CS4501 Cryptographic Protocols
Lecture 9: BGW Example, Multiplication Protocols

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Recap: The $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

- *Hybrid Models* are models of the world that *blend* aspects of the Ideal and Real world models. The parties run a non-trivial protocol, like in the real world, but one or more functionalities also participate in the protocol. This is another world on which the Distinguisher can run experiments!
- The usefulness of hybrid models comes from the *transitivity* of indistinguishability: if no algorithm can distinguish the real world from a hybrid world, and no algorithm can distinguish the same hybrid world from the ideal world, then the real and ideal worlds must also be indistinguishable.
- Looking at it another way, hybrid models allow us to break down protocols *and their security proofs* into self-contained, reusable components rather than constructing and proving them monolithically.
- *For now*, we will make a key simplifying assumption: that functionalities do not run *concurrently*. To make this easier, we will assume non-reactivity.

Recap: How to *Use* the $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathcal{F}_{\text{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathcal{F}_{\text{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\text{mul}}$.
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul}.

Lemma 1: Let $p > n > t$, let $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be any *randomized* n-ary function, and let C be a circuit that computes f. Assuming synchronicity and secure channels, $\pi_{\text{BGW}}(n, t, p, C)$ perfectly realizes $\mathcal{F}_{\text{SFE}}(n, f, \mathbb{F}_p, \ldots, \mathbb{F}_p)$ in the presence of a semi-honest $\mathcal{A}$ that statically corrupts up to $n - 1$ parties in the $\mathcal{F}_{\text{mul}}$-hybrid model.

Proved in Lecture 8!

Recap: How to *Use* the $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathcal{F}_{\text{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathcal{F}_{\text{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\text{mul}}$.
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul}.

Lemma 2: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels there exists an n-party protocol that perfectly realizes $\mathcal{F}_{\text{mul}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

Lecture 8: why this is nontrivial. **Today: Two Protocols!**

Recap: How to *Use* the $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul}. **Next Class!**

Perfect MPC Composition Theorem: Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that perfectly realizes $\mathcal{F}_{\text{inner}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

Recap: How to *Use* the $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

1. Construct the BGW protocol π_{BGW} for any (possibly nonlinear) circuit in the $\mathcal{F}_{\text{mul}}$-hybrid model (we just need to add multiplication gates!)
2. Prove that π_{BGW} realizes $\mathcal{F}_{\text{SFE}}$.
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\text{mul}}$.
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul}.

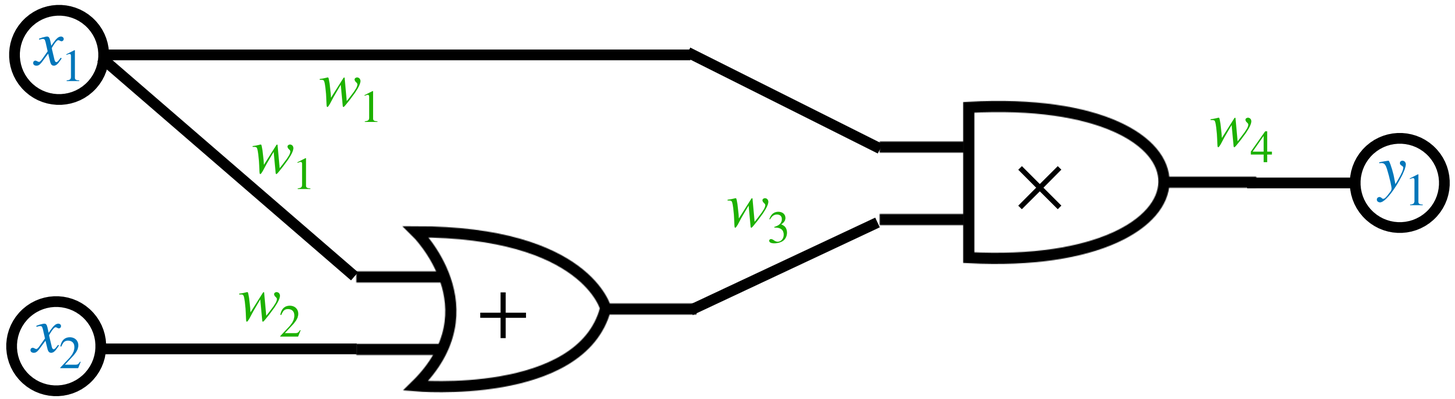
Putting it together...

Perfect MPC Feasibility Theorem: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels there exists an n-party protocol that perfectly realizes $\mathcal{F}_{\text{SFE}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

1+2. Recap: BGW Protocol and Proof (A Worked Example)

The Setting for Our Example

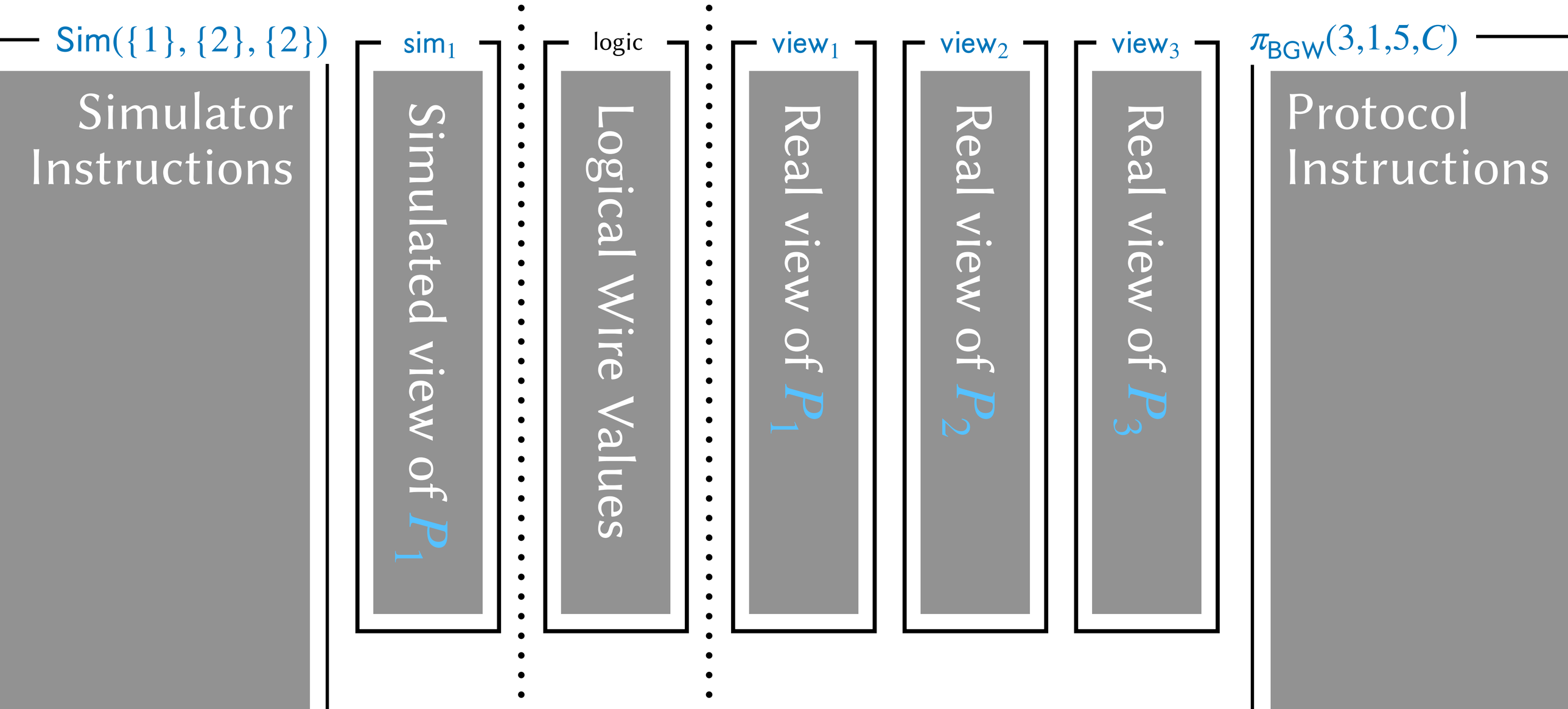
- $t = 1$, $n = 3$, $p = 5$, so we have 3 parties (P_1, P_2, P_3), with at least two honest, and they work over $\mathbb{F}_5$. They wish to compute $f(x_1, x_2, \lambda) = ((x_1 + x_2) \cdot x_1, \lambda, \lambda)$.
- The circuit that computes f is $C = \{(\text{in},1,1), (\text{in},2,2), (+,1,2,3), (\times,1,3,4), (\text{out},1,4)\}$:

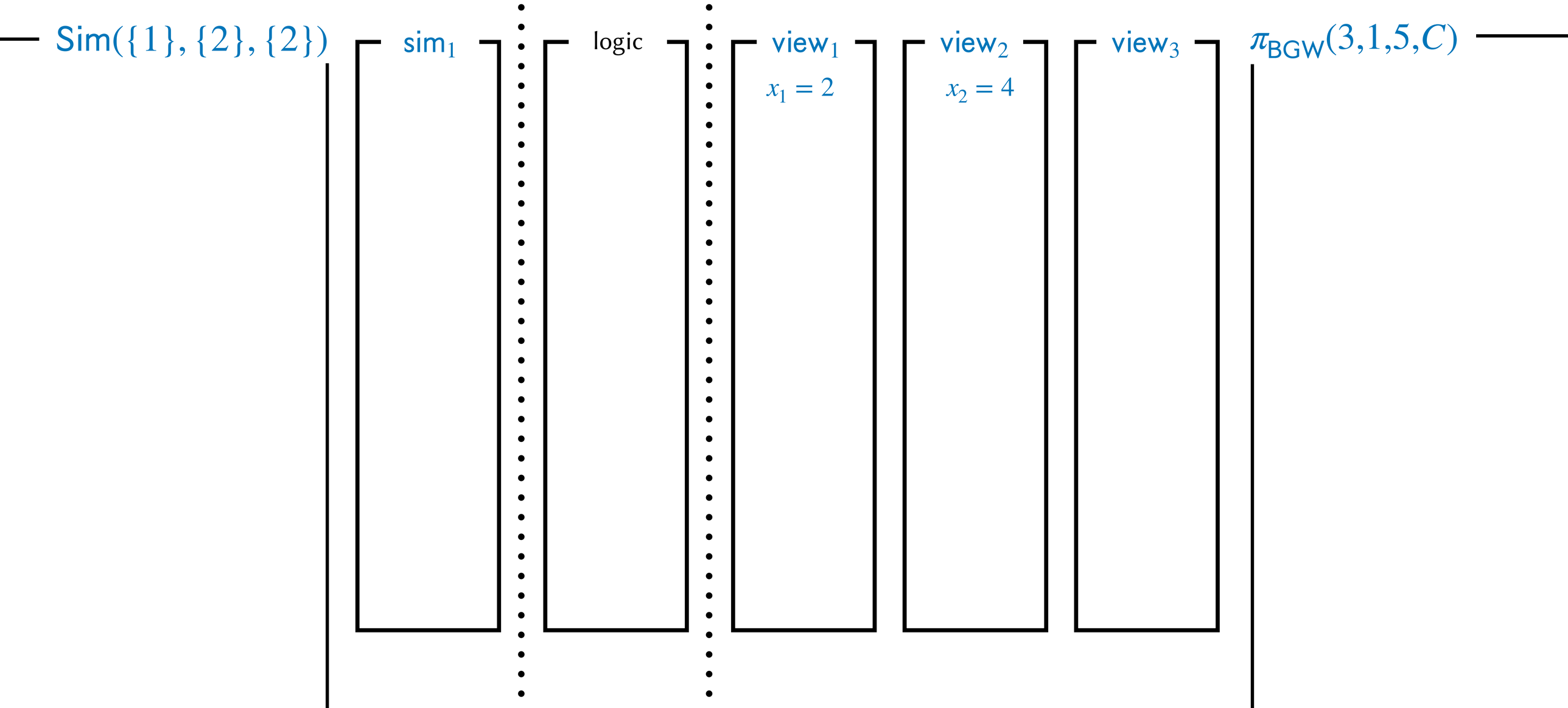


  Notice P_3 has no input, and only P_1 has output (y_1). They will compute the gates in the order listed above.

- We will fix $x_1 = 2$ and $x_2 = 4$, and P_1 will be corrupted. Notice that $f(2,4,\lambda) = (2,\lambda,\lambda)$.

How We Will Show Things

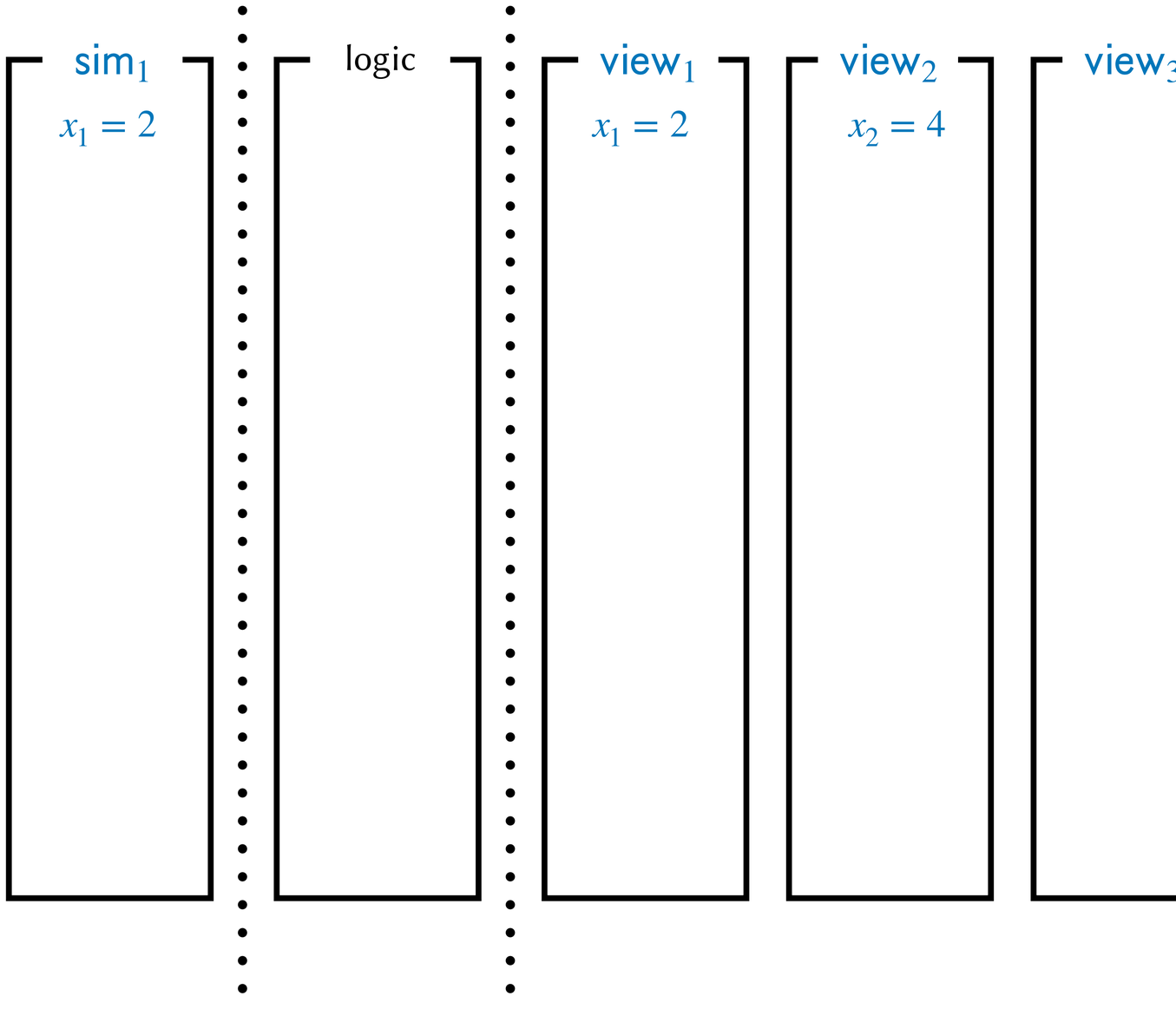



Sim({1}, {2}, {2})
sim1
logic
view1
x1 = 2
view2
x2 = 4
view3
πBGW(3,1,5,C)

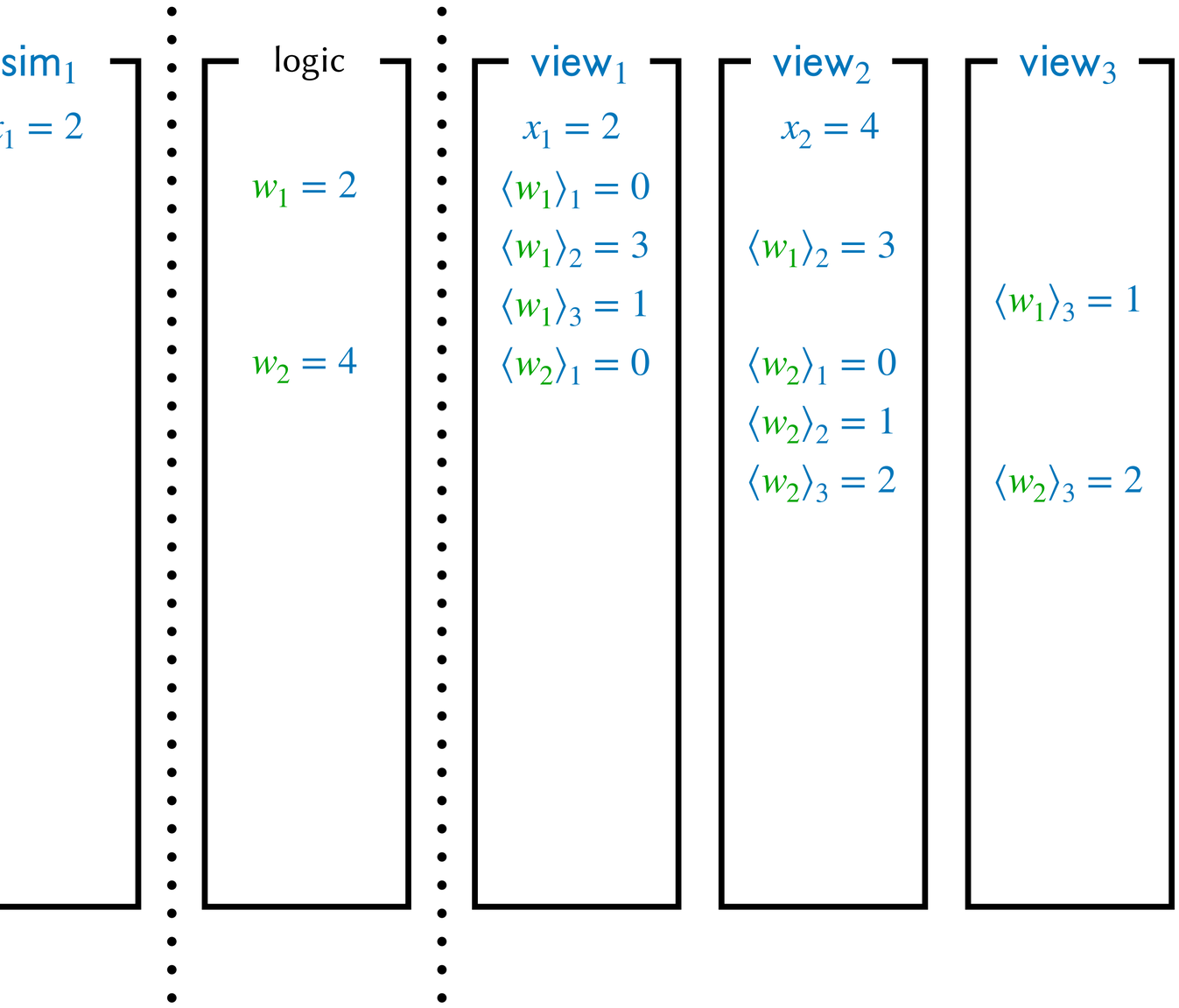
Initialization

Sim($\{1\}, \{2\}, \{2\}$)

1. Copy the inputs of the corrupt parties into their views.



Gates: (in,1,1), (in,2,2)



$\pi_{\mathsf{BGW}}(3,1,5,C)$

1. **Input Sharing:** every P_i with input x_i finds the entry $(\mathsf{in}, i, o) \in C$, computes $\langle w_o \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$ and sends $\langle w_o \rangle_j$ to every P_j for $j \in [n] \setminus \{i\}$.

P_1 samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + x_1$.
w.p. 1/5, $f_1(x) = 3x + 2 \implies \langle w_1 \rangle = (0,3,1)$

P_2 samples $a_2 \leftarrow \mathbb{F}_5$ and sets $f_2(x) = a_2 \cdot x + x_2$.
w.p. 1/5, $f_2(x) = 1x + 4 \implies \langle w_2 \rangle = (0,1,2)$

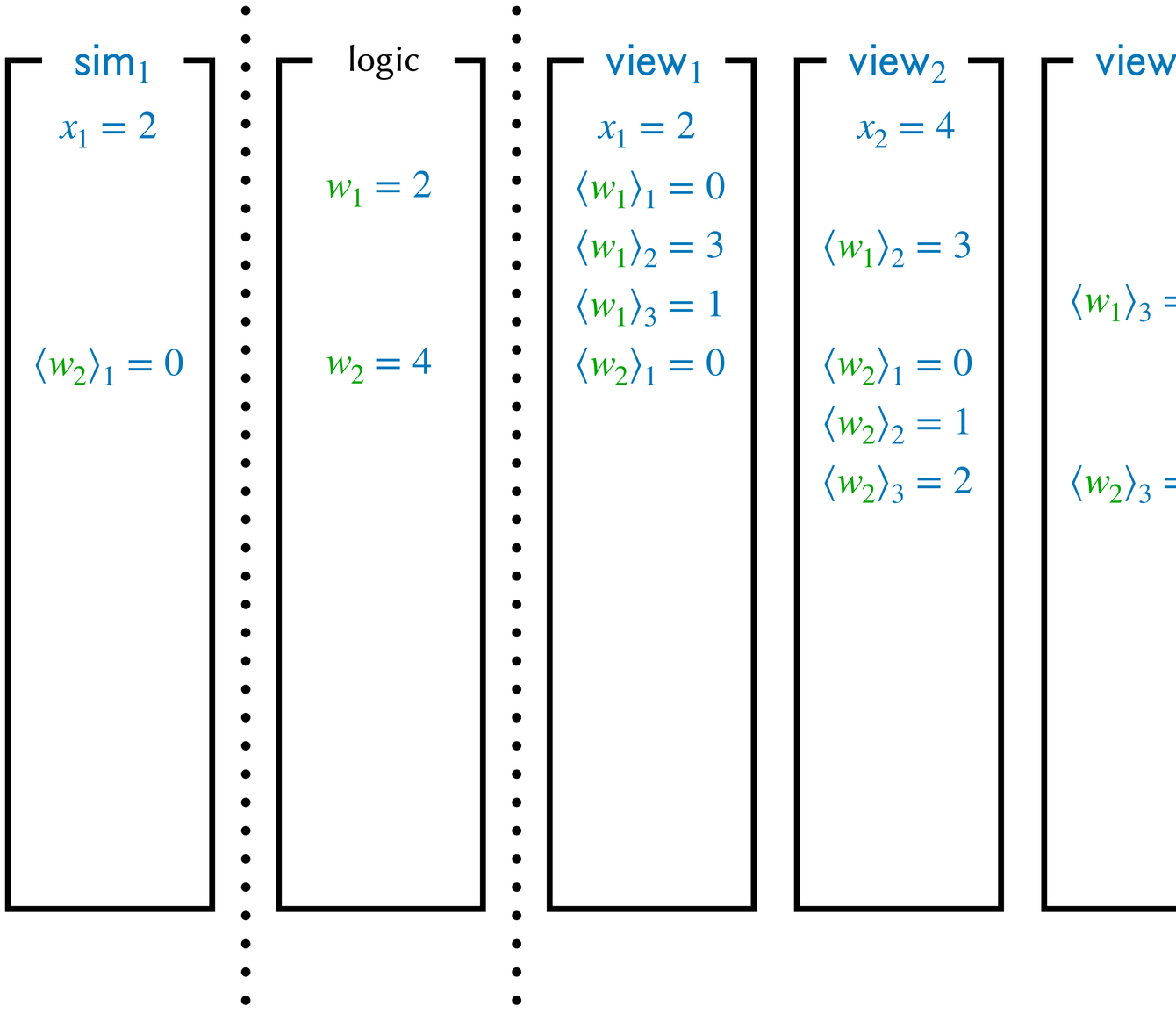
Note: $a_2 = 0 \implies \langle w_2 \rangle_1 = 4$,
$a_2 = 2 \implies \langle w_2 \rangle_1 = 1$,
$a_2 = 3 \implies \langle w_2 \rangle_1 = 2$,
$a_2 = 4 \implies \langle w_2 \rangle_1 = 3$,
Thus $\langle w_2 \rangle_1$ is uniformly distributed in $\mathbb{F}_5$.

Gates: (in,1,1), (in,2,2)

Sim$(\{1\},\{2\},\{2\})$

1. Copy the inputs of the corrupt parties into their views.
2. For every $h \in [n] \backslash I$, find $(\mathsf{in}, h, j) \in C$. For every $i \in I$ sample $\langle w_j \rangle_i \leftarrow \mathbb{F}_p$ and copy this value into the view of corrupted P_i.

Sim samples $\langle w_2 \rangle_1 \leftarrow \mathbb{F}_5$. w.p. 1/5, $\langle w_2 \rangle_1 = 0$



Gates: (in,1,1), (in,2,2)

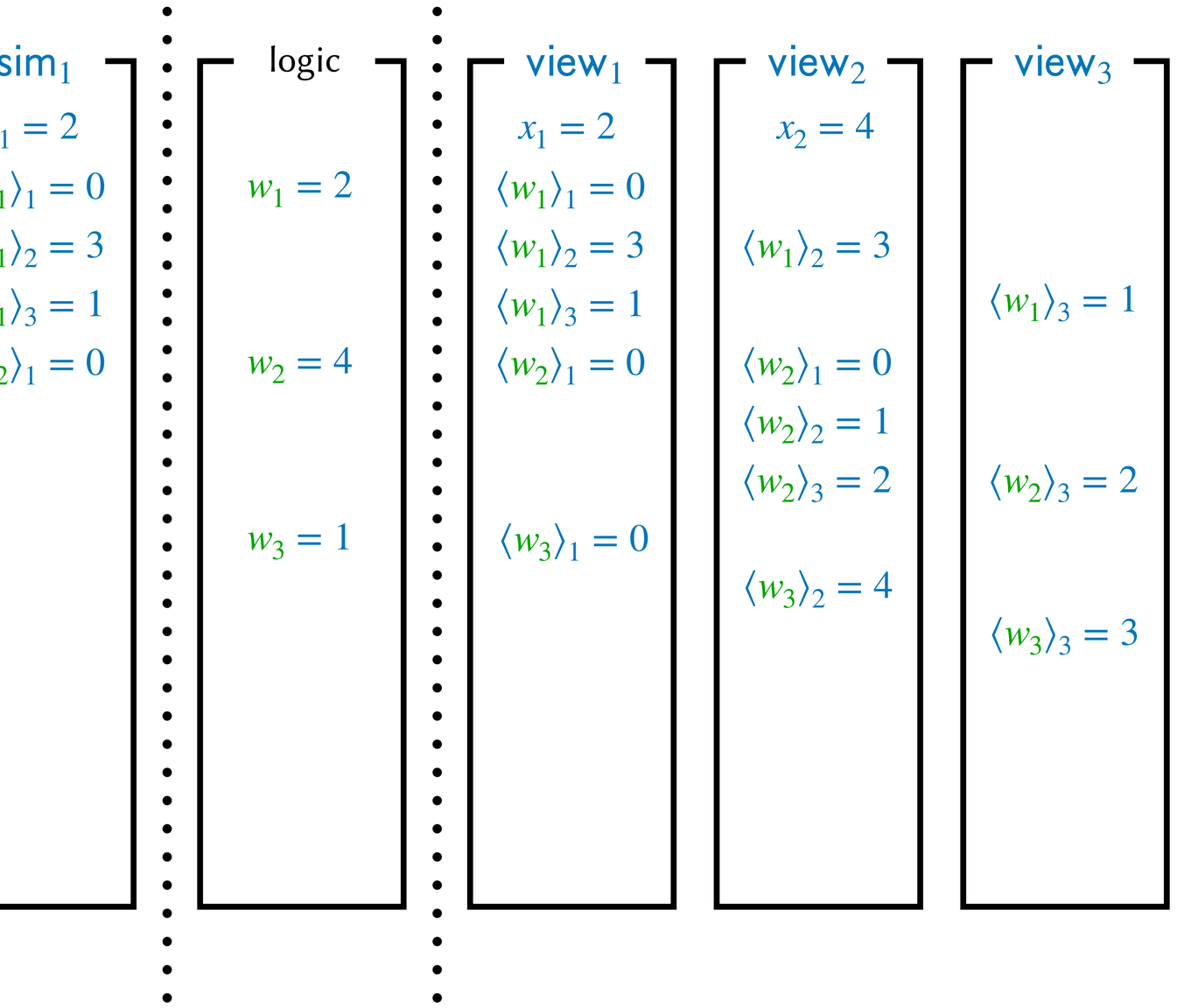
Sim($\{1\},\{2\},\{2\}$)

1. Copy the inputs of the corrupt parties into their views.
2. For every $h \in [n]\backslash I$, find $(\mathsf{in}, h, j) \in C$. For every $i \in I$ sample $\langle w_j \rangle_i \leftarrow \mathbb{F}_p$ and copy this value into the view of corrupted P_i.
3. For every $i \in I$, find $(\mathsf{in}, i, j) \in C$. Sample $\langle w_j \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$. Copy the entirety of $\langle w_j \rangle$ into the view of P_i, and for every $c \in I\backslash\{i\}$, copy $\langle w_j \rangle_c$ into the view of P_c.

Sim samples $a_1 \leftarrow \mathbb{F}_5$ and sets $f_1(x) = a_1 \cdot x + x_1$.
w.p. 1/5, $f_1(x) = 3x + 2 \implies \langle w_1 \rangle = (0,3,1)$

sim_1	logic	view_1	view_2	view
$x_1 = 2$		$x_1 = 2$	$x_2 = 4$	
$\langle w_1 \rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$		
$\langle w_1 \rangle_2 = 3$		$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_2 = 3$	
$\langle w_1 \rangle_3 = 1$		$\langle w_1 \rangle_3 = 1$		$\langle w_1 \rangle_3 =$
$\langle w_2 \rangle_1 = 0$	$w_2 = 4$	$\langle w_2 \rangle_1 = 0$	$\langle w_2 \rangle_1 = 0$	
			$\langle w_2 \rangle_2 = 1$	
			$\langle w_2 \rangle_3 = 2$	$\langle w_2 \rangle_3 =$

Gate: (+,1,2,3)



$\pi_{\mathsf{BGW}}(3,1,5,C)$

1. **Input Sharing:** every P_i with input x_i finds the entry $(\mathsf{in}, i, o) \in C$, computes $\langle w_o \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$ and sends $\langle w_o \rangle_j$ to every P_j for $j \in [n] \setminus \{i\}$.

2. **Circuit Eval:** the parties traverse the circuit C in topological order, jointly evaluating each gate, using shares of its input wires to produce shares of its output wire.

   - If the parties arrive at gate $(+, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$.

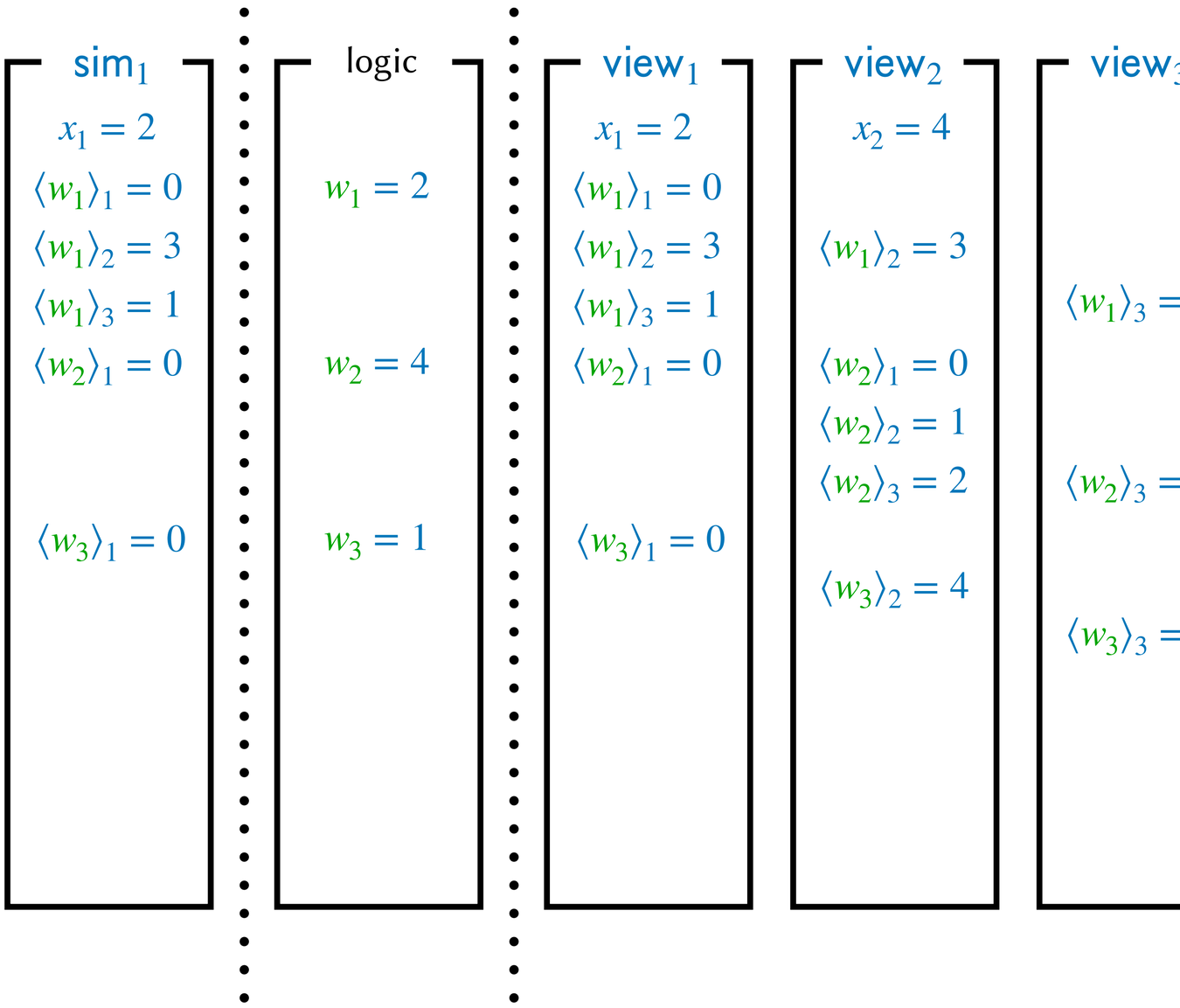
The parties do this deterministic operation locally.

Sim$(\{1\},\{2\},\{2\})$

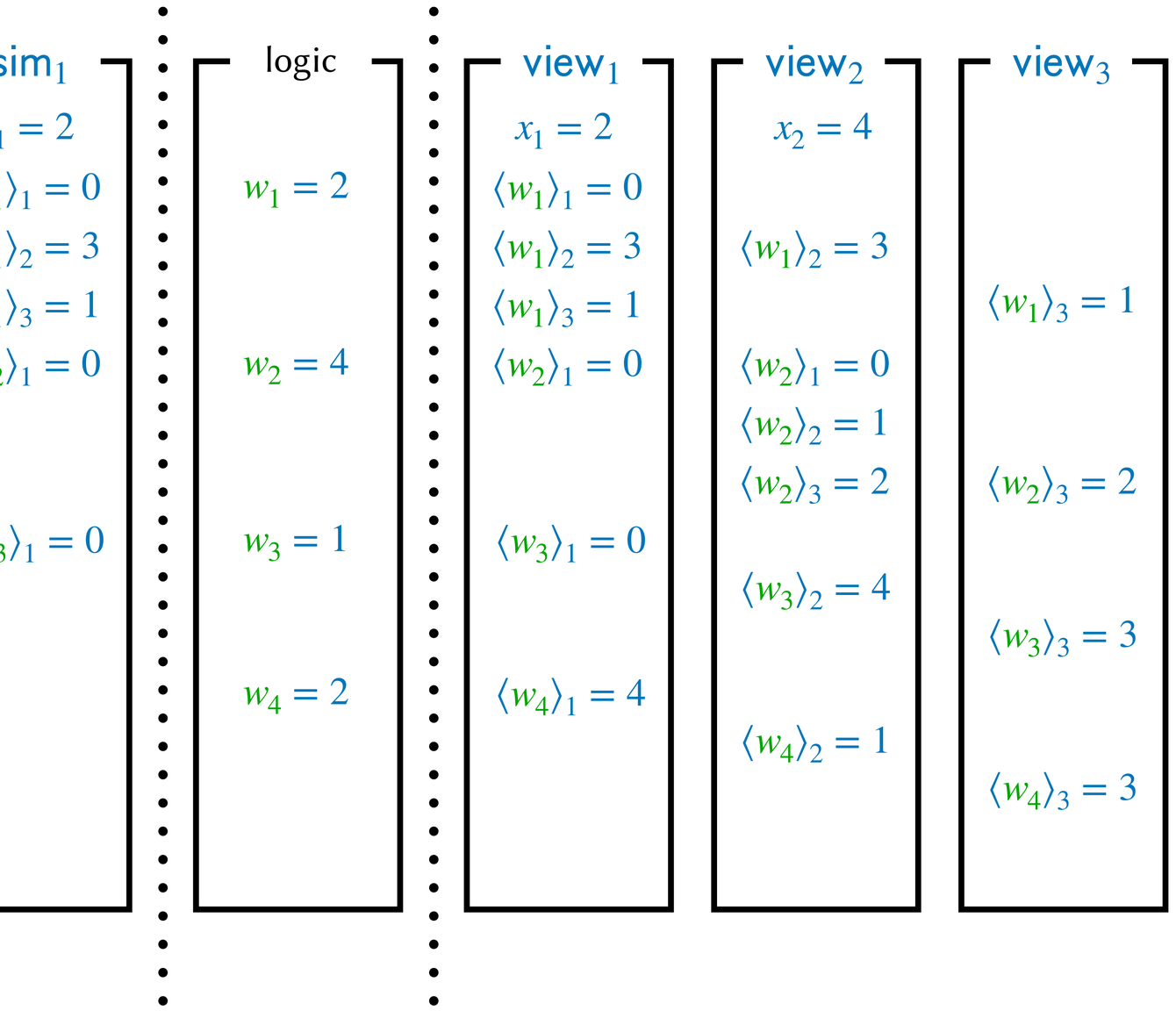
1. Copy the inputs of the corrupt parties into their views.
2. For every $h \in [n]\backslash I$, find $(\mathsf{in}, h, j) \in C$. For every $i \in I$ sample $\langle w_j \rangle_i \leftarrow \mathbb{F}_p$ and copy this value into the view of corrupted P_i.
3. For every $i \in I$, find $(\mathsf{in}, i, j) \in C$. Sample $\langle w_j \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$. Copy the entirety of $\langle w_j \rangle$ into the view of P_i, and for every $c \in I\backslash\{i\}$, copy $\langle w_j \rangle_c$ into the view of P_c.
4. For every $(+, j, k, o) \in C$ and every $i \in I$, compute $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$ and copy this value into the view of corrupted P_i.

Sim does this deterministic operation.

Gate: (+,1,2,3)



Gate: $(\times,1,3,4)$



- If the parties arrive at gate $(+,j,k,o) \in C$. Each party P_i individually computes $\langle w_o\rangle_i := \langle w_j\rangle_i + \langle w_k\rangle_i$.

- If the parties arrive at gate $(\times,j,k,o) \in C$. Each party P_i sends $(\langle w_j\rangle_i, \langle w_k\rangle_i)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o\rangle_i$ in response.

$\mathcal{F}_{\mathsf{mul}}$ recovers w_1, w_3, computes $w_4 := w_1 \cdot w_3$, samples $a_4 \leftarrow \mathbb{F}_5$, and sets $f_4(x) = a_4 \cdot x + w_4$.
w.p. 1/5, $f_4(x) = 2x + 2 \implies \langle w_4\rangle = (4,1,3)$

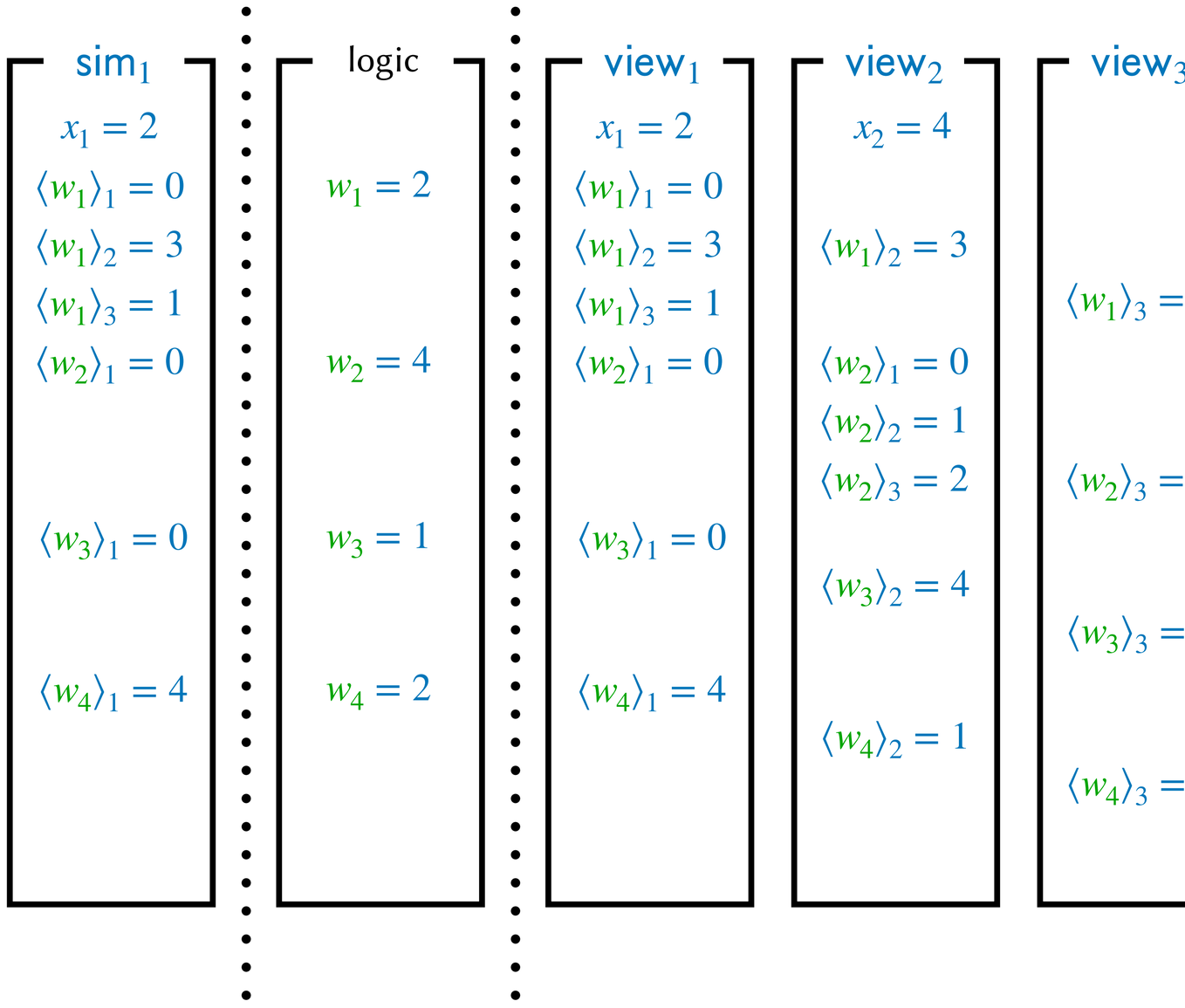
Note: $a_4 = 0 \implies \langle w_4\rangle = (2,2,2)$,
$a_4 = 1 \implies \langle w_4\rangle = (3,4,0)$,
$a_4 = 3 \implies \langle w_4\rangle = (0,3,1)$,
$a_4 = 4 \implies \langle w_4\rangle = (1,0,4)$,
Thus $\langle w_4\rangle_1$ is uniformly distributed in $\mathbb{F}_5$.

4. For every $(+, j, k, o) \in C$ and every $i \in I$, compute $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$ and copy this value into the view of corrupted P_i.

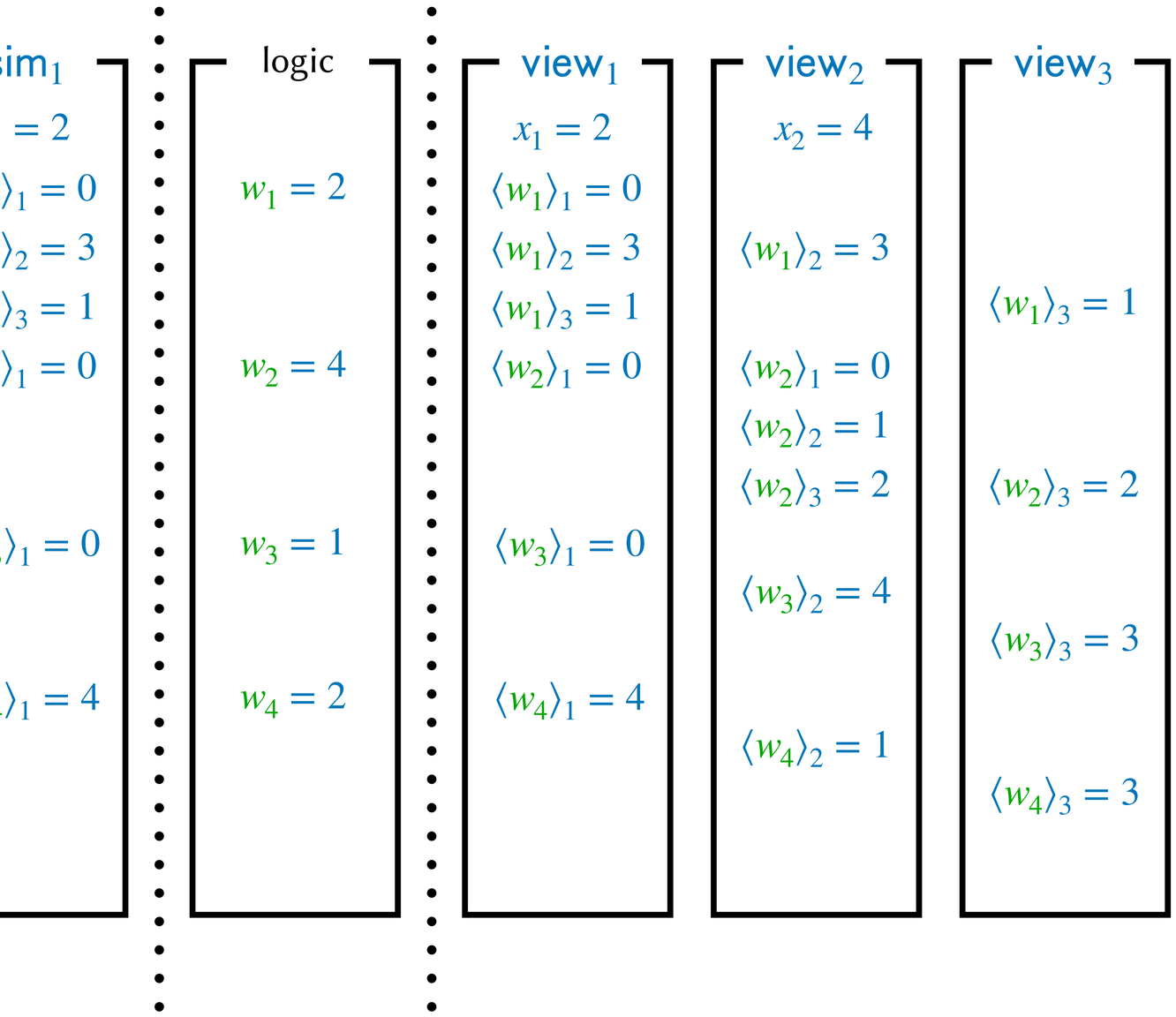
5. For every $(\times, j, k, o) \in C$ and every $i \in I$, sample $\langle w_o \rangle_i \leftarrow \mathbb{F}_p$ and copy this value into the view of corrupted P_i.

Sim samples $\langle w_4 \rangle_1 \leftarrow \mathbb{F}_5$. w.p. 1/5, $\langle w_4 \rangle_1 = 4$

Gate: $(\times,1,3,4)$

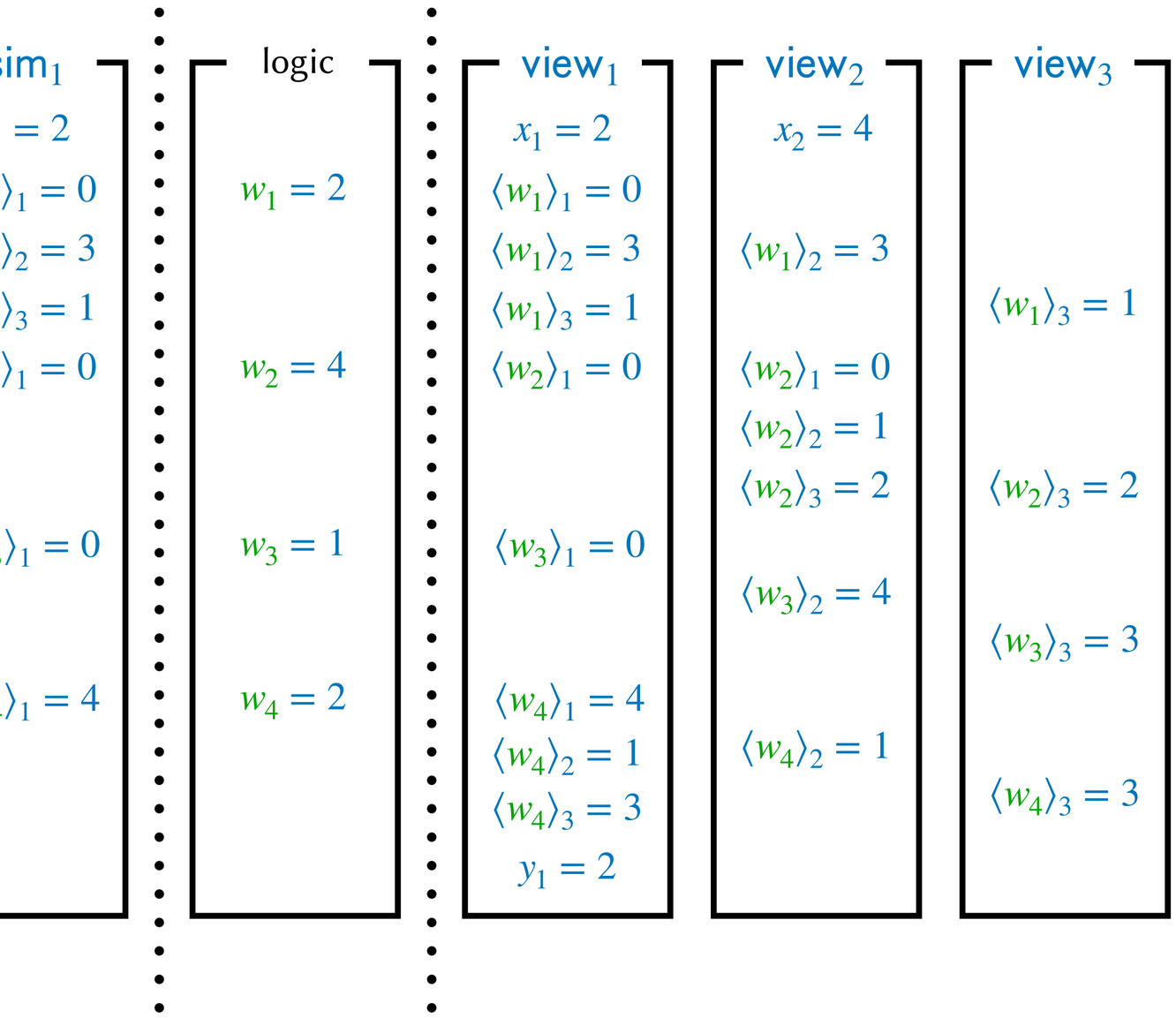


There is no rand Gate



- If the parties arrive at gate $(+, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$.
- If the parties arrive at gate $(\times, j, k, o) \in C$. Each party P_i sends $(\langle w_j \rangle_i, \langle w_k \rangle_i)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o \rangle_i$ in response.
- If the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \mathbb{F}_p$, the parties use a sequence of in gates and $+$ gates to securely compute $\langle w_o \rangle$ such that $w_o = \sum_{k \in [n]} r_{o,k}$.

Gate: (out,1,4)



if the parties arrive at gate $(\times, j, k, o) \in C$. Each party P_i sends $(\langle w_j \rangle_i, \langle w_k \rangle_i)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o \rangle_i$ in response.

- If the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \mathbb{F}_p$, the parties use a sequence of in gates and $+$ gates to securely compute $\langle w_o \rangle$ such that $w_o = \sum_{k \in [n]} r_{o,k}$.

3. **Output Reconstruction:** Each P_i finds every output wire $(\mathsf{out}, k, j) \in C$ and sends $\langle w_j \rangle_i$ to P_k. P_k receives $\langle w_j \rangle$, computes $y_k := \mathsf{Recon}_{p,n,t}([n], \langle w_j \rangle)$, and outputs y_k.

P_2 and P_3 send $\langle w_4 \rangle_2$ and $\langle w_4 \rangle_3$ to P_1.

P_1 recovers $f_4(x) = 2x + 2$ and sets $y_1 := f(0) = 2$.

7. For $i \in I$, find $(\text{out}, i, j) \in C$. Sim received y_i as input, and computed $\langle w_j \rangle_c \ \forall c \in I$ in the previous steps. Sample a degree-t polynomial g uniformly from the set of all degree-t polynomials over $\mathbb{F}_p$ that pass through the points $(0, y_i)$ and $(c, \langle w_j \rangle_c) \ \forall c \in I$. For every $h \in [n] \backslash I$, interpolate $\langle w_j \rangle_h := g(h)$ and copy $\langle w_j \rangle_h$ into the view of P_i.

Sim finds the only degree-1 polynomial g that satisfies $g(0) = y_1$ and $g(1) = \langle w_4 \rangle_1$. Since $\langle w_4 \rangle_1 = 4$ and $y_1 = 2$, we have $g(x) = 2x + 2$ which implies $\langle w_4 \rangle = (4,1,3)$.

Recall that Sim sampled $\langle w_4 \rangle_1 \leftarrow \mathbb{F}_5$ uniformly.

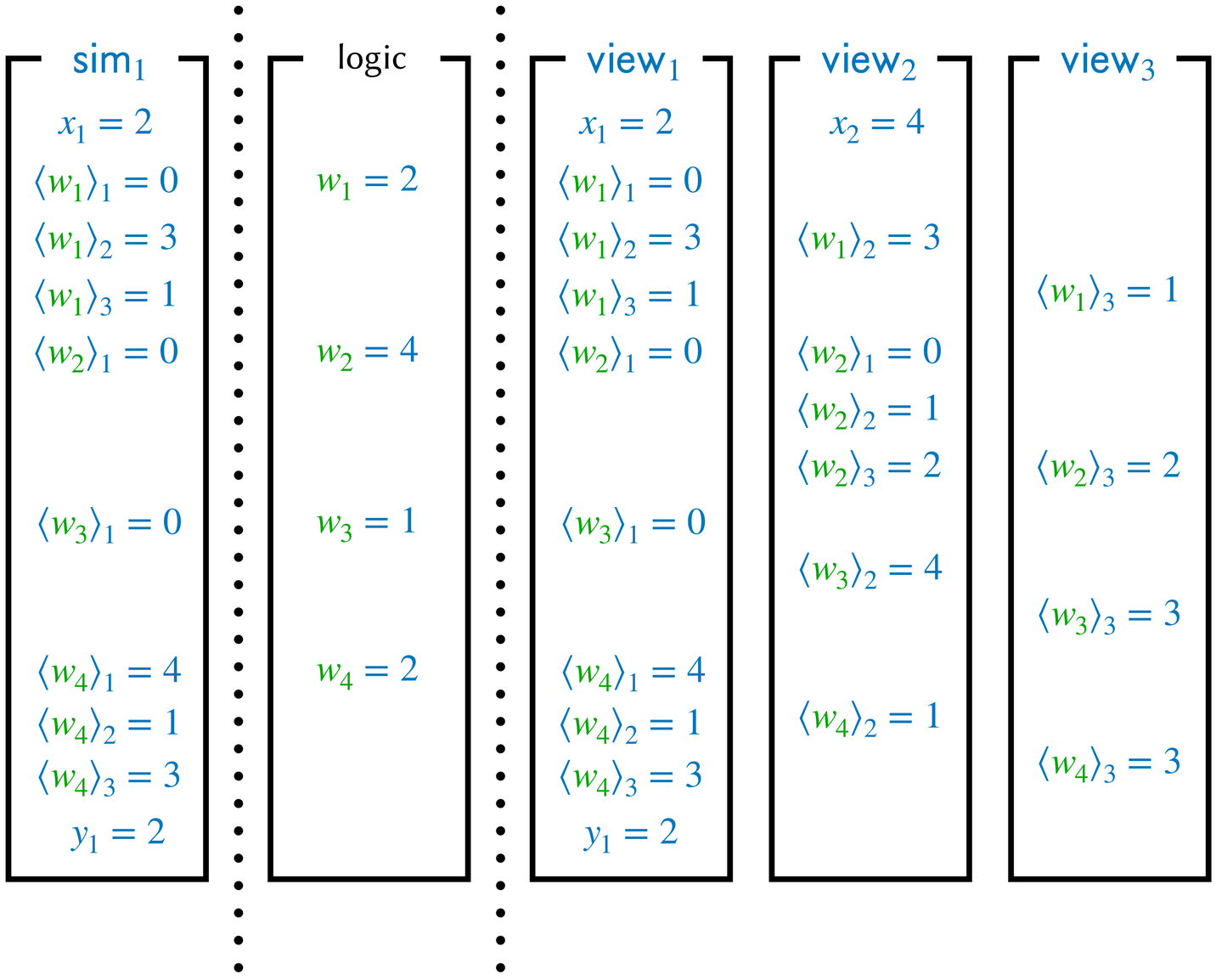
$\langle w_4 \rangle_1 = 0 \implies g(x) = 3x + 2 \implies \langle w_4 \rangle = (0,3,1)$,
$\langle w_4 \rangle_1 = 1 \implies g(x) = 4x + 2 \implies \langle w_4 \rangle = (1,0,4)$,
$\langle w_4 \rangle_1 = 2 \implies g(x) = 0x + 2 \implies \langle w_4 \rangle = (2,2,2)$,
$\langle w_4 \rangle_1 = 3 \implies g(x) = 1x + 2 \implies \langle w_4 \rangle = (3,4,0)$.

Thus the entire sharing $\langle w_4 \rangle$ is distributed identically to its the real world counterpart

Gate: (out,1,4)

sim_1	logic	view_1	view_2	view_3
$x_1 = 2$		$x_1 = 2$	$x_2 = 4$	
$\langle w_1 \rangle_1 = 0$	$w_1 = 2$	$\langle w_1 \rangle_1 = 0$		
$\langle w_1 \rangle_2 = 3$		$\langle w_1 \rangle_2 = 3$	$\langle w_1 \rangle_2 = 3$	
$\langle w_1 \rangle_3 = 1$		$\langle w_1 \rangle_3 = 1$		$\langle w_1 \rangle_3 =$
$\langle w_2 \rangle_1 = 0$	$w_2 = 4$	$\langle w_2 \rangle_1 = 0$	$\langle w_2 \rangle_1 = 0$	
			$\langle w_2 \rangle_2 = 1$	
			$\langle w_2 \rangle_3 = 2$	$\langle w_2 \rangle_3 =$
$\langle w_3 \rangle_1 = 0$	$w_3 = 1$	$\langle w_3 \rangle_1 = 0$		
			$\langle w_3 \rangle_2 = 4$	
				$\langle w_3 \rangle_3 =$
$\langle w_4 \rangle_1 = 4$	$w_4 = 2$	$\langle w_4 \rangle_1 = 4$		
$\langle w_4 \rangle_2 = 1$		$\langle w_4 \rangle_2 = 1$	$\langle w_4 \rangle_2 = 1$	
$\langle w_4 \rangle_3 = 3$		$\langle w_4 \rangle_3 = 3$		$\langle w_4 \rangle_3 =$
$y_1 = 2$		$y_1 = 2$		

Final Views at Experiment End



BGW Protocol Round Costs

- Recall: a round can only contain communication between parties, or a single invocation of $\mathcal{F}_{\mathsf{mul}}$, but not both.
- We can't have multiple *simultaneous* invocations of $\mathcal{F}_{\mathsf{mul}}$ in our model, but we can define a *single* functionality $\mathcal{F}_{\mathsf{mul}}^{m}$ that multiplies m shared secrets at once, and realize it in a similar way to the regular $\mathcal{F}_{\mathsf{mul}}$ (we will see this next). Thus we can evaluate many multiplication gates at once (but not at the same time as other things).
- We can evaluate all input gates at once, and load all random values at the same time (in advance of when they're needed). We can also evaluate all output gates at once.
- If we organize the circuit C into *layers* that alternate between containing only $\times$ gates and containing only $+$ and rand gates, and we use the multiplication protocols we will see later to realize $\mathcal{F}_{\mathsf{mul}}^{m}$, then we get a total round count of $\Omega(d)$ where d is the *multiplicative depth* of C.

3. Realize $\mathscr{F}_{\text{mul}}$

Reminder: Multiplying Shamir-Shared Values

A First Idea: we have $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$, and we need $g \leftarrow \mathscr{P}_{p,t,w\cdot w'}$. Can we set $g := f \cdot f'$?

This set contains degree $\leq t$

Consider the entire polynomial:

$$g(x) = f(x) \cdot f'(x) = \left(w + \sum_{i\in[t]} a_i \cdot x^i\right) \cdot \left(w' + \sum_{i\in[t]} a'_i \cdot x^i\right)$$

Actual degree could be up to $2t$

$$= w \cdot w' + w \cdot \sum_{i\in[t]} a'_i \cdot x^i + w' \cdot \sum_{i\in[t]} a_i \cdot x^i + \sum_{i,j\in[t]} a_i \cdot a'_j \cdot x^{i+j}$$

There are two problems with g.

Problem #1: This polynomial is degree $2t$. i.e. $g \in \mathscr{P}_{p,2t,w\cdot w'}$.

Problem #2: This polynomial is not *uniformly distributed* in $\mathscr{P}_{p,2t,w\cdot w'}$.

How to Get $g \leftarrow \mathscr{P}_{p,t,w \cdot w'}$?

A Second Idea, given $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$:

1. Compute $\hat{g} := f \cdot f' \in \mathscr{P}_{p,2t,w \cdot w'}$.
   Since $\hat{g}$ is not uniformly distributed, so we cannot safely reconstruct it.
   Since $\text{degree}(\hat{g}) = 2t$, we need $2t + 1$ points in order to completely represent it.

   *We **must** have $n > 2t$ or else we lose correctness!*

2. Jointly *rerandomize* $\hat{g}$ to a new polynomial $\tilde{g} \leftarrow \mathscr{P}_{p,2t,w \cdot w'}$. If we can achieve this, we solve **Problem #2**, and reconstruction is safe. The degree is still too high though (multiplying by another sharing will cause a correctness violation).

3. Reduce the degree of $\tilde{g}$ to obtain another polynomial $g \leftarrow \mathscr{P}_{p,t,w \cdot w'}$. This solves **Problem #1** and finally gives us shares that we can use for future gates!

4. All of the above operations must be performed without knowledge of the coefficients, using only the individual points on each of the polynomials.

Rerandomizing $\hat{g}$ to Obtain $\tilde{g} \leftarrow \mathscr{P}_{p,2t,w \cdot w'}$.

Observations:

1. Any function that takes an input in $\mathbb{F}_p$ and adds another fixed value in $\mathbb{F}_p$ is bijective. This implies that vector addition with a fixed value in $\mathbb{F}_p^{2t}$ is bijective.
2. $|\mathscr{P}_{p,2t,x}| = p^{2t}$ for every $x \in \mathbb{F}_p$.
3. Since any member of $\mathscr{P}_{p,2t,x}$ for fixed x can be represented using $2t$ values in $\mathbb{F}_p^{2t}$ (the y-coordinates at any $2t$ distinct locations on the x-axis), and polynomials can be added by adding their points, the function that takes a member of $\mathscr{P}_{p,2t,x}$ and adds a fixed member of $\mathscr{P}_{p,2t,y}$ is bijective onto $\mathscr{P}_{p,2t,x+y}$.
4. Thus, if we sample a uniform $h \leftarrow \mathscr{P}_{p,2t,0}$ that is not known to anyone and add any fixed $\hat{g} \in \mathscr{P}_{p,2t,w \cdot w'}$ to it, we get $\tilde{g} \leftarrow \mathscr{P}_{p,2t,w \cdot w'}$ that is uniformly distributed as we need!

Rerandomizing $\hat{g}$ to Obtain $\tilde{g} \leftarrow \mathscr{P}_{p,2t,w \cdot w'}$.

Observations:

5. How to sample a uniform $h \leftarrow \mathscr{P}_{p,2t,0}$ that is not known to anyone, and then distribute $\langle 0 \rangle_i := h(i)$ to every P_i for $i \in [n]$?

   Let every P_i for $i \in [n]$ sample $\langle 0_i \rangle \leftarrow \mathsf{Share}_{p,n,2t}(0)$ and send $\langle 0_i \rangle_j$ to every P_j for $j \in [n] \setminus \{i\}$.

   Then every P_i computes $\langle 0 \rangle_i = \sum_{k \in [n]} \langle 0_k \rangle_i$.

Degree Reduction of $\tilde{g}$ to Get $g \in \mathscr{P}_{p,t,w\cdot w'}$.

Observations:

1. Input of P_i: $\widetilde{\langle w \cdot w' \rangle}_i = \tilde{g}(i)$ where $\tilde{g} \in \mathscr{P}_{p,2t,w\cdot w'}$ can be written in the form:
$$\tilde{g}(x) = w \cdot w' + a_1 \cdot x + \ldots + a_t \cdot x^t + \ldots + a_{2t} \cdot x^{2t}.$$
2. If $\tilde{g}$ is uniformly distributed in $\mathscr{P}_{p,2t,w\cdot w'}$ (i.e. it has uniform coefficients $a_1, \ldots, a_t, \ldots, a_{2t}$), then $g(x) = w \cdot w' + a_1 \cdot x + \ldots + a_t \cdot x^t$ is uniform in $\mathscr{P}_{p,t,w\cdot w'}$.
3. So it suffices for each party P_i to output $\langle w \cdot w' \rangle_i = g(i)$. How can they compute it?

Definition 1 (Vandermonde Matrix): Given $\vec{i} \in \mathbb{N}^*$ with pairwise distinct entries such that $|\vec{i}| = \ell$, the Vandermonde matrix $V_{\vec{i}} \in M_{|\vec{i}|}(\mathbb{F}_p)$ is given by

$$V_{\vec{i}} = \begin{bmatrix} 1 & i_1 & \cdots & i_1^{\ell-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & i_\ell & \cdots & i_\ell^{\ell-1} \end{bmatrix}. \text{ Thus we have } V_{[n]} = \begin{bmatrix} 1 & 1 & \cdots & 1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & n & \cdots & n^{n-1} \end{bmatrix}.$$

Degree Reduction of $\tilde{g}$ to Get $g \in \mathscr{P}_{p,t,w\cdot w'}$.

4. Recall that given $a_1, \ldots, a_t, \ldots, a_{2t}$, we can express $\mathsf{Share}_{p,n,t}(w \cdot w')$ in matrix form as

$$\begin{bmatrix} \langle w \cdot w' \rangle_1 \\ \langle w \cdot w' \rangle_2 \\ \vdots \\ \langle w \cdot w' \rangle_n \end{bmatrix} := V_{[n]} \begin{bmatrix} w \cdot w' \\ a_1 \\ \vdots \\ a_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Output shares

Similarly, $\mathsf{Share}_{p,n,2t}(w \cdot w')$ is

$$\begin{bmatrix} \langle \widetilde{w \cdot w'} \rangle_1 \\ \langle \widetilde{w \cdot w'} \rangle_2 \\ \vdots \\ \langle \widetilde{w \cdot w'} \rangle_n \end{bmatrix} := V_{[n]} \begin{bmatrix} w \cdot w' \\ a_1 \\ \vdots \\ a_{2t} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Input shares

Which implies

$$\begin{bmatrix} w \cdot w' \\ a_1 \\ \vdots \\ a_{2t} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = V_{[n]}^{-1} \begin{bmatrix} \langle \widetilde{w \cdot w'} \rangle_1 \\ \langle \widetilde{w \cdot w'} \rangle_2 \\ \vdots \\ \langle \widetilde{w \cdot w'} \rangle_n \end{bmatrix}$$

...so we just need a secure way to *truncate* the coefficient vector.

Degree Reduction of $\tilde{g}$ to Get $g \in \mathscr{P}_{p,t,w\cdot w'}$.

5. This kind of truncation can be achieved by defining a *projection matrix*:

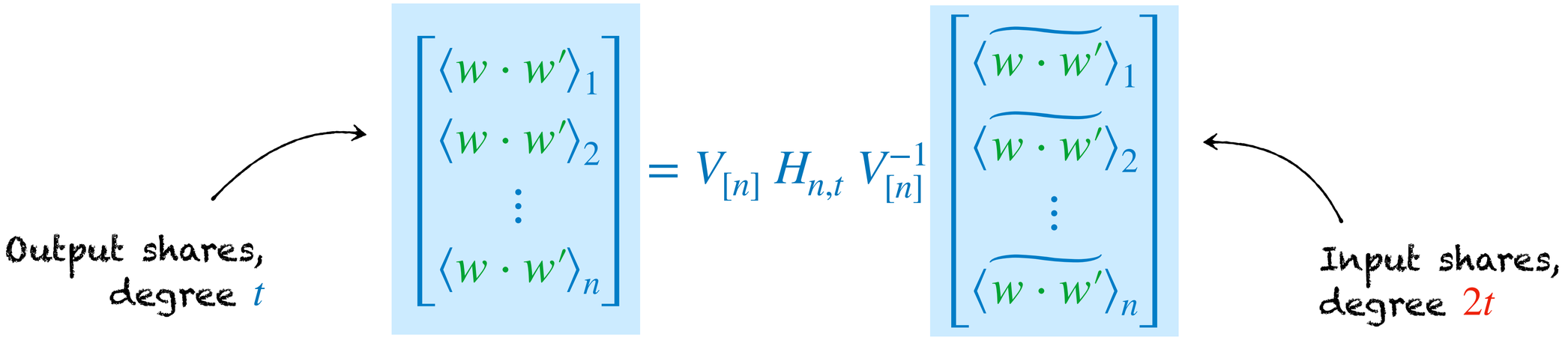
$$H_{n,t} := \begin{bmatrix} \overbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}^{t+1} & \\ & \overbrace{\begin{matrix} 0 & & \\ & \ddots & \\ & & 0 \end{matrix}}^{n-t-1} \end{bmatrix}$$

... and then multiplying by it

$$\begin{bmatrix} w \cdot w' \\ a_1 \\ \vdots \\ a_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} = H_{n,t} \begin{bmatrix} w \cdot w' \\ a_1 \\ \vdots \\ a_{2t} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Degree Reduction of $\tilde{g}$ to Get $g \in \mathscr{P}_{p,t,w\cdot w'}$.

6. So, finally, putting the pieces together, we have:



7. This is a *deterministic*, *linear* n-ary function on $\mathbb{F}_p$ where every party knows one input and receives one output. *What can we do?*
8. We can use BGW for deterministic linear functions (which we saw in **Lecture 7**) to securely compute it!

The BGW Multiplication Protocol:

Inputs: Each P_i begins with $\langle w \rangle_i$ and $\langle w' \rangle_i$.

1. Without interacting, every P_i computes $\langle \widehat{w \cdot w'} \rangle_i = \langle w \rangle_i \cdot \langle w' \rangle_i$.
2. Every P_i samples $\langle 0_i \rangle \leftarrow \mathsf{Share}_{p,n,2t}(0)$ and sends $\langle 0_i \rangle_j$ to every P_j for $j \in [n] \backslash \{i\}$.
3. Every P_i computes $\langle \widetilde{w \cdot w'} \rangle_i = \langle \widehat{w \cdot w'} \rangle_i + \sum_{k \in [n]} \langle 0_k \rangle_i$.
4. The parties invoke $\mathscr{F}_{\mathsf{SFE}}(n, f_{\mathsf{reduce}}, \mathbb{F}_p, \ldots, \mathbb{F}_p)$ where $f_{\mathsf{reduce}}(\vec{x}) = V_{[n]}\, H_{n,t}\, V_{[n]}^{-1}\, \vec{x}$. Each P_i supplies $\langle \widetilde{w \cdot w'} \rangle_i$ as its input and receives $\langle w \cdot w' \rangle_i$ as its output.
5. Because f_{reduce} is *linear,* we can realize $\mathscr{F}_{\mathsf{SFE}}(n, f_{\mathsf{reduce}}, \mathbb{F}_p, \ldots, \mathbb{F}_p)$ using the BGW protocol *without* any multiplication gates.

Outputs: Each P_i ends with $\langle w \cdot w' \rangle_i$.

Total bandwidth cost: n inputs + n outputs + n zero-sharings $= 3n^2|p|$.
Total Rounds: 3.

3b. A Simpler Protocol for $\mathcal{F}_{\text{mul}}$ (Gennaro-Rabin-Rabin Multiplication)

Here is our Degree Reduction Formula for $\tilde{g}$

$$\begin{bmatrix} \langle w \cdot w' \rangle_1 \\ \langle w \cdot w' \rangle_2 \\ \vdots \\ \langle w \cdot w' \rangle_n \end{bmatrix} = V_{[n]} \, H_{n,t} \, V_{[n]}^{-1} \begin{bmatrix} \widetilde{\langle w \cdot w' \rangle}_1 \\ \widetilde{\langle w \cdot w' \rangle}_2 \\ \vdots \\ \widetilde{\langle w \cdot w' \rangle}_n \end{bmatrix}$$

Input shares, degree $2t$

Linear wrt shares; inversion of public matrix is computationally costly

Remember, we also have to do rerandomization!

Output shares, degree t

$$= \begin{bmatrix} 1^1 & \cdots & 1^{n-1} \\ \vdots & \ddots & \vdots \\ n^1 & \cdots & n^{n-1} \end{bmatrix} \begin{bmatrix} \overbrace{1 & & & & \\ & \ddots & & & \\ & & 1 & & }^{t+1} \overbrace{& & \\ & 0 & & \\ & & \ddots & \\ & & & 0 }^{n-t-1} \end{bmatrix} \begin{bmatrix} 1^1 & \cdots & 1^{n-1} \\ \vdots & \ddots & \vdots \\ n^1 & \cdots & n^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} \widetilde{\langle w \cdot w' \rangle}_1 \\ \widetilde{\langle w \cdot w' \rangle}_2 \\ \vdots \\ \widetilde{\langle w \cdot w' \rangle}_n \end{bmatrix}$$

Let's take a step back and think...

Multiplication: we have $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$, and we need $g \leftarrow \mathscr{P}_{p,t,w \cdot w'}$.

We need 2 things from g:

1. To achieve privacy, it must be uniform among polynomials of degree t.
2. To achieve correctness, it must have degree t and $g(0) = w \cdot w'$.

What we have from $\hat{g} = f \cdot f'$:

3. Although $\hat{g}$ isn't uniform and it has the wrong degree, it *does* encode the right value (i.e. $\hat{g}(0) = w \cdot w'$). If we define the n Lagrange bases $\ell_i(x) \forall i \in [n]$ with respect to the x-coordinates $[n]$, then we have

$$w \cdot w' = \sum_{i \in [n]} \ell_i(0) \cdot \hat{g}(i)$$

Observation: looking at it another way, we are trying to find a degree-t Shamir sharing of *this* equation. *Is there an easy way using tools we know?*

Shares of Shares?

Suppose we wanted to securely compute $y = \sum_{i \in [n]} c_i \cdot x_i$ where each party P_i for $i \in [n]$ has input $x_i \in \mathbb{F}_p$ and the c_i are public constants. *How would we do it?*

Using the pieces of the BGW protocol:

1. Every P_i samples $\langle x_i \rangle \leftarrow \mathsf{Share}_{p,n,t}(x_i)$ and sends $\langle x_i \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$
2. Every P_i computes $\langle y \rangle_i = \sum_{j \in [n]} c_j \cdot \langle x_j \rangle_i$. This ls linear, and thus local.
3. The parties reconstruct y from $\langle y \rangle$.

Notice:

- The security of this protocol has nothing to do with the distribution of the x_i values. They can be *completely arbitrary*, and yet $\langle y \rangle$ is a *uniform* sharing.
- If we let $c_i = \ell_i(0)$ and $x_i = \hat{g}(i)$ then $\langle y \rangle$ is a *uniform* degree-t sharing of $\hat{g}(0)$!

Putting it Together: $\pi_{\mathsf{GRR}}(n, t, p)$.

Inputs: Every P_i for $i \in [n]$ has input $\langle w \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w}$ and input $\langle w' \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w'}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w \rangle_i \cdot \langle w' \rangle_i$. Note that $\hat{g} \in \mathscr{P}_{p,2t,w \cdot w'}$.
2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \setminus \{i\}$.
3. Every P_i computes $\langle w \cdot w' \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$.

Outputs: Each P_i ends with $\langle w \cdot w' \rangle_i$.

Total bandwidth cost: $n^2 |p|$.
Total Rounds: 1.
Computation is also less than BGW mul!

Another Look at $\pi_{\mathsf{GRR}}(n, t, p)$.

	Local Values/Ops	Values Transmitted By				
		P_1	$\cdots$	P_i	$\cdots$ P_n	
Polynomial	$f(x) \cdot f'(x) =: \hat{g}(x)$					$g(x)$
Encoded Value	$w \cdot w' =: w \cdot w'$	$\ell_1(0) \cdot \hat{g}(1)$	$\cdots + \ell_i(0) \cdot \hat{g}(i)$		$\cdots + \ell_n(0) \cdot \hat{g}(n)$	$= w \cdot w'$
Share of P_1	$\langle w \rangle_1 \cdot \langle w' \rangle_1 =: \hat{g}(1)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_1$	$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_1$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_1$	$= \langle w \cdot w' \rangle_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Share of P_i	$\langle w \rangle_i \cdot \langle w' \rangle_i =: \hat{g}(i)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_i$	$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_i$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_i$	$= \langle w \cdot w' \rangle_i$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Share of P_n	$\langle w \rangle_n \cdot \langle w' \rangle_n =: \hat{g}(n)$	$\ell_1(0) \cdot \langle \hat{g}(1) \rangle_n$	$\cdots + \ell_i(0) \cdot \langle \hat{g}(i) \rangle_n$		$\cdots + \ell_n(0) \cdot \langle \hat{g}(n) \rangle_n$	$= \langle w \cdot w' \rangle_n$

Security Proof for $\pi_{\mathsf{GRR}}(n, t, p)$.

Lemma 2: Let $2t < n < p$. Assuming synchrony and secure point-to-point channels, $\pi_{\mathsf{GRR}}(n, t, p)$ perfectly realizes $\mathscr{F}_{\mathsf{mul}}(\mathbb{F}_p)$ in the presence of a semi-honest adversary that statically corrupts up to t parties.

Pf Sketch: because $\mathscr{A}$ is semi-honest we can use our simplified security definition.

Recall that $\mathscr{F}_{\mathsf{mul}}$ securely computes $\mathsf{mul}((x_1, y_1), \ldots, (x_n, y_n)) = (z_1, \ldots, z_n)$ where $\vec{z} \leftarrow \mathsf{Share}_{p,n,t}(\mathsf{Recon}_{p,n,t}([n], x_1, \ldots, x_n)) \cdot \mathsf{Recon}_{p,n,t}([n], y_1, \ldots, y_n)))$.

We begin by enumerating the view of the corrupt parties, and then we will define the algorithm Sim. Afterward we will prove that for every $I \subseteq [n]$ of size $|I| \leq t$ and every input vector $((x_1, y_1), \ldots, (x_n, y_n))$ such that $(x_1, \ldots, x_n) \in \mathrm{image}(\mathsf{Share}_{p,n,t})$ and $(y_1, \ldots, y_n) \in \mathrm{image}(\mathsf{Share}_{p,n,t})$ it holds that

$$\left(\mathsf{Sim}\left(I, (\vec{x}_I, \vec{y}_I), \mathsf{mul}_I((x_1, y_1), \ldots, (x_n, y_n))\right), \mathsf{mul}((x_1, y_1), \ldots, (x_n, y_n))\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_{\pi_{\mathsf{GRR}}})$$

Simulated View

$\mathsf{Sim}(I, \langle w \rangle_I, \langle w' \rangle_I, \langle w \cdot w' \rangle_I)$ for $\pi_{\mathsf{GRR}}(n, t, p)$

Inputs: Copy the inputs of the corrupt parties into their views.

1. For every $i \in I$, compute $\hat{g}(i) := \langle w \rangle_i \cdot \langle w' \rangle_i$ and copy this value into the view of corrupted P_i.
2. For $i \in I$, sample $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$, and for $h \in [n] \backslash I$, sample $\langle \hat{g}(h) \rangle_i \leftarrow \mathbb{F}_p$, subject to the *restriction* that $\langle w \cdot w' \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$
3. Copy the outputs of the corrupt parties into their views.

Real-World VIEW_I

$\pi_{\mathsf{GRR}}(n, t, p)$

Inputs: Every P_i for $i \in [n]$ has input $\langle w \rangle_i = f(i)$ where $f \in \mathscr{P}_{p,t,w}$ and input $\langle w' \rangle_i = f'(i)$ where $f' \in \mathscr{P}_{p,t,w'}$.

1. Every P_i locally computes $\hat{g}(i) = f(i) \cdot f'(i) = \langle w \rangle_i \cdot \langle w' \rangle_i$.
2. Every P_i samples $\langle \hat{g}(i) \rangle \leftarrow \mathsf{Share}_{p,n,t}(\hat{g}(i))$ and sends $\langle \hat{g}(i) \rangle_j$ to P_j for $j \in [n] \backslash \{i\}$.
3. Every P_i computes $\langle w \cdot w' \rangle_i = \sum_{j \in [n]} \ell_j(0) \cdot \langle \hat{g}(j) \rangle_i$.

Outputs: Each P_i ends with $\langle w \cdot w' \rangle_i$.

Argument for Perfect Simulation

- The inputs of the corrupt parties and all values they compute prior to interacting are distributed identically in the real and ideal worlds, because they are computed the same way in both.

- In the real world, the shares dealt by honest parties are distributed uniformly within their domain (from $\mathscr{A}$'s point of view) by the privacy property of Shamir sharing. It follows that $\langle w \cdot w' \rangle_I$ is distributed uniformly in $\mathbb{F}_p^{|I|}$.

- In the ideal world, $\langle w \cdot w' \rangle_I$ is distributed uniformly in $\mathbb{F}_p^{|I|}$ because mul samples these values as Shamir shares of degree $\geq |I|$, and the simulator "programs" $\langle w \cdot w' \rangle_I$ to be the values that mul produced.

- In the ideal world, the shares dealt by honest parties are sampled uniformly (from $\mathscr{A}$'s point of view) from the set of possible shares that yield the correct value of $\langle w \cdot w' \rangle_I$. Taking this together with the above two points, we can conclude that the honest parties shares are identically distributed to their real-world counterparts.

Argument for Perfect Simulation

- In the ideal world, $\langle w \cdot w' \rangle_I$ is distributed uniformly in $\mathbb{F}_p^{|I|}$ because mul samples these values as Shamir shares of degree $\geq |I|$, and the simulator "programs" $\langle w \cdot w' \rangle_I$ to be the values that mul produced.

- In the ideal world, the shares dealt by honest parties are sampled uniformly (from $\mathscr{A}$'s point of view) from the set of possible shares that yield the correct value of $\langle w \cdot w' \rangle_I$. Taking this together with the above two points, we can conclude that the honest parties shares are identically distributed to their real-world counterparts.

- Output consistency is guaranteed by the fact that the protocol is correct (as we saw by example a few slides ago), and the simulator "programs" $\langle w \cdot w' \rangle_I$ to be the values that mul produced.

Argument for Perfect Simulation

- In the ideal world, the shares dealt by honest parties are sampled uniformly (from $\mathscr{A}$'s point of view) from the set of possible shares that yield the correct value of $\langle w \cdot w' \rangle_I$. Taking this together with the above two points, we can conclude that the honest parties shares are identically distributed to their real-world counterparts.
- Output consistency is guaranteed by the fact that the protocol is correct (as we saw by example a few slides ago), and the simulator "programs" $\langle w \cdot w' \rangle_I$ to be the values that mul produced.

Thus we have

$$\left(\mathsf{Sim}\left(I, (\vec{x}_I, \vec{y}_I), \mathsf{mul}_I((x_1, y_1), \ldots, (x_n, y_n))\right), \mathsf{mul}((x_1, y_1), \ldots, (x_n, y_n))\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_{\pi_{\mathsf{GRR}}})$$

CS4501 Cryptographic Protocols
Lecture 9: BGW Example, Multiplication Protocols

https://jackdoerner.net/teaching/#2026/Spring/CS4501