Secure Multi-Party Computation

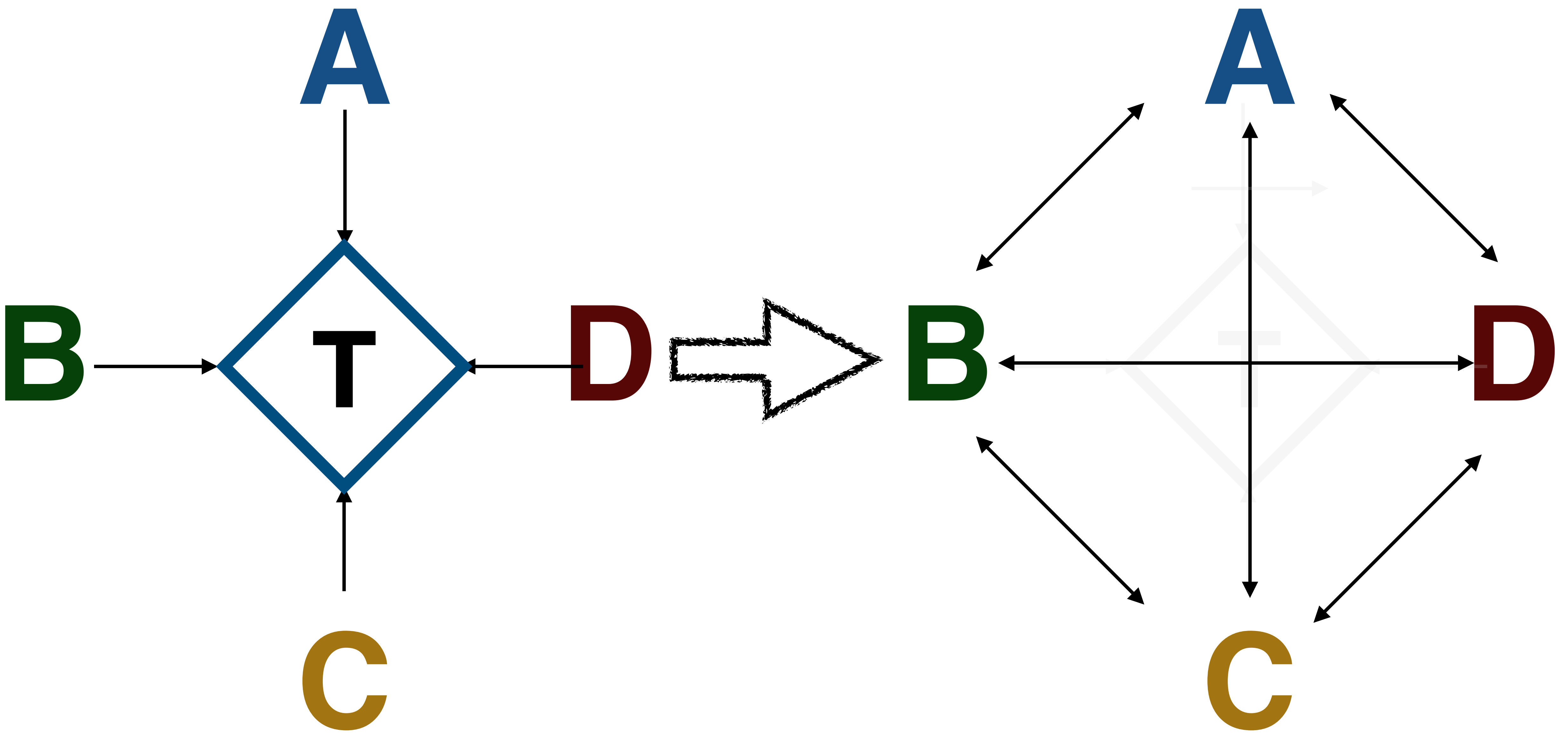
The BGW Protocol

Gilad Asharov
Bar-Ilan University (BIU)

The 10th Bar-Ilan Winter School on Cryptography, Information Theoretic Cryptography

Center for Research in Applied
Cryptography and Cyber Security

Secure Computation



Secure Computation

- Set of parties $P_1, \ldots, P_n$
- Each holds some private input $x_1, \ldots, x_n$
- The parties wish to compute a joint function $f(x_1, \ldots, x_n)$ while keeping their inputs private

- Some parties might be corrupted:
  - **Semi-honest**: Follow the protocol specifications' but try to gain some extra information by pooling their views
  - **Malicious**: Might act arbitrarily

- **Correctness**:
  - The output of the parties is $f(x_1, \ldots, x_n)$
- **Privacy:**
  - The corrupted parties do not learn anything about the honest parties' inputs
- **Guaranteed output delivery:**
  - The adversary should not prevent the honest parties from obtaining output

Main Theorem

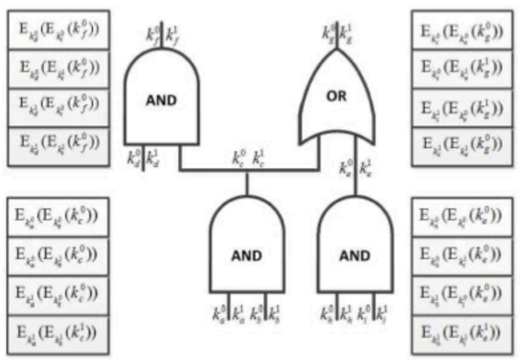
- For every n-ary function $f(x_1, \ldots, x_n)$, there exists a protocol for computing f with ***perfect security*** in the presence of **a semi-honest** adversary controlling $t < n/2$ parties

- For every n-ary function $f(x_1, \ldots, x_n)$, there exists a protocol for computing f with **perfect security** in the presence of **a malicious** adversary controlling $t < n/3$ parties

4 Approaches to MPC

Garbled Circuits

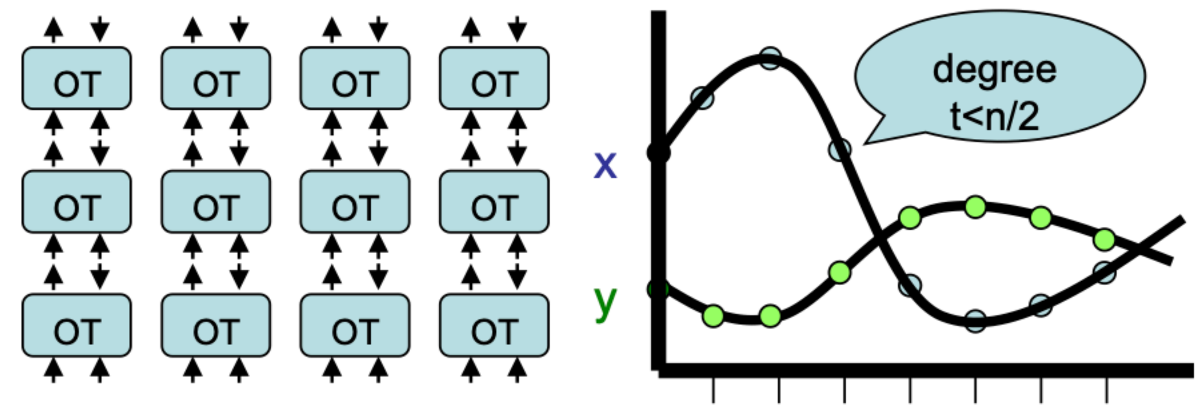
[Yao 86,…]



Linear Secret Sharing

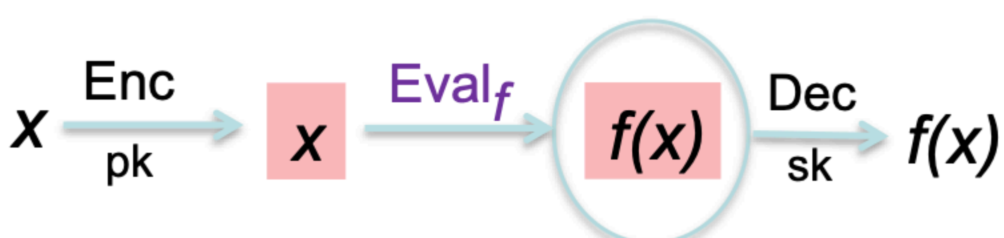
[Goldreich-Micali-Wigderson 87]

[BenOr-Goldwasser-W88, Chaum-Crépeau-Damgård88, …]



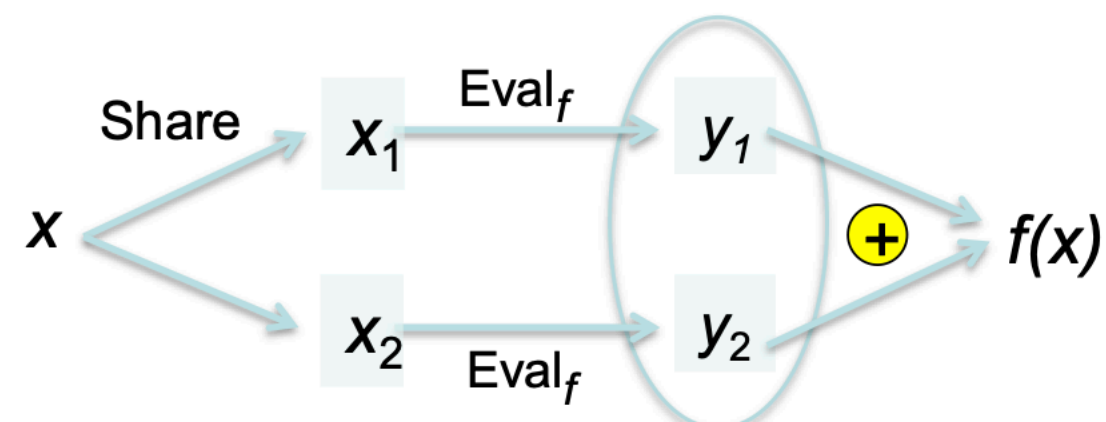
Fully Homomorphic Encryption

[Gentry 09,…]



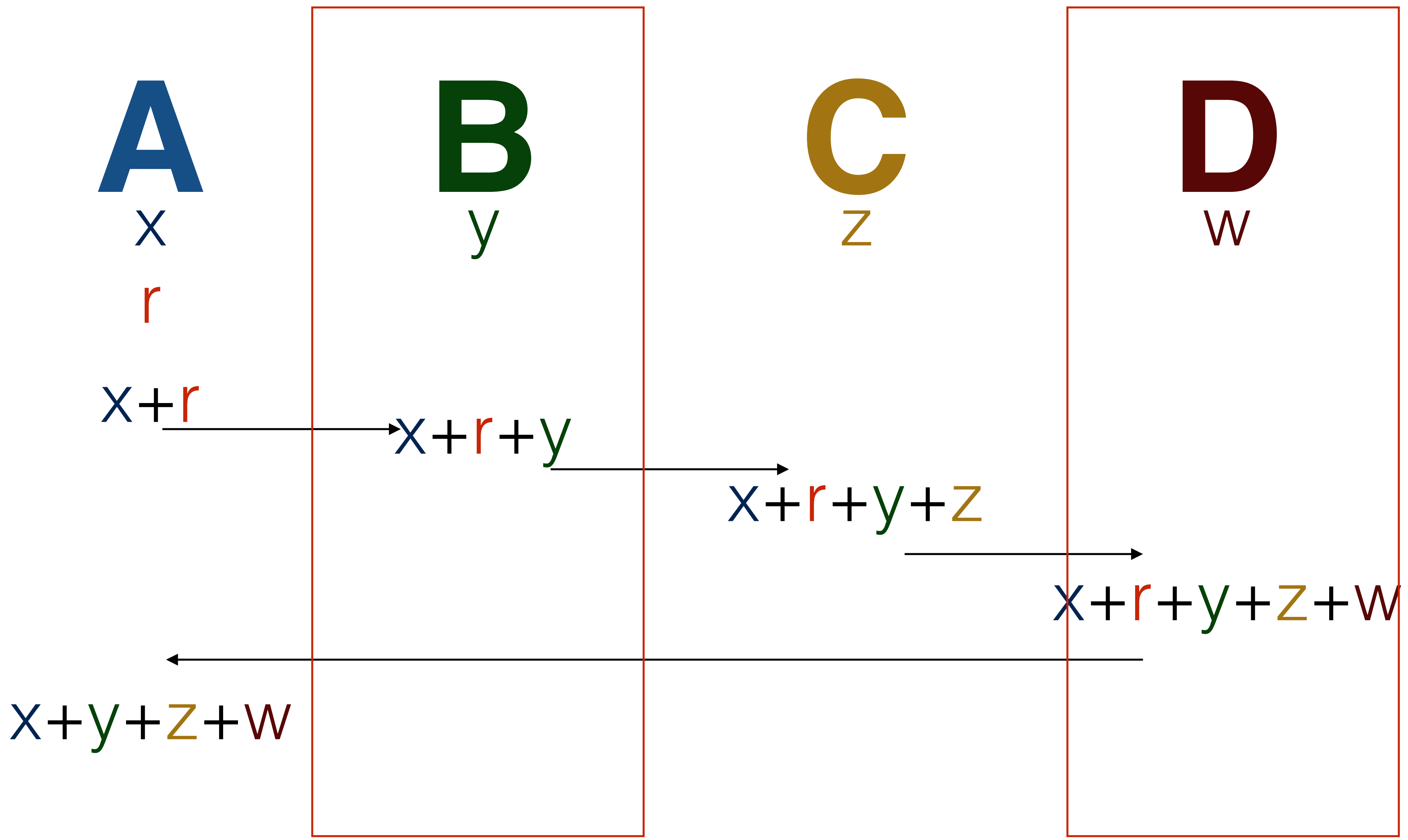
Homomorphic Secret Sharing

[Boyle-Gilboa-I 15,…]



The Semi-Honest Case

Warmup:
Average of Salaries (or Sum..)



Warmup:
Average of Salaries (or Sum..)

A	B	C	D
x	y	z	w
$x_1+x_2+x_3+x_4=x$	$y_1+y_2+y_3+y_4=y$	$z_1+z_2+z_3+z_4=z$	$w_1+w_2+w_3+w_4=w$
x_1	x_2	x_3	x_4
y_1	y_2	y_3	y_4
z_1	z_2	z_3	z_4
w_1	w_2	w_3	w_4
$x_1+y_1+z_1+w_1$	$x_2+y_2+z_2+w_2$	$x_3+y_3+z_3+w_3$	$x_4+y_4+z_4+w_4$
$=s_1$	$=s_2$	$=s_3$	$=s_4$

x
y
z
w

$$S=s_1+s_2+s_3+s_4= \begin{matrix} x_1+x_2+x_3+x_4 \\ y_1+y_2+y_3+y_4 \\ z_1+z_2+z_3+z_4 \\ w_1+w_2+w_3+w_4 \end{matrix}$$

Warmup: Average of Salaries (or Sum..)

Input Sharing Phase

“the actual computation”

Output Reconstruction

A

x

$x_1+x_2+x_3+x_4=x$

x
y
z
w

x_1
y_1
z_1
w_1

$x_1+y_1+z_1+w_1$

$=s_1$

B

y

$y_1+y_2+y_3+y_4=y$

x_2
y_2
z_2
w_2

$x_2+y_2+z_2+w_2$

$=s_2$

$x_3+y_3+z_3+w_3$

$=s_3$

D

w

$w_1+w_2+w_3+w_4=w$

x_4
y_4
z_4
w_4

$x_4+y_4+z_4+w_4$

$S=s_1+s_2+s_3+s_4=$

$x_1+x_2+\ldots$

$y_1+y_2+y_3+\ldots$

$z_1+z_2+z_3+z_4$

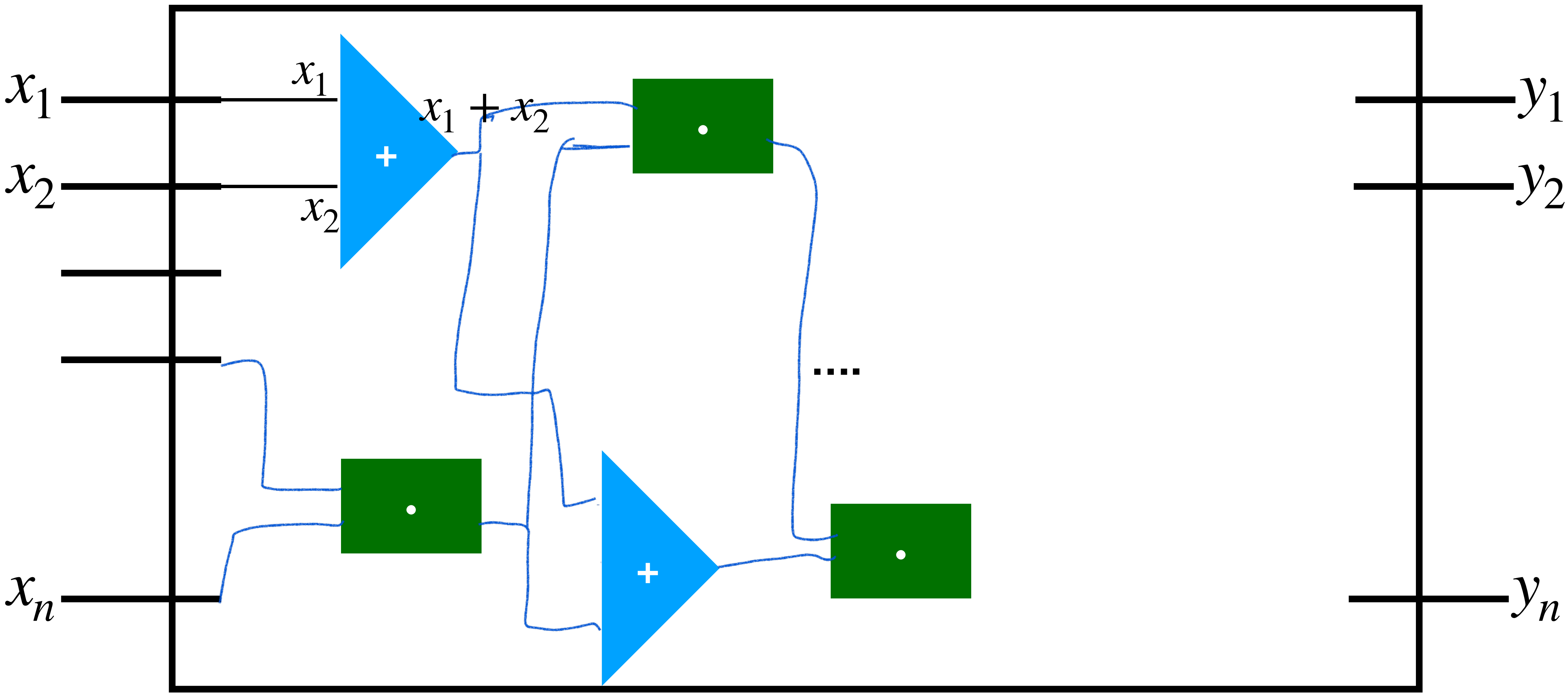
$w_1+w_2+w_3+w_4$

Overview of the BGW Protocol

- It is enough to assume that f is deterministic
  - $g(x_1, \ldots, x_n; r)$ can be computed using the deterministic function $f((x_1, r_1), \ldots, (x_n, r_n)) := g(x_1, \ldots, x_n; \oplus r_i)$

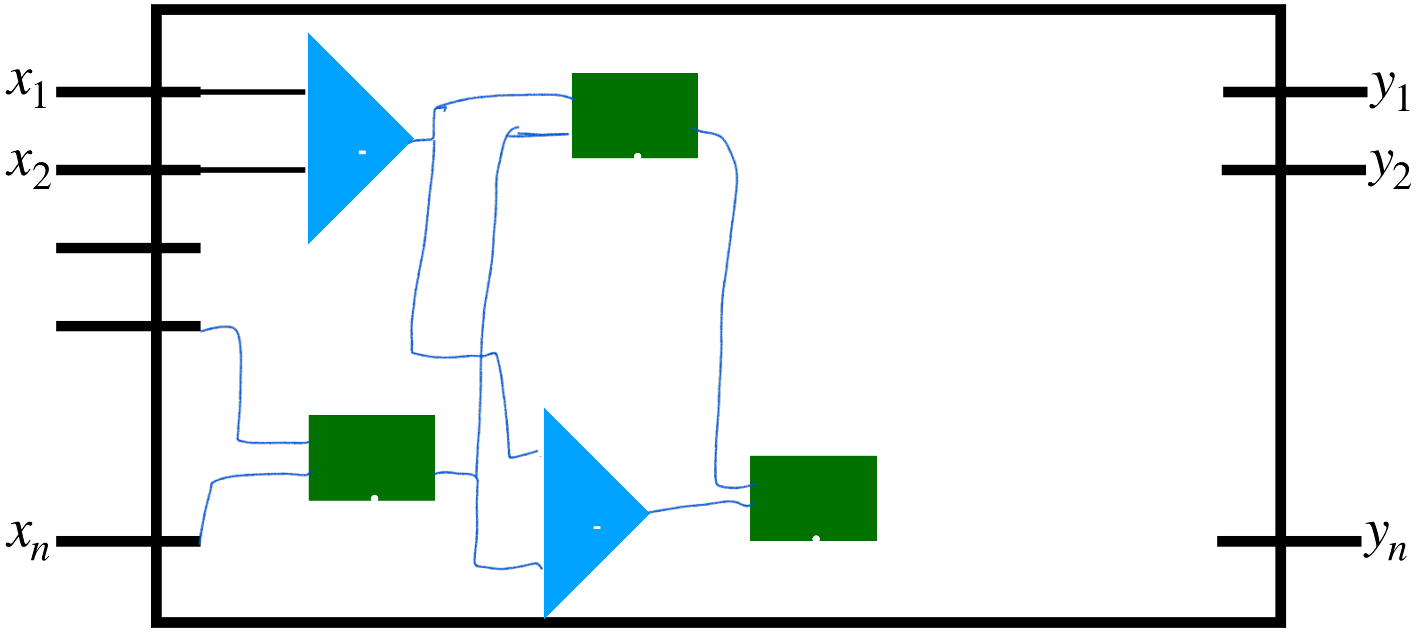
- We represent f using an **arithmetic** circuit over a field $\mathbb{F}$ $(|\mathbb{F}| > n)$
  - A circuit where each wire gets a value in $\mathbb{F}$
  - **Gates**:
    - Addition gate: $g(a, b) = a + b$
      - Multiplication with a constant gate: $g_c(a) = c \cdot a$
    - Multiplication gate: $g(a, b) = a \cdot b$

Circuit Evaluation



Evaluating C Privately

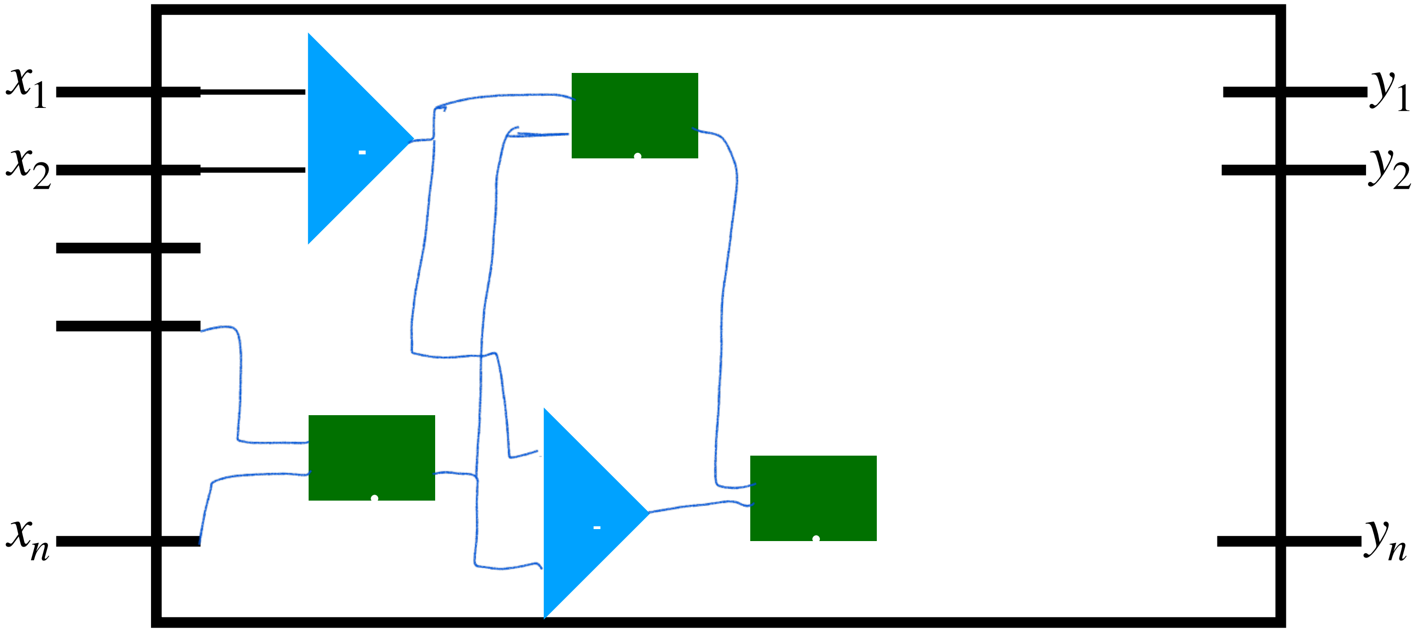
- In the secure protocol, each **input wire** is known to only one party
    - And that party wants to keep it private!
- Moreover, we ***cannot*** reveal any **intermediate** values
    - All values on all wires during the evaluation should be hidden
- Only values on the output wires should be revealed



The Key Idea

- The parties will emulate a computation of the circuit C on the inputs $x_1, \ldots, x_n$

Invariant: The value of each wire is hidden using *a random polynomial of degree t* (i.e., secret shared among the parties)



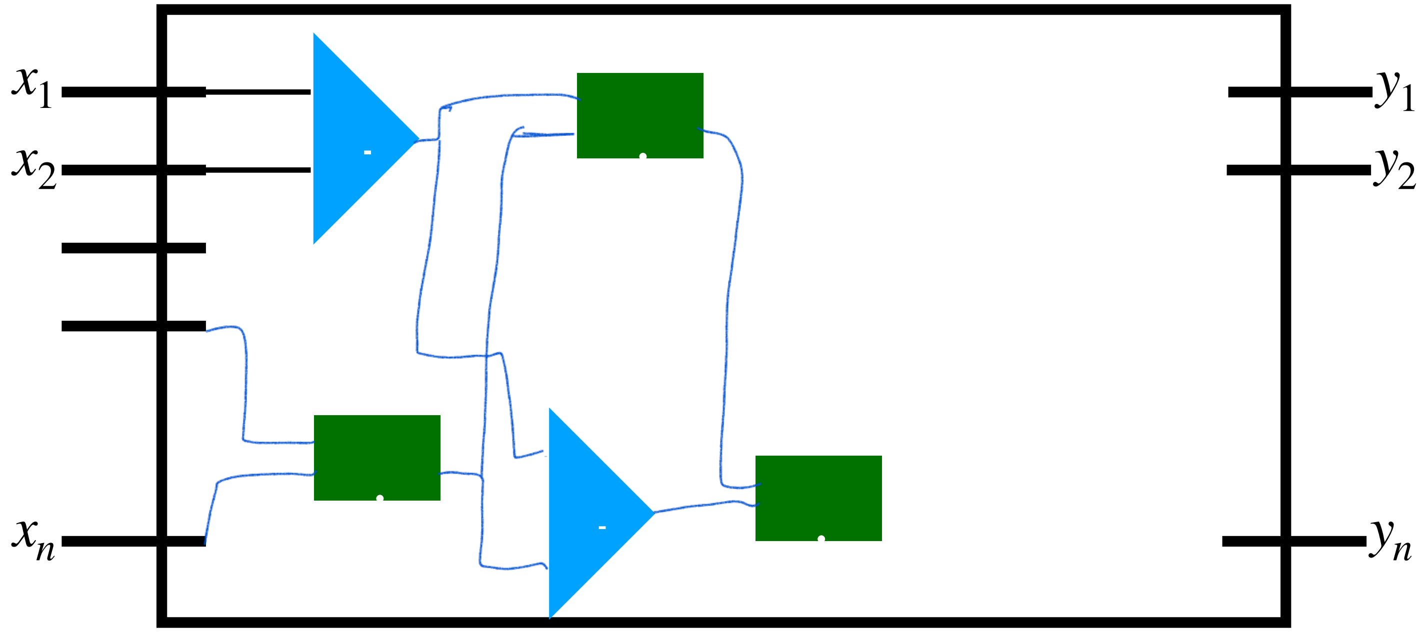
A Reminder:
Shamir's Secret Sharing Scheme

- Sharing$_{t+1,n}(s)$:
  - Choose a random degree t polynomial with s as its constant term
  - $p(x) = s + p_1 x + \ldots, p_t x^t$
  - Party P_i receives $(\alpha_i, p(\alpha_i))$

- **Properties**:
  - Every set of $t + 1$ participants can **recover** the secret
  - Every set of t shares **does not reveal** any information about s

Protocol Overview

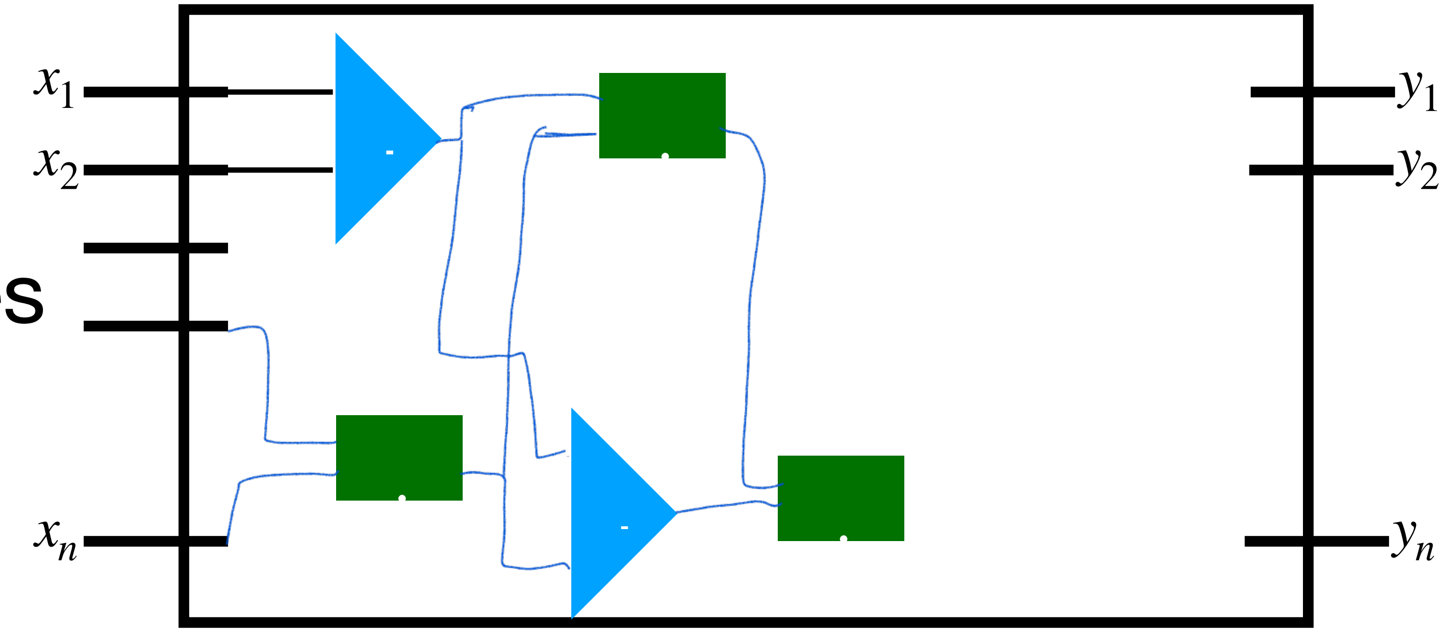
- **Stage I**: Input sharing phase
- **Stage II**: Circuit emulation phase
- **Stage III**: Output reconstruction phase



Stage I: Input Sharing Phase

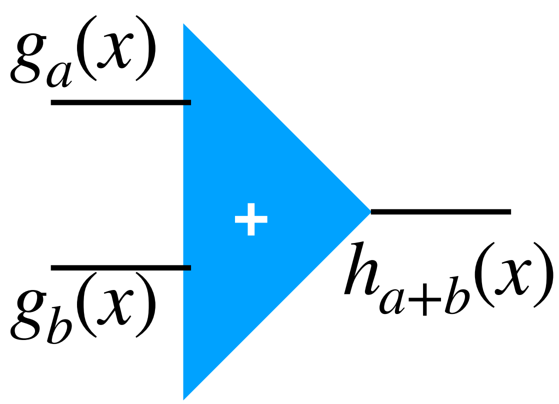
- Each party P_i shares its input x_i
  - It chooses a random polynomial $g_i(x)$ of degree-t for which $g_i(0) = x_i$
  - It sends to each party P_j the share $g_i(\alpha_j)$

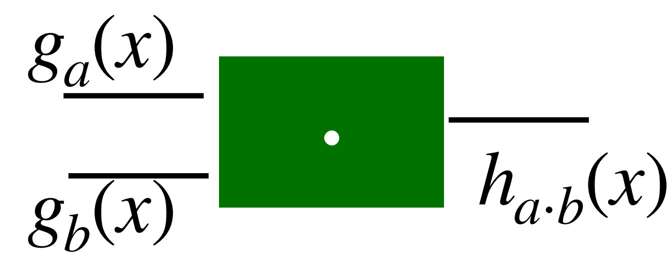
- At the end of this stage each party P_i holds shares $g_1(\alpha_i), \ldots, g_n(\alpha_i)$



Stage II: Circuit Emulation Phase

- We will show secure protocols for two specific functions:





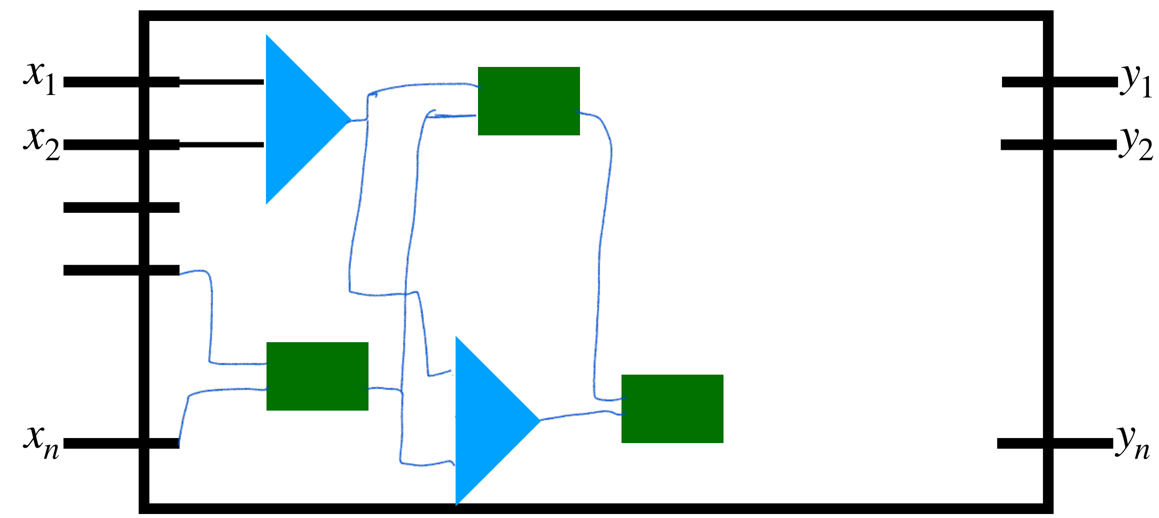
$$f_{\text{add}}\Big(\big(g_a(\alpha_1), g_b(\alpha_1)\big), \ldots, \big(g_a(\alpha_n), g_b(\alpha_n)\big)\Big) = \big(h_{a+b}(\alpha_1), \ldots, h_{a+b}(\alpha_n)\big)$$

$$f_{\text{mult}}\Big(\big(g_a(\alpha_1), g_b(\alpha_1)\big), \ldots, \big(g_a(\alpha_n), g_b(\alpha_n)\big)\Big) = \big(h_{a\cdot b}(\alpha_1), \ldots, h_{a\cdot b}(\alpha_n)\big)$$

- Computing the circuit **gate-by-gate**:
  Computing shares of the output wire of a gate
  from the shares of its input wires

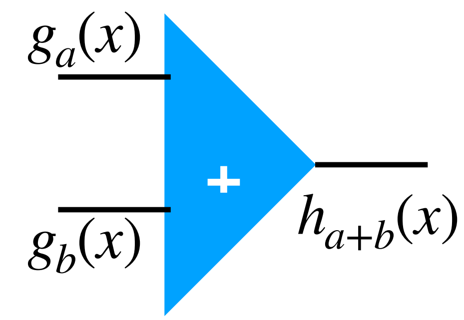
Stage III: Output Reconstruction Phase

- The parties hold shares of all output wires
- Each party P_i holds shares $g_{y_1}(\alpha_i), \ldots, g_{y_n}(\alpha_i)$
  - P_1 is supposed to learn y_1
  - P_2 is supposed to learn y_2
  - …



- All parties send their shares $g_{y_j}(\alpha_1), \ldots, g_{y_j}(\alpha_n)$ to P_j
  - P_j can reconstruct y_j

How to Compute f_{add}?

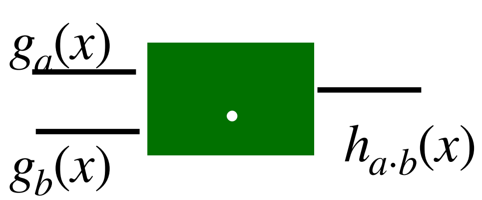


$$f_{\text{add}}\Big(\big(g_a(\alpha_1), g_b(\alpha_1)\big), \ldots, \big(g_a(\alpha_n), g_b(\alpha_n)\big)\Big)$$
$$= \big(h_{a+b}(\alpha_1), \ldots, h_{a+b}(\alpha_n)\big)$$

- Each P_i knows:
  - $g_a(\alpha_i), g_b(\alpha_i)$
  - Simply output $g_a(\alpha_i) + g_b(\alpha_i)$
    - No interaction!
- All parties obtain shares of the polynomial $h_{a+b}(x) := g_a(x) + g_b(x)$
  - Polynomial of degree-t
  - Constant term: $h_{a+b}(0) = g_a(0) + g_b(0) = a + b$

How to Compute f_{mult}?

- Each party P_i holds shares $g_a(\alpha_i), g_b(\alpha_i)$
- Can we simply output $g_a(\alpha_i) \cdot g_b(\alpha_i)$?
  - The parties will obtain shares of the polynomial $h(x) := g_a(x) \cdot g_b(x)$
  - It's constant term is $h(0) = g_a(0) \cdot g_b(0) = a \cdot b$
    - Looks good
- **But…**
  - What is the degree of h?
  - Is h random?



$$f_{\text{mult}}\Big(\big(g_a(\alpha_1), g_b(\alpha_1)\big), \ldots, \big(g_a(\alpha_n), g_b(\alpha_n)\big)\Big) = \big(h_{a\cdot b}(\alpha_1), \ldots, h_{a\cdot b}(\alpha_n)\big)$$

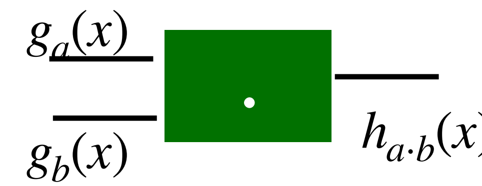
Reminder:

- For any polynomial $h(x)$ with degree $t < n$, there exist constants $\lambda_1, \ldots, \lambda_n$ such that:

$$\lambda_1 \cdot h(\alpha_1) + \ldots + \lambda_n \cdot h(\alpha_n) = h(0) = a \cdot b$$

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{2t} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{2t} \\ \vdots & & & & \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{2t} \end{pmatrix} \begin{pmatrix} ab \\ h_1 \\ \vdots \\ h_{2t} \end{pmatrix} = \begin{pmatrix} h(\alpha_1) \\ h(\alpha_2) \\ \vdots \\ h(\alpha_n) \end{pmatrix}$$

$$\begin{pmatrix} ab \\ h_1 \\ \vdots \\ h_{2t} \end{pmatrix} = \begin{pmatrix} \lambda_1 & \ldots & \lambda_n \\ \vdots & & \\ \ldots & & \end{pmatrix} \begin{pmatrix} h(\alpha_1) \\ h(\alpha_2) \\ \vdots \\ h(\alpha_n) \end{pmatrix}$$



Computing f_{mult}

- Let's take a look again at $h(x) := g_a(x) \cdot g_b(x)$
- Each party P_i can compute $h(\alpha_i)$
- Can we reveal $h(\alpha_i)$ to other parties, or it should be kept secret?
- We know that
  $ab = \lambda_1 \cdot h(\alpha_1) + \ldots + \lambda_n \cdot h(\alpha_n)$
- The protocol for P_i:
  - Compute $h(\alpha_i) := g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - Share $h(\alpha_i)$ using a degree-t polynomial $H_i(x)$
  - Given all the shares that were received $H_1(\alpha_i), \ldots, H_n(\alpha_i)$, output $\lambda_1 \cdot H_1(\alpha_i) + \ldots + \lambda_n \cdot H_n(\alpha_i)$

Simplification of BGW by [GenaroRabinRabin96]

Why Does It Work?

- The parties compute a share on the polynomial
- $H(x) := \lambda_1 H_1(x) + \ldots \lambda_n H_n(x)$
  - Each P_i outputs $H(\alpha_i)$
- **This is a polynomial of degree** t
  - Each one of $H_1(x), \ldots, H_n(x)$ is of degree-t
- **It is random**
  - Each one of $H_1(x), \ldots, H_n(x)$ is random
- **Its constant term is** ab
  - $$\begin{aligned} H(0) &= \lambda_1 H_1(0) + \ldots + \lambda_n H_n(0) \\ &= \lambda_1 h(\alpha_1) + \ldots + \lambda_n h(\alpha_n) = a \cdot b \end{aligned}$$
- Perfect.

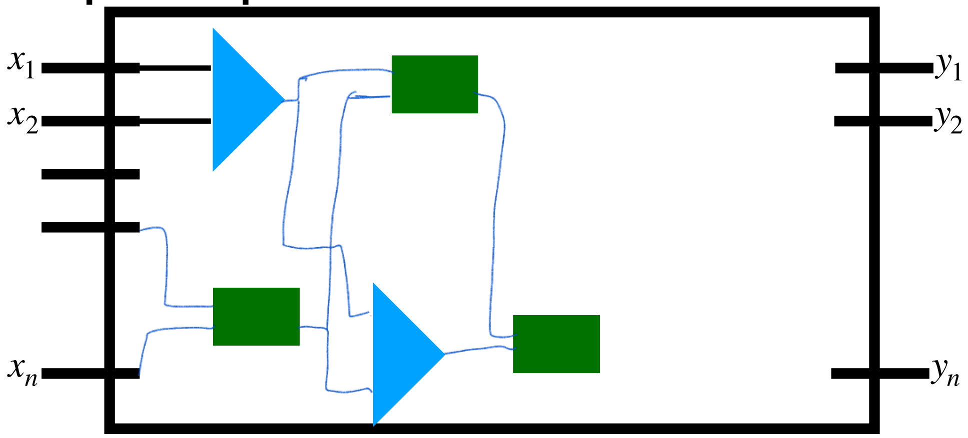
Semi-Honest: Conclusion

- For every n-ary function $f(x_1, \ldots, x_n)$, there exists a protocol for computing f with ***perfect security*** in the presence of **a semi-honest** adversary controlling $t < n/2$ parties

Why do we need honest majority?

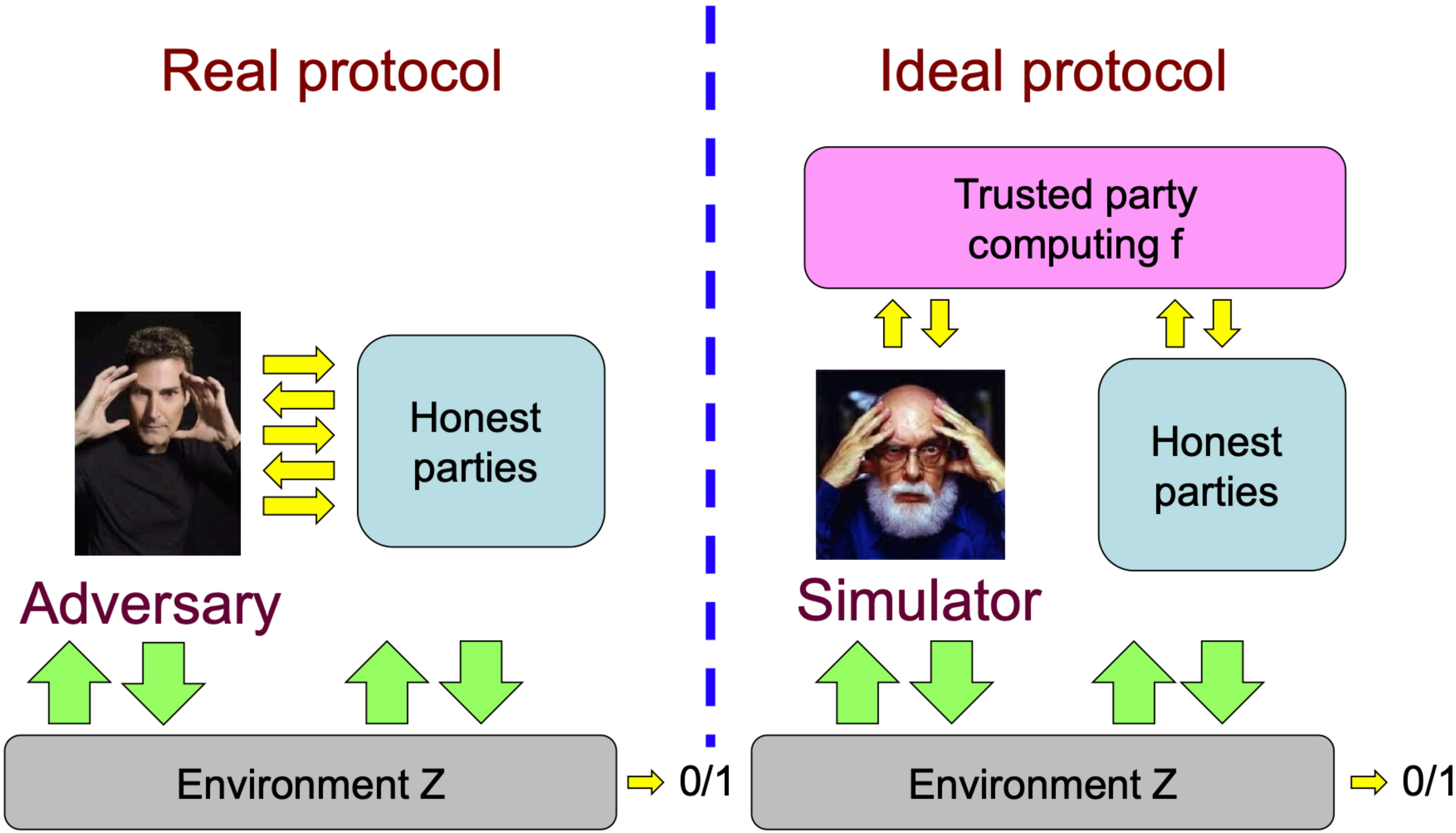
Security

- What is the view of the corrupted parties?
  - **Input sharing phase:**
    t shares on polynomials of honest parties
  - **Circuit emulation phase:**
    In each multiplication, the adversary receives t shares on each one of the polynomials $H_1(x), \ldots, H_n(x)$
  - **Output reconstruction phase:**
    Given the t shares on the output wires of the corrupted parties + the outputs of the corrupted parties to the simulator as input $\Longrightarrow$ reconstruct the polynomial and send the remaining shares



The Malicious Case

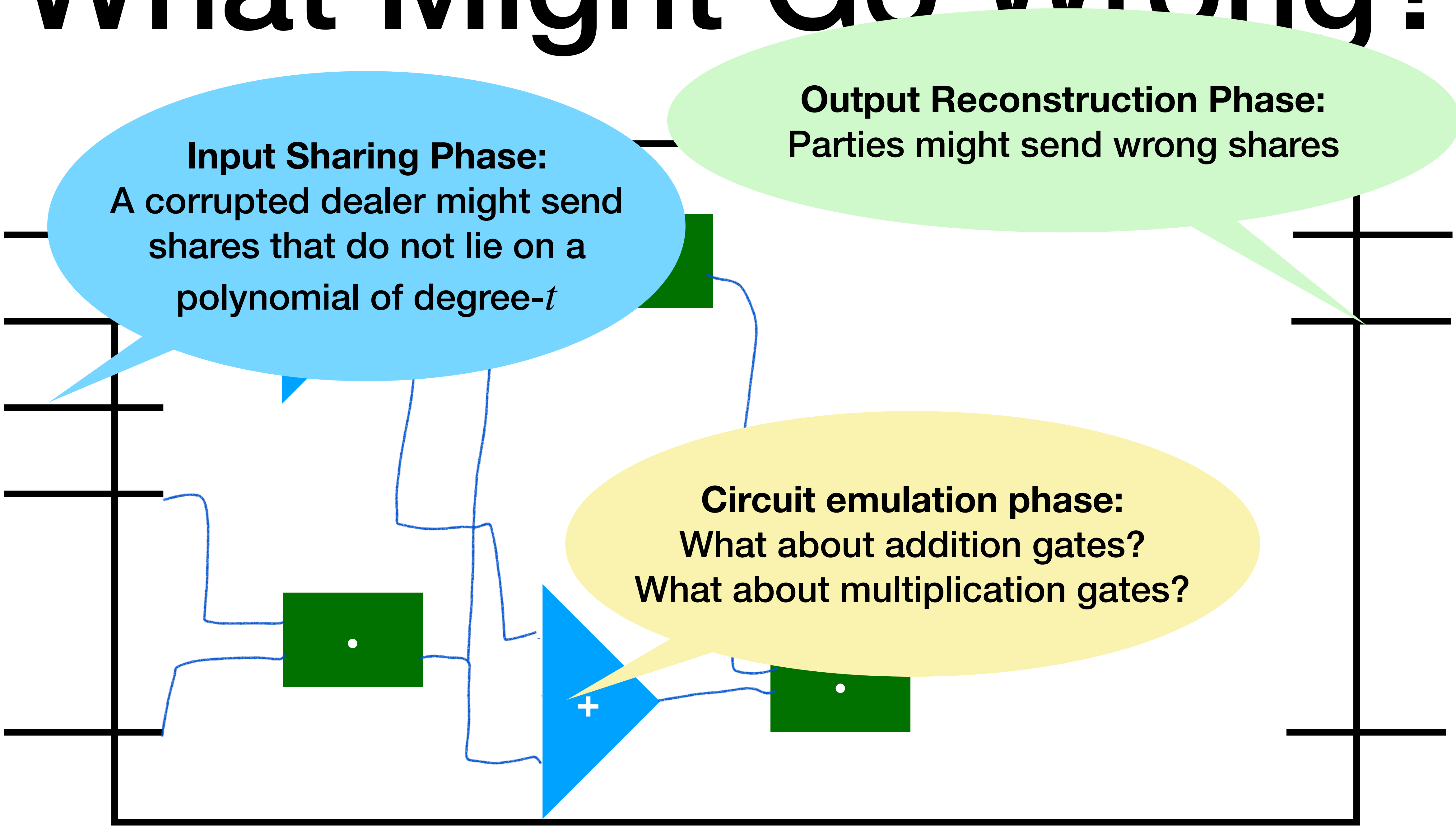
Real/Ideal Paradigm



Malicious Security

- The parties jointly compute $f(x_1, \ldots, x_n)$:
  - The honest parties provide true inputs
  - The corrupted parties might provide any input they like
    - If do not cooperating, the honest parties can choose some default inputs for them
- **Privacy:** The adversary does not learn any information on the honest parties' inputs
- **Guaranteed output delivery**: The adversary cannot prevent the honest parties from obtaining outputs

What Might Go Wrong?



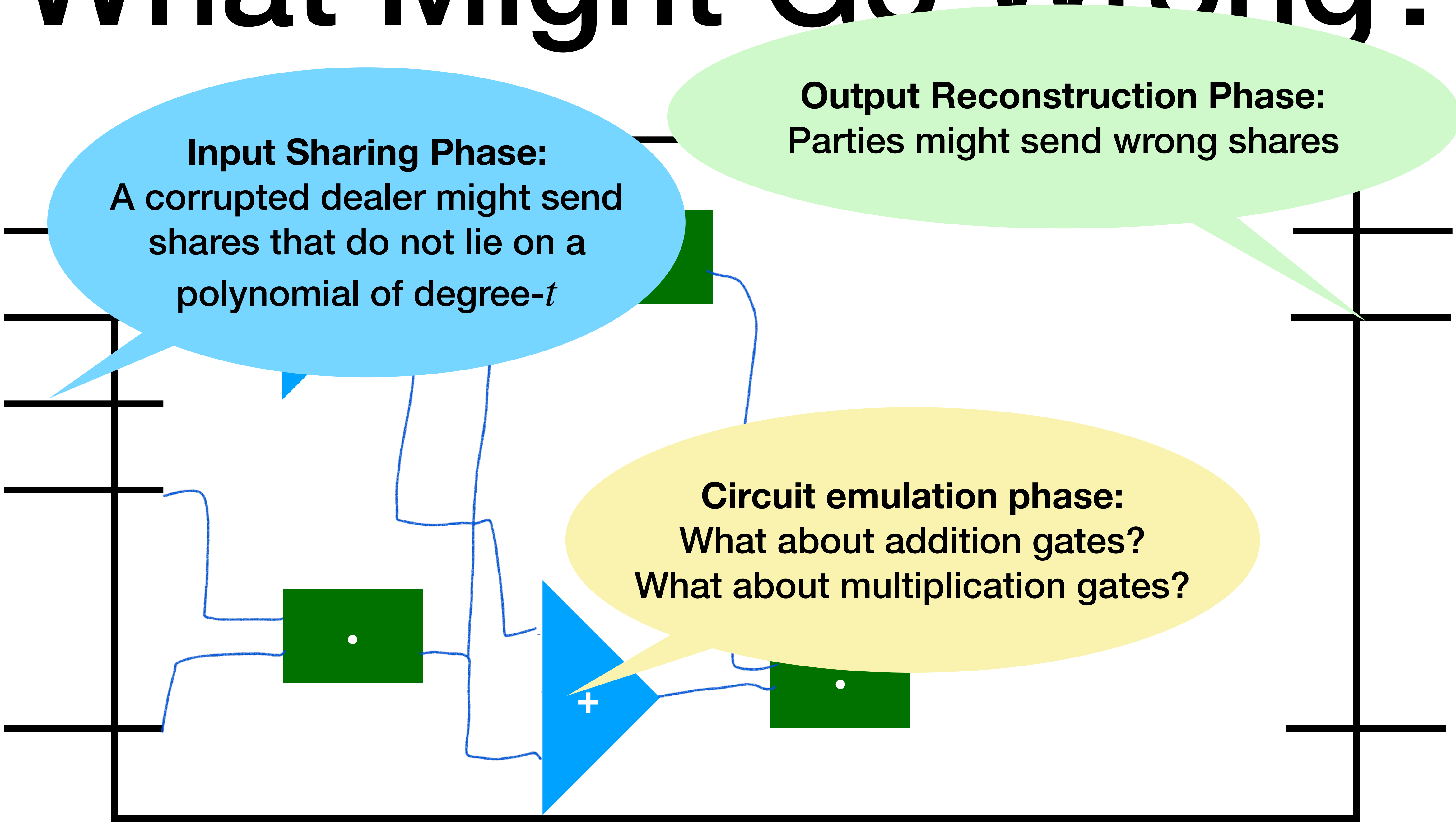
Reminder - VSS

- We saw on Monday:
  Let $t < n/3$. There exists a perfectly secure Verifiable Secret Sharing protocol in the presence of a malicious adversary
  - **Privacy:**
    For an honest dealer, the adversary learns nothing about s
  - **Consistency:**
    The outputs of the honest party are consistent with some s^* even if the adversary is corrupted (agreement)
  - **Correctness:**
    For an honest dealer, consistency holds with $s^* = s$
- **Reconstruction:**
  Even if corrupted parties send wrong shares, honest parties can still recover the secret

Before We Proceed

- Note that if the function f does not contain any multiplication gates - we are done!
- Which functions do not contain multiplication gates?
  - All linear functions!
    - **Multiplication with a vector:**
      For a public vector $(a_1, \ldots, a_n)$
      $(x_1, \ldots, x_n) \to a_1x_1 + \ldots + a_nx_n$
    - **Multiplication with a matrix:**
      For a public matrix $A \in \mathbb{F}^{n \times t}$:
      $(x_1, \ldots, x_n) \to A \cdot (x_1, \ldots, x_n) = (y_1, \ldots, y_n)$

What Might Go Wrong?



What Might Go Wrong?

Circuit Emulation Phase

- **Multiplication gate:**

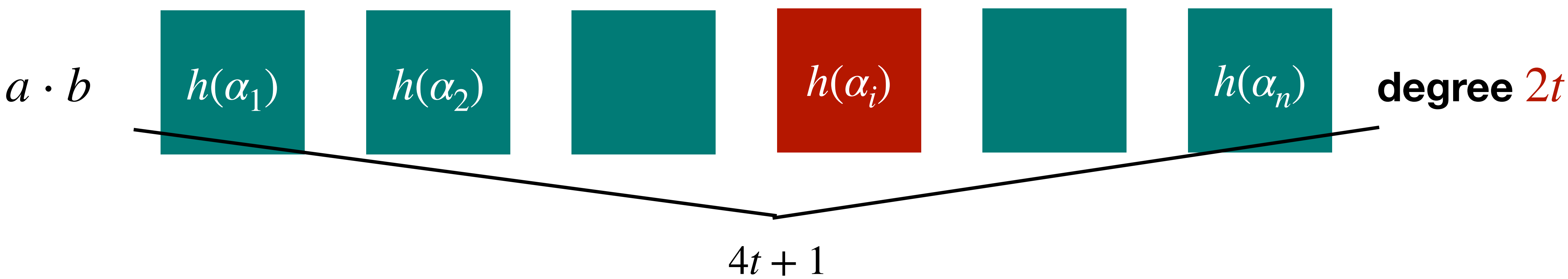
  The protocol for P_i

  Input: $g_a(\alpha_i), g_b(\alpha_i)$

  - Compute $h(\alpha_i) := g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - Share $h(\alpha_i)$ using a degree-t polynomial $H_i(x)$
  - Given all the shares that were received $H_1(\alpha_i), \ldots, H_n(\alpha_i)$
  - Output $\lambda_1 \cdot H_1(\alpha_i) + \ldots + \lambda_n \cdot H_n(\alpha_i)$

Simplified Case: $t < n/4$

- Let's take a look again at the polynomial $h(x) := g_a(x) \cdot g_b(x)$
- This is a polynomial of degree $2t$
- Each party computes a share on this polynomial by just computing $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$

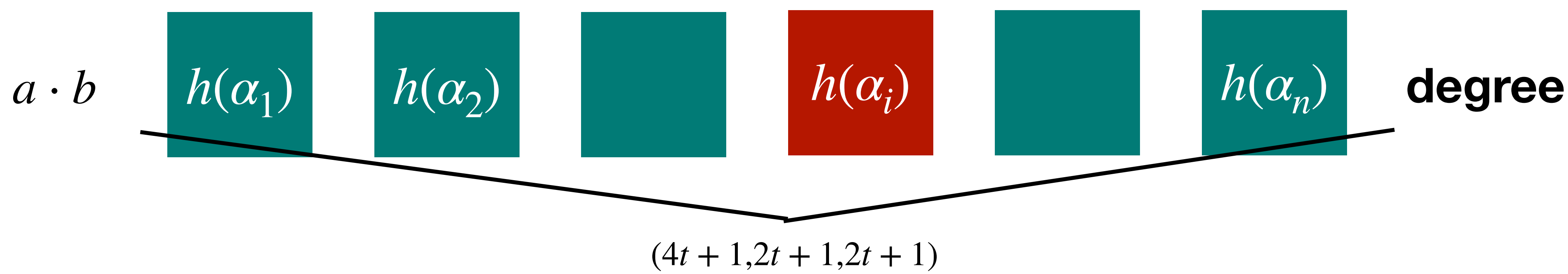


- Can we somehow correct the wrong shares?
- **Recall**: Reed Solomon code is $(n, k+1, n-k)$-code, can correct $(n-k-1)/2$ errors
  - When $n = 3t+1$, for $k = 2t$ we have $(3t+1, 2t+1, t+1)$-code, can correct $t/2$ errors
  - When $n = 4t+1$, for $k = 2t$ we have $(4t+1, 2t+1, 2t+1)$-code, can correct t errors

Facts From Error Correcting Code

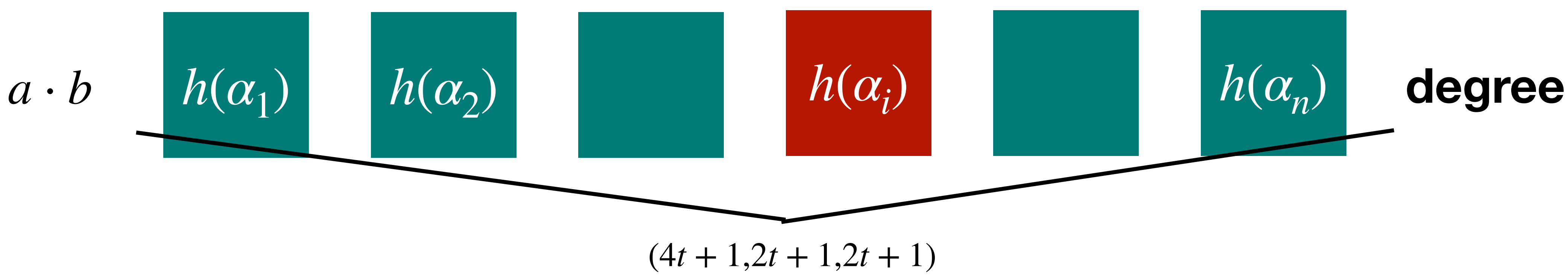
- Let $C \subset \Sigma^n$ be a (n, k, d)-linear code
- **Generator matrix:** $G \in \Sigma^{k \times n}$: maps "messages" into codewords
  For $\mathbf{m} \in \Sigma^k$, we have that $\mathbf{m} \cdot G \in \mathbb{F}^n$ is a codeword
- **A parity check matrix:** $H \in \Sigma^{(n-k) \times n}$ matrix
  - Satisfies $G \cdot H^T = 0^{k \times (n-k)}$
- For every codeword $\mathbf{c} \in C$ (i.e., there exists some $\mathbf{m} \in \Sigma^k$ such that $\mathbf{m} \cdot G = \mathbf{c}$):
  - $\mathbf{c} \cdot H^T = 0$
- For every "noise" codeword $\widetilde{\mathbf{c}} = \mathbf{c} + \mathbf{e} \in \Sigma^n$ where $\mathbf{c} \in C$ and $\mathbf{e} \in \Sigma^n$ is of distance $(d-1)/2$ from $\mathbf{0}$
  - $\widetilde{\mathbf{c}} \cdot H^T = (\mathbf{c} + \mathbf{e}) \cdot H^T = \mathbf{e} \cdot H^T$
  - It is possible to find $\mathbf{e}$ from $\mathbf{e} \cdot H^T$
  - $\mathbf{e} \cdot H^T$ does not contain any information about $\mathbf{m}$

In Our Simplified Case ($n = 4t + 1$)

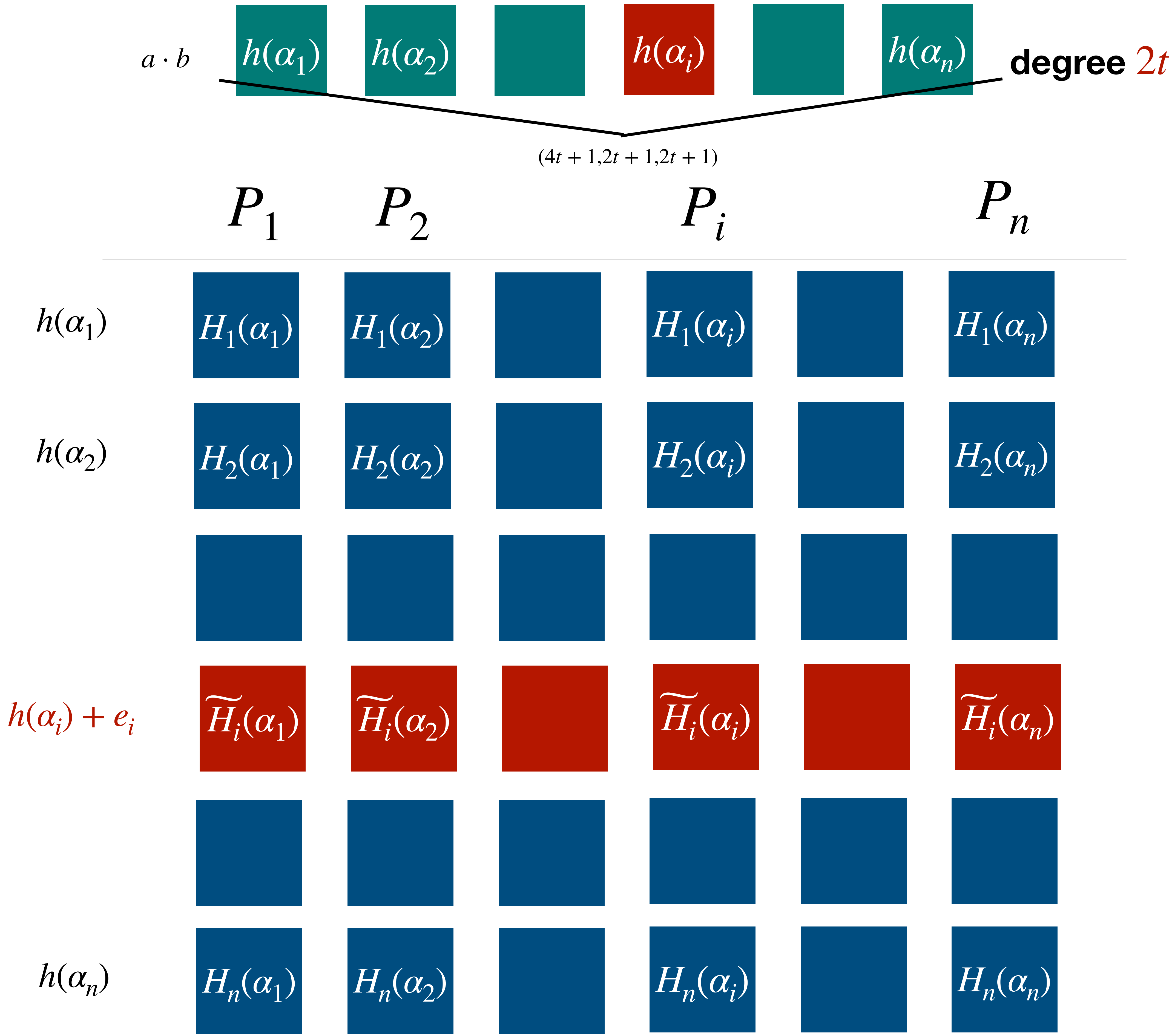


- Each party computes $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$ and sub-shares it
- Let $\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} = (h(\alpha_1), \ldots, h(\alpha_n))$ and the distance of $\mathbf{e}$ from $\mathbf{0}$ is at most t
- We run a check. If some P_i inputs something wrong, we want to identify it, and "correct" it
    - I.e., the honest parties will change their sub-shares of P_i to $h(\alpha_i)$

The Check



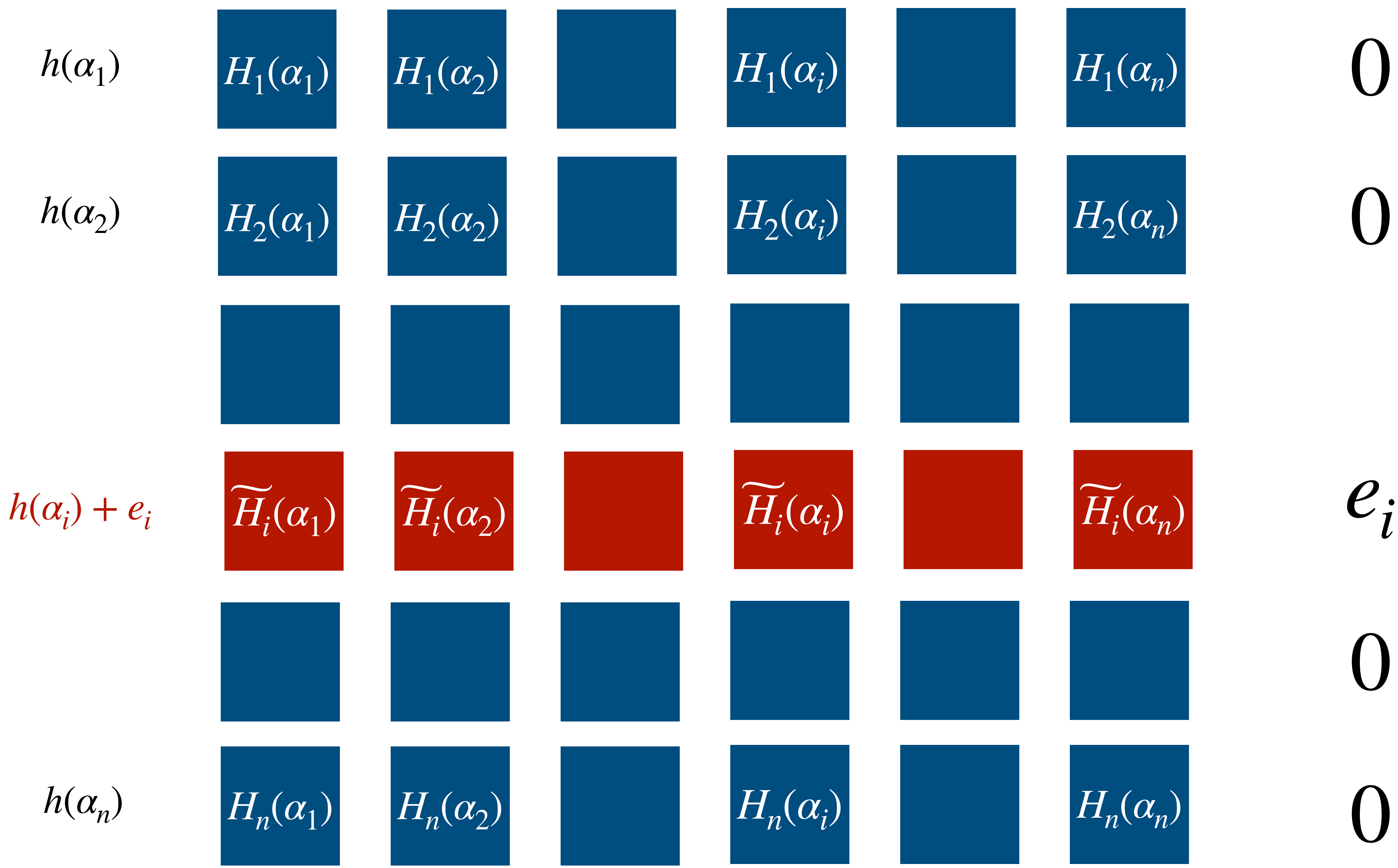
- Each party P_i sub-share its input using some $H_i(x)$
  - $H_i(x)$ hides $h(\alpha_i)$
- Parties compute the "circuit" $\widetilde{\mathbf{c}} \cdot H^T$
  - Reconstruct $\mathbf{e} = (e_1, \ldots, e_n)$
- The parties can see if there are errors, where, and what
  - For every $e_i \neq 0$:
    - Reconstruct $H_i(0)$
    - "Correct" the sub-share to $H_i(0) - e_i$



$a \cdot b$
$h(\alpha_1)$
$h(\alpha_2)$
$h(\alpha_i)$
$h(\alpha_n)$
degree $2t$
$(4t+1, 2t+1, 2t+1)$
P_1
P_2
P_i
P_n
$h(\alpha_1)$
$H_1(\alpha_1)$
$H_1(\alpha_2)$
$H_1(\alpha_i)$
$H_1(\alpha_n)$
$h(\alpha_2)$
$H_2(\alpha_1)$
$H_2(\alpha_2)$
$H_2(\alpha_i)$
$H_2(\alpha_n)$
$h(\alpha_i) + e_i$
$\widetilde{H}_i(\alpha_1)$
$\widetilde{H}_i(\alpha_2)$
$\widetilde{H}_i(\alpha_i)$
$\widetilde{H}_i(\alpha_n)$
$h(\alpha_n)$
$H_n(\alpha_1)$
$H_n(\alpha_2)$
$H_n(\alpha_i)$
$H_n(\alpha_n)$

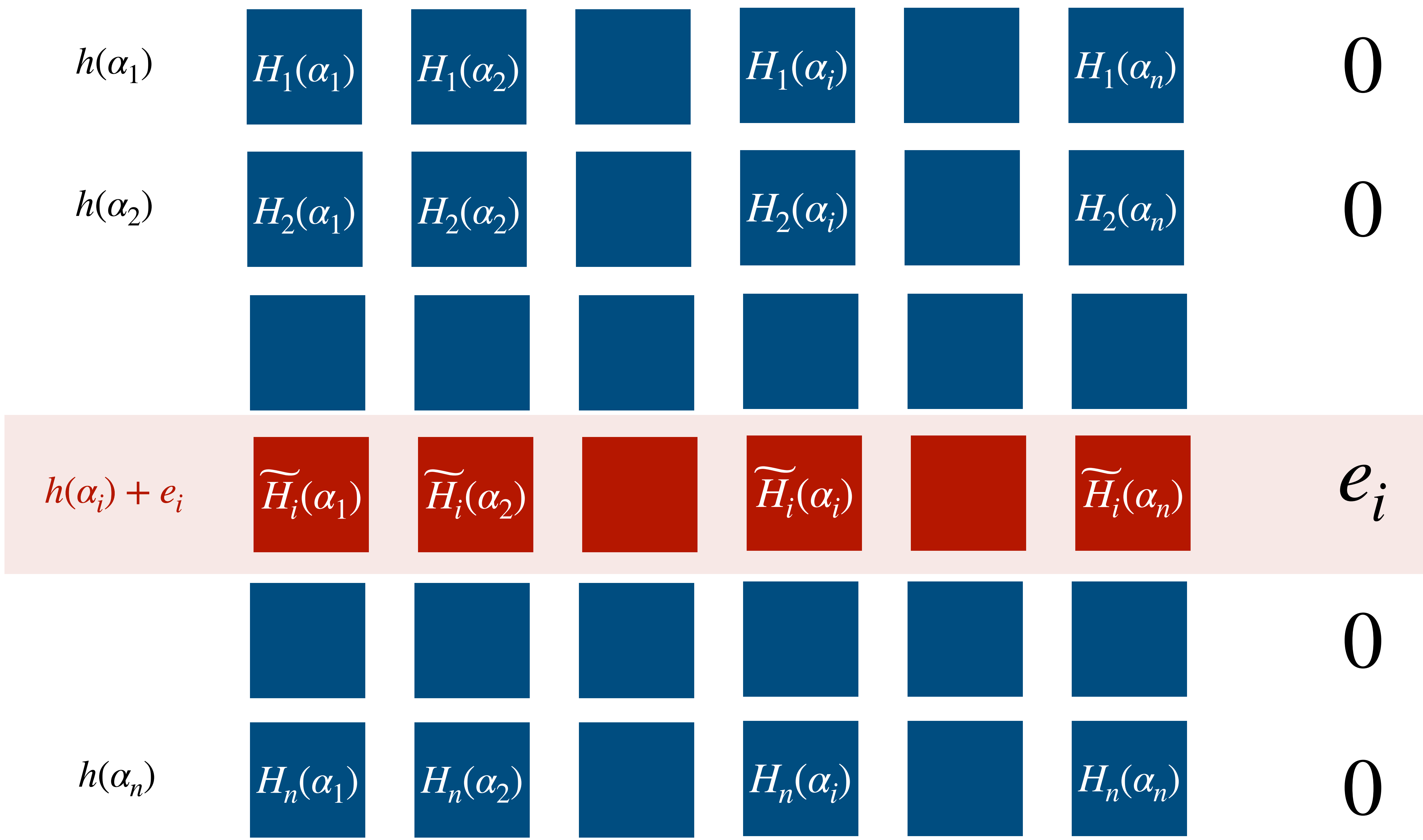
Multiply with the parity-check matrix H^T

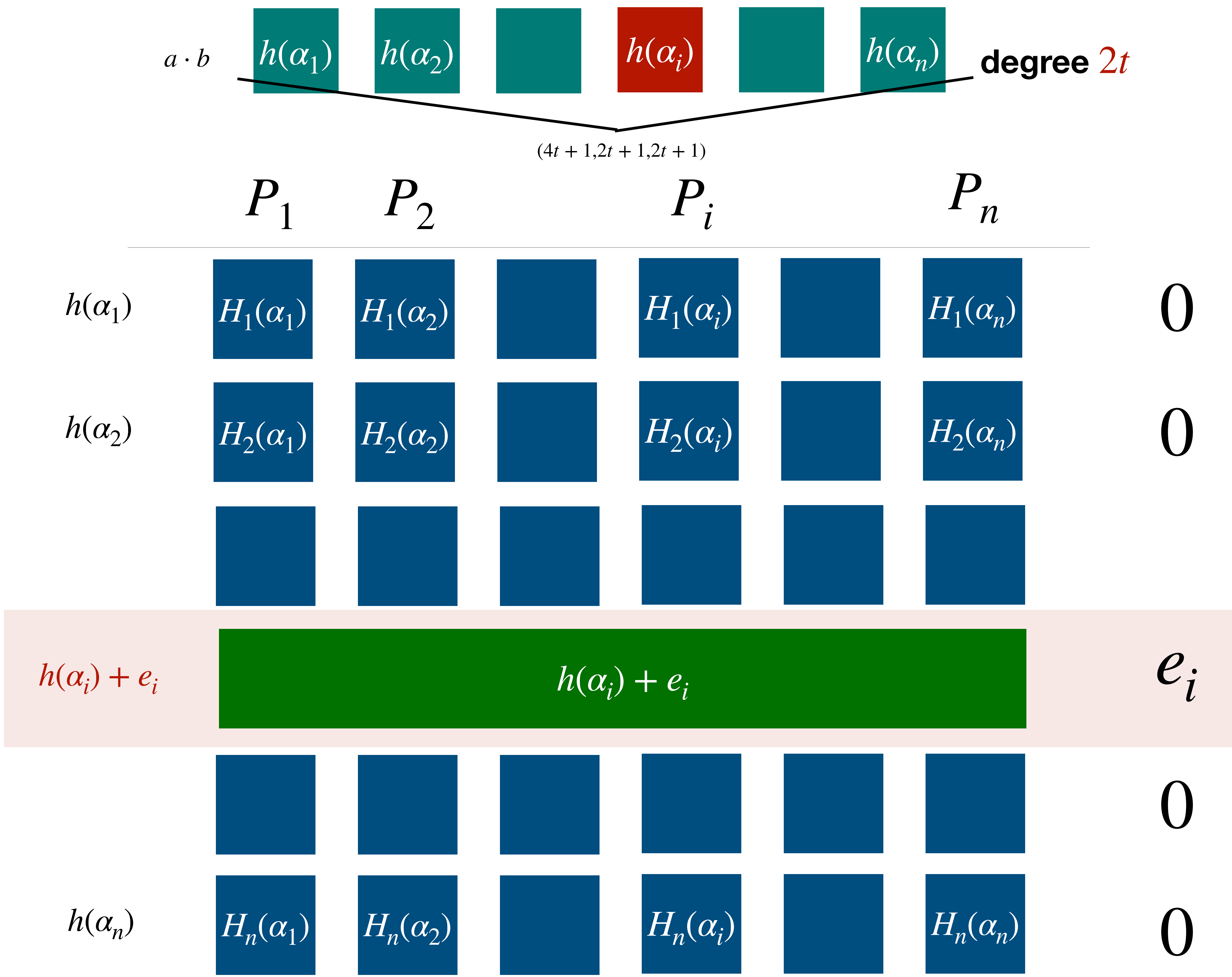
Reconstruct $\mathbf{e} = (e_1, \ldots, e_n)$

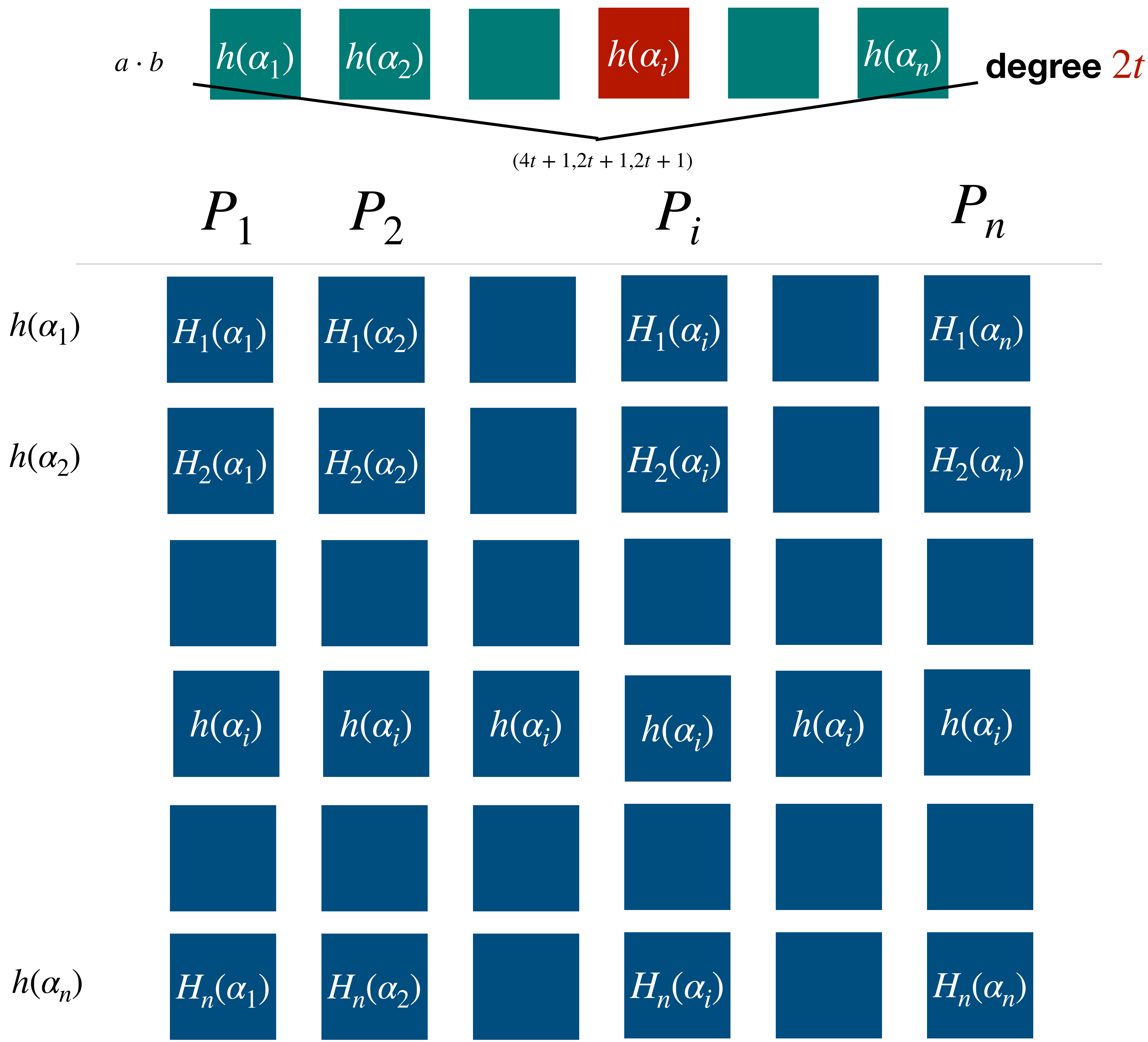


Multiply with the parity-check matrix H^T

Reconstruct $\mathbf{e} = (e_1, \ldots, e_n)$




$a \cdot b$
$h(\alpha_1)$
$h(\alpha_2)$
$h(\alpha_i)$
$h(\alpha_n)$
degree $2t$
$(4t+1, 2t+1, 2t+1)$
P_1
P_2
P_i
P_n
$h(\alpha_1)$
$H_1(\alpha_1)$ $H_1(\alpha_2)$ $H_1(\alpha_i)$ $H_1(\alpha_n)$
0
$h(\alpha_2)$
$H_2(\alpha_1)$ $H_2(\alpha_2)$ $H_2(\alpha_i)$ $H_2(\alpha_n)$
0
$h(\alpha_i) + e_i$
$h(\alpha_i) + e_i$
e_i
0
$h(\alpha_n)$
$H_n(\alpha_1)$ $H_n(\alpha_2)$ $H_n(\alpha_i)$ $H_n(\alpha_n)$
0


$a \cdot b$
$h(\alpha_1)$
$h(\alpha_2)$
$h(\alpha_i)$
$h(\alpha_n)$
degree $2t$
$(4t+1, 2t+1, 2t+1)$
P_1
P_2
P_i
P_n
$h(\alpha_1)$
$H_1(\alpha_1)$
$H_1(\alpha_2)$
$H_1(\alpha_i)$
$H_1(\alpha_n)$
$h(\alpha_2)$
$H_2(\alpha_1)$
$H_2(\alpha_2)$
$H_2(\alpha_i)$
$H_2(\alpha_n)$
$h(\alpha_i)$
$h(\alpha_i)$
$h(\alpha_i)$
$h(\alpha_i)$
$h(\alpha_i)$
$h(\alpha_i)$
$h(\alpha_n)$
$H_n(\alpha_1)$
$H_n(\alpha_2)$
$H_n(\alpha_i)$
$H_n(\alpha_n)$

Conclusion - Multiplication with $n = 4t + 1$

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- Each party multiplies $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$
- The parties sub-share $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - And then they check and "correct" wrong inputs
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

What About $n = 3t + 1$?

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- Each party multiplies $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$
- The parties sub-share $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - And then they check and "correct" wrong inputs
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

What About $n = 3t + 1$?

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- Each party multiplies $h(\alpha_i) = g_a(\alpha_i) \cdot$
- The parties sub-share $h(\alpha_i) = g_a(\alpha_i) \cdot g_b$
  - **And then they check and "correct" wrong inputs**
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

When $n = 3t + 1$, we can correct only $t/2$ errors for a polynomial of degree $2t$

What About $n = 3t + 1$?

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- Each party sub-shares $g_a(\alpha_i)$ and $g_b(\alpha_i)$
  - Since $g_a(x), g_b(x)$ are of degree t,
    we can guarantee that right values where shared
- Each party sub-shares $h(\alpha_i) = g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - And "proves" that those sub-shares agree with the sub-shares of $g_a(x), g_b(x)$
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

Main Theorems

- We saw:
  - ***Perfectly secure*** protocol in the **semi-honest** model, for $t < n/2$ [BGW88,CCD88]
  - **Perfectly secure** protocol in the **malicious** model, for $t < n/4$
- It holds:
  - **Perfectly secure** protocol in the **malicious** model, for $t < n/3$ [BGW88]
    - Statistically secure [CCD88]
  - **Statistically secure** protocol in the **malicious** model, for $t < n/2$ (assuming broadcast) [RB89]

What About $n = 3t + 1$?

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- Each party sub-shares $g_a(\alpha_i)$ and $g_b(\alpha_i)$
  - Since $g_a(x), g_b(x)$ are of degree t,
    we can guarantee that right values where shared
- Each party sub-shares $g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - And "proves" that those sub-shares agree with the sub-shares of $g_a(x), g_b(x)$
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

What About $n = 3t + 1$?

- **Input:** Each party holds $g_a(\alpha_i), g_b(\alpha_i)$
- ~~Each party sub-shares $g_a(\alpha_i)$ and $g_b(\alpha_i)$~~
  - ~~Since $g_a(x), g_b(x)$ are of degree t, we can guarantee that right values where shared~~
- Each party sub-shares $g_a(\alpha_i) \cdot g_b(\alpha_i)$
  - And “proves” that those sub-shares agree with the sub-shares of $g_a(x), g_b(x)$
- Now each party P_j holds a share on each one of the polynomials $H_1(x), \ldots, H_n(x)$ that hide $h(\alpha_1), \ldots, h(\alpha_n)$, resp.
  - That is, P_j holds $H_1(\alpha_j), \ldots, H_n(\alpha_j)$
- **Output:** $\lambda_1 \cdot H_1(\alpha_j) + \ldots + \lambda_n \cdot H_n(\alpha_j)$

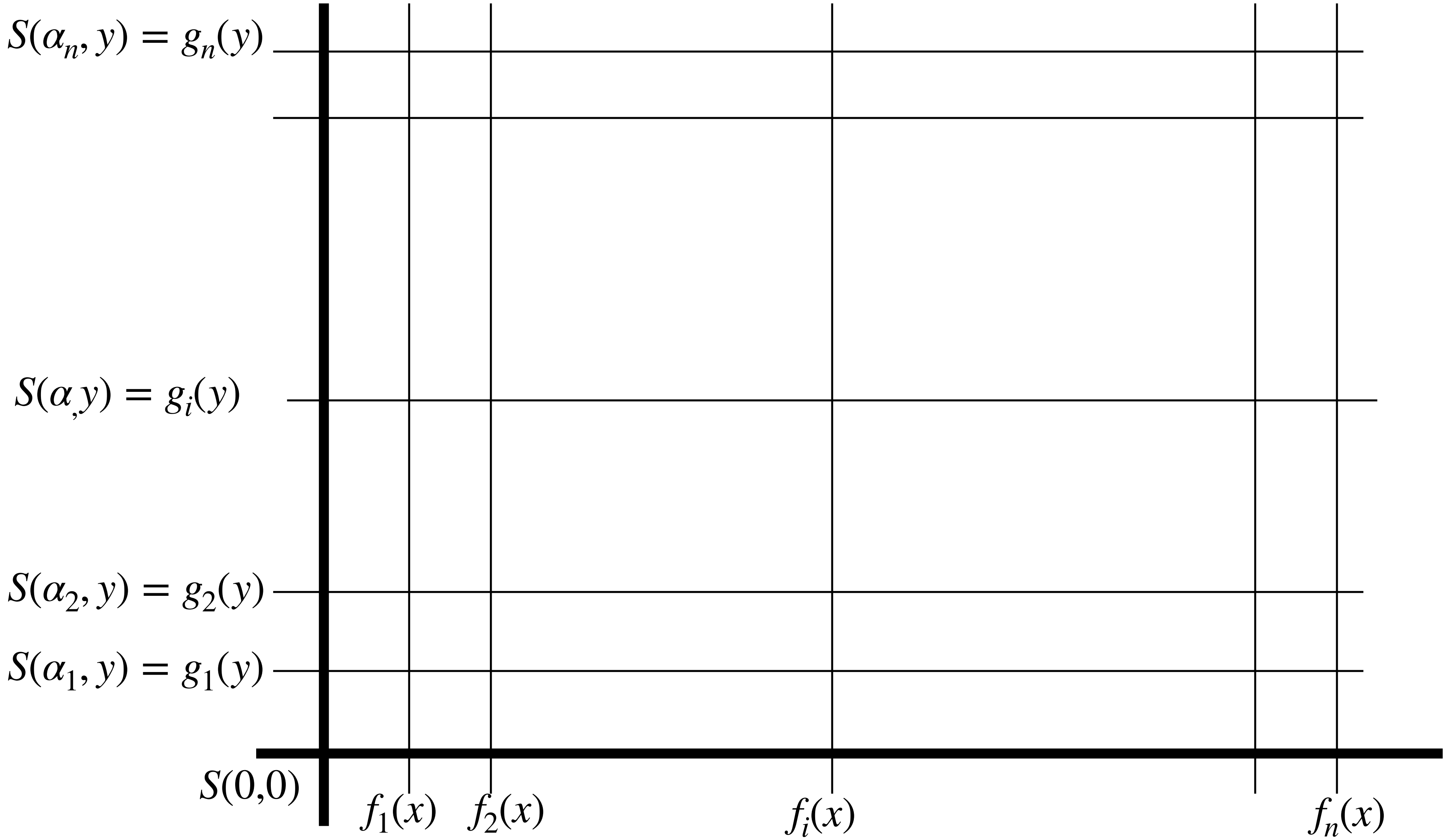
Changing the Invariant

[A-Lindell-Rabin11]

- Instead of having:
  "Each value on a wire is hidden with a univariate polynomial of degree-t"

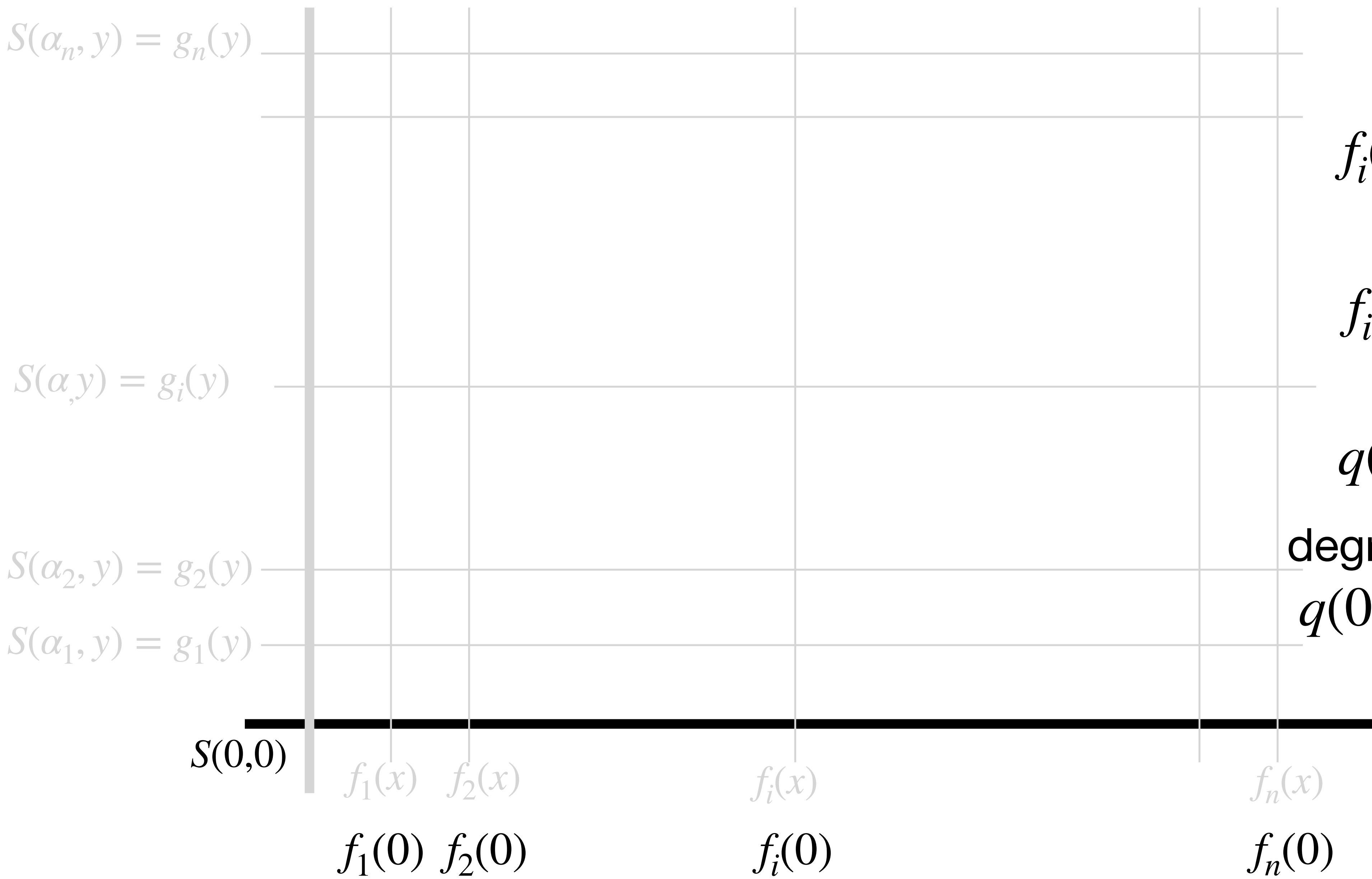
- We can work with:
  *"Each value on a wire is hidden with a **bivariate** polynomial of degree-t"*

Recall VSS



$S(\alpha_n, y) = g_n(y)$
$S(\alpha_i y) = g_i(y)$
$S(\alpha_2, y) = g_2(y)$
$S(\alpha_1, y) = g_1(y)$
$S(0,0)$
$f_1(x)$
$f_2(x)$
$f_i(x)$
$f_n(x)$

Recall VSS



$f_i(x) = S(x, \alpha_i)$

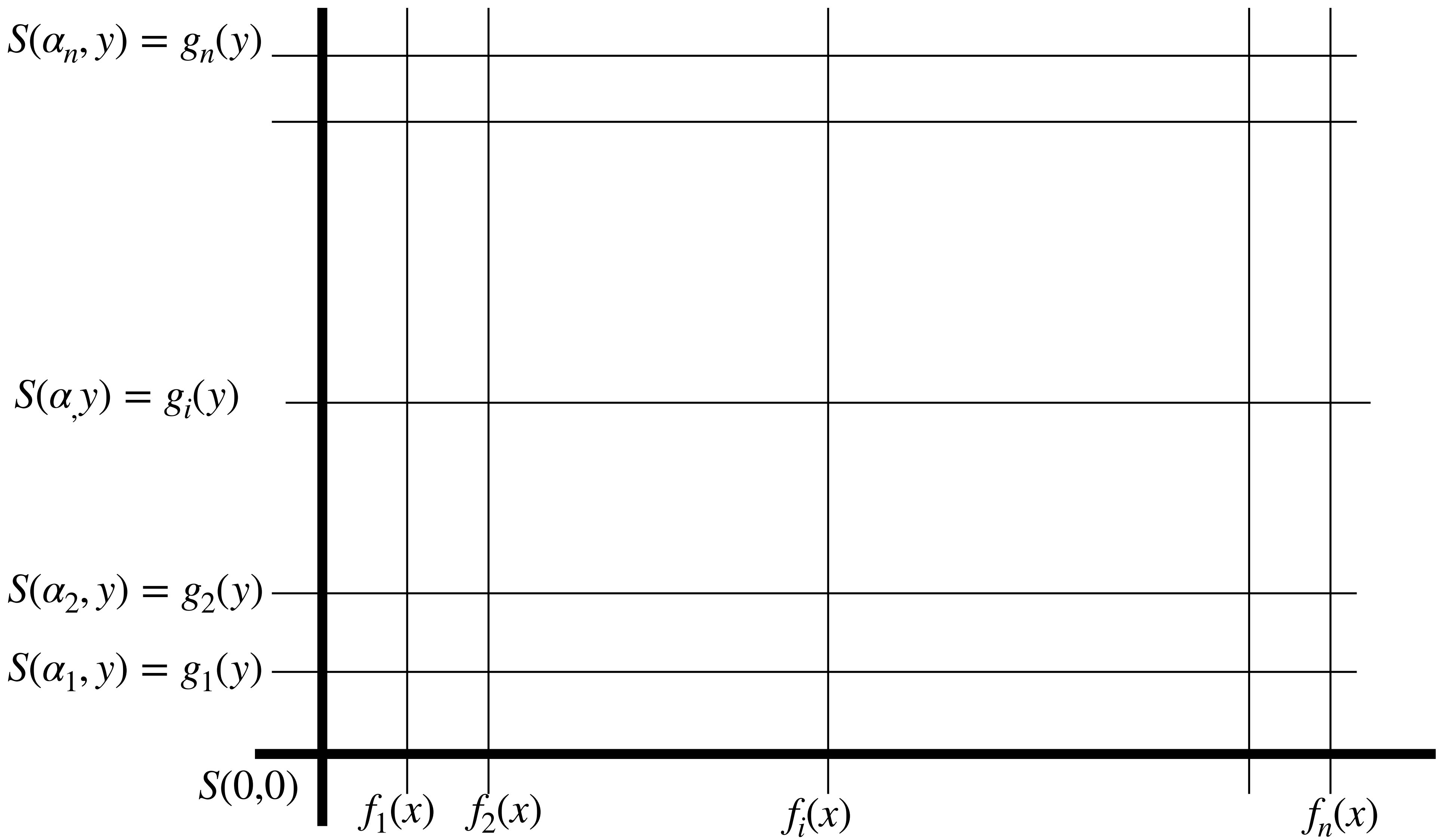
$f_i(0) = S(0,\alpha_i)$

$q(y) = S(0,y)$

degree-t polynomial

$q(0) = S(0,0) = s$

But… This is Exactly Sub-Share



Conclusion

- We saw:
  - ***Perfectly secure*** protocol in the **semi-honest** model, for $t < n/2$ [BGW88,CCD88]
  - **Perfectly secure** protocol in the **malicious** model, for $t < n/4$
- It holds:
  - **Perfectly secure** protocol in the **malicious** model, for $t < n/3$ [BGW88]
    - Statistically secure [CCD88]
  - **Statistically secure** protocol in the **malicious** model, for $t < n/2$ (assuming broadcast) [RB89]

Thank You!!

References

- Michael Ben-Or, Shafi Goldwasser, Avi Widgerson:
  Completeness theorems for non-cryptographic fault-tolerant distributed computation
- David Chaum, Claude Crépeau, Ivan Damgård:
  Multiparty Unconditionally Secure Protocols
- Tal Rabin, Michael Ben-Or:
  Verifiable Secret Sharing and Multiparty Protocols with Honest Majority
- Gilad Asharov, Yehuda Lindell:
  A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation.
- Gilad Asharov, Yehuda Lindell, Tal Rabin:
  Perfectly-Secure Multiplication for Any t < n/3.
- Ronald Cramer, Ivan Damgård, Jesper Buus Nielsen:
  Secure Multiparty Computation and Secret Sharing. Cambridge University Press 2015, ISBN 9781107043053