Course Announcements

- Assignments
  - Homework 9 is due 4/8, no HW for the following 2 weeks
  - Class project has been posted (see Piazza post 302), due Wednesday 4/22
  - Reading: Secure Multiparty Computation for Privacy-Preserving Data Mining
- Notes
  - This course is moving to fall-only starting this fall, tell your friends!

Lecture 18: Protecting Data in Use against Mallory

1. MPC against Eve
2. MPC for Boolean circuits
3. MPC with preprocessing
4. MPC against Mallory

Objective of secure multi-party computation (MPC)

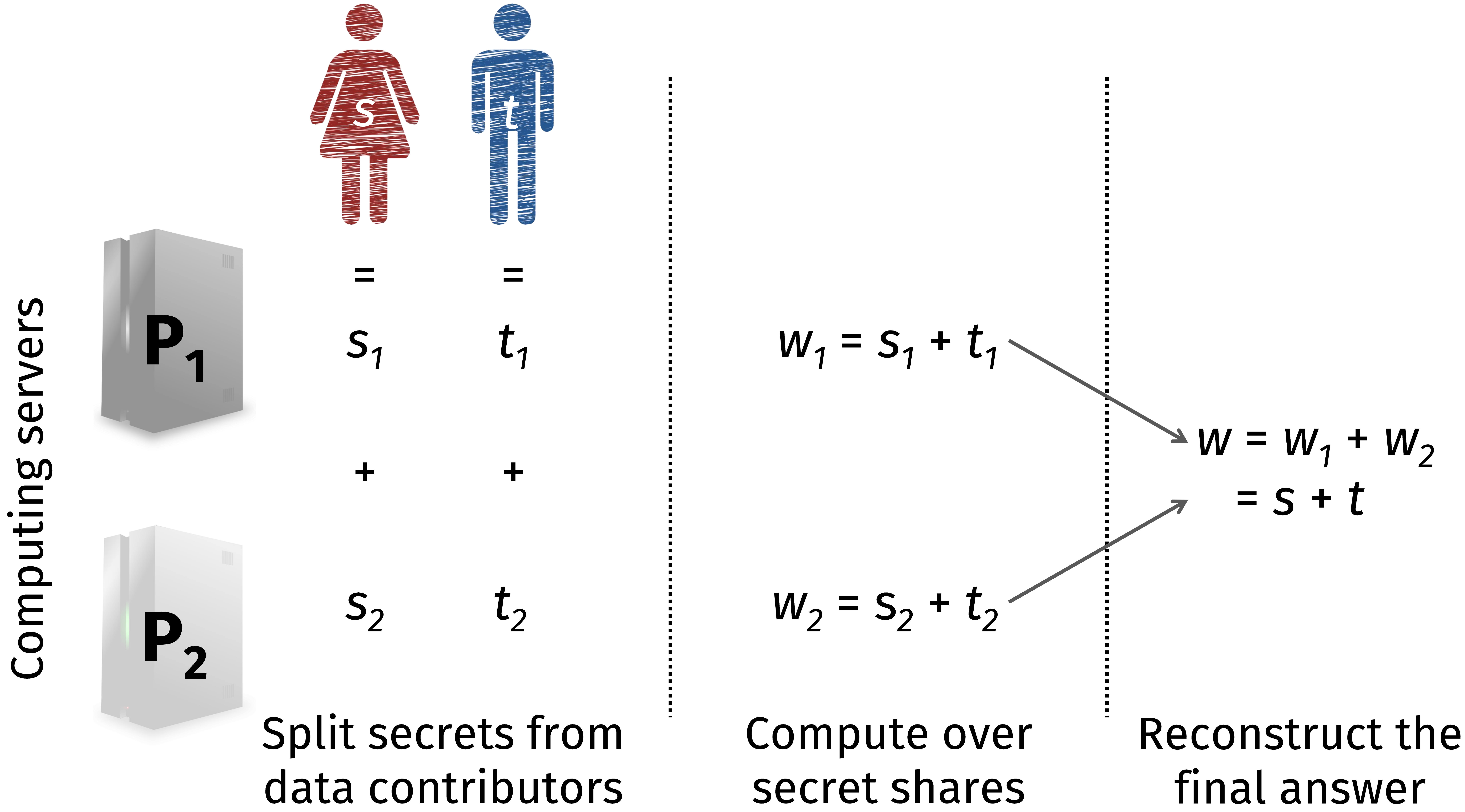
- Given multiple parties P_1, P_2, ..., P_n each with private data x_1, x_2, ..., x_n
- Parties engage in computing a publicly-known function *f*

$$\mathrm{y} = f(x_1, x_2, x_3, \dots)$$

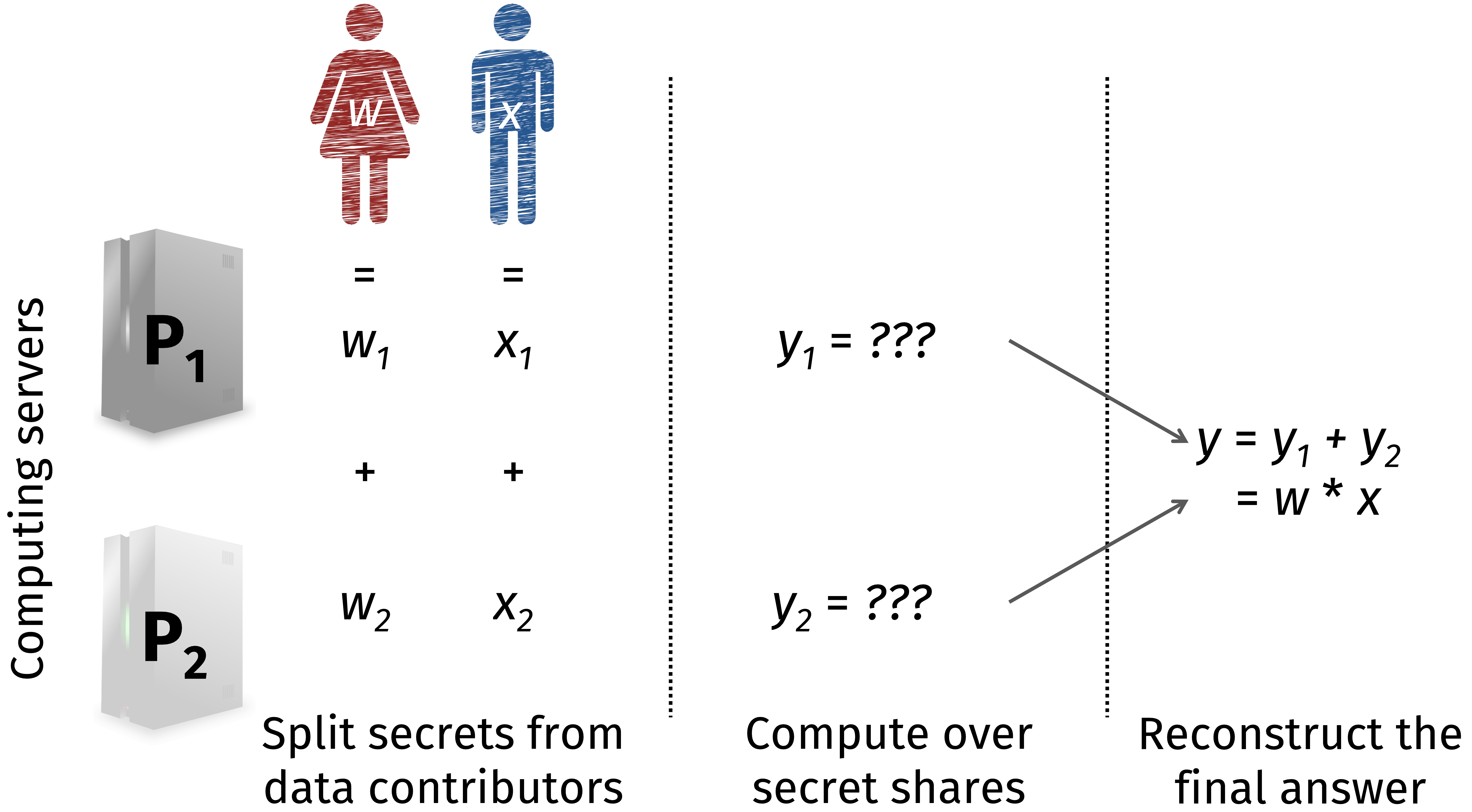
- Assume that at most *t* of the *n* parties are adversarial
  - They might collectively be acting as a passive Eve or an active Mallory
- Then, nothing is revealed about the inputs beyond what can be inferred from the output *y* (note: this inference problem can be challenging)
- Special case: *zero knowledge proofs* in which prover P(x, w) wants to convince verifier V(x) that $x \in L$ without revealing *w*

1. MPC against Eve

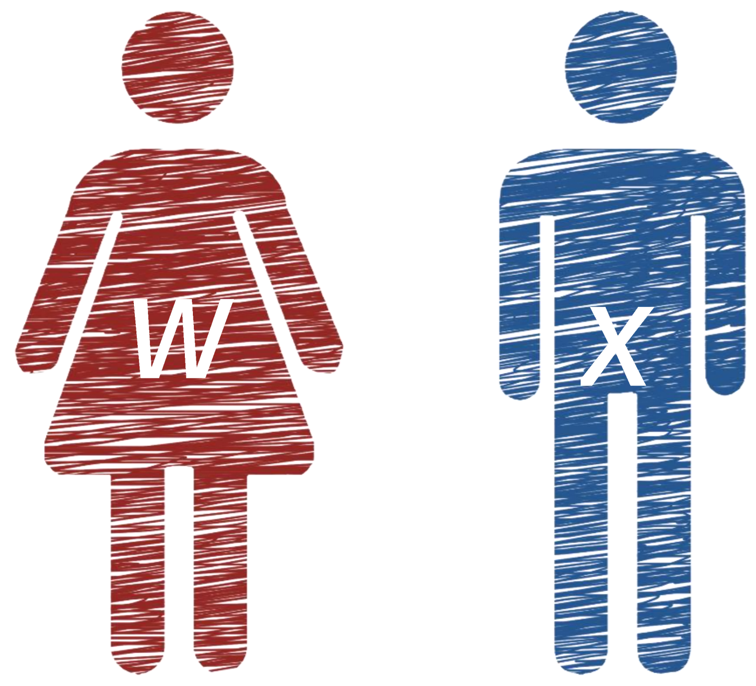
Review: secure addition [s + t] = [s] + [t]



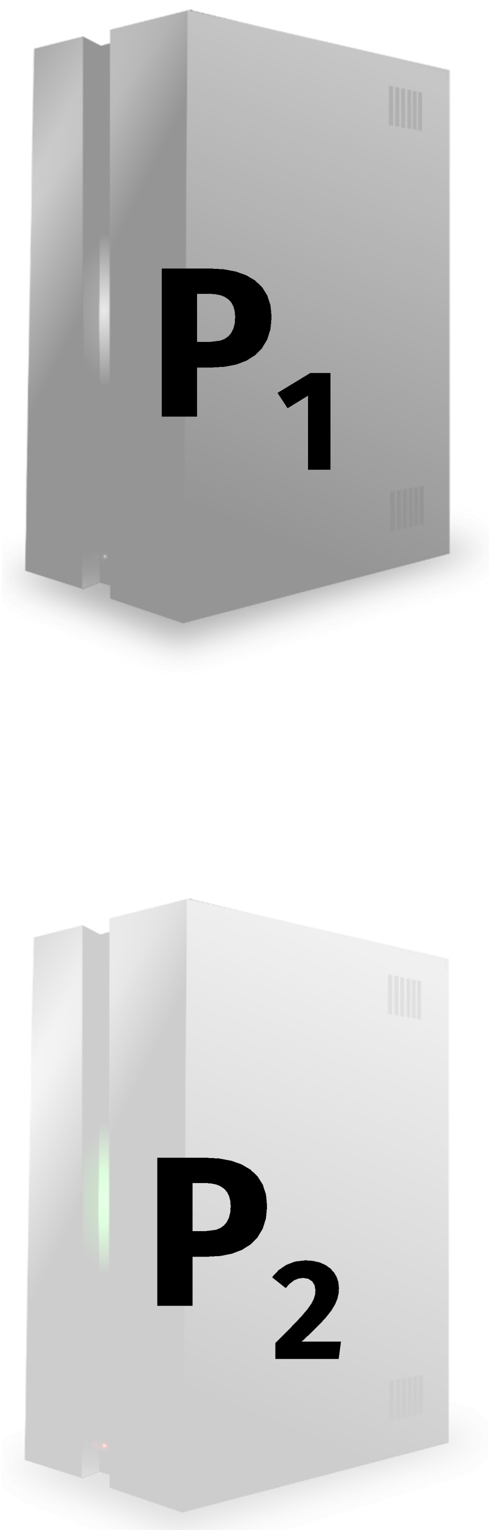
Review: secure multiplication?



Review: secure multiplication with help



Computing servers



=	=			
w_1	x_1	a_1	b_1	c_1
+	+			
w_2	x_2	a_2	b_2	c_2

Split secrets … and
random a, b, c with $a * b = c$

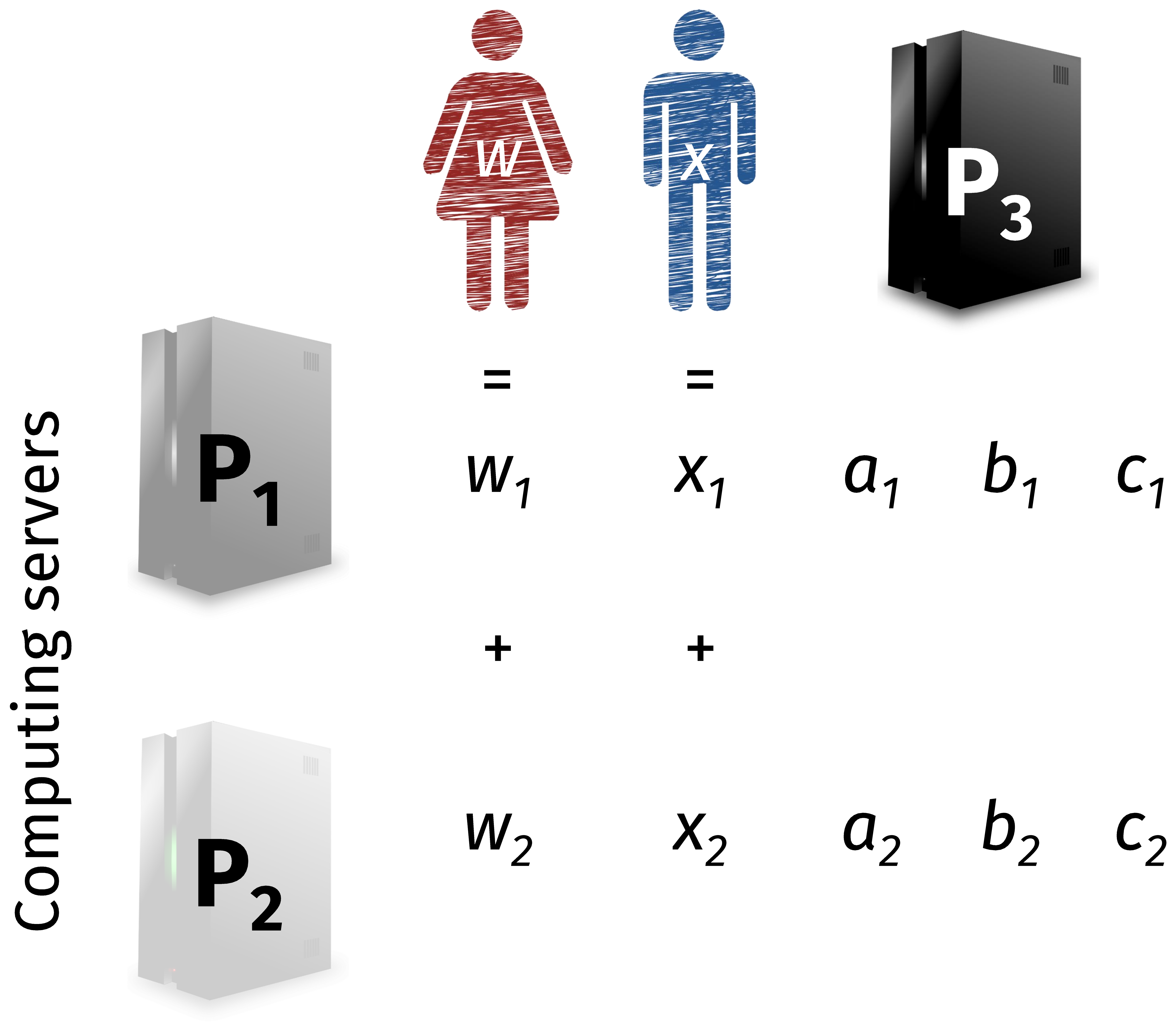
$[d] = [w] - [a]$
$[e] = [x] - [b]$

open d, e

$[y] = de + d[b] + e[a] + [c]$

Compute over
secret shares

Add a third party P_3 to generate hints

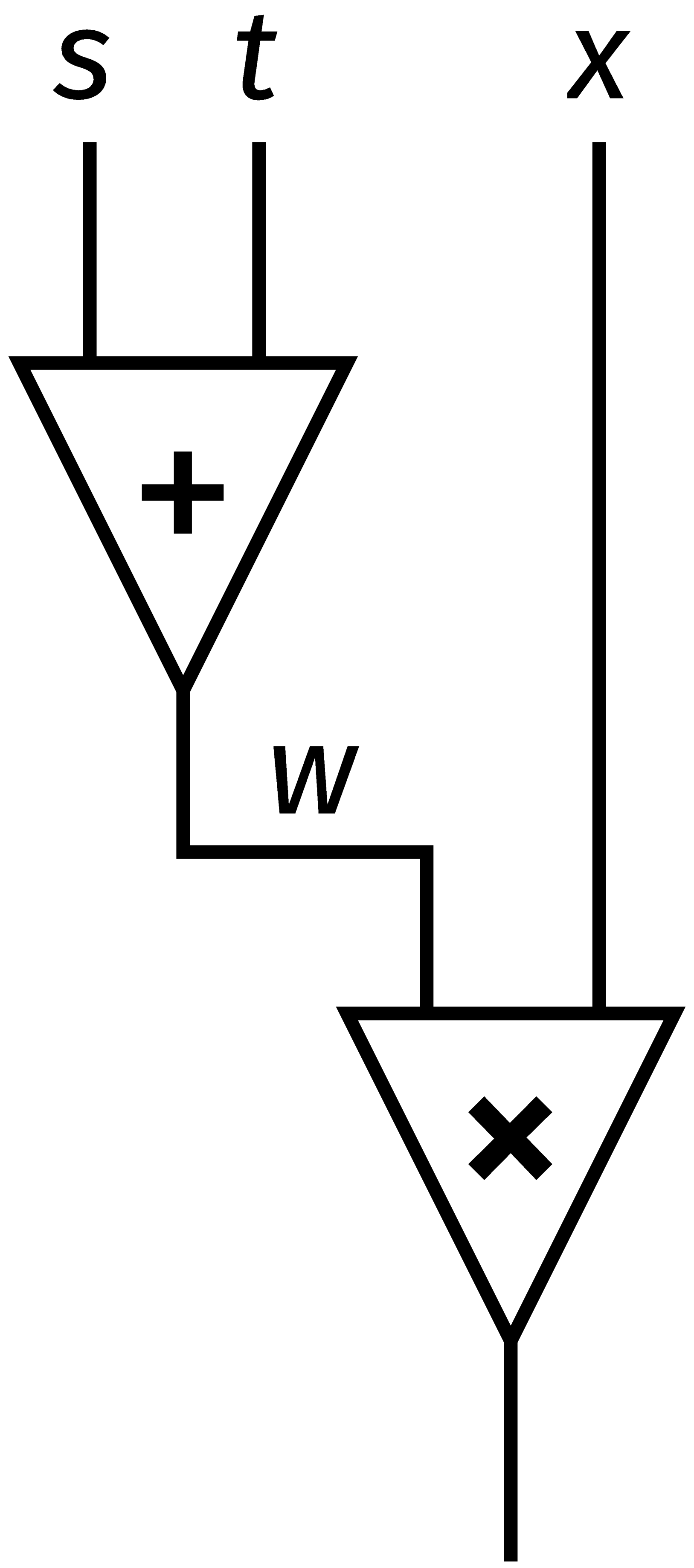


$[d] = [w] - [a]$
$[e] = [x] - [b]$

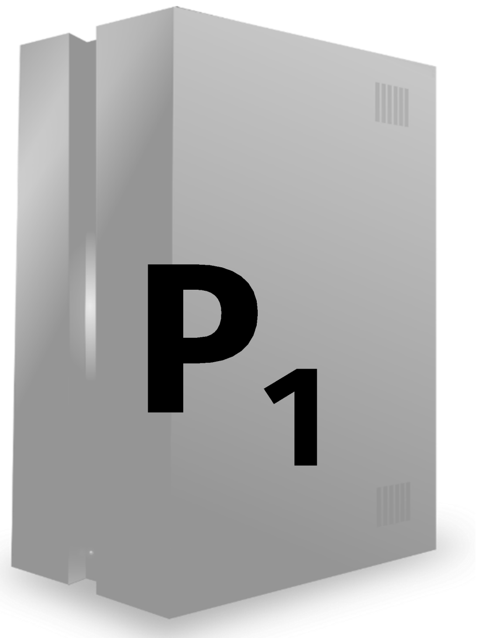
open d, e

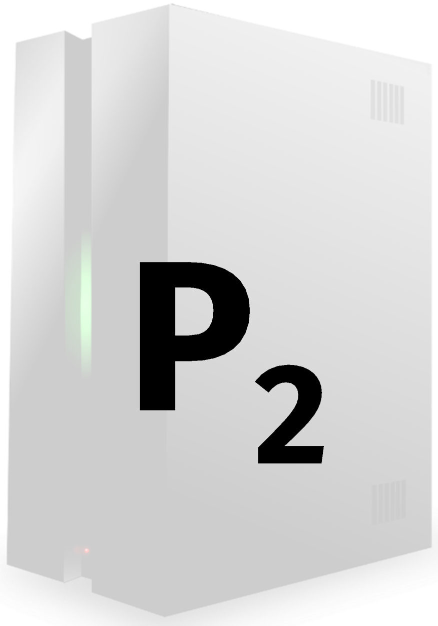
$[y] = de + d[b] + e[a] + [c]$

Putting it all together



$y = (s + t) * x$





P_1		P_2		P_3
start: s_1, t_1, x_1		start: s_2, t_2, x_2		start: nothing
$w_1 = s_1 + t_1$		$w_2 = s_2 + t_2$		
$d_1 = w_1 - a_1$ $e_1 = x_1 - b_1$	⟷	$d_2 = w_2 - a_2$ $e_2 = x_2 - b_2$	←	pick a, b, $c{=}ab$ split a, b, c
$y_1 = de + db_1 + ea_1 + c_1$		$y_2 = db_2 + ea_2 + c_2$		

Security against Eve

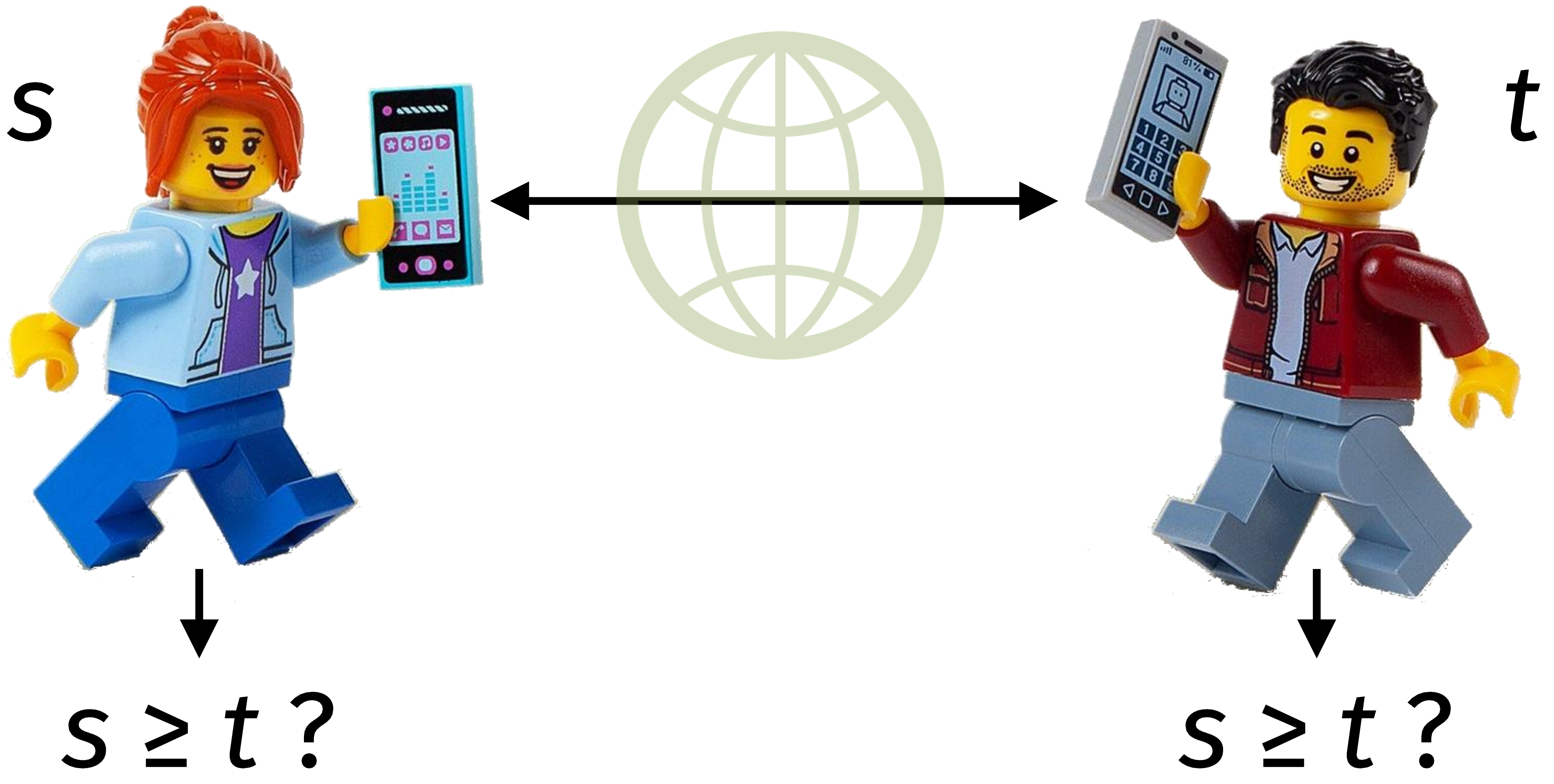
- Claim: if all three servers follow the protocol, no server learns any data
  - P_1 and P_2 each hold 1 share of each secret, the other serves as a one-time pad
  - P_3 never receives any information in the entire protocol
- However, protocol is unsafe if one server is an active Mallory
  - Bad: If Mallory = P_1, she can tamper with the output. Calculating a bad share $y_1' = y_1 + 1$ causes a corresponding change to the hidden value $y' = y + 1$
  - Worse: Some protocols that are only secure against Eve might permit Mallory to learn secrets as well (see this week's reading assignment)

Recall: Secure computation of everything

- + and * form a Turing-complete set of gates
- Ergo, we can compose them to do secure computation of any function *f*
- (This may not be the *fastest* way to compute *f* securely, however…)

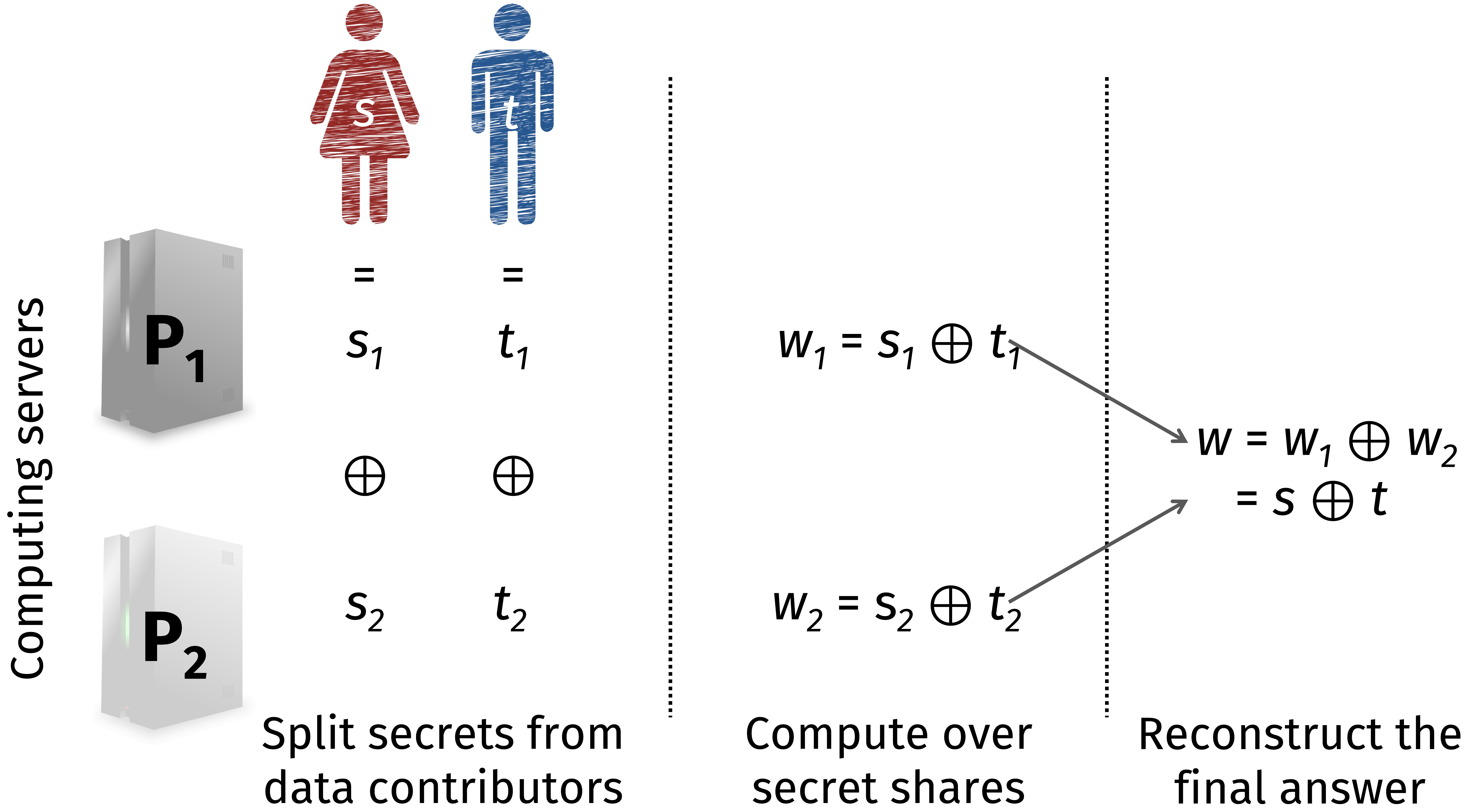
2. MPC for Boolean circuits

Yao's millionaires problem

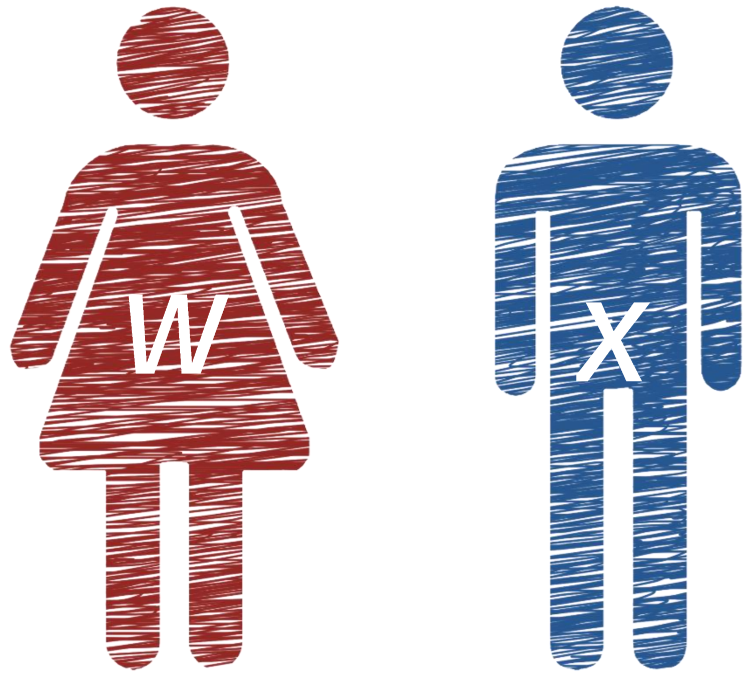


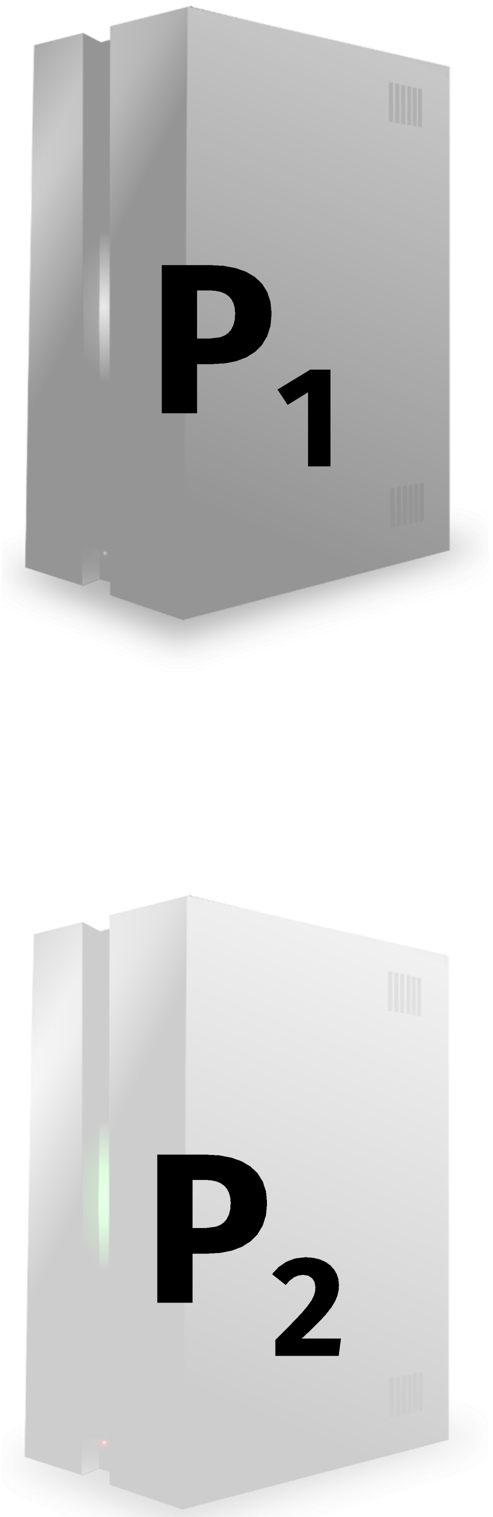
- Alice and Bob know their own salaries (s and t, respectively)
- They want to know if $s \geq t$
- You *can* convert ≥ into an arithmetic circuit… but it's large
- Much easier to compute ≥ on the bit representation of s and t

Secure Boolean XOR: a new way to split secrets!



Secure Boolean AND... with help from P_3





Computing servers

	=	=			
P_1	w_1	x_1	a_1	b_1	c_1
	$\oplus$	$\oplus$			
P_2	w_2	x_2	a_2	b_2	c_2

Split secrets … and
random a, b, c with $a \wedge b = c$

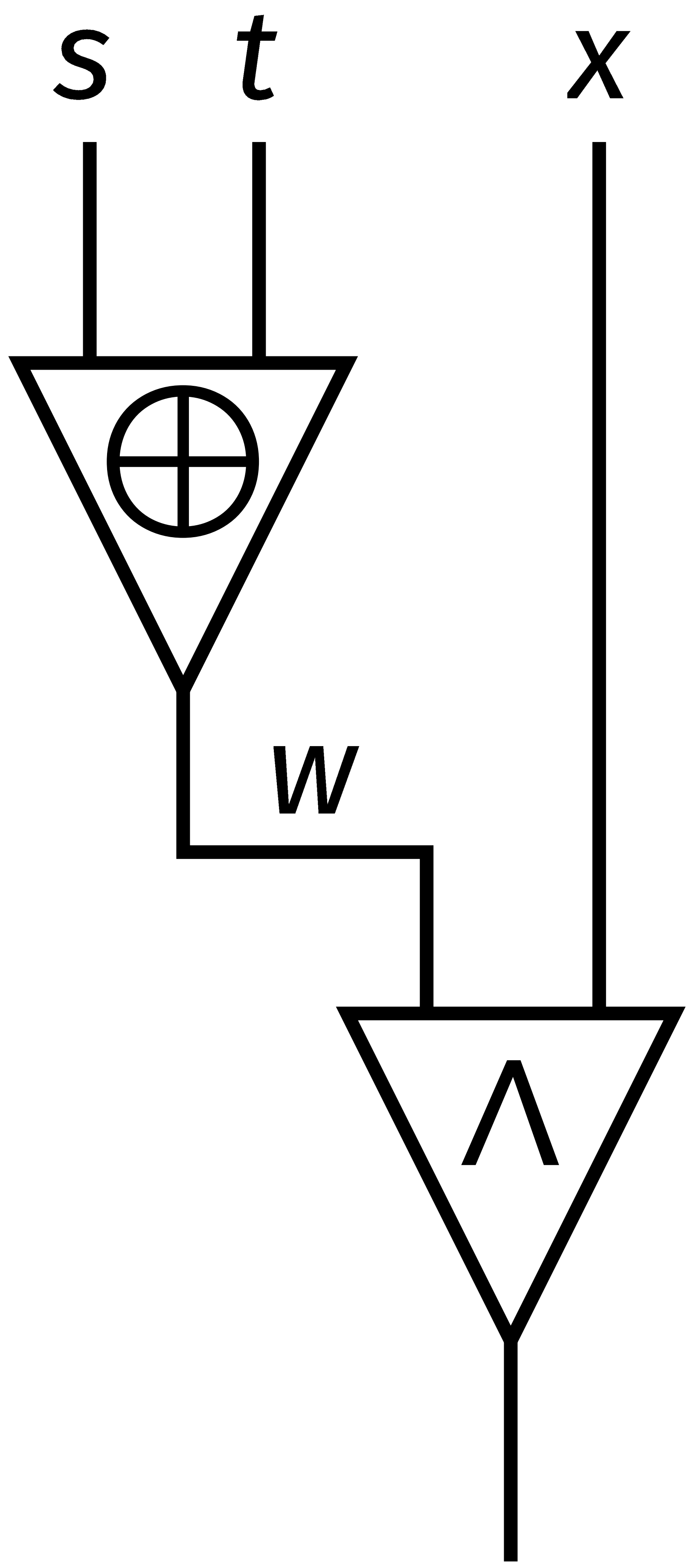
$[d] = [w] \oplus [a]$
$[e] = [x] \oplus [b]$

open d, e

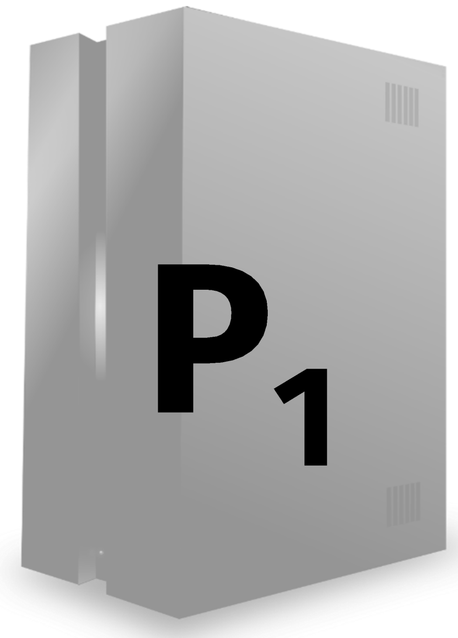
$[y] = de \oplus d[b] \oplus e[a] \oplus [c]$

Compute over
secret shares

Combined MPC for a Boolean circuit



$y = (s \oplus t) \wedge x$

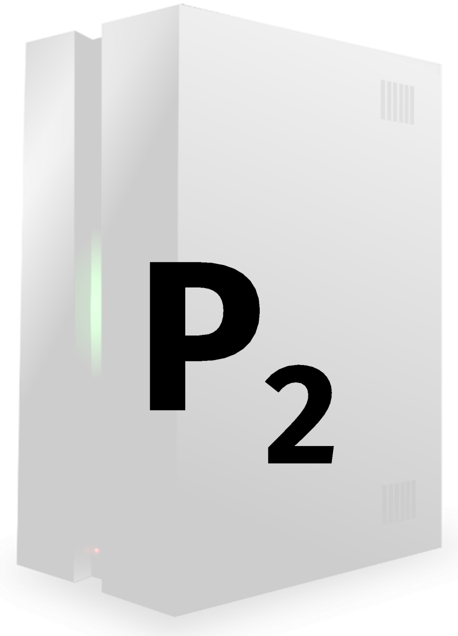


start: s_1, t_1, x_1

$w_1 = s_1 \oplus t_1$

$d_1 = w_1 \oplus a_1$
$e_1 = x_1 \oplus b_1$ ⟷

$y_1 = de \oplus db_1 \oplus ea_1 \oplus c_1$



start: s_2, t_2, x_2

$w_2 = s_2 \oplus t_2$

$d_2 = w_2 \oplus a_2$
$e_2 = x_2 \oplus b_2$

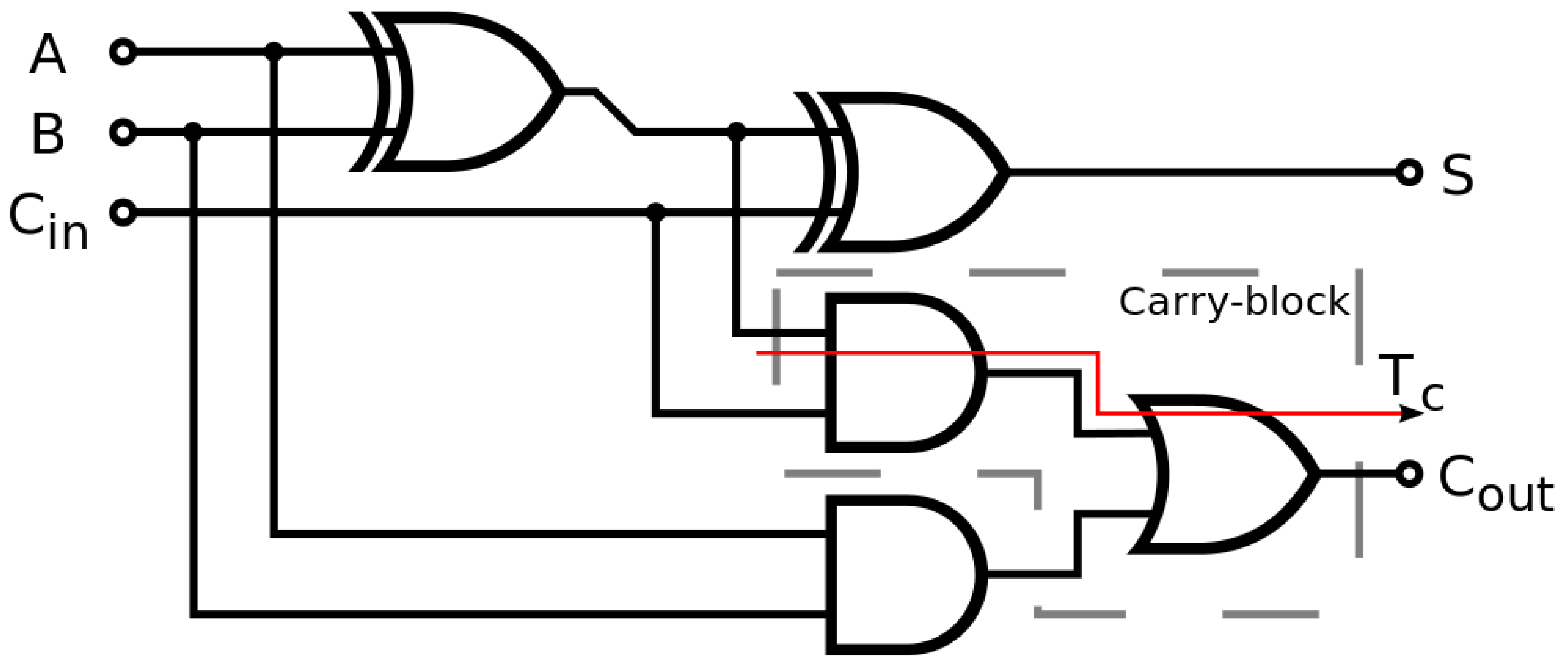
$y_2 = db_2 \oplus ea_2 \oplus c_2$

start: nothing

pick a, b, c
⟵ s.t. $c = a \wedge b$
split a, b, c

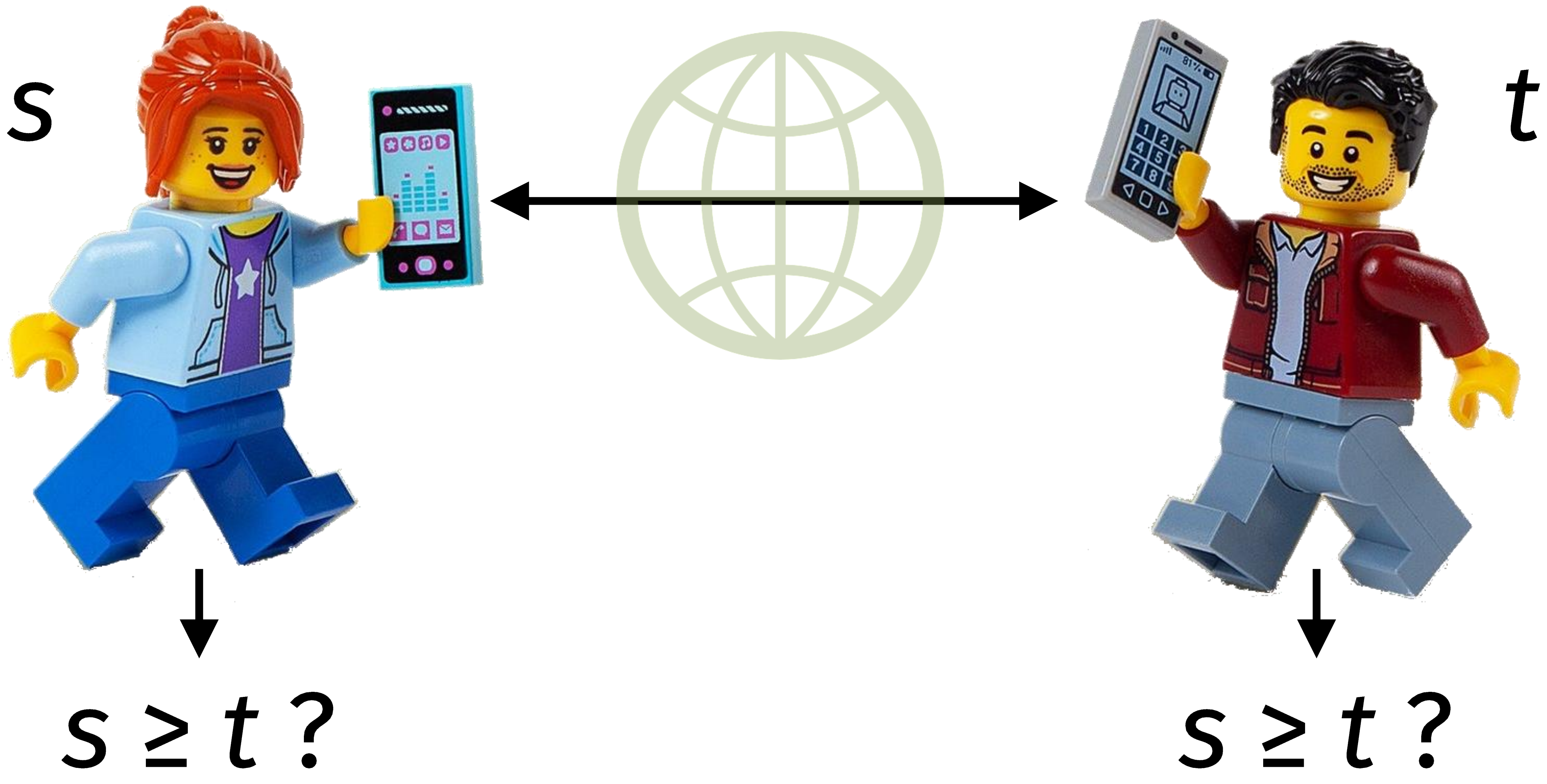
Converting between arithmetic and boolean

- Problem
  - Compute parties have an additive sharing $[x]$ of secret $x = x_1 \oplus x_2$
  - Want a Boolean sharing $x = x'_1 \oplus x'_2$
- Solution
  - P_1 imagines that x_1 is a fresh secret, makes Boolean splitting $x_1 = x_{12} \oplus x_{12}$
  - P_2 does the same: $x_2 = x_{22} \oplus x_{22}$
  - Securely compute the Boolean circuit that does ripple-carry addition of x_i
  - Result: Boolean sharing of the sum x



Source: https://en.wikipedia.org/wiki/Adder_(electronics)

Solving Yao's millionaires problem



- If salaries are one bit long:
  answer = s

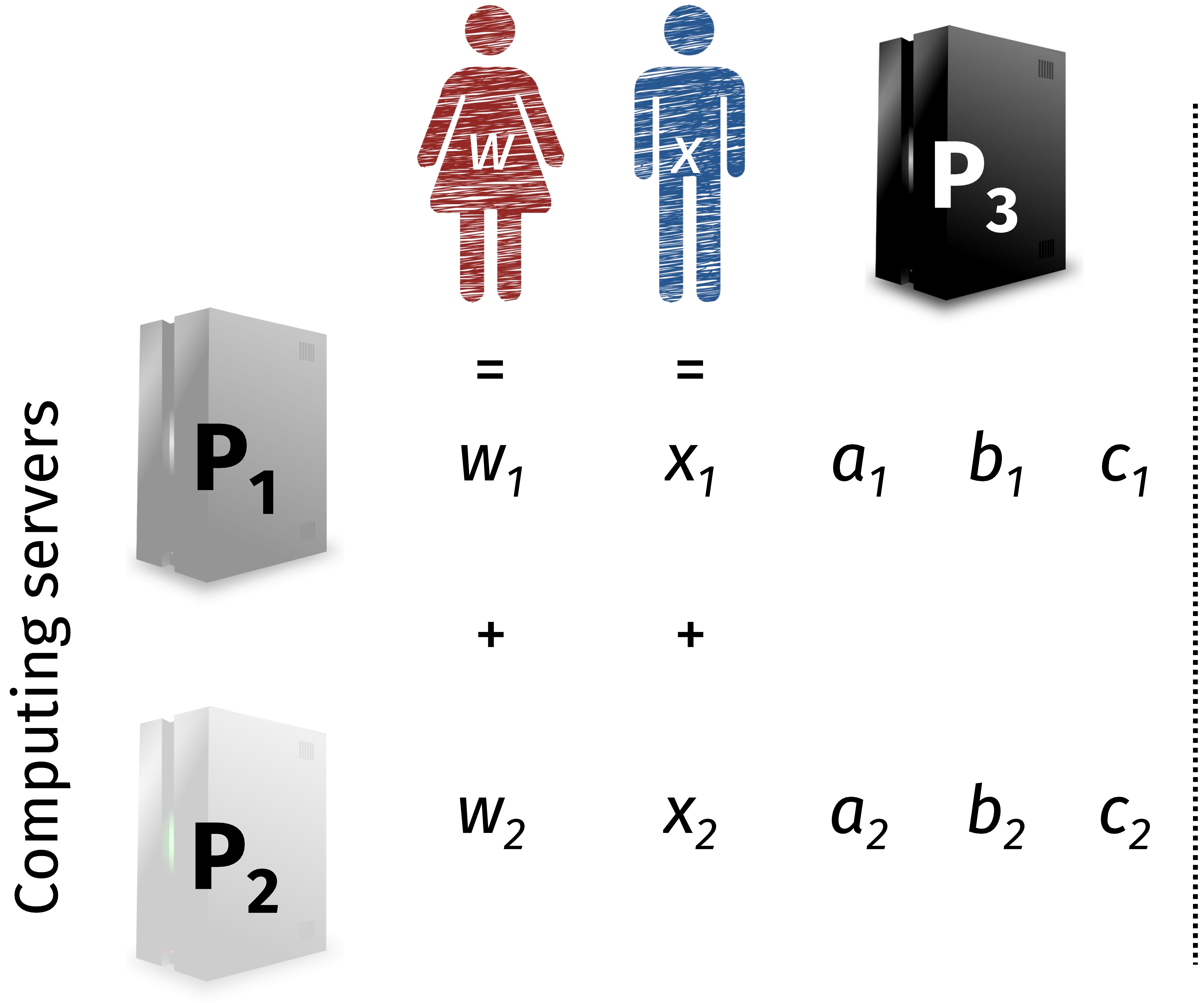
- Given 2-bit salaries $s = s^2 s^1$:
  answer = $(s^2 \oplus t^2) \wedge s^2$
  $\oplus\ (s^2 \oplus t^2 \oplus 1) \wedge s^1$

- Given 3-bit salaries: same idea...

- Important: cannot 'short circuit' a secure computation

3. MPC with preprocessing

Observe: “hints” from P_3 are data-independent

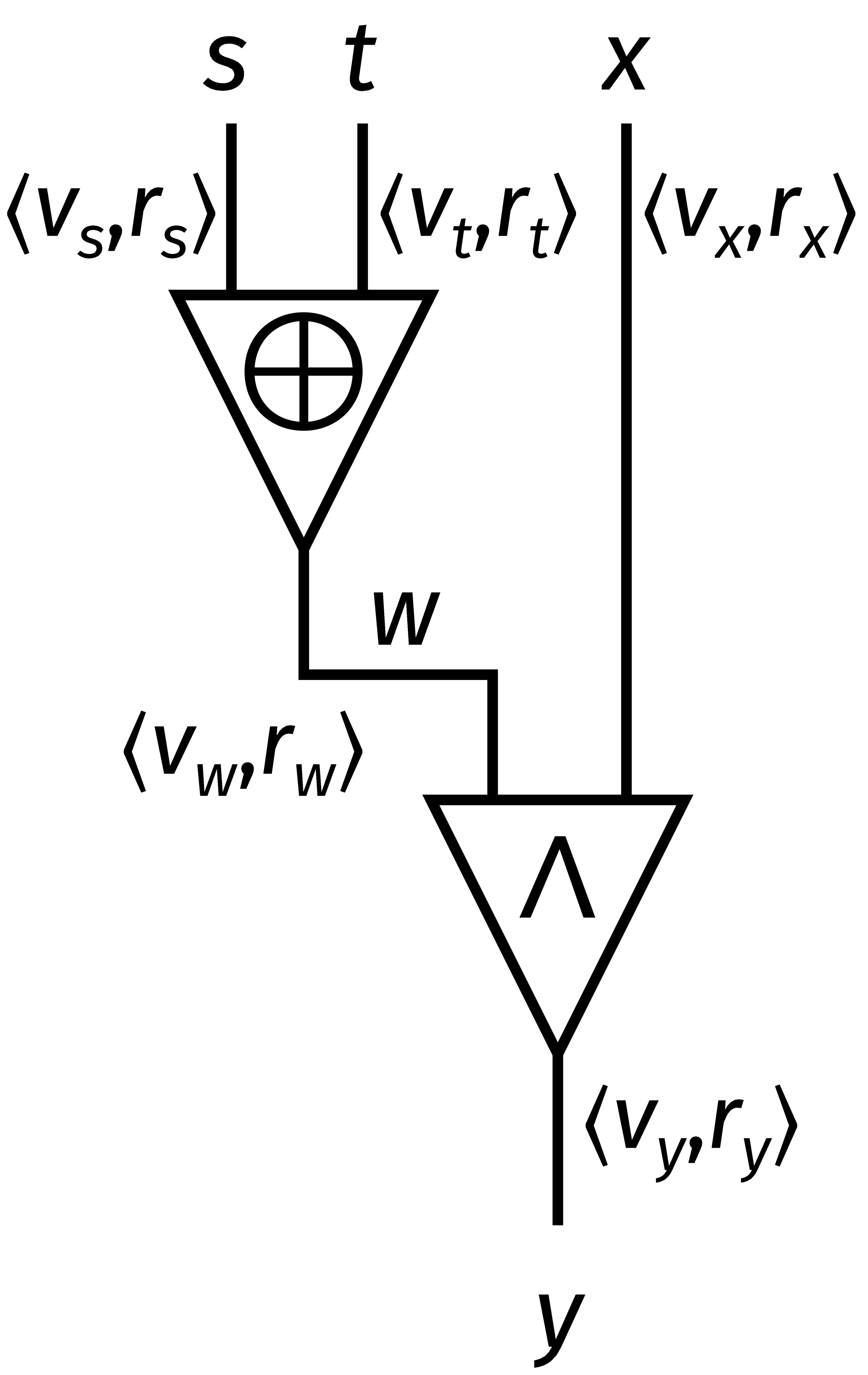


$[d] = [w] - [a]$
$[e] = [x] - [b]$

open d, e

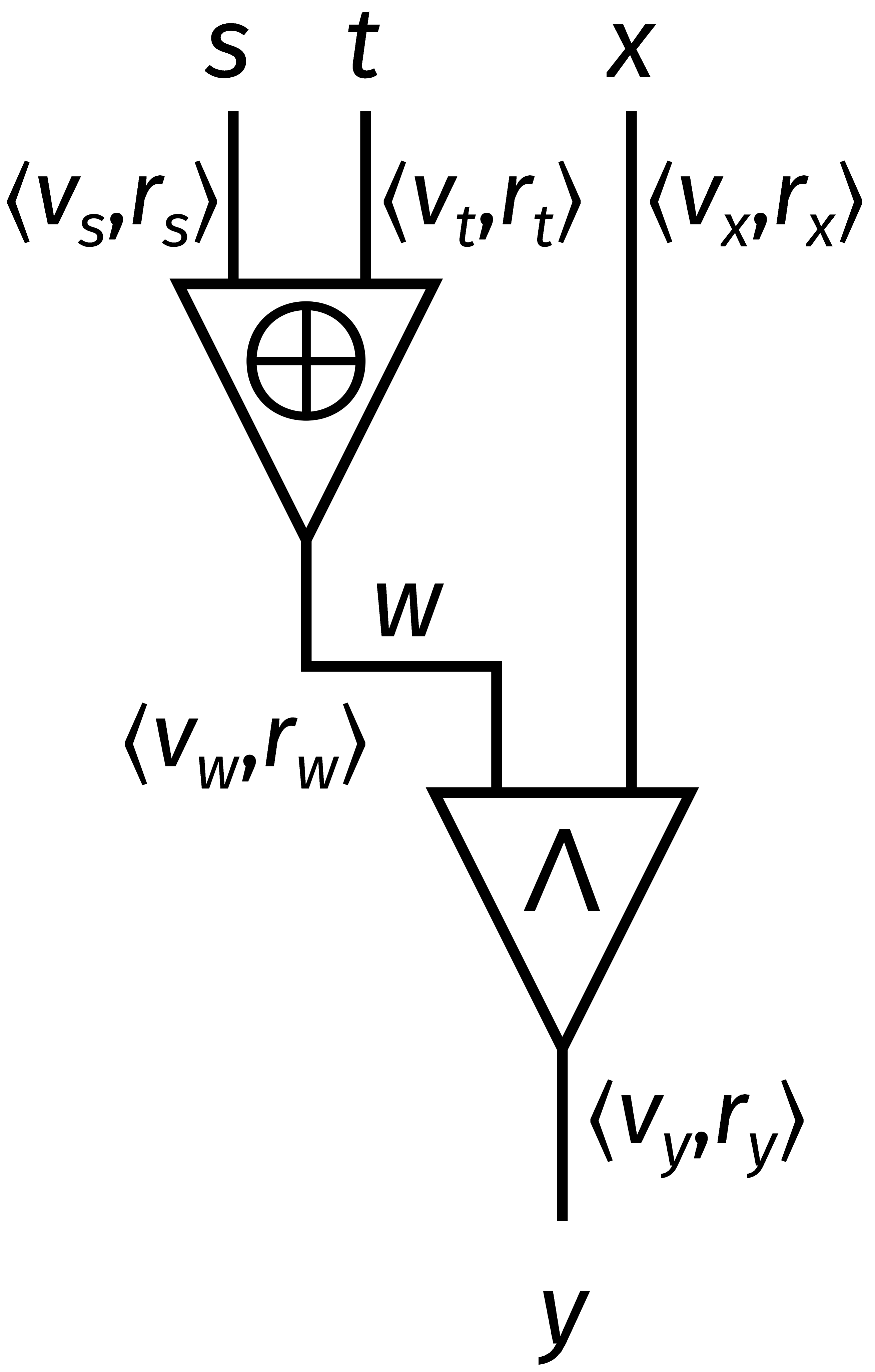
$[y] = de + d[b] + e[a] + [c]$

Pre-processing



- New plan: we will consider two values for every wire *s*: the secret value v_s and an independent, random r_s
- P_3 can pre-compute the entire circuit on the r_w
  - Sample r_s, r_t, r_x uniformly at random
  - Compute $r_w = r_s + r_t$
  - Compute $r_y = r_w * r_x$ (note: mult has one extra detail…)
- P_3 gives P_1 and P_2 one share $[r]$ of each random value

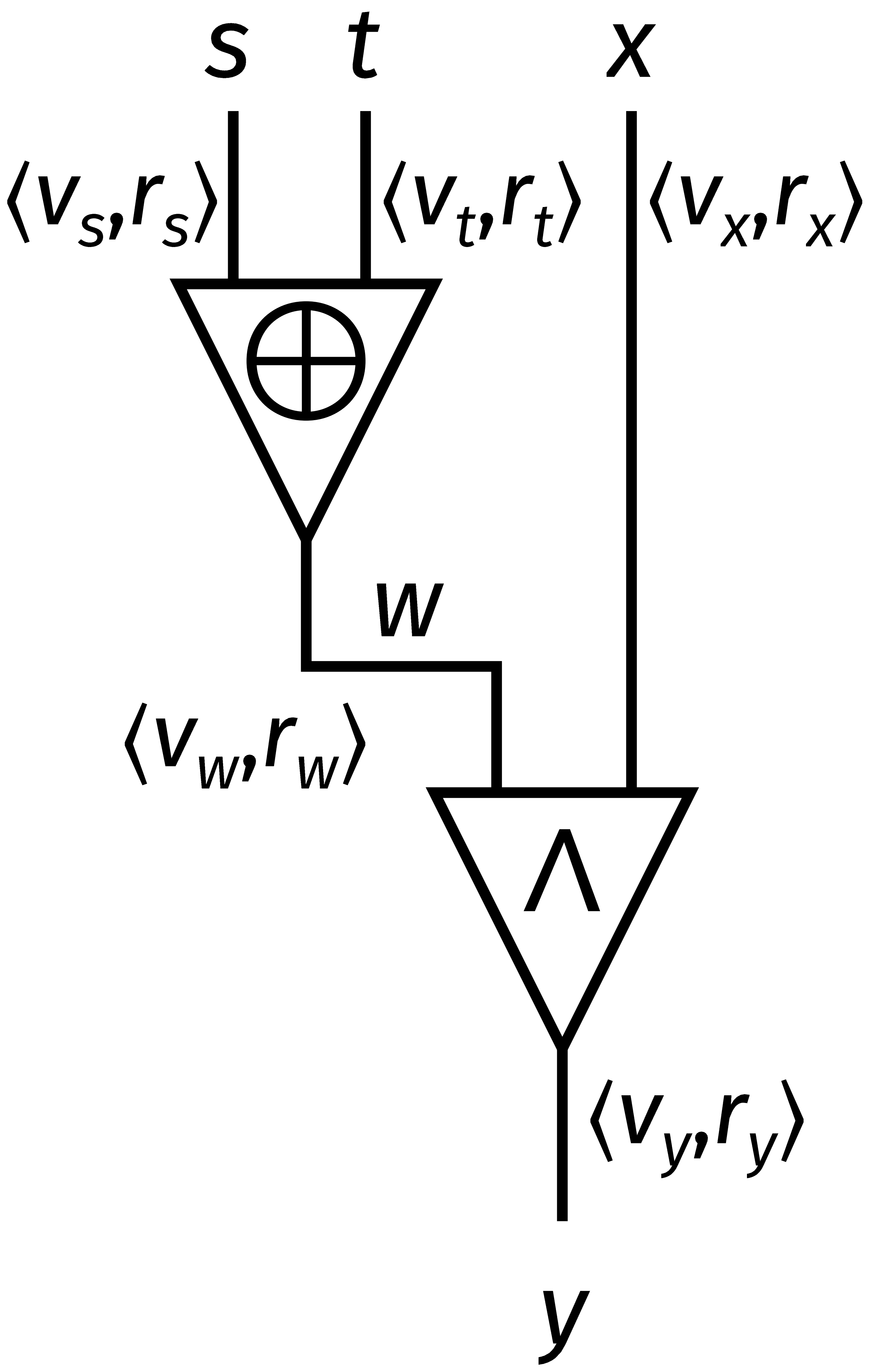
Compute on masked data

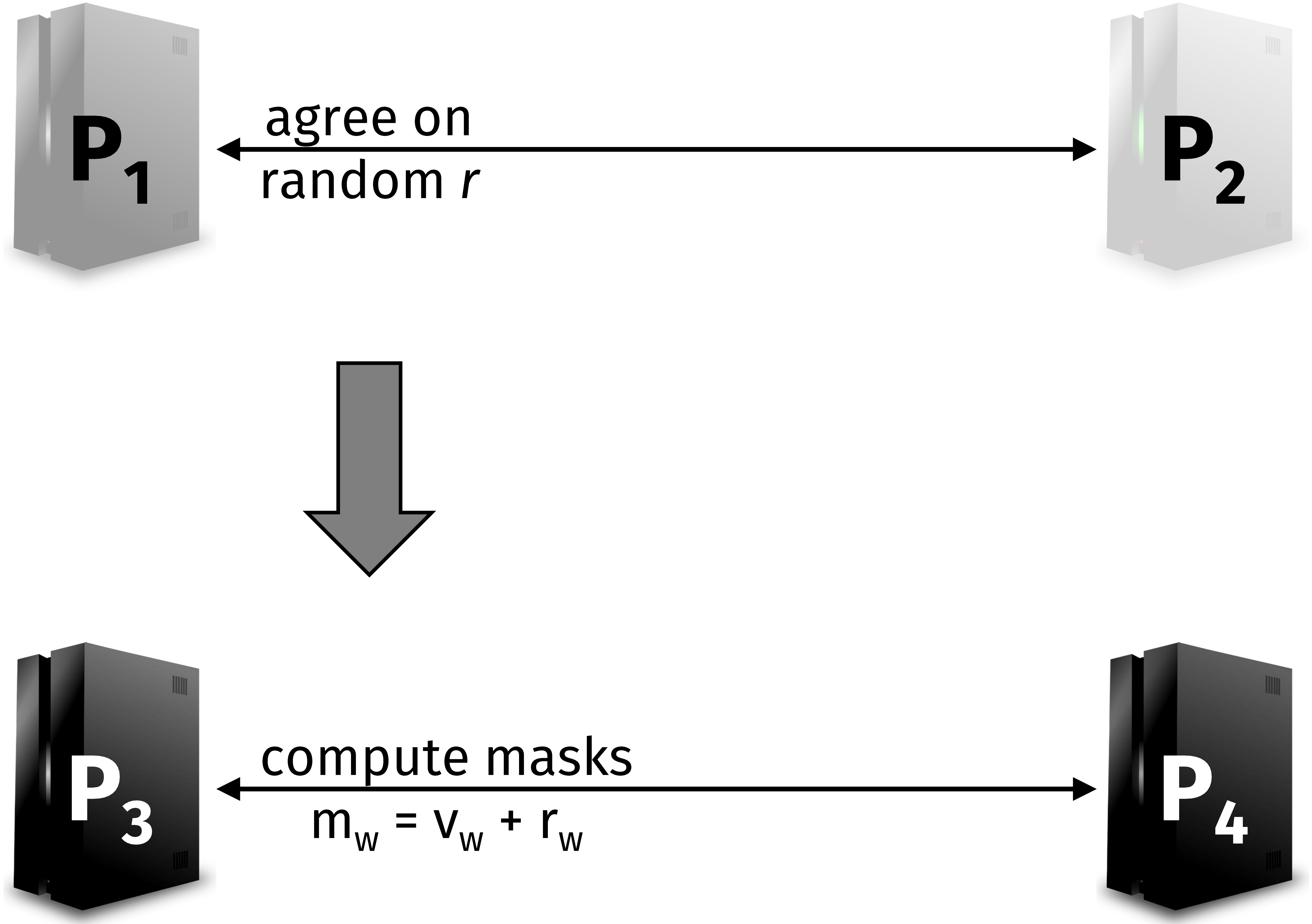


- Data holders use random r as a one-time pad, P_1 and P_2 are given "masked" wire values $m_s = v_s + r_s$
- P_1 and P_2 compute all masks in the clear (no shares!)
- Addition of masks gives addition of real values: set $m_w \triangleq m_s + m_t$, then $v_w = (m_s - r_s) + (m_t - r_t) = v_s + v_t$
- Multiplication of masks follows our algebra trick: set $[m_y] \triangleq m_w m_x - m_w[r_x] - m_w[r_x] + r_y$ and open m_y
- Invariant: none of the compute parties learn any v_s
- Reveal y to the output party by providing v_y and r_y

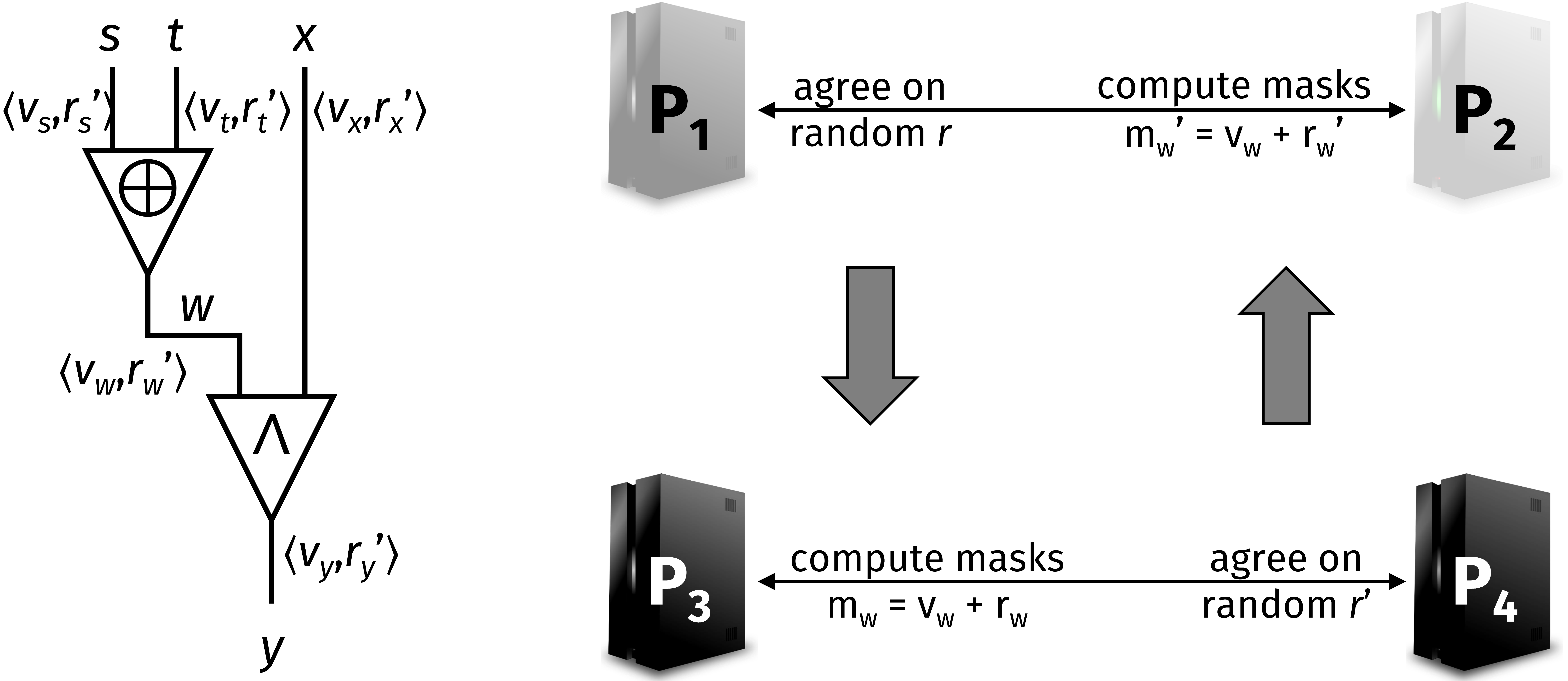
4. MPC against Mallory

Add a fourth party P_4 for redundancy

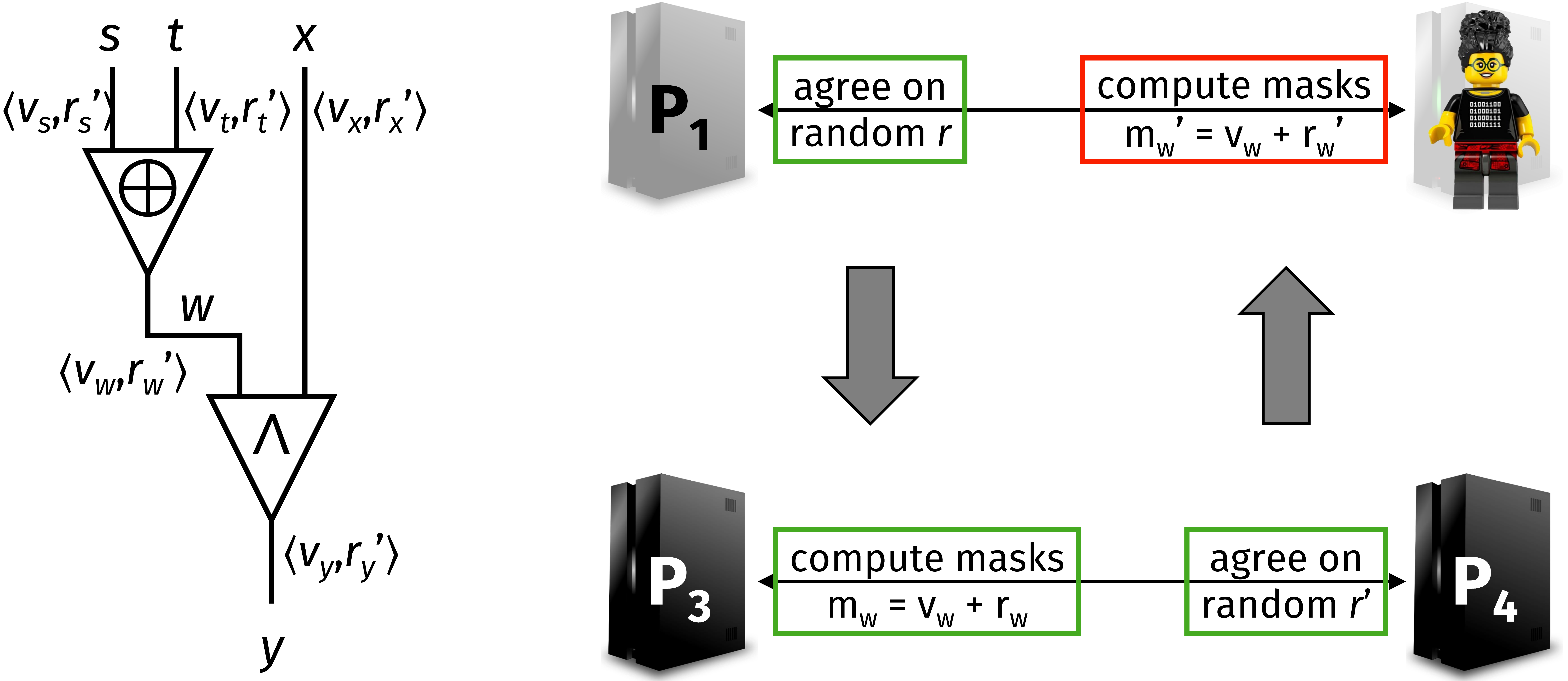




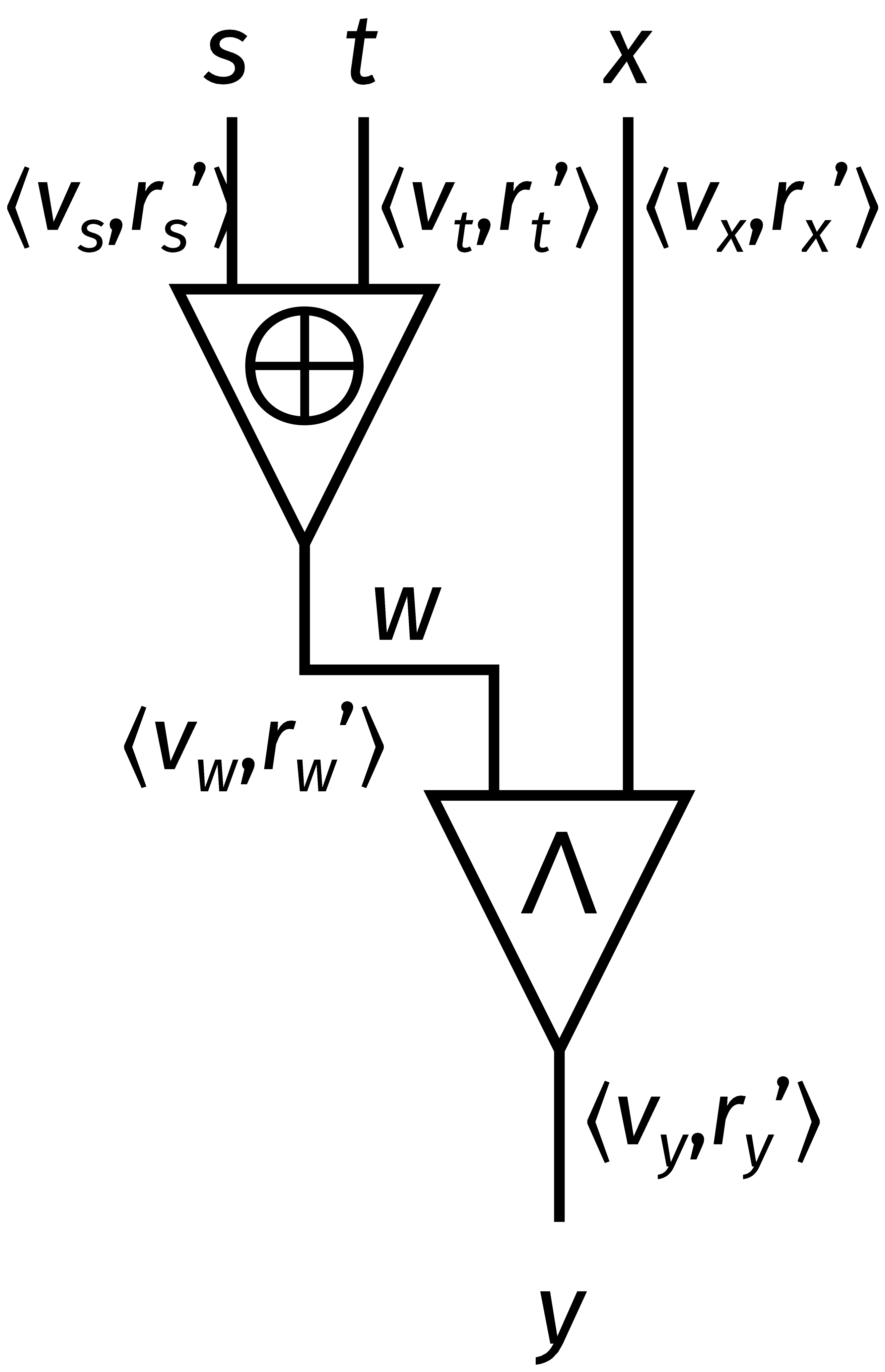
Add a fourth party P_4 for redundancy

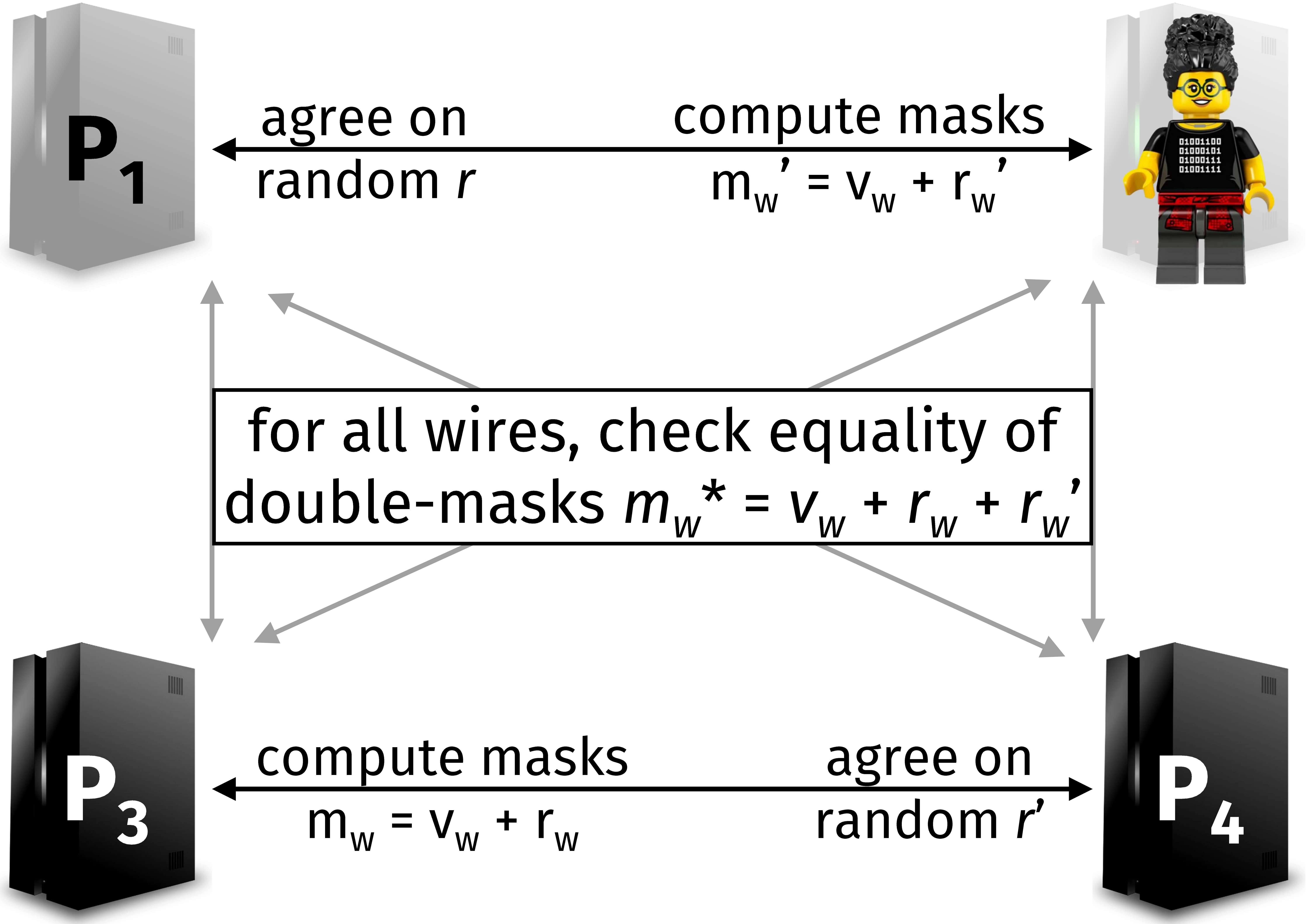


Secure against Mallory?



Secure against Mallory!

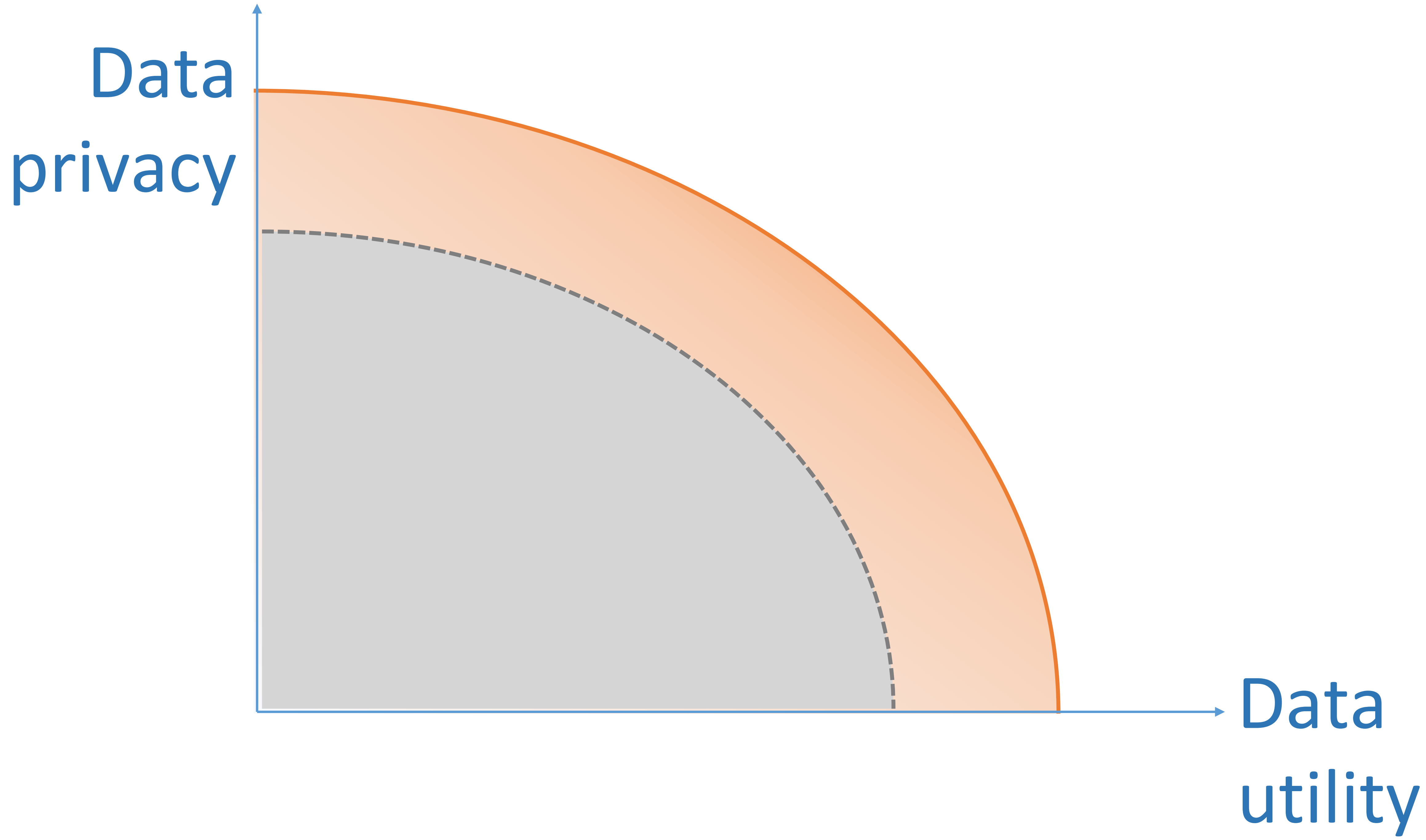




5. Final thoughts

Benefit of cryptographically secure computation

- MPC says nothing about which data analyses are worthwhile to compute
- MPC de-couples discussion of *what* to compute from *how* to do so
- MPC expands the Pareto frontier of possible data analyses

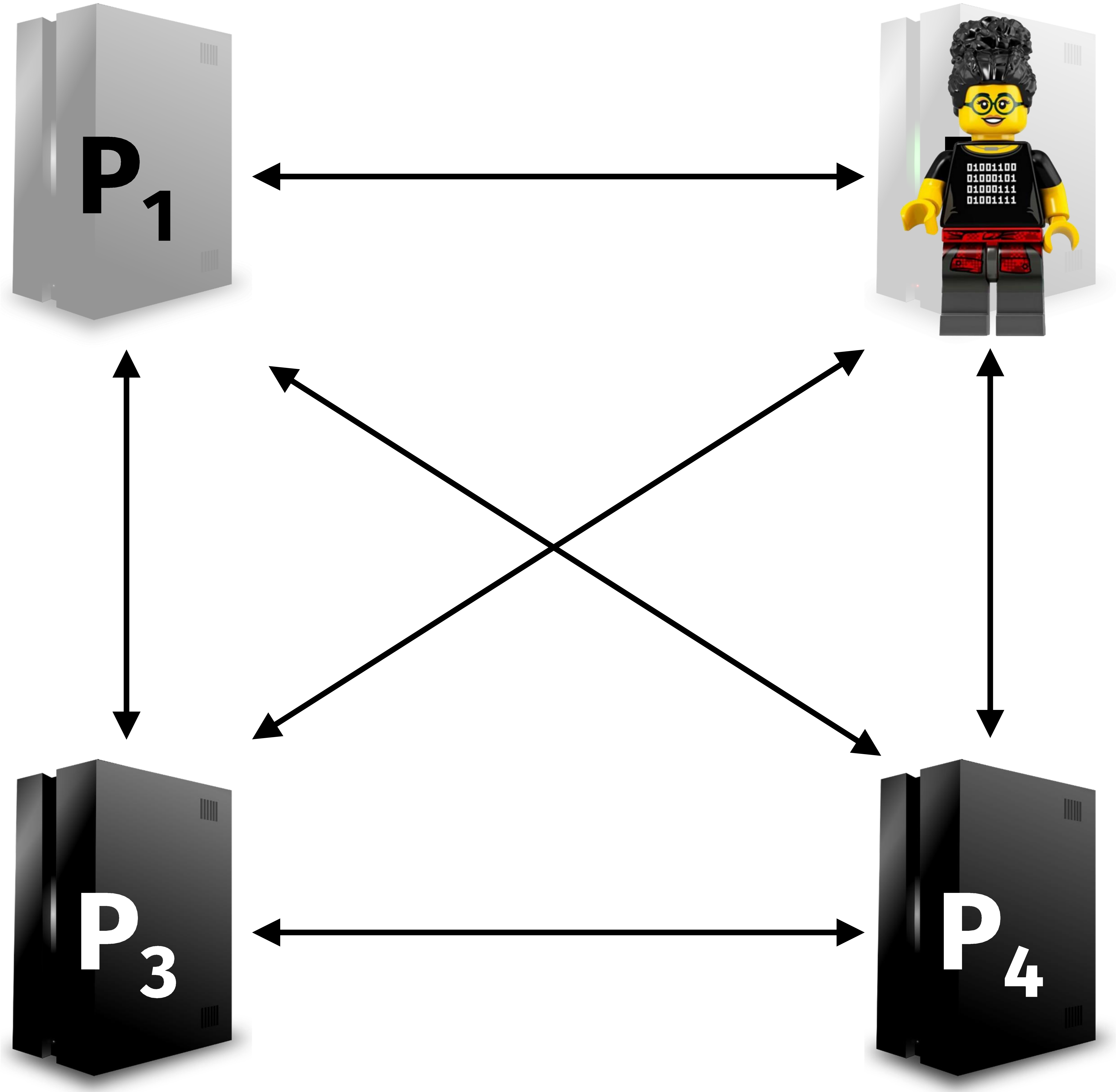


Special case: zero-knowledge proofs

- Consider two parties: a prover P and a verifier V
- There is a public statement x that is claimed to be in an NP language L, and the prover knows a witness w such that $R(x, w) = \text{True}$
- P wants to convince V that $x \in L$, but without revealing w
- Prover and verifier can execute a 2-party secure computation of R

Zero knowledge via "MPC in the head"

- P wants to convince V that $x \in L$, but without revealing w
- Prover securely computes R(x,w)
  - Prover acts as *all* compute parties
- Let the verifier choose t parties and receive their complete state
  - Privacy: observing the view of t parties gives V no information
  - Accuracy: if P deviates from the protocol, Pr[V catches] = t/n



Next week: securely computing *specific* functions