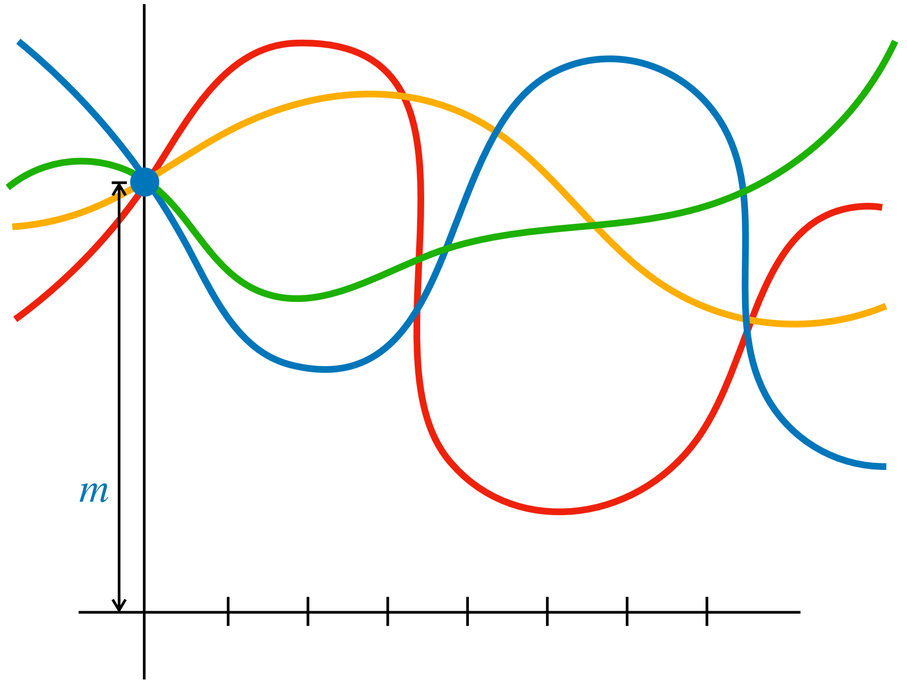
CS4501 Cryptographic Protocols
Lecture 8: The BGW Protocol

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

Recap: Polynomials over $\mathbb{F}_p$

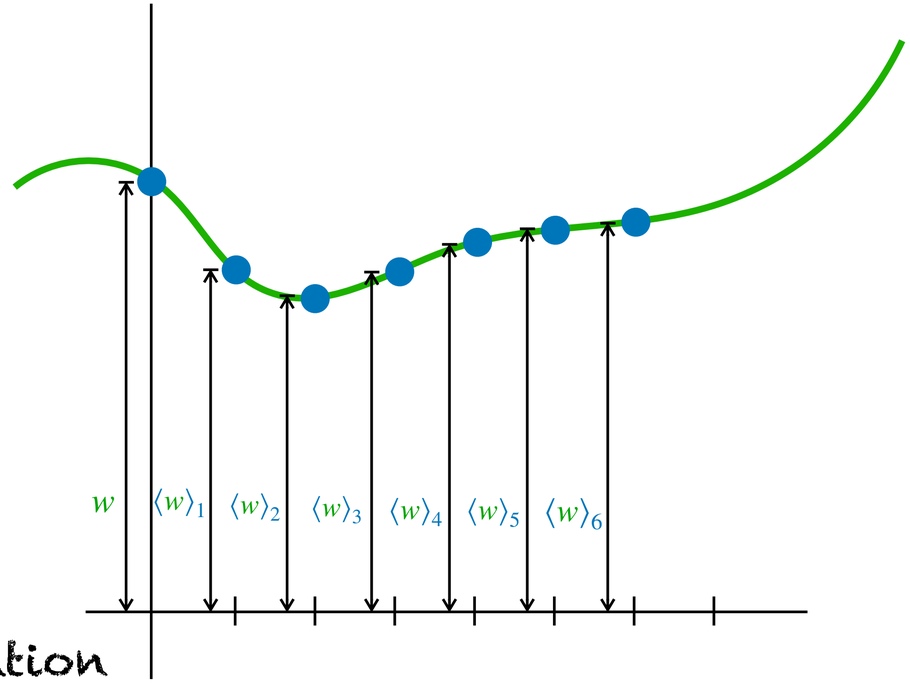
- Let $0 \leq t < p$ where p is a prime.
- Let $\mathscr{P}_{p,t,w} = \{f \in \mathbb{F}_p[x] : \deg(f) \leq t \wedge f(0) = w\}$.
- Every $f \in \mathscr{P}_{p,t,w}$ is of the form
  $f(x) = w + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_t \cdot x^t$.
  where $a_1, \ldots, a_t \in \mathbb{F}_p$.



Recap: $(t+1)$-of-n Shamir Sharing over $\mathbb{F}_p$

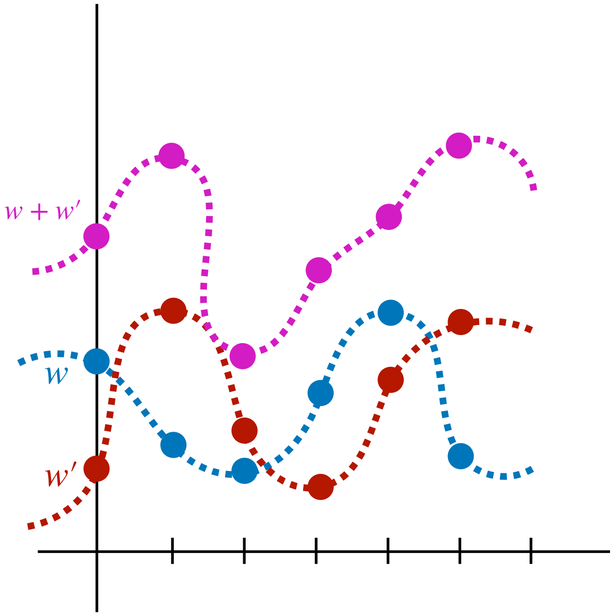
- Let $0 \leq t < n < p$ where t is the corruption limit and n is the number of parties, and let $\mathcal{M} = \mathbb{F}_p$
- $\mathsf{Share}_{p,n,t}(w)$:
  1. Sample $f \leftarrow \mathcal{P}_{p,t,w}$ uniformly (by sampling the coefficients of f uniformly from $\mathbb{F}_p$).
  2. Output $\langle w \rangle$ where $\langle w \rangle_i = f(i)\ \forall i \in [n]$.
- $\mathsf{Recon}_{p,n,t}\left((i_1, \ldots, i_{t+1}), \left(\langle w \rangle_{i_1}, \ldots, \langle w \rangle_{i_{t+1}}\right)\right)$:
  1. Interpolate $f(0) \in \mathbb{F}_p$.
  2. Output $w = f(0)$.

Lagrange Interpolation
Lecture 7



Recap: Adding Shamir-Shared Values

- Every $f \in \mathscr{P}_{p,t,w}$ is of the form
  $f(x) = w + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_t \cdot x^t$.
  where $a_1, \ldots, a_t \in \mathbb{F}_p$.



Addition: we have $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$, and we need $g \leftarrow \mathscr{P}_{p,t,w+w'}$. Setting $g := f + f'$ suffices!

Consider the entire polynomial:
$g(x) = f(x) + f'(x) = w + w' + (a_1 + a'_1) \cdot x + (a_2 + a'_2) \cdot x^2 + \ldots + (a_t + a'_t) \cdot x^t$

Consider the individual shares:
If $\langle w \rangle_i := f(i)$, $\langle w' \rangle_i := f'(i)$, $\langle w + w' \rangle_i := g(i)$ $\forall i \in [n]$ then $\langle w + w' \rangle_i = \langle w \rangle_i + \langle w' \rangle_i$.

In other words, addition of secrets can be performed by adding shares.

Recap: BGW for Linear Functions

- In **Lecture 7**, we introduced the notion of *arithmetic circuits*, which are circuits with wires that carry values in $\mathbb{F}_p$ and gates that compute the $+$ and $\times$ operations.
- *Linear functions* can be represented using circuits that contain only $+$ gates.
- We sketched a proof of the following theorem:

Theorem 1: Let $p > n > t$. Assuming synchronicity and secure channels, every *linear deterministic* n-ary function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ with a single output can be securely computed in the presence of a semi-honest $\mathscr{A}$ that statically corrupts up to $n - 1$ parties.

We Want to *Multiply* Shamir-Shared Values

- Every $f \in \mathscr{P}_{p,t,w}$ is of the form
  $f(x) = w + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_t \cdot x^t$.
  where $a_1, \ldots, a_t \in \mathbb{F}_p$.

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

Consider the entire polynomial:

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

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

We Want to *Multiply* Shamir-Shared Values

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

Consider the entire polynomial:

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

The degree 0 term is what we need it to be

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

Good News #1: $g(0) = w \cdot w'$ which means this is a sharing of the product!

Good News #2: If $\langle w \rangle_i := f(i)$, $\langle w' \rangle_i := f'(i)$, $\langle w \cdot w' \rangle_i := g(i)$ $\forall i \in [n]$ then $\langle w \cdot w' \rangle_i = \langle w \rangle_i \cdot \langle w' \rangle_i$ which means every party can multiply its own shares to compute a share of $w \cdot w'$. No communication! *Right?*

We Want to *Multiply* Shamir-Shared Values

Multiplication: we have $f \leftarrow \mathscr{P}_{p,t,w}$ and $f' \leftarrow \mathscr{P}_{p,t,w'}$, and we need $g \leftarrow \mathscr{P}_{p,t,w\cdot w'}$. 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. *Can you see them?*

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

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

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) = 2$ where $g \leftarrow \mathscr{P}_{p,2t,w\cdot w'}$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,\quad 2 + 1x,\quad 1 + 2x,\quad 0 + 3x,\quad 4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,\quad 3 + 1x,\quad 2 + 2x,\quad 1 + 3x,\quad 0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$$2 + 0x + 0x^2,\quad 1 + 1x + 0x^2,\quad 0 + 2x + 0x^2,\quad 4 + 3x + 0x^2,\quad 3 + 4x + 0x^2$$
$$1 + 0x + 1x^2,\quad 0 + 1x + 1x^2,\quad 4 + 2x + 1x^2,\quad 3 + 3x + 1x^2,\quad 2 + 4x + 1x^2$$
$$0 + 0x + 2x^2,\quad 4 + 1x + 2x^2,\quad 3 + 2x + 2x^2,\quad 2 + 3x + 2x^2,\quad 1 + 4x + 2x^2$$
$$4 + 0x + 3x^2,\quad 3 + 1x + 3x^2,\quad 2 + 2x + 3x^2,\quad 1 + 3x + 3x^2,\quad 0 + 4x + 3x^2$$
$$3 + 0x + 4x^2,\quad 2 + 1x + 4x^2,\quad 1 + 2x + 4x^2,\quad 0 + 3x + 4x^2,\quad 4 + 4x + 4x^2$$

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) := f(1) \cdot f'(1) = 3 \cdot 4 = 2$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,\quad 2 + 1x,\quad 1 + 2x,\quad 0 + 3x,\quad 4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,\quad 3 + 1x,\quad 2 + 2x,\quad 1 + 3x,\quad 0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$2 + 0x + 0x^2,\quad 1 + 1x + 0x^2,\quad 0 + 2x + 0x^2,\quad 4 + 3x + 0x^2,\quad 3 + 4x + 0x^2$

$1 + 0x + 1x^2,\quad 0 + 1x + 1x^2,\quad 4 + 2x + 1x^2,\quad 3 + 3x + 1x^2,\quad 2 + 4x + 1x^2$

$0 + 0x + 2x^2,\quad 4 + 1x + 2x^2,\quad 3 + 2x + 2x^2,\quad 2 + 3x + 2x^2,\quad 1 + 4x + 2x^2$

$4 + 0x + 3x^2,\quad 3 + 1x + 3x^2,\quad 2 + 2x + 3x^2,\quad 1 + 3x + 3x^2,\quad 0 + 4x + 3x^2$

$3 + 0x + 4x^2,\quad 2 + 1x + 4x^2,\quad 1 + 2x + 4x^2,\quad 0 + 3x + 4x^2,\quad 4 + 4x + 4x^2$

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) := f(1) \cdot f'(1) = 3 \cdot 4 = 2$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,$ $2 + 1x,$ $1 + 2x,$ $0 + 3x,$ $4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,$ $3 + 1x,$ $2 + 2x,$ $1 + 3x,$ $0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$2 + 0x + 0x^2,$ $1 + 1x + 0x^2,$ $0 + 2x + 0x^2,$ $4 + 3x + 0x^2,$ $3 + 4x + 0x^2$

$1 + 0x + 1x^2,$ $0 + 1x + 1x^2,$ $4 + 2x + 1x^2,$ $3 + 3x + 1x^2,$ $2 + 4x + 1x^2$

$0 + 0x + 2x^2,$ $4 + 1x + 2x^2,$ $3 + 2x + 2x^2,$ $2 + 3x + 2x^2,$ $1 + 4x + 2x^2$

$4 + 0x + 3x^2,$ $3 + 1x + 3x^2,$ $2 + 2x + 3x^2,$ $1 + 3x + 3x^2,$ $0 + 4x + 3x^2$

$3 + 0x + 4x^2,$ $2 + 1x + 4x^2,$ $1 + 2x + 4x^2,$ $0 + 3x + 4x^2,$ $4 + 4x + 4x^2$

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) := f(1) \cdot f'(1) = 3 \cdot 4 = 2$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,$ $2 + 1x,$ $1 + 2x,$ $0 + 3x,$ $4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,$ $3 + 1x,$ $2 + 2x,$ $1 + 3x,$ $0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$2 + 0x + 0x^2,$ $1 + 1x + 0x^2,$ $0 + 2x + 0x^2,$ $4 + 3x + 0x^2,$ $3 + 4x + 0x^2$

$1 + 0x + 1x^2,$ $0 + 1x + 1x^2,$ $4 + 2x + 1x^2,$ $3 + 3x + 1x^2,$ $2 + 4x + 1x^2$

$0 + 0x + 2x^2,$ $4 + 1x + 2x^2,$ $3 + 2x + 2x^2,$ $2 + 3x + 2x^2,$ $1 + 4x + 2x^2$

$4 + 0x + 3x^2,$ $3 + 1x + 3x^2,$ $2 + 2x + 3x^2,$ $1 + 3x + 3x^2,$ $0 + 4x + 3x^2$

$3 + 0x + 4x^2,$ $2 + 1x + 4x^2,$ $1 + 2x + 4x^2,$ $0 + 3x + 4x^2,$ $4 + 4x + 4x^2$

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) := f(1) \cdot f'(1) = 3 \cdot 4 = 2$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,$ $2 + 1x,$ $1 + 2x,$ $0 + 3x,$ $4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,$ $3 + 1x,$ $2 + 2x,$ $1 + 3x,$ $0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$2 + 0x + 0x^2,$	$1 + 1x + 0x^2,$	$0 + 2x + 0x^2,$	$4 + 3x + 0x^2,$	$3 + 4x + 0x^2$
$1 + 0x + 1x^2,$	$0 + 1x + 1x^2,$	$4 + 2x + 1x^2,$	$3 + 3x + 1x^2,$	$2 + 4x + 1x^2$
$0 + 0x + 2x^2,$	$4 + 1x + 2x^2,$	$3 + 2x + 2x^2,$	$2 + 3x + 2x^2,$	$1 + 4x + 2x^2$
$4 + 0x + 3x^2,$	$3 + 1x + 3x^2,$	$2 + 2x + 3x^2,$	$1 + 3x + 3x^2,$	$0 + 4x + 3x^2$
$3 + 0x + 4x^2,$	$2 + 1x + 4x^2,$	$1 + 2x + 4x^2,$	$0 + 3x + 4x^2,$	$4 + 4x + 4x^2$

A Product of 2 Uniform Polynomials is Biased

- Let $n = 3,\ t = 1,\ p = 5$ so we work over $\mathbb{F}_5$.
- Party P_1 gets $f(1) = 3$ and $f'(1) = 4$ and $g(1) := f(1) \cdot f'(1) = 3 \cdot 4 = 2$
- P_1 can deduce that $f(x)$ must be one of: $3 + 0x,$ $2 + 1x,$ $1 + 2x,$ $0 + 3x,$ $4 + 4x$
- P_1 can deduce that $f'(x)$ must be one of: $4 + 0x,$ $3 + 1x,$ $2 + 2x,$ $1 + 3x,$ $0 + 4x$
- P_1 can deduce that $g(x)$ must be one of:

$2 + 0x + 0x^2,$	$1 + 1x + 0x^2,$	$0 + 2x + 0x^2,$	$4 + 3x + 0x^2,$	$3 + 4x + 0x^2$
$1 + 0x + 1x^2,$	$0 + 1x + 1x^2,$	$4 + 2x + 1x^2,$	$3 + 3x + 1x^2,$	$2 + 4x + 1x^2$
$0 + 0x + 2x^2,$	$4 + 1x + 2x^2,$	$3 + 2x + 2x^2,$	$2 + 3x + 2x^2,$	$1 + 4x + 2x^2$
$4 + 0x + 3x^2,$	$3 + 1x + 3x^2,$	$2 + 2x + 3x^2,$	$1 + 3x + 3x^2,$	$0 + 4x + 3x^2$
$3 + 0x + 4x^2,$	$2 + 1x + 4x^2,$	$1 + 2x + 4x^2,$	$0 + 3x + 4x^2,$	$4 + 4x + 4x^2$

A Product of 2 Uniform Polynomials is Biased

What does this mean? $g := f \cdot f'$ is twice as likely to be a polynomial with two highlights. It cannot be a polynomial that is not highlighted.

Question: Suppose the parties reconstruct and find out $g(0) = 4$. *What can they infer?*

Answer: There are 4 combinations of f, f' that yield $g(0) = 4$. There are 5 combinations of $m, m' \in \mathbb{F}_5$ s.t. $m \cdot m' = 4$. So we rule out one input combination! An attack!

- P_1 can deduce that $g(x)$ must be one of:

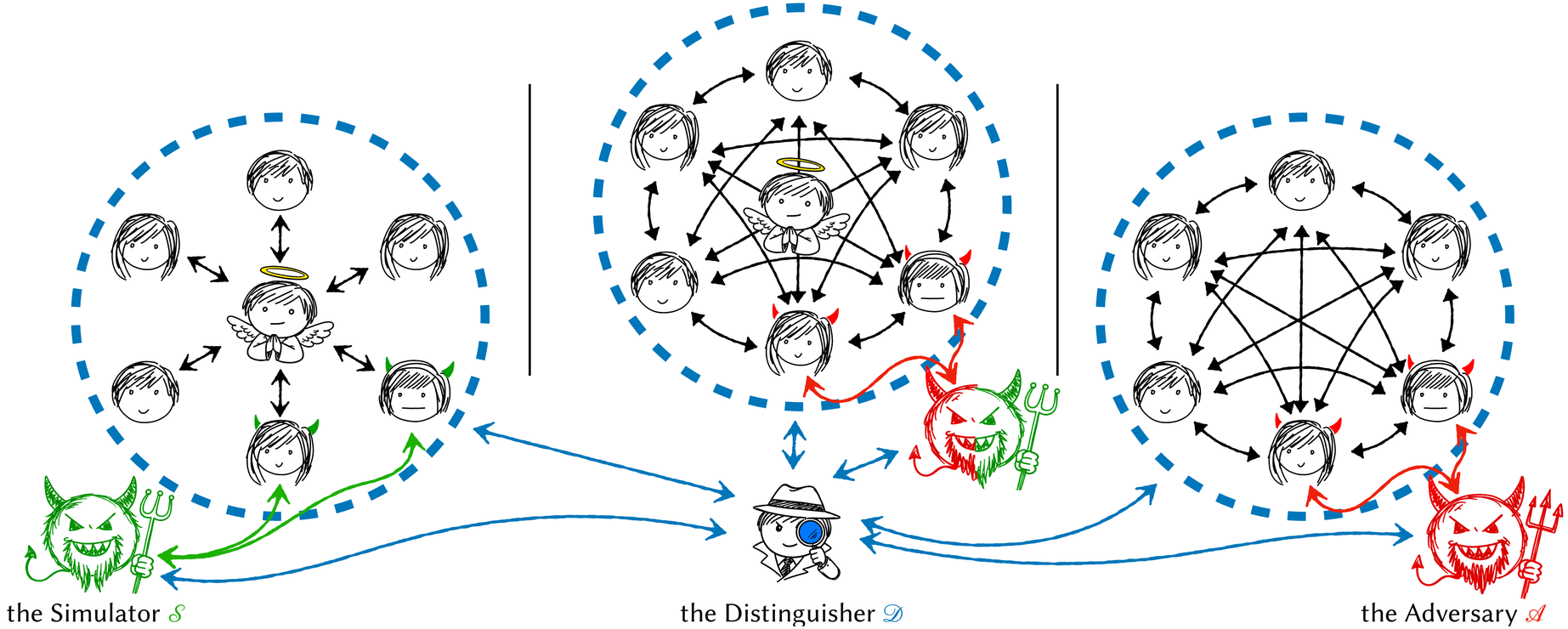
$2 + 0x + 0x^2$,	$1 + 1x + 0x^2$,	$0 + 2x + 0x^2$,	$4 + 3x + 0x^2$,	$3 + 4x + 0x^2$
$1 + 0x + 1x^2$,	$0 + 1x + 1x^2$,	$4 + 2x + 1x^2$,	$3 + 3x + 1x^2$,	$2 + 4x + 1x^2$
$0 + 0x + 2x^2$,	$4 + 1x + 2x^2$,	$3 + 2x + 2x^2$,	$2 + 3x + 2x^2$,	$1 + 4x + 2x^2$
$4 + 0x + 3x^2$,	$3 + 1x + 3x^2$,	$2 + 2x + 3x^2$,	$1 + 3x + 3x^2$,	$0 + 4x + 3x^2$
$3 + 0x + 4x^2$,	$2 + 1x + 4x^2$,	$1 + 2x + 4x^2$,	$0 + 3x + 4x^2$,	$4 + 4x + 4x^2$

What Would the Ideal Process Be?

- Suppose our old friend the (trusted) Functionality is willing to help…
  but only with multiplication. The parties have sharings of two wires w_1 and w_2.
  What would we want the Multiplication Functionality ($\mathcal{F}_{\mathsf{mul}}$) to do?
- Each P_i for $i \in [n]$ sends inputs $\langle w_1 \rangle_i$ and $\langle w_2 \rangle_i$ to $\mathcal{F}_{\mathsf{mul}}$.
- $\mathcal{F}_{\mathsf{mul}}$ performs the following steps:
  1. Reconstruct w_1 from $\langle w_1 \rangle$ and w_2 from $\langle w_2 \rangle$ (abort if this fails).
  2. Compute $w_3 := w_1 \cdot w_2$.
  3. Sample $g \leftarrow \mathcal{P}_{p,t,w_3}$ and let $\langle w_3 \rangle := (g(1), \ldots, g(n))$. i.e. let $\langle w_3 \rangle \leftarrow \mathsf{Share}_{p,n,t}(w_3)$.
  4. Send each $\langle w_3 \rangle_i$ to P_i.
- **Next Question:** So how do we incorporate $\mathcal{F}_{\mathsf{mul}}$ into BGW? *Any ideas?*

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!

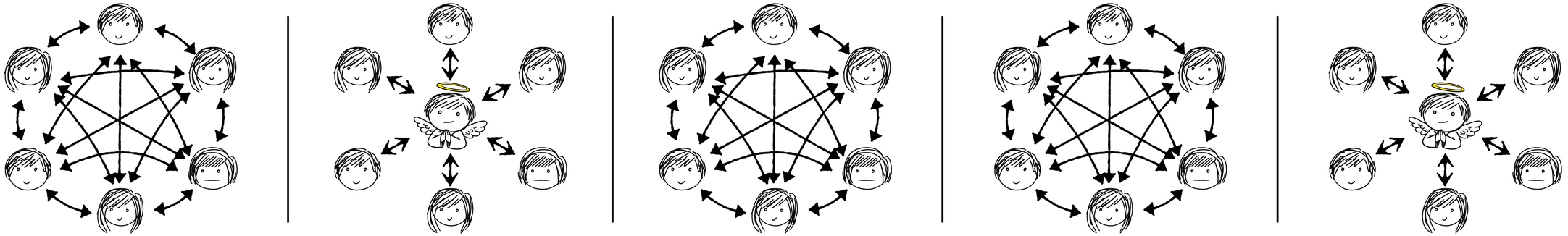


The $\mathcal{F}_{\text{mul}}$-*Hybrid* Model

- *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.

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

- In particular, we will assume that in a hybrid model, all communication happens in rounds (i.e. we have synchrony) and that in every round, the parties communicate in one of the following *mutually exclusive* ways:
  - They send messages to one another over secure point-to-point channels, just like in the real world.
  - They send messages to a single Functionality, and receive a reply, just like in the ideal world.



Round 1 | Round 2 | Round 3 | Round 4 | Round 5

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 $\mathcal{F}_{\text{SFE}}$. *Are we done? What's left?*
3. Design another protocol π_{mul} that realizes $\mathcal{F}_{\text{mul}}$. *Howabout now?*
4. Prove that security is maintained if we *replace* $\mathcal{F}_{\text{mul}}$ with π_{mul} in π_{BGW}.

   Why isn't this obvious?

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 $t < n < p$ and let f be an n-ary function over $\mathbb{F}_p$. Assuming synchrony and secure point-to-point channels in the $\mathcal{F}_{\text{mul}}$-hybrid model there exists an n-party protocol that perfectly securely computes f in the presence of a semi-honest adversary that statically corrupts up to t parties.

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.

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}.

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.

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. Construct the BGW Protocol

Specifying Arithmetic Circuits over $\mathbb{F}_p$

Simplifying Assumption: each party has exactly 1 input + output. Easy to generalize.

A randomized circuit $C \subset \{(\mathsf{in}, \mathbb{N}, \mathbb{N}), (\mathsf{rand}, \mathbb{N}), (+, \mathbb{N}, \mathbb{N}, \mathbb{N}), (\times, \mathbb{N}, \mathbb{N}, \mathbb{N})), (\mathsf{out}, \mathbb{N}, \mathbb{N})\}^*$ over $\mathbb{F}_p$ specifies a set of gates. We say a C computes f if $\forall x \in \text{domain}(f)$, $C(x) = f(x)$.

The gates are:

- (in, i, o) sets the value of wire o equal to the input of P_i.
- (rand, o) sets samples value of wire o uniformly from $\mathbb{F}_p$.
- $(+, j, k, o)$ sets the value of wire o to the sum of the values of wires j and k.
- $(\times, j, k, o)$ sets the value of wire o to the product of the values of wires j and k.
- (out, i, j) sets outputs of P_i equal to the values of wire j.

Specifying Arithmetic Circuits over $\mathbb{F}_p$

The gates are:

- (in, i, o) sets the value of wire o equal to the input of P_i.
- (rand, o) sets samples value of wire o uniformly from $\mathbb{F}_p$.
- $(+, j, k, o)$ sets the value of wire o to the sum of the values of wires j and k.
- $(\times, j, k, o)$ sets the value of wire o to the product of the values of wires j and k.
- (out, i, j) sets outputs of P_i equal to the values of wire j.

Definition 1 (Circuit Well-Formedness): a circuit C is well-formed if and only if every gate's input wires (i.e. j, k for $+, \times$ and j for out) are the output wires of some other gate (i.e. o for $+, \times$, rand, and in), and each wire is the output of at most one gate, and the graph these restrictions induce is acyclic. We also insist that the out gates have distinct output indices i.

The BGW Protocol $\pi_{\mathsf{BGW}}(n, t, p, C)$

Let $t < n < p \in \mathbb{N}$. There are n parties $P_1, \ldots, P_n$, with inputs $x_1, \ldots, x_n \in \mathbb{F}_p$ respectively. π_{BGW} computes a well-formed n-ary arithmetic circuit $C : \mathbb{F}_p^n \to \mathbb{F}_p^n$. The parties output $y_1, \ldots, y_n \in \mathbb{F}_p$ respectively.

There are Three Phases:

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 $j \in [n] \backslash \{i\}$. This takes 1 round.

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.

   - Suppose the parties arrive at gate $(+, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$. This takes 0 rounds.

The BGW Protocol $\pi_{\mathsf{BGW}}(n, t, p, C)$

Let $t < n < p \in \mathbb{N}$. There are n parties $P_1, \ldots, P_n$, with inputs $x_1, \ldots, x_n \in \mathbb{F}_p$ respectively. π_{BGW} computes a well-formed n-ary arithmetic circuit $C : \mathbb{F}_p^n \to \mathbb{F}_p^n$. The parties output $y_1, \ldots, y_n \in \mathbb{F}_p$ respectively.

There are Three Phases:

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.

   - Suppose the parties arrive at gate $(+, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$. This takes 0 rounds.
   - Suppose the parties arrive at gate $(\times, j, k, o) \in C$. Each party P_i sends $\left(\langle w_j \rangle_i, \langle w_k \rangle_i\right)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o \rangle_i$ in response. This takes 1 round.

The BGW Protocol $\pi_{\mathsf{BGW}}(n, t, p, C)$

Let $t < n < p \in \mathbb{N}$. There are n parties $P_1, \ldots, P_n$, with inputs $x_1, \ldots, x_n \in \mathbb{F}_p$ respectively. π_{BGW} computes a well-formed n-ary arithmetic circuit $C : \mathbb{F}_p^n \to \mathbb{F}_p^n$. The parties output $y_1, \ldots, y_n \in \mathbb{F}_p$ respectively.

There are Three Phases:

- Suppose the parties arrive at gate $(+, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$. This takes 0 rounds.
- Suppose the parties arrive at gate $(\times, j, k, o) \in C$. Each party P_i sends $\left(\langle w_j \rangle_i, \langle w_k \rangle_i\right)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o \rangle_i$ in response. This takes 1 round.
- Suppose the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \mathbb{F}_p$, and shares it to the others. The parties use a sequence of $+$ gates to securely compute $\langle w_o \rangle = \sum_{k \in [n]} \langle r_{o,k} \rangle$. This takes 1 round.

The BGW Protocol $\pi_{\mathsf{BGW}}(n, t, p, C)$

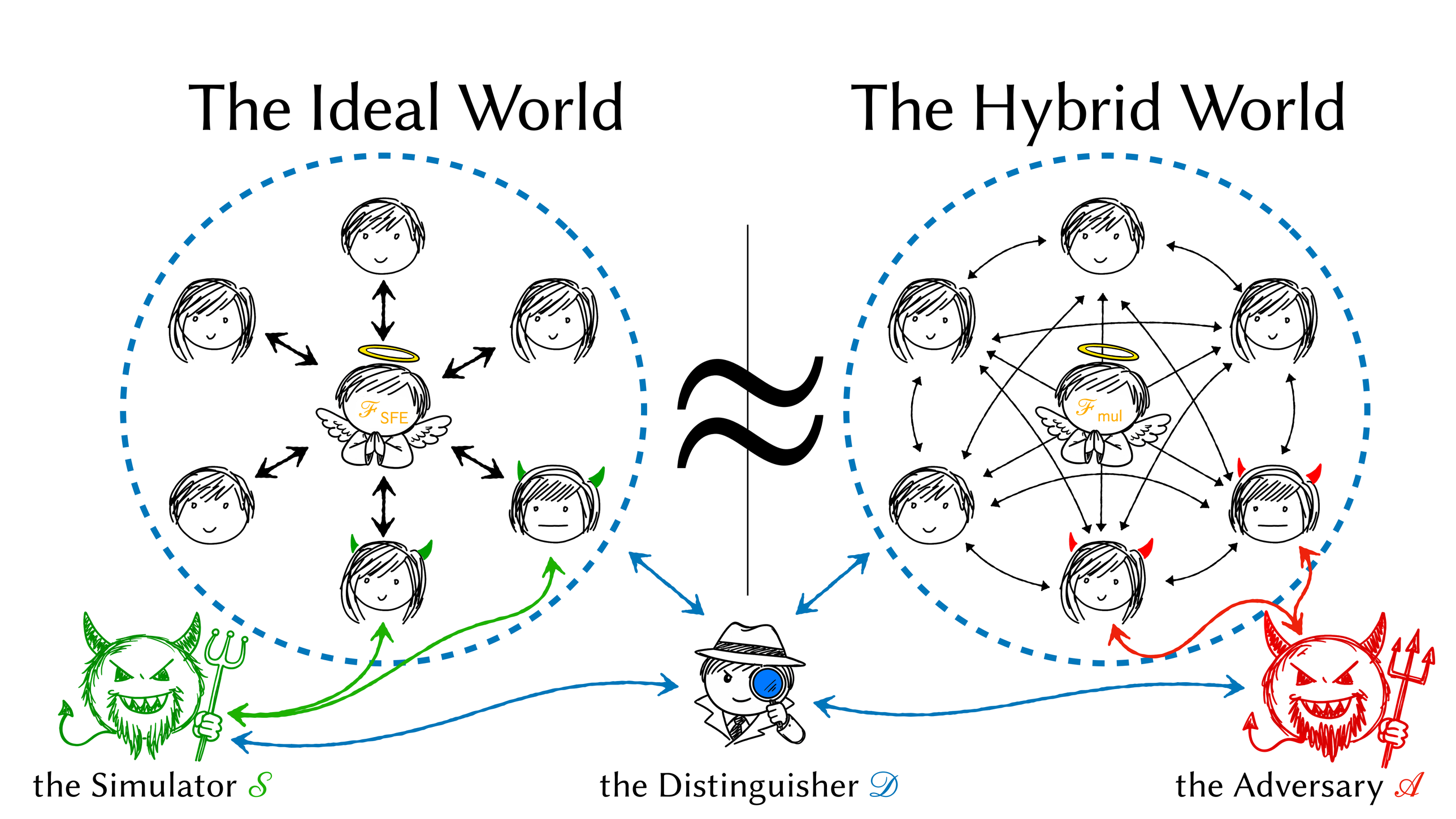
Let $t < n < p \in \mathbb{N}$. There are n parties $P_1, \ldots, P_n$, with inputs $x_1, \ldots, x_n \in \mathbb{F}_p$ respectively. π_{BGW} computes a well-formed n-ary arithmetic circuit $C : \mathbb{F}_p^n \to \mathbb{F}_p^n$. The parties output $y_1, \ldots, y_n \in \mathbb{F}_p$ respectively.

There are Three Phases:

- Suppose the parties arrive at gate $(\times, j, k, o) \in C$. Each party P_i sends $\left(\langle w_j \rangle_i, \langle w_k \rangle_i\right)$ to $\mathcal{F}_{\mathsf{mul}}$ and receives $\langle w_o \rangle_i$ in response. This takes 1 round.
- Suppose the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \mathbb{F}_p$, and shares it to the others. The parties use a sequence of $+$ gates to securely compute $\langle w_o \rangle = \sum_{k \in [n]} \langle r_{o,k} \rangle$. This takes 1 round.

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. This takes 1 round.

2. Prove BGW Secure in the Hybrid Model


The Ideal World
The Hybrid World
$\mathcal{F}_{\text{SFE}}$
$\mathcal{F}_{\text{mul}}$
the Simulator $\mathcal{S}$
the Distinguisher $\mathcal{D}$
the Adversary $\mathcal{A}$

A Simplified Security Def for Semi-Honest $\mathcal{A}$

Definition 2: Perfect Semi-Honest Secure Function Evaluation (Simplified).

Let $n, t \in \mathbb{N}$ such that $t < n$, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be an n-ary function, and let π be a n-party protocol (in the $\mathcal{F}_{\mathsf{mul}}$-hybrid model).

We say that π *perfectly securely computes* f in the presence of a semi-honest adversary that statically corrupts up to t parties if there exists a *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every input vector $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ it holds that

$$\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)$$

A Slightly Less Sketchy Proof of Security

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, $\pi_{\mathsf{BGW}}(n, t, p, C)$ perfectly realizes $\mathscr{F}_{\mathsf{SFE}}(n, f, \mathbb{F}_p, \ldots, \mathbb{F}_p)$ in the presence of a semi-honest $\mathscr{A}$ that statically corrupts up to $n - 1$ parties in the $\mathscr{F}_{\mathsf{mul}}$-hybrid model.

Pf Sketch: because $\mathscr{A}$ is semi-honest we can use our simplified security definition. 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 $\vec{x} \in \mathrm{domain}(f)$ it holds that

$$\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_{\pi_{\mathsf{BGW}}})$$

General Guideline for Semi-Honest Sims

If $\mathscr{A}$ is semi-honest, then Sim can simulate the corrupt parties' contributions to VIEW_I by running their code (since they will not deviate) on their inputs (which Sim is given as input). Note that Sim essentially *must* do this because it can be detected if it alters the behavior of the corrupted parties.

The main task, then, is to simulate the contributions of the honest parties and functionalities. Sim does *not* know the inputs of these participants.

Sim also ensure that the corrupt parties output the exact values that they are supposed to (which are supplied as inputs to Sim). Since it is not allowed to change the behavior of the corrupt parties, it can *only* do this by manipulating the contributions of the honest parties and functionalities.

Thus, Sim must "program" the output of the corrupt parties without altering the distribution of the protocol transcript in a way that can be detected by a distinguisher.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

Hybrid-World VIEW_I

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

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 $j \in [n] \backslash \{i\}$.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

Hybrid-World VIEW_I

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

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 $j \in [n] \backslash \{i\}$.

   Corrupt inputs and all shares of corrupt inputs. Up to t received shares of honest party inputs, which are uniform by the privacy property of Shamir sharing.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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

Hybrid-World VIEW_I

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

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 $j \in [n] \backslash \{i\}$.

   Corrupt inputs and all shares of corrupt inputs. Up to t received shares of honest party inputs, which are uniform by the privacy property of Shamir sharing.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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.

Hybrid-World VIEW_I

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

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 $j \in [n] \backslash \{i\}$.

Corrupt inputs and all shares of corrupt inputs. Up to t received shares of honest party inputs, which are uniform by the privacy property of Shamir sharing.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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.

Hybrid-World VIEW_I

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

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 $j \in [n] \backslash \{i\}$.

Corrupt inputs and all shares of corrupt inputs. Up to t received shares of honest party inputs, which are uniform by the privacy property of Shamir sharing.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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.

Hybrid-World VIEW_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 $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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.

Hybrid-World VIEW_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 $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$.

   Deterministic sums of shares already in corrupt parties view.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{BGW}}(n, t, p, C)$ and $\mathscr{A}$

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.

Hybrid-World VIEW_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 $\langle w_o \rangle_i := \langle w_j \rangle_i + \langle w_k \rangle_i$.

   Deterministic sums of shares already in corrupt parties view.

Simulated View

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.

Hybrid-World VIEW_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.

Simulated View

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.

Hybrid-World VIEW_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.

Up to t shares received from $\mathcal{F}_{\mathsf{mul}}$. By the privacy property of Shamir sharing, these are uniform.

Simulated View

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.

Hybrid-World VIEW_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 $\mathcal{F}_{\text{mul}}$ and receives $\langle w_o \rangle_i$ in response.

Up to t shares received from $\mathcal{F}_{\text{mul}}$. By the privacy property of Shamir sharing, these are uniform.

Notice: the functionality is treated like an honest party for the purposes of simulation. Sim must "play the role"of $\mathcal{F}_{\text{mul}}$.

Simulated View

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.

Hybrid-World VIEW_I

- 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 $w_o = \sum_{k \in [n]} r_{o,k}$.

Simulated View

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.

Hybrid-World VIEW_I

- 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 $w_o = \sum_{k \in [n]} r_{o,k}$.

Corrupt randomness and all shares of corrupt randomness. Up to t received shares of honest party randomness, which are uniform by the privacy property of Shamir sharing. Plus a deterministic sum.

Simulated View

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.
6. For every $(\mathsf{rand}, o) \in C$ and $i \in I$, sample $r_{o,i} \leftarrow \mathbb{F}_p$ and $\langle r_{o,i} \rangle \leftarrow \mathsf{Share}_{p,n,t}(r_{o,i})$, and for $h \in [n] \backslash I$, sample $\langle r_{o,h} \rangle_i \leftarrow \mathbb{F}_p$. Compute $\langle w_o \rangle_i := \sum_{k \in [n]} \langle r_{o,k} \rangle_i$ and copy $r_{o,i}$, $\langle r_{o,i} \rangle$, $\langle w_o \rangle_i$ and $\langle r_{o,k} \rangle_i \; \forall k \in [n] \backslash \{i\}$ into the view of corrupted P_i.

Hybrid-World VIEW_I

- 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 $w_o = \sum_{k \in [n]} r_{o,k}$.

Corrupt randomness and all shares of corrupt randomness. Up to t received shares of honest party randomness, which are uniform by the privacy property of Shamir sharing. Plus a deterministic sum.

Simulated View

6. For every $(\mathsf{rand}, o) \in C$ and $i \in I$, sample $r_{o,i} \leftarrow \mathbb{F}_p$ and $\langle r_{o,i} \rangle \leftarrow \mathsf{Share}_{p,n,t}(r_{o,i})$, and for $h \in [n] \backslash I$, sample $\langle r_{o,h} \rangle_i \leftarrow \mathbb{F}_p$. Compute $\langle w_o \rangle_i := \sum_{k \in [n]} \langle r_{o,k} \rangle_i$ and copy $r_{o,i}$, $\langle r_{o,i} \rangle$, $\langle w_o \rangle_i$ and $\langle r_{o,k} \rangle_i \; \forall k \in [n] \backslash \{i\}$ into the view of corrupted P_i.

Hybrid-World VIEW_I

Corrupt randomness and all shares of corrupt randomness. Up to t received shares of honest party randomness, which are uniform by the privacy property of Shamir sharing. Plus a deterministic sum.

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.

Simulated View

6. For every $(\mathsf{rand}, o) \in C$ and $i \in I$, sample $r_{o,i} \leftarrow \mathbb{F}_p$ and $\langle r_{o,i} \rangle \leftarrow \mathsf{Share}_{p,n,t}(r_{o,i})$, and for $h \in [n] \backslash I$, sample $\langle r_{o,h} \rangle_i \leftarrow \mathbb{F}_p$. Compute $\langle w_o \rangle_i := \sum_{k \in [n]} \langle r_{o,k} \rangle_i$ and copy $r_{o,i}$, $\langle r_{o,i} \rangle$, $\langle w_o \rangle_i$ and $\langle r_{o,k} \rangle_i \; \forall k \in [n] \backslash \{i\}$ into the view of corrupted P_i.

Hybrid-World VIEW_I

Corrupt randomness and all shares of corrupt randomness. Up to t received shares of honest party randomness, which are uniform by the privacy property of Shamir sharing. Plus a deterministic sum.

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.

All shares of each corrupt output. The received shares must be consistent with the logical output of the circuit and all corrupt shares that are already fixed.

Simulated View

6. For every $(\mathsf{rand}, o) \in C$ and $i \in I$, sample $r_{o,i} \leftarrow \mathbb{F}_p$ and $\langle r_{o,i} \rangle \leftarrow \mathsf{Share}_{p,n,}$ for $h \in [n] \backslash I$, sample $\langle r_{o,h} \rangle_i \leftarrow$ Compute $\langle w_o \rangle_i := \sum_{k \in [n]} \langle r_{o,}$ $r_{o,i}$, $\langle r_{o,i} \rangle$, $\langle w_o \rangle_i$ and $\langle r_{o,k} \rangle_i \; \forall k \in [n] \backslash \{i\}$ into the view of corrupted P_i.

7. For $i \in I$, find $(\mathsf{out}, i, j) \in C$. 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.

Notice: here Sim is "programming" the output of the corrupt parties to match what it received as input. We must prove that the distribution of $\langle w_j \rangle_h$ is correct!

Hybrid-World VIEW_I

Corrupt randomness and all shares of ndomness. Up to t received honest party randomness, uniform by the privacy of Shamir sharing. Plus a deterministic sum.

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.

All shares of each corrupt output. The received shares must be consistent with the logical output of the circuit and all corrupt shares that are already fixed.

Argument for Perfect Simulation

- The input and random gate shares dealt by honest parties are identically distributed in the hybrid and ideal worlds, by the privacy property of Shamir sharing.
- The random coins and sharings of the corrupt parties are identically distributed in the hybrid and ideal worlds, because those parties are semi-honest and the simulator simply runs their code.
- The shares of the output wires of multiplication gates are identically distributed in the hybrid and ideal worlds by the privacy property of Shamir sharing, and the fact that they are sampled independently of all topologically-earlier sharings.
- All other gate output wires are derived in an identical, deterministic way from identically distributed input wires. Thus they must be identically distributed in both worlds.

Argument for Perfect Simulation

- By the correctness of Shamir sharing (for the input and output phases), the linearity of Shamir sharing (for addition gates) and the ideal behavior of $\mathcal{F}_{\text{mul}}$, the hybrid world correctly computes the circuit C.

  By the correctness of the interpolation procedure, the corrupt parties outputs in the ideal world are equal to the outputs of f, and because C computes f, this implies the corrupt parties outputs are identically distributed in the hybrid and ideal worlds.

  Finally, consider the honest parties' shares of the output wires. If there are exactly t corruptions, then these shares are fixed by the logical output wire values and by the shares of the corrupt parties. Since the latter values are identically distributed in both world, the honest parties shares must be as well.

  If there are fewer than t corruptions, then the set to which the honest parties' shares of the output wires are constrained is identically distributed in both worlds, and in both worlds, the shares are uniform within that set (and thus identically distributed).

Argument for Perfect Simulation

Finally, consider the honest parties' shares of the output wires. If there are exactly t corruptions, then these shares are fixed by the logical output wire values and by the shares of the corrupt parties. Since the latter values are identically distributed in both world, the honest parties shares must be as well.

If there are fewer than t corruptions, then the set to which the honest parties' shares of the output wires are constrained is identically distributed in both worlds, and in both worlds, the shares are uniform within that set (and thus identically distributed).

Thus we have $\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_{\pi_{\mathsf{BGW}}})$ ∎

Take a deep breath.

Next Time:

3. Multiplication Protocols!

4. Composition Theorem!

CS4501 Cryptographic Protocols
Lecture 8: The BGW Protocol

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