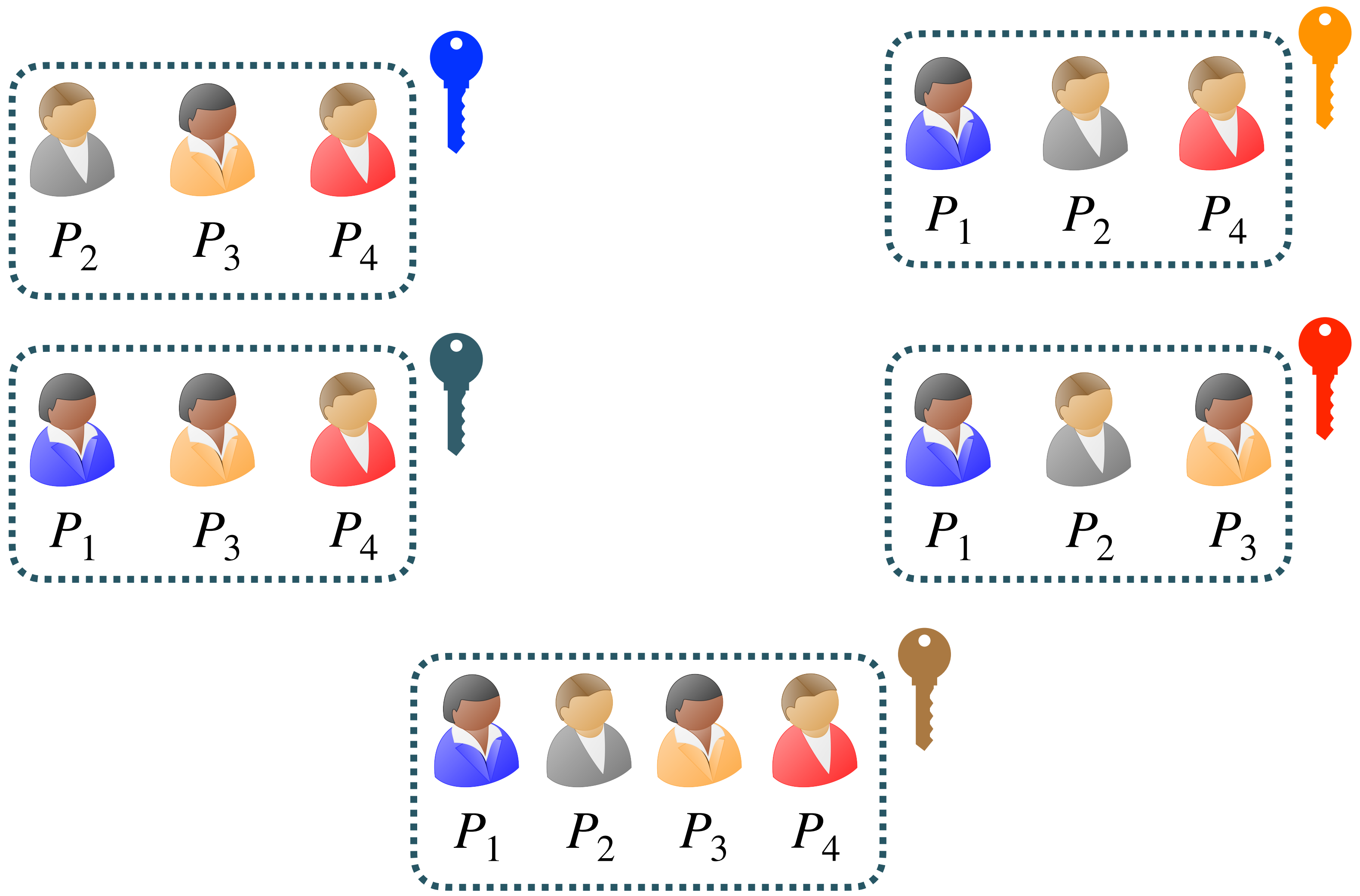
Cryptographically Secure HMPC - Secret Sharing and Reconstruction

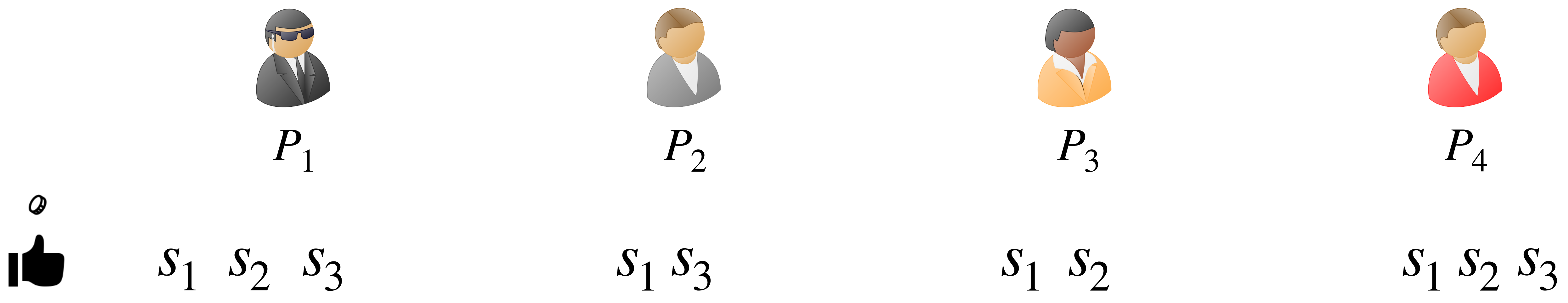




- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

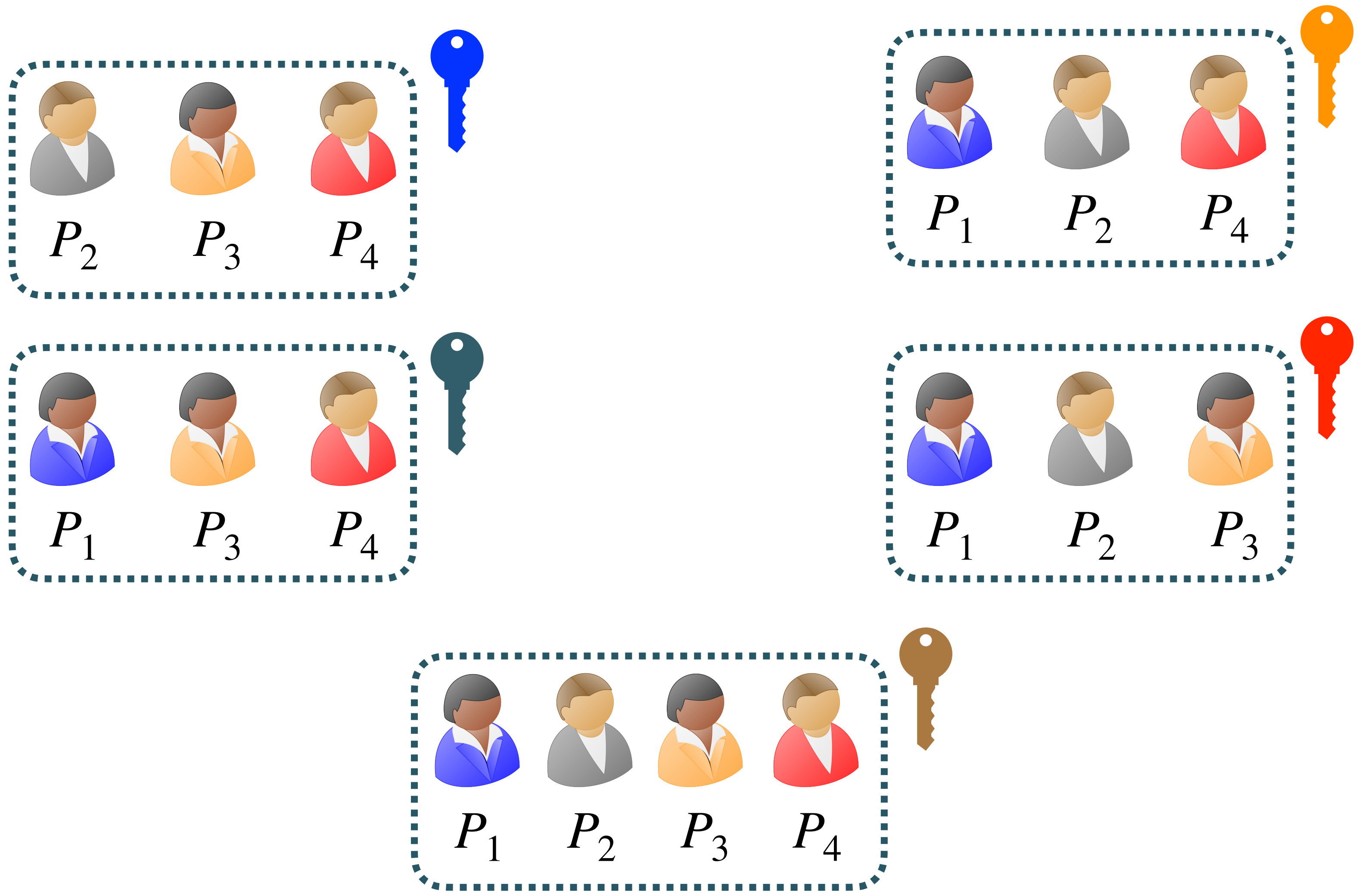
Cryptographically Secure HMPC - Secret Sharing and Reconstruction

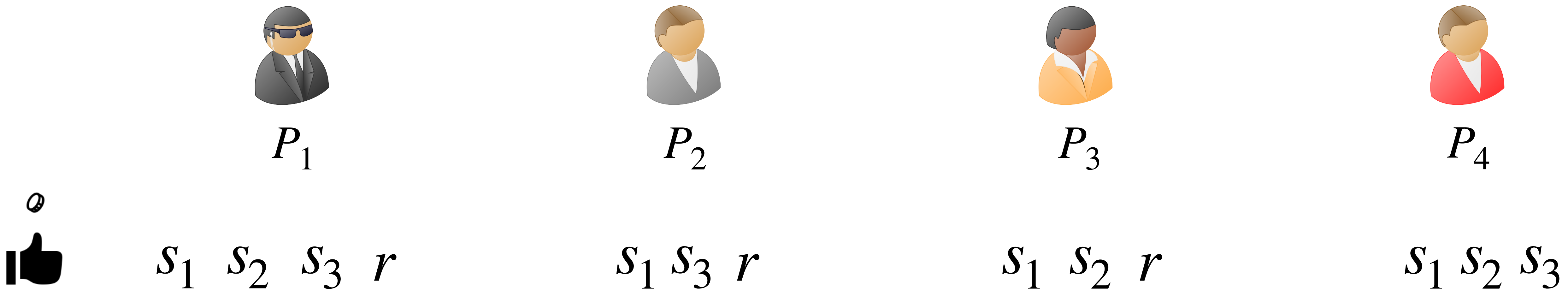




- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

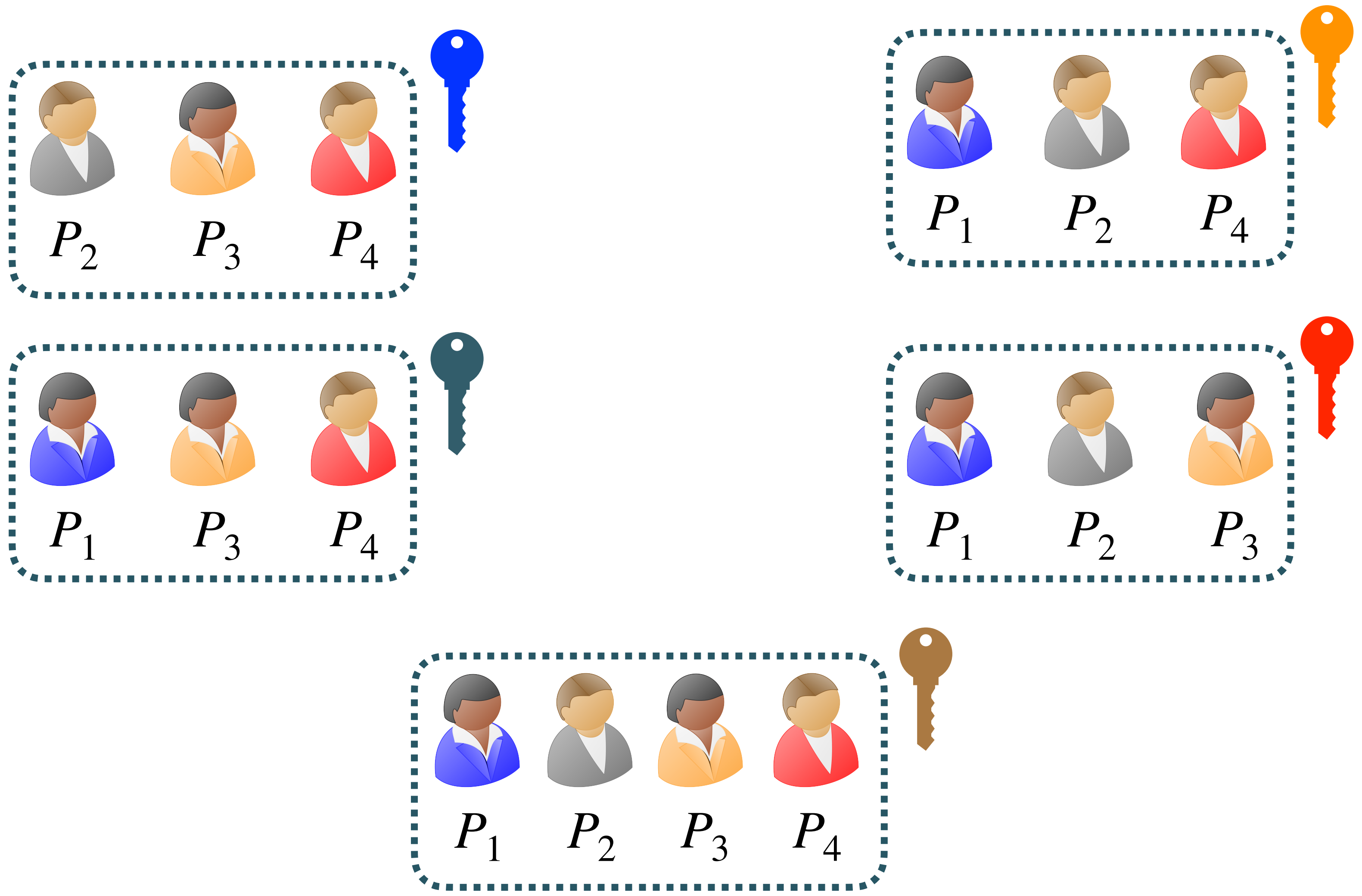
Cryptographically Secure HMPC - Secret Sharing and Reconstruction

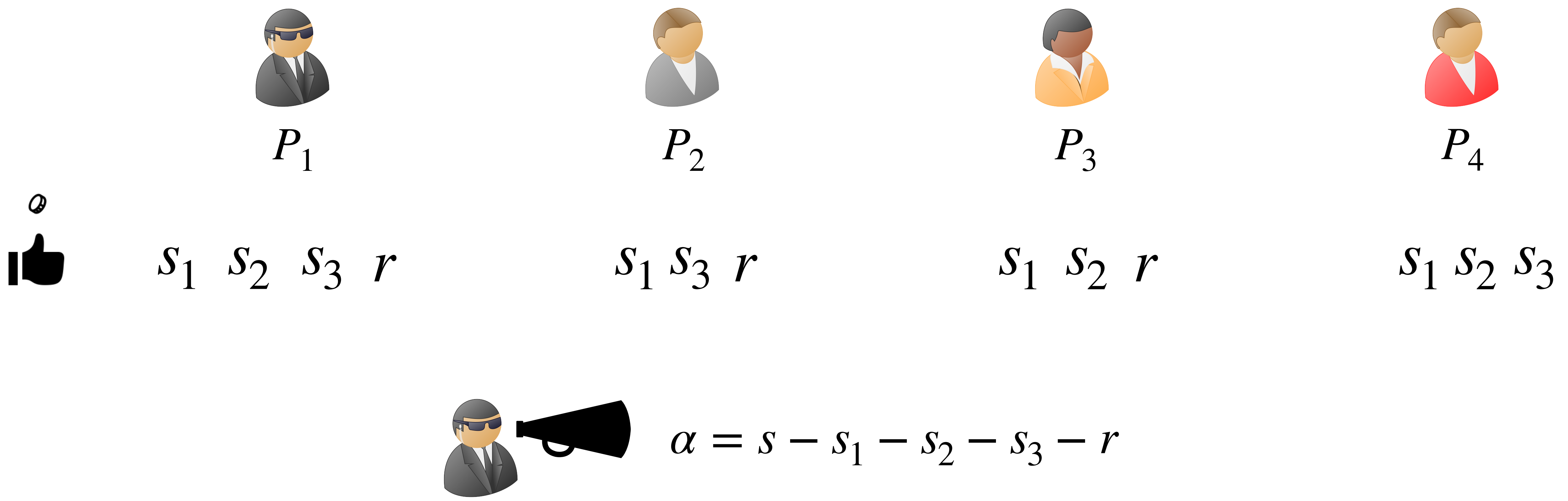




- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

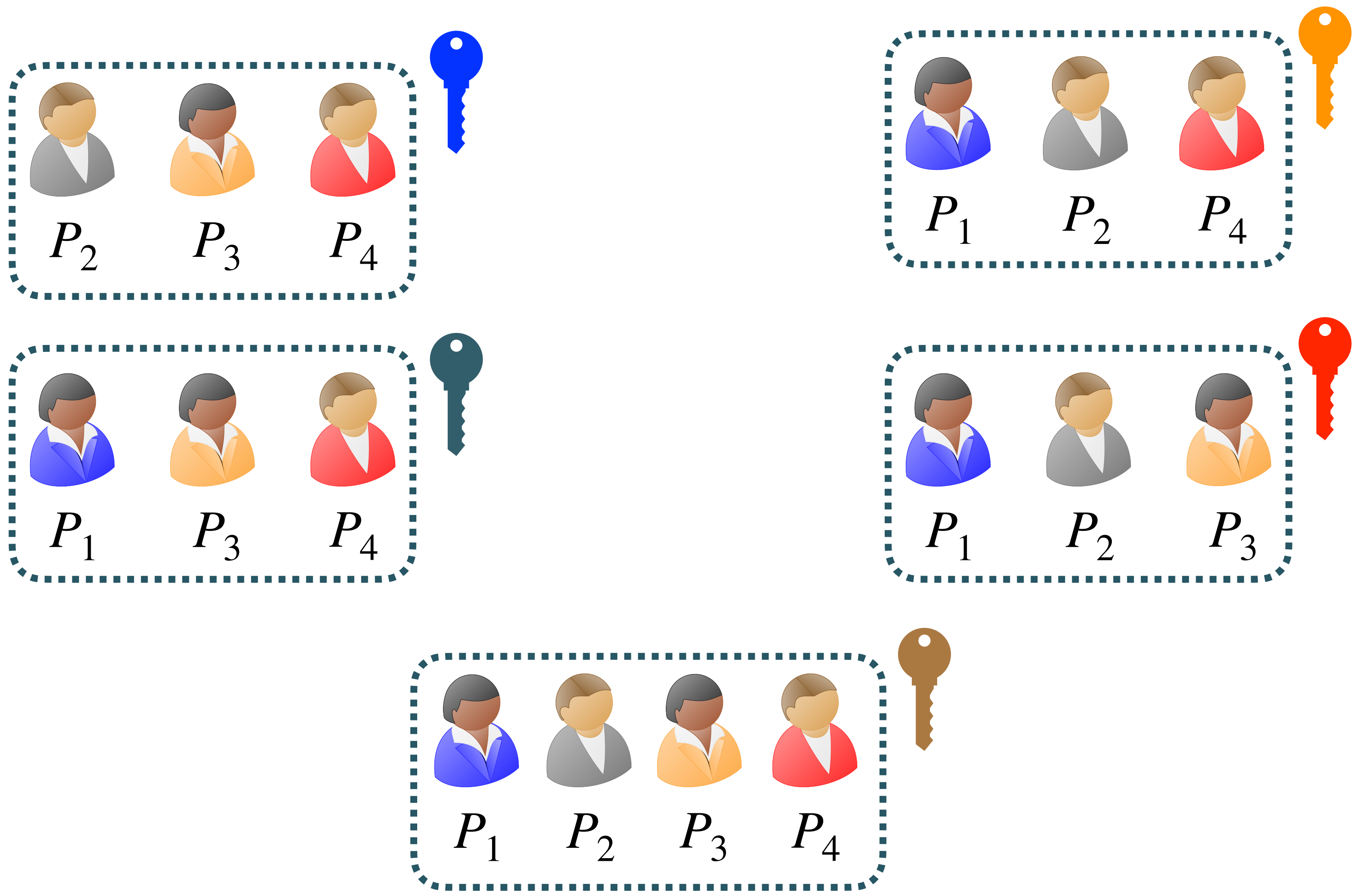
Cryptographically Secure HMPC - Secret Sharing and Reconstruction

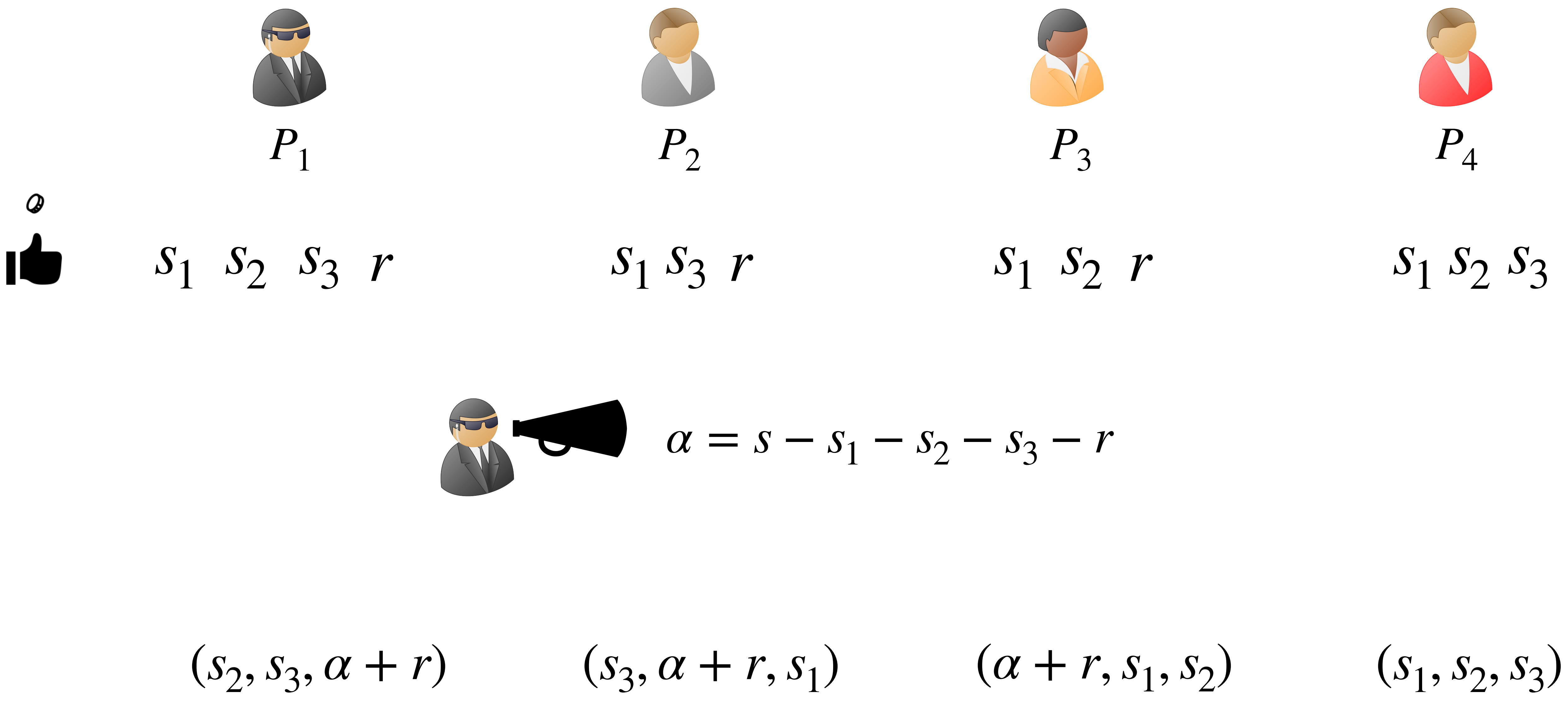




- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
    - One synchronous round VSS protocol

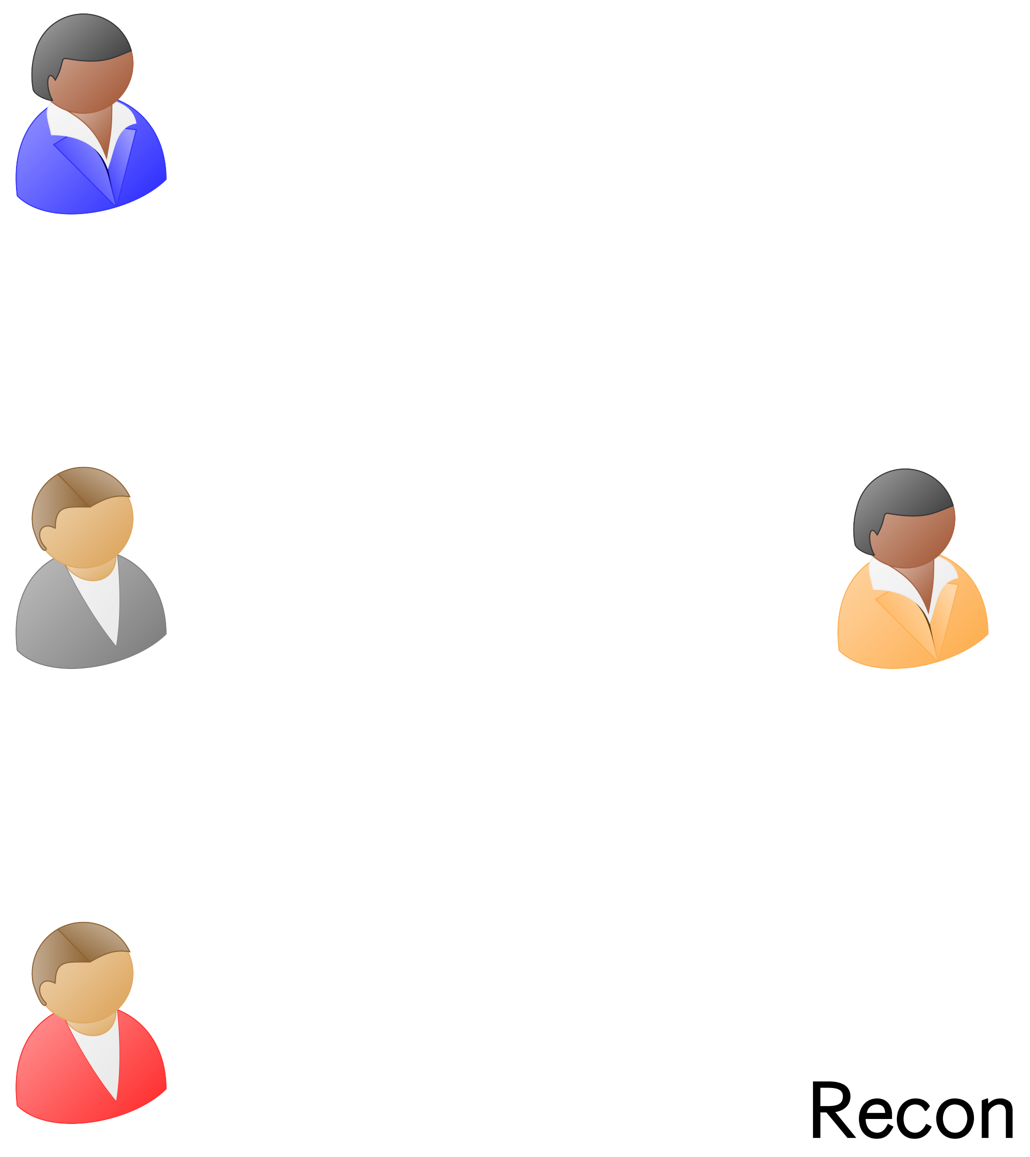
Cryptographically Secure HMPC - Secret Sharing and Reconstruction





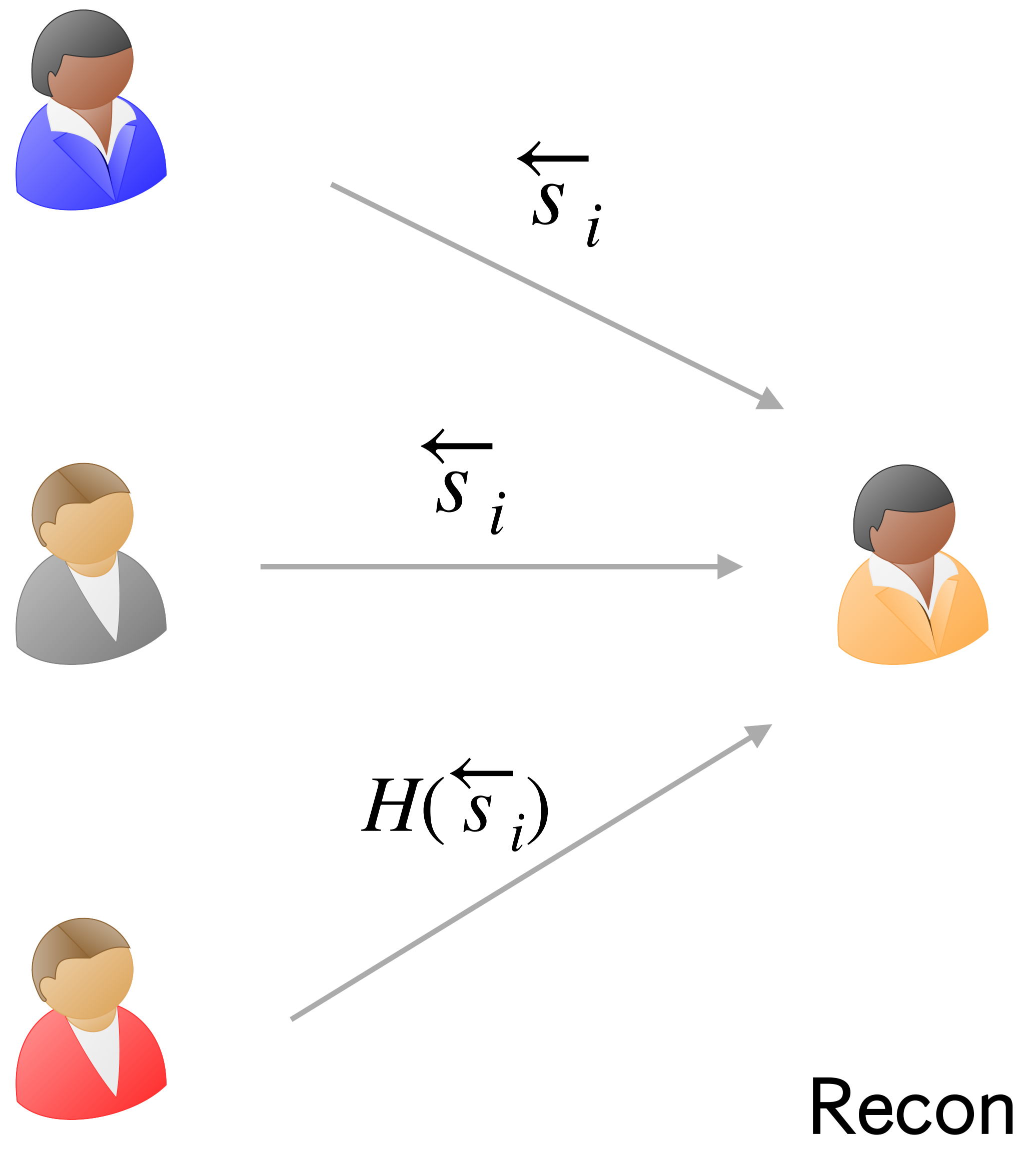
- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol

Cryptographically Secure HMPC - Secret Sharing and Reconstruction



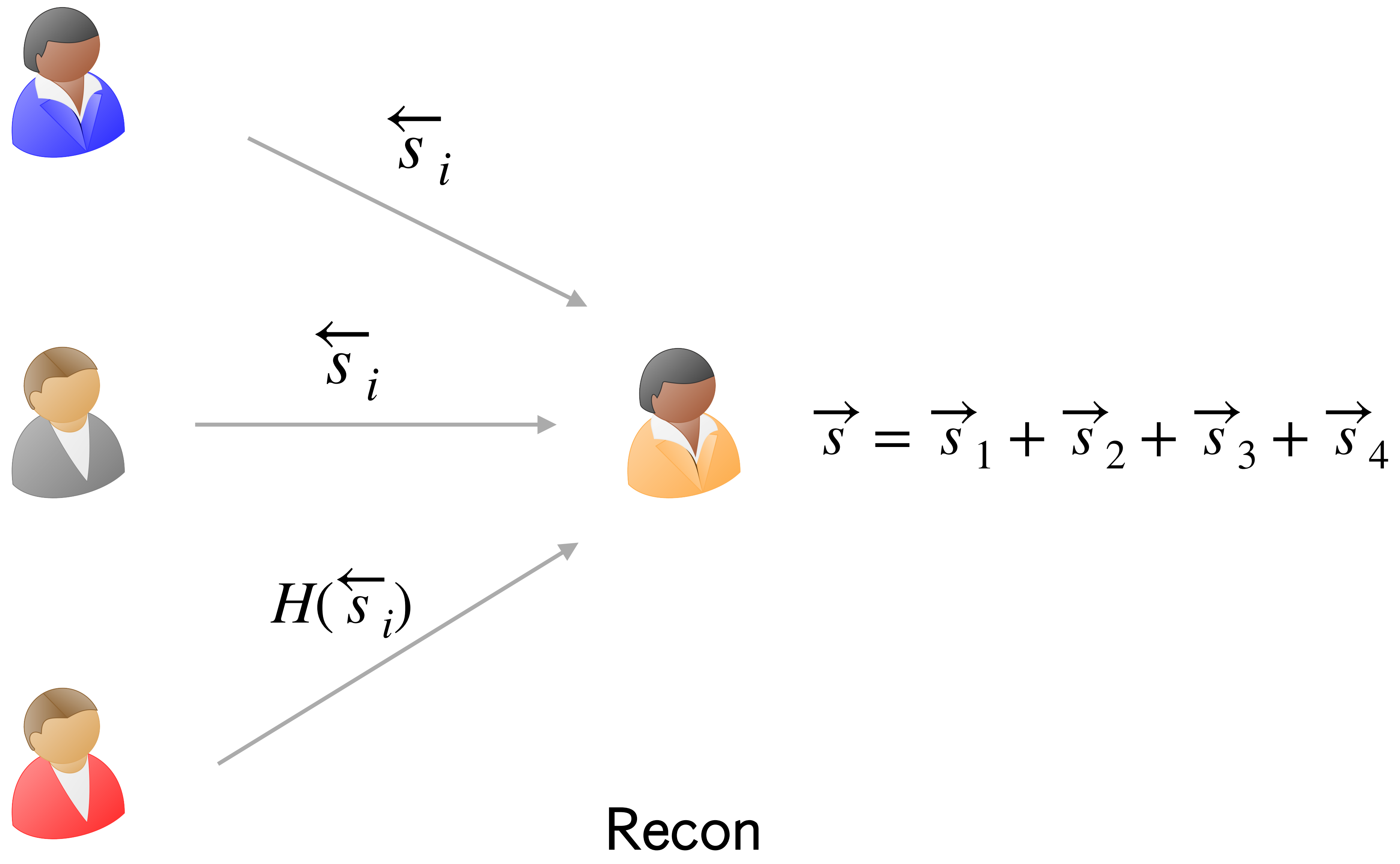
- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
    - One synchronous round VSS protocol
- Efficient reconstruction protocol

Cryptographically Secure HMPC - Secret Sharing and Reconstruction



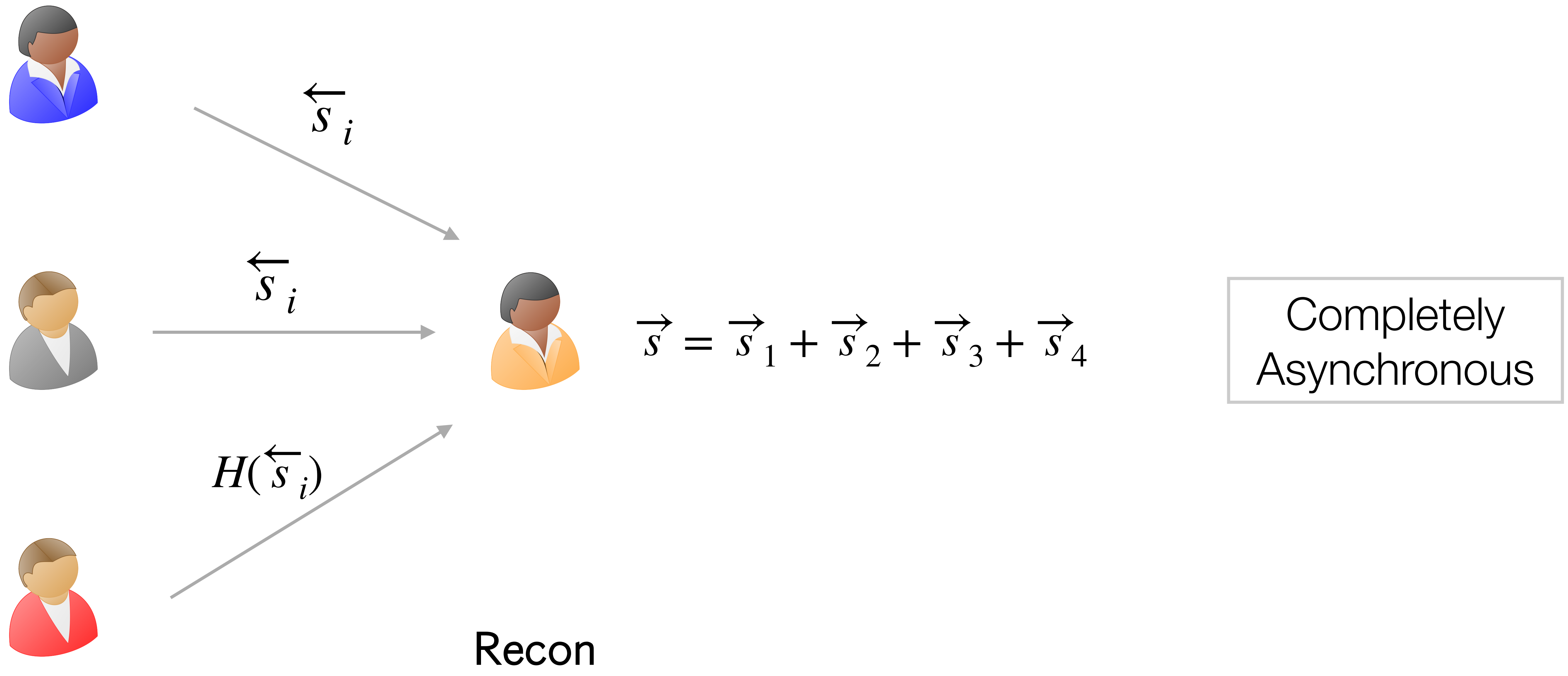
- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol
- Efficient reconstruction protocol

Cryptographically Secure HMPC - Secret Sharing and Reconstruction



- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
    - One synchronous round VSS protocol
- Efficient reconstruction protocol

Cryptographically Secure HMPC - Secret Sharing and Reconstruction



$\overleftarrow{s}_i$

$\overleftarrow{s}_i$

$H(\overleftarrow{s}_i)$

$\vec{s} = \vec{s}_1 + \vec{s}_2 + \vec{s}_3 + \vec{s}_4$

Completely Asynchronous

Recon

- Assume symmetric-key setup for PRF [AFL+16,CCP+19,MR18]
  - One synchronous round VSS protocol
- Efficient reconstruction protocol

Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to TripSh
    - Dealer shares $2l + 1$ triples instead of 3 triples
    - Other parties don’t share triples

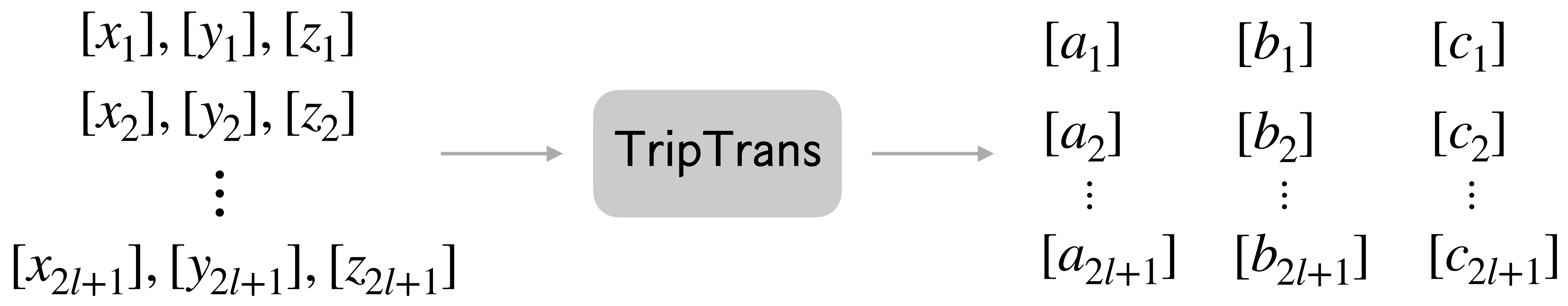
Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to TripSh
  - Dealer shares $2l + 1$ triples instead of 3 triples
  - Other parties don't share triples

$$[x_1], [y_1], [z_1]$$
$$[x_2], [y_2], [z_2]$$
$$\vdots$$
$$[x_{2l+1}], [y_{2l+1}], [z_{2l+1}]$$

Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to TripSh
  - Dealer shares $2l+1$ triples instead of 3 triples
  - Other parties don't share triples



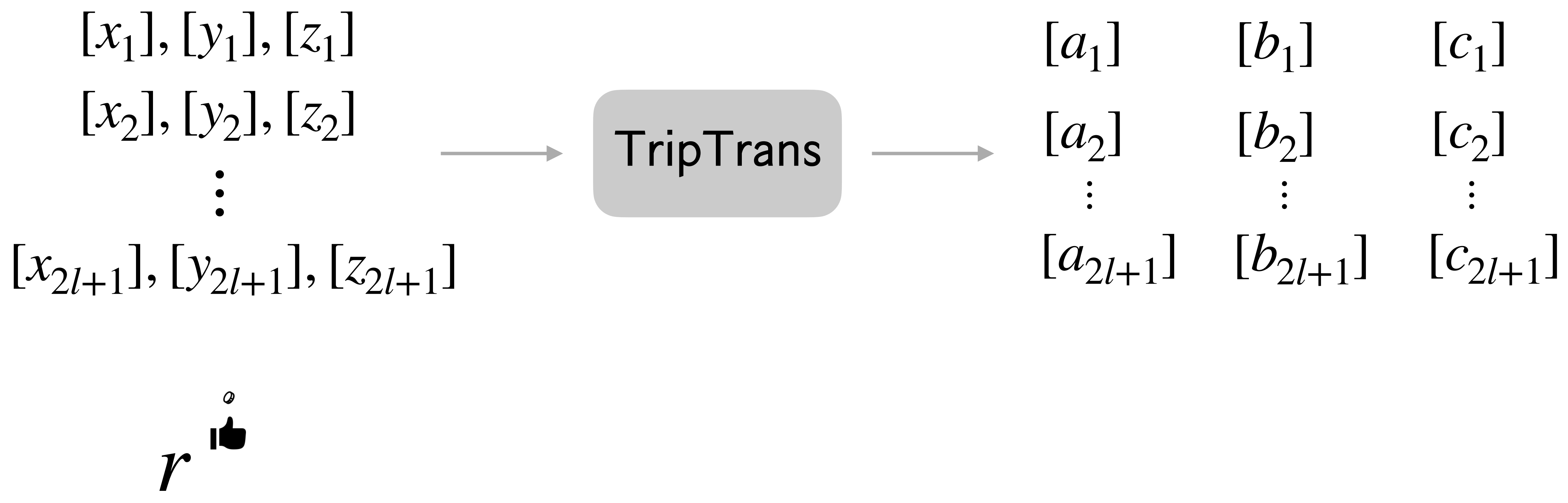
$[x_1], [y_1], [z_1]$
$[x_2], [y_2], [z_2]$
$\vdots$
$[x_{2l+1}], [y_{2l+1}], [z_{2l+1}]$

TripTrans

$[a_1]$ $[b_1]$ $[c_1]$
$[a_2]$ $[b_2]$ $[c_2]$
$\vdots$ $\vdots$ $\vdots$
$[a_{2l+1}]$ $[b_{2l+1}]$ $[c_{2l+1}]$

Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to TripSh
  - Dealer shares $2l+1$ triples instead of 3 triples
  - Other parties don't share triples



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2l+1}], [y_{2l+1}], [z_{2l+1}]$

$\longrightarrow$ TripTrans $\longrightarrow$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

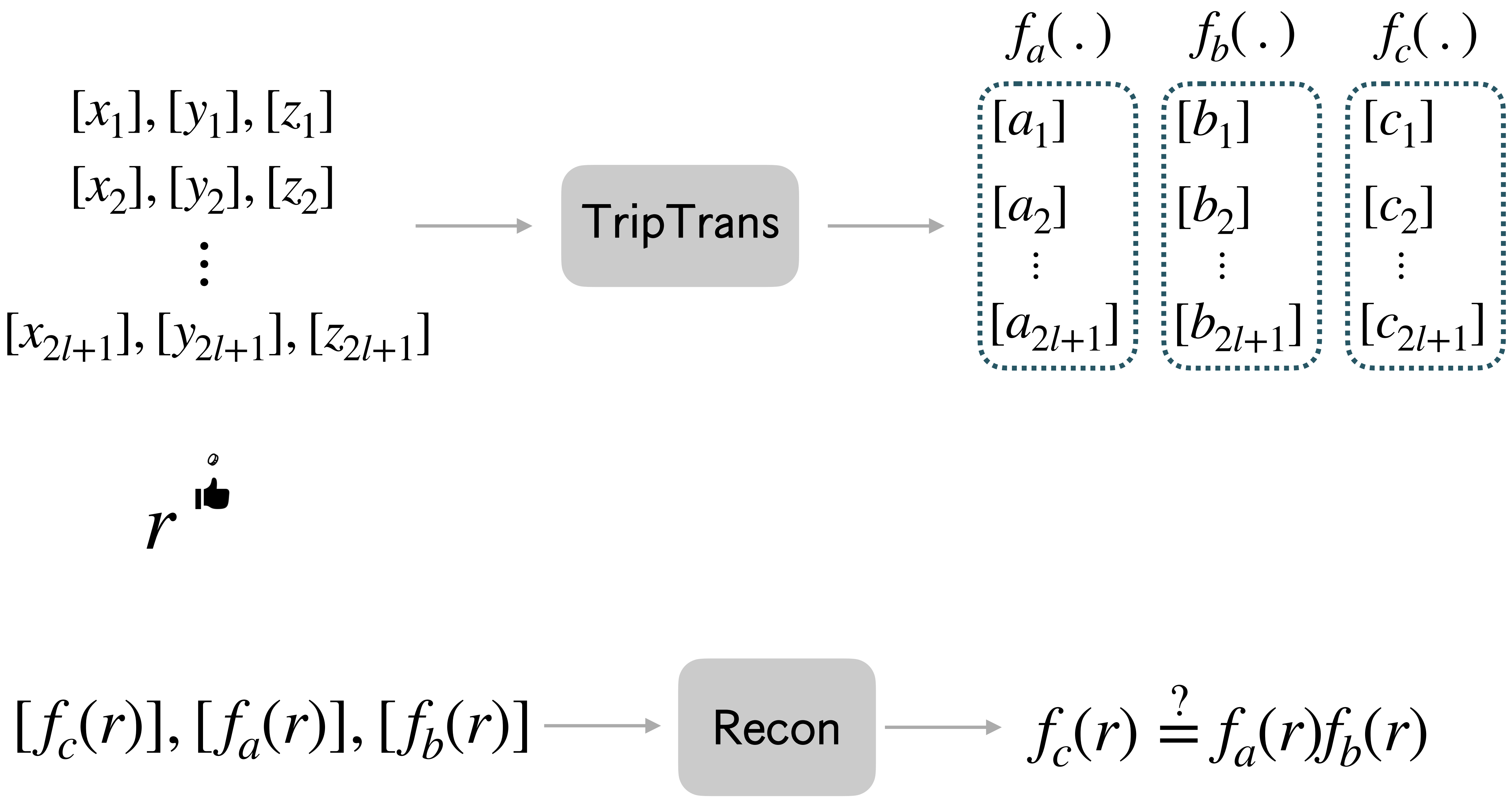
$\vdots$ $\vdots$ $\vdots$

$[a_{2l+1}]$ $[b_{2l+1}]$ $[c_{2l+1}]$

r 👍

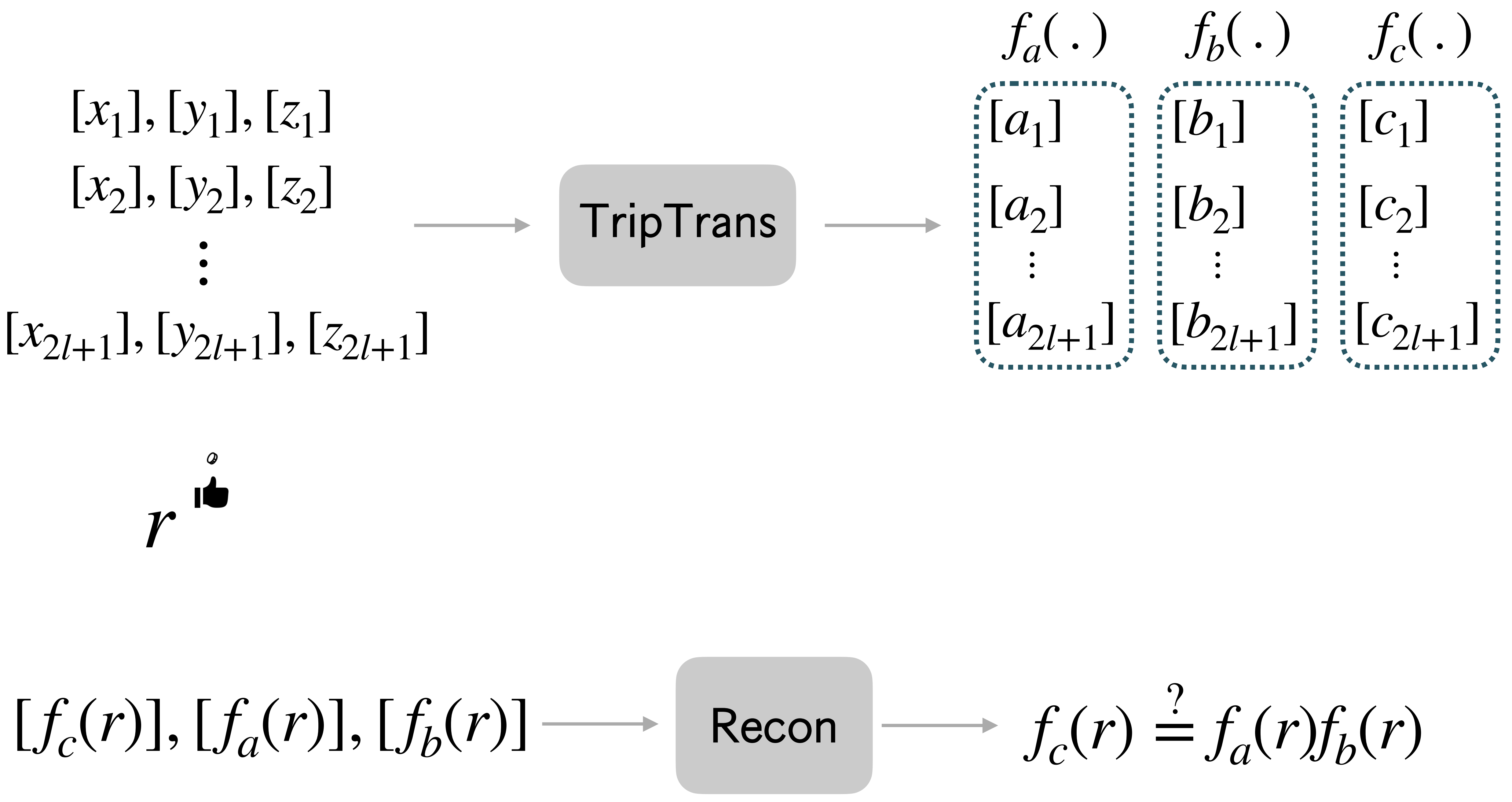
Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to **TripSh**
  - Dealer shares $2l+1$ triples instead of 3 triples
  - Other parties don't share triples



Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to **TripSh**
  - Dealer shares $2l + 1$ triples instead of 3 triples
  - Other parties don't share triples



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2l+1}], [y_{2l+1}], [z_{2l+1}]$ → TripTrans →

$f_a(\,.\,)$: $[a_1]$, $[a_2]$, $\vdots$, $[a_{2l+1}]$

$f_b(\,.\,)$: $[b_1]$, $[b_2]$, $\vdots$, $[b_{2l+1}]$

$f_c(\,.\,)$: $[c_1]$, $[c_2]$, $\vdots$, $[c_{2l+1}]$

r 👍

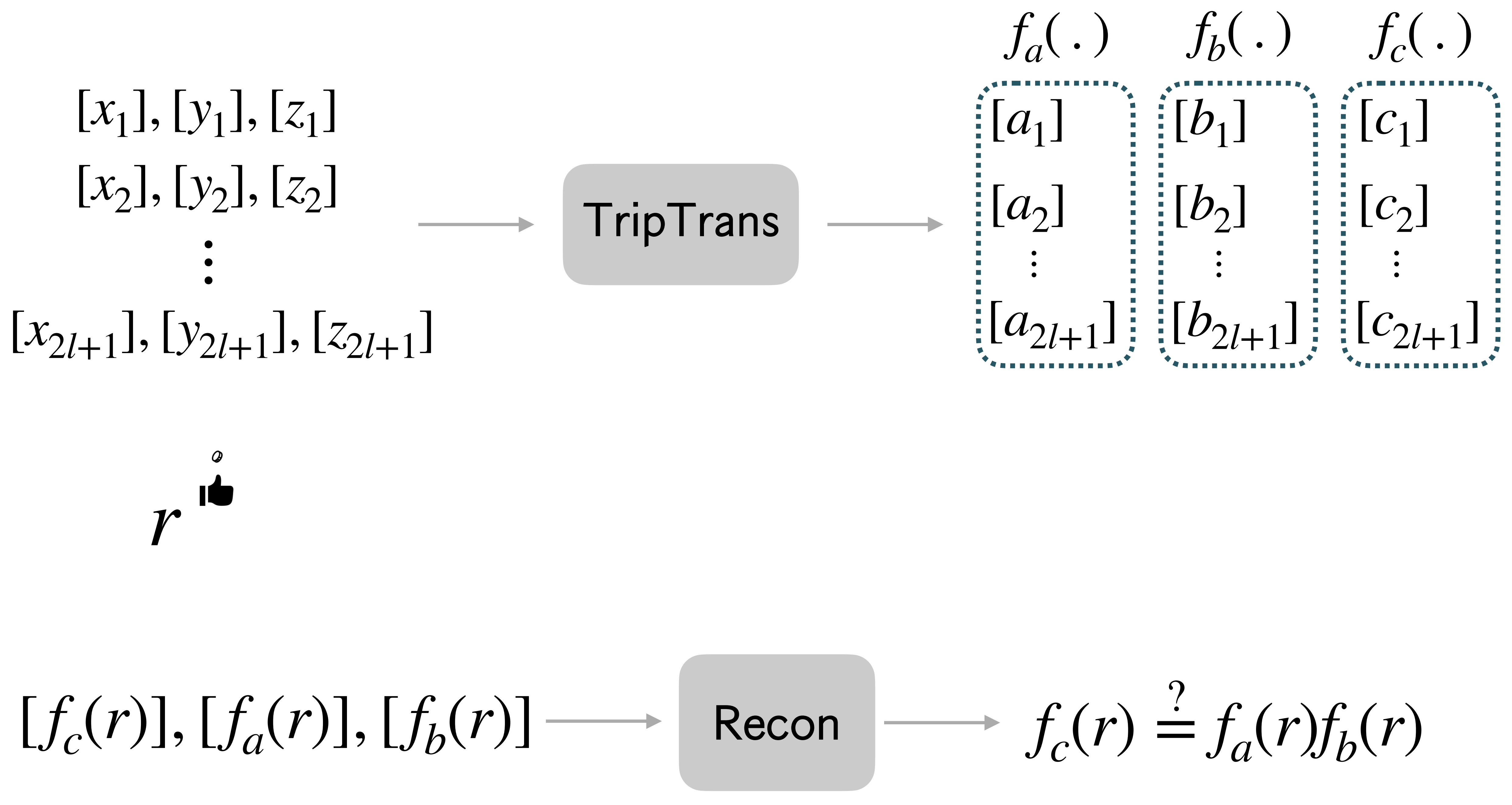
$[f_c(r)], [f_a(r)], [f_b(r)]$ → Recon → $f_c(r) \overset{?}{=} f_a(r) f_b(r)$

✓ l multiplication triples

✗ l shares of $([0], [0], [0])$

Cryptographically Secure HMPC - Triple Generation Protocol

- Triple sharing similar to **TripSh**
  - Dealer shares $2l + 1$ triples instead of 3 triples
  - Other parties don't share triples
- Triple generation similar to **TripGen**
  - Each instance outputs l triples



$[x_1], [y_1], [z_1]$

$[x_2], [y_2], [z_2]$

$\vdots$

$[x_{2l+1}], [y_{2l+1}], [z_{2l+1}]$

TripTrans

$f_a(.)$ $f_b(.)$ $f_c(.)$

$[a_1]$ $[b_1]$ $[c_1]$

$[a_2]$ $[b_2]$ $[c_2]$

$\vdots$

$[a_{2l+1}]$ $[b_{2l+1}]$ $[c_{2l+1}]$

r

$[f_c(r)], [f_a(r)], [f_b(r)]$ → Recon → $f_c(r) \overset{?}{=} f_a(r) f_b(r)$

✓ l multiplication triples

✗ l shares of $([0], [0], [0])$

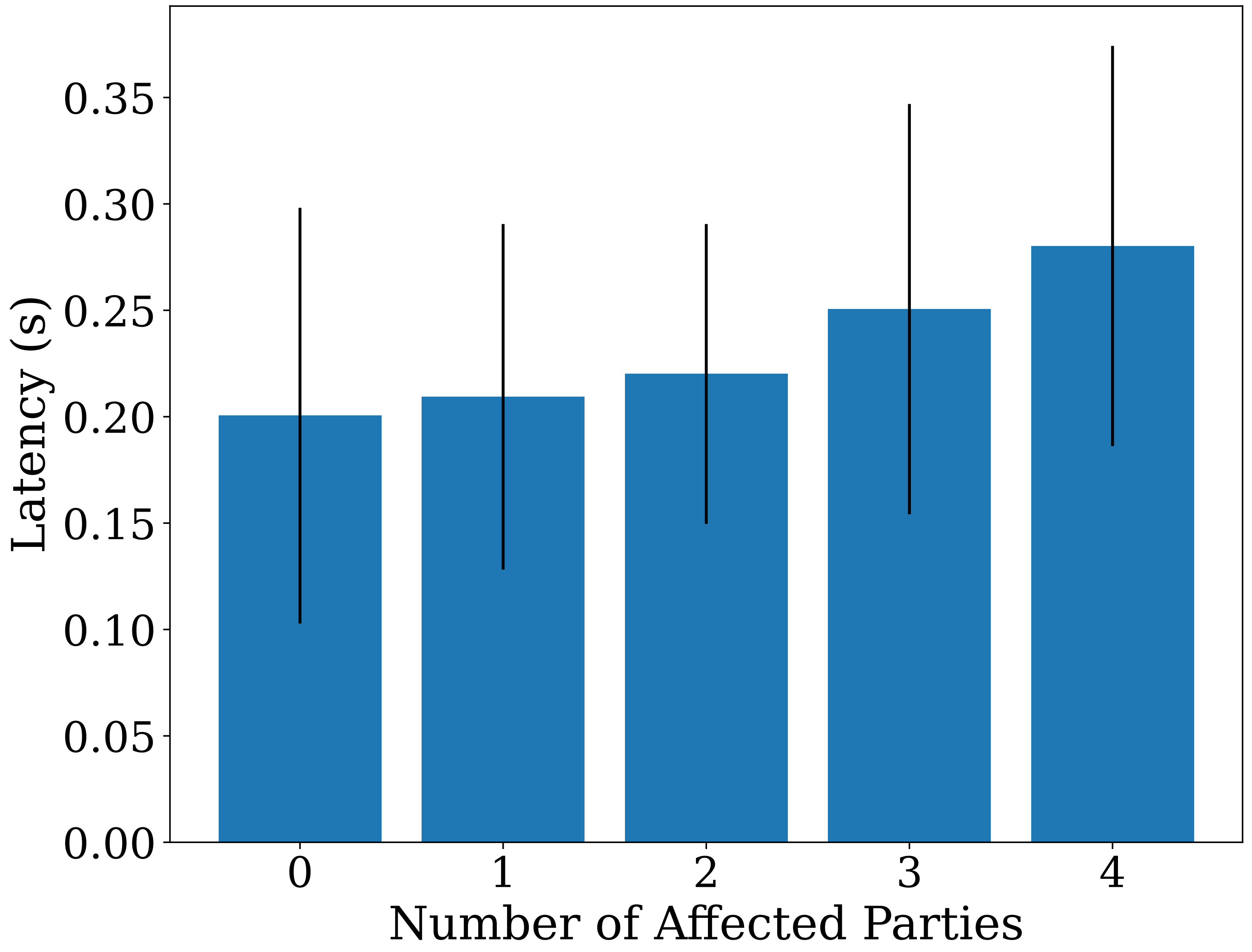
Cryptographically Secure HMPC and AMPC

- Cryptographically secure HMPC
  - Triple generation phase and input phase use 1 synchronous round
  - Circuit evaluation is completely asynchronous
  - Input provision

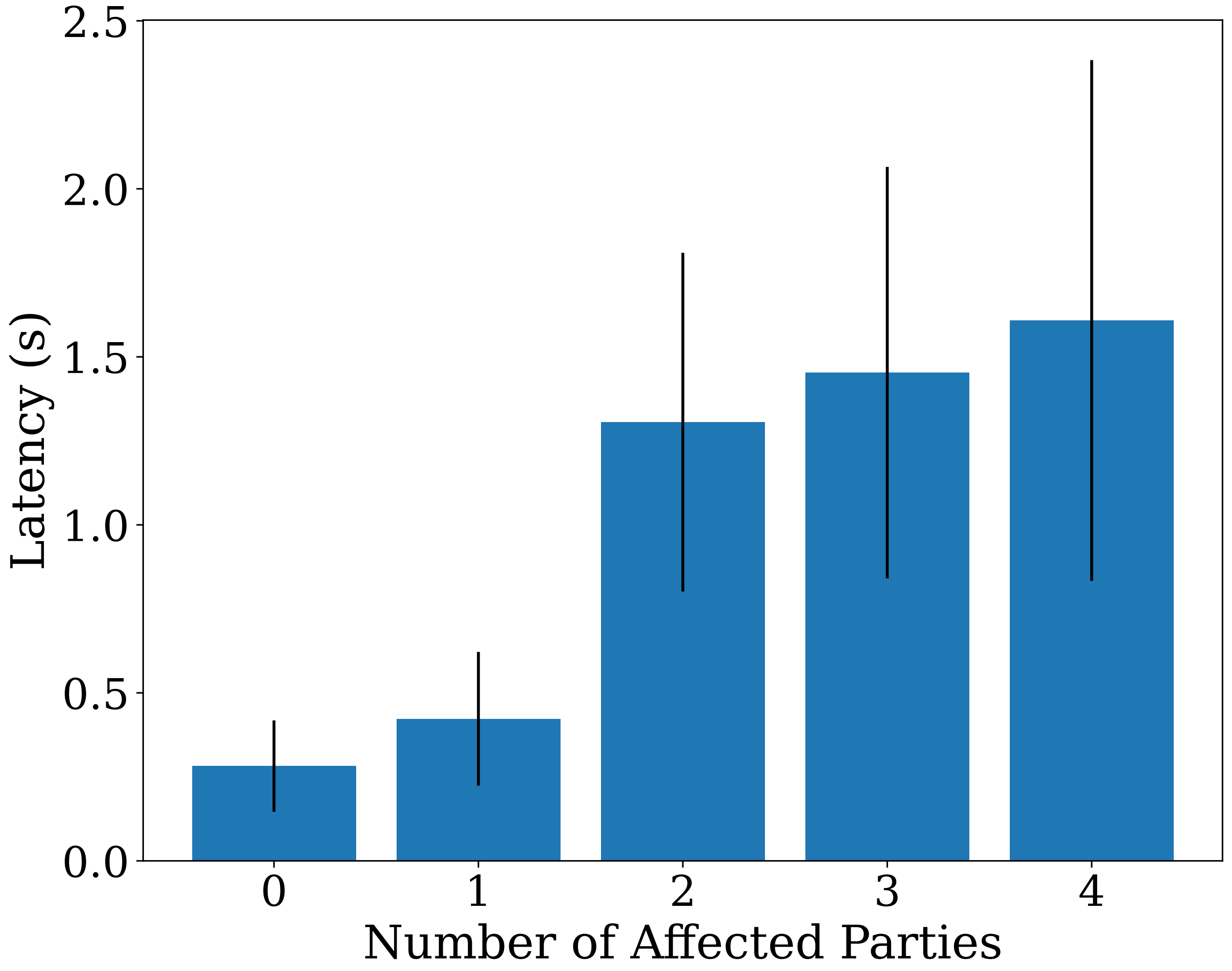
Cryptographically Secure HMPC and AMPC

- Cryptographically secure HMPC
  - Triple generation phase and input phase use 1 synchronous round
  - Circuit evaluation is completely asynchronous
  - Input provision
- Cryptographically secure AMPC
  - Similar to Cryptographically secure HMPC
  - No synchronous broadcast $\Longrightarrow$ ACast and ACS
  - No input provision

Conclusion

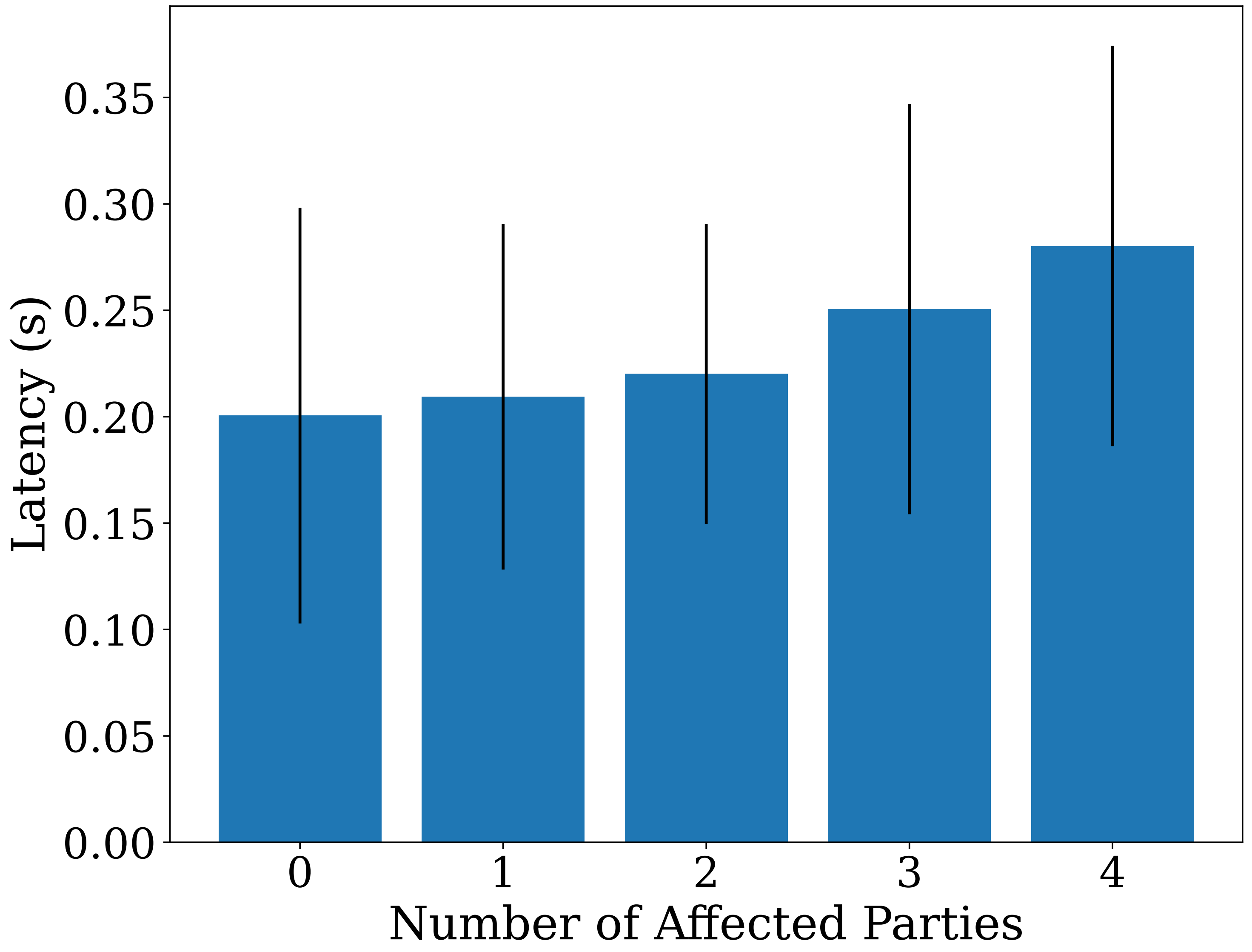


LAN

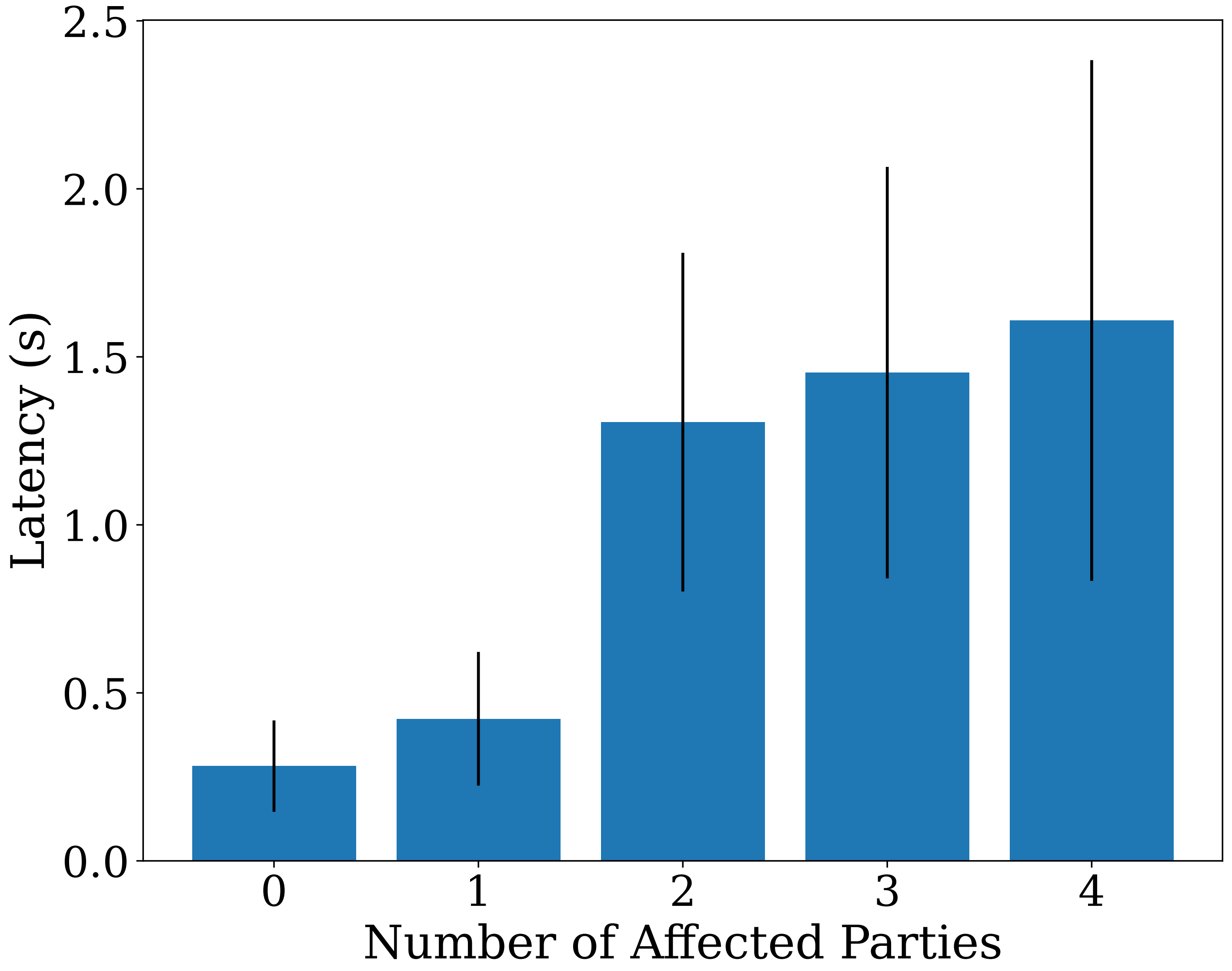


WAN

Conclusion



LAN



WAN

- Open problems
  - Perfect HMPC protocol for general case
  - Bridging the gap between synchronous and asynchronous MPC protocols

References

- [Beaver91a] - D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. CRYPTO 1991.
- [BHN10] - Z. Beerliová-Trubíniová, M. Hirt, and J. B. Nielsen. On the Theoretical Gap between Synchronous and Asynchronous MPC Protocols. PODC 2010.
- [CHP13] - A. Choudhury, M. Hirt, and A. Patra. Asynchronous multiparty computation with linear communication complexity. *DISC 2013*.
- [AFL+16] - T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. CCS 2016.
- [CP17] - A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. IEEE Trans. Information Theory 2017.
- [MR18] - P. Mohassel and P. Rindal. ABY3: A Mixed Protocol Framework for Machine Learning. CCS 2018.
- [PR18] - A. Patra and D. Ravi. On the Power of Hybrid Networks in Multi-Party Computation. IEEE Trans. Information Theory 2018.
- [CCP+19] - H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In CCSW@CCS 2019.

Thank You