CS 65500
Advanced Cryptography

Lecture 13: Reducing Communication in Semi-Honest BGW

Instructor: Aarushi Goel

Spring 2025

Communication Complexity of Semi-Honest of BGW

→ Input Sharing: For every input, the party holding that input sends a share to every other party : $\underbrace{n}_{\#\text{parties}} \times \underbrace{|I|}_{\#\text{inputs}}$ field elements

→ Circuit Evaluation: For each gate in the circuit:

* Addition gates: No communication
* Multiplication gate: Each party sends a share to every other party : n^2 field elements

→ Output Reconstruction: For each output wire, each party sends their share to every other party : $n^2 \times |O|$ field elements

Communication Complexity of Semi-Honest of BGW

→ Input Sharing: For every input, the party holding that input sends a share to every other party : $\underbrace{n}_{\#\text{parties}} \times \underbrace{|I|}_{\#\text{inputs}}$ field elements

→ Circuit Evaluation: For each gate in the circuit:

* Addition gates: No communication
* Multiplication gate: Each party sends a share to every other party : n^2 field elements ← Can we reduce this?

→ Output Reconstruction: For each output wire, each party sends their share to every other party : $n^2 \times |O|$ field elements

Agenda

Two ways of reducing communication:

1. Leveraging (input-independent) pre-processing
2. Ammortization when Computing Multiple copies of a function.

Approach - I

Leveraging Input-Independent Preprocessing

BGW with PreProcessing

* Pre-processing Phase
* Input-Sharing Phase
* Circuit-Evaluation Phase
* Output-Reconstruction Phase

Input-Independent Preprocessing (Beaver Triples)

→ Let us assume that before beginning the computation, parties *magically* get secret shares of correlated random field elements. — Known as Beaver triples

→ In particular, for each multiplication gate m, the parties obtain secret shares of (r_A, r_B, r_C), where $r_A, r_B \xleftarrow{\$} \mathbb{F}$ and $r_C = r_A \cdot r_B$

→ Crucially, none of the parties know r_A, r_B, r_C for any multiplication gate

Multiplication using Beaver Triples

given $[a]_t$, $[b]_t$ and a Beaver triple $([r_A]_t, [r_B]_t, [r_C]_t)$, parties need to compute $[c]_t$.

1. $\forall i \in [n]$, party P_i computes and sends $d_i = a_i - r_{A_i}$ and $e_i = b_i - r_{B_i}$ to party 1.

2. Party 1 uses $d_1 \ldots d_n$ & $e_1 \ldots e_n$ to reconstruct d & e. It sends d & e to all parties.

3. $\forall i \in [n]$, party P_i computes $c_i = d \cdot e + r_{C_i} + d \cdot r_{B_i} + e \cdot r_{A_i}$

Is c_i a valid secret sharing of c?

$$d = a - r_A \qquad\qquad e = b - r_B$$

$$
\begin{aligned}
& d \cdot e + r_c + d \cdot r_B + e \cdot r_A \\
= \;& (a - r_A)(b - r_B) + r_c + (a - r_A)\, r_B + (b - r_B)\, r_A \\
= \;& ab - \cancel{r_A b} - \cancel{r_B \cdot a} + \cancel{r_A \cdot r_B} + \cancel{r_c} + \cancel{a\, r_B} - \cancel{r_A \cdot r_B} + \cancel{b\, r_A} - \cancel{r_B\, r_A} \\
= \;& ab = c
\end{aligned}
$$

Hence,

$$d \cdot e + [r_c]_t + d \cdot [r_B]_t + e \cdot [r_A]_t = [c]_t$$

Exercise: Think if this idea can also be used in the GMW Protocol with additive secret sharing.

Communication Complexity (per multiplication)

1. $\forall i \in [n]$, party P_i computes and sends $d_i = a_i - r_{Ai}$ and $e_i = b_i - r_{Bi}$ to party 1. : 2n field elements

2. Party 1 uses $d_1 \dots d_n$ & $e_1 \dots e_n$ to reconstruct d & e. It sends d & e to all parties : 2n field elements

3. $\forall i \in [n]$, party P_i computes $c_i = d \cdot e + r_{Ci} + d \cdot r_{Bi} + e \cdot r_{Ai}$

Overall, the communication complexity is 4n field elements as compared to n^2 field elements in vanilla BGW

BGW with PreProcessing

* **Pre-processing Phase**: generate a Beaver triple for each multiplication gate. (We will learn how to do this efficiently later in the course)

* **Input-Sharing Phase**: Similar to vanilla BGW

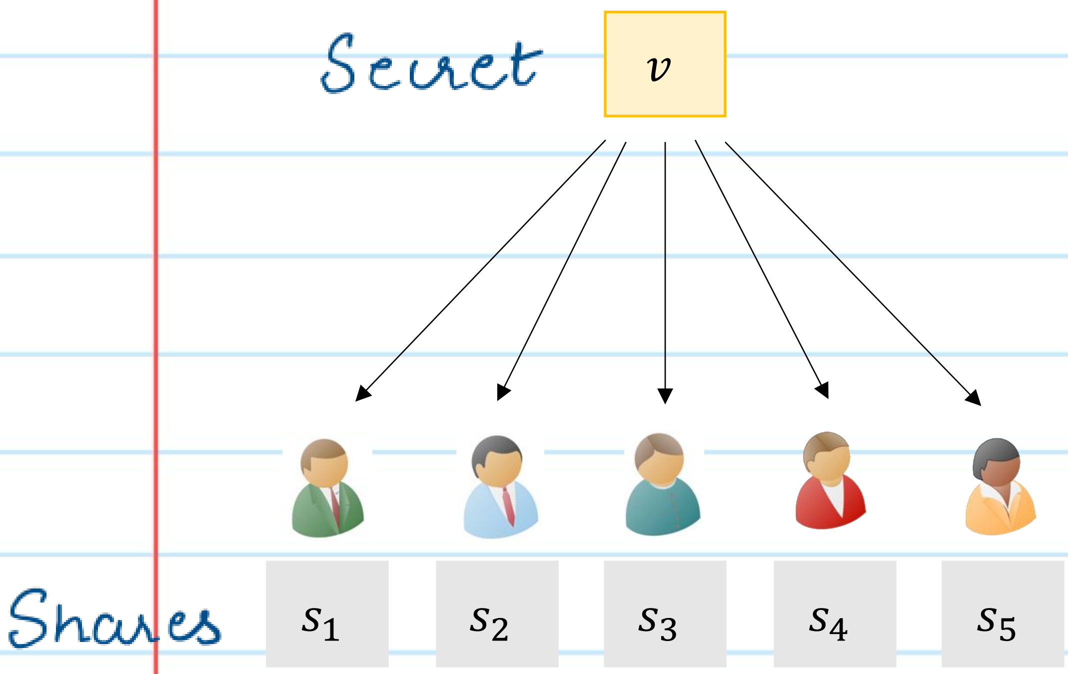
* **Circuit-Evaluation Phase**: Addition is similar to vanilla BGW. Multiplication using Beaver triples

* **Output-Reconstruction Phase**: Similar to vanilla BGW.

Approach - II

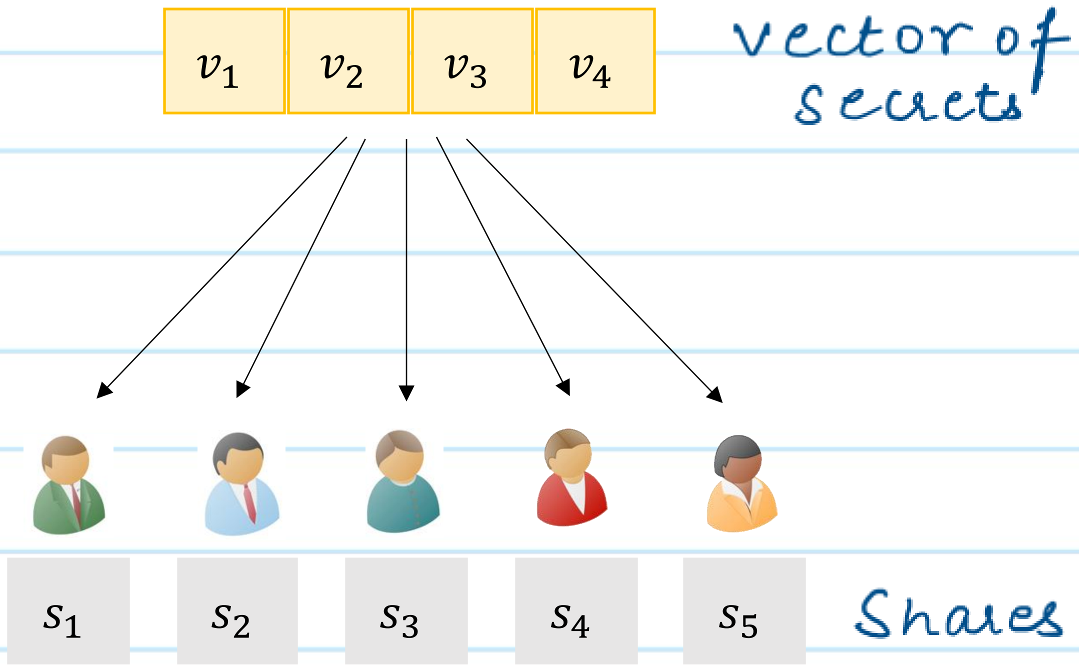
Ammortization when computing multiple copies of a function.

Packed Secret Sharing



Shamir Secret Sharing

1 Value $\rightarrow n$ shares



Packed Secret Sharing

$O(n)$ Values $\rightarrow n$ shares

(ℓ, t, n) - Packed Secret Sharing

Message space: finite field $\mathbb{F}$

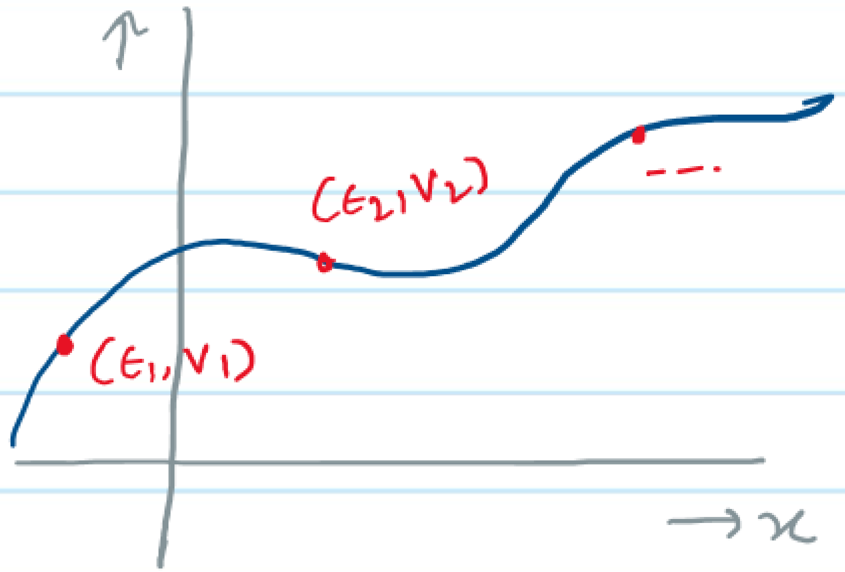
Let $\epsilon_1, \ldots, \epsilon_\ell, \alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be some fixed field elements

→ Share$(v_1, \ldots, v_\ell)$: pick a random degree $t+\ell-1$ polynomial, s.t.,

$p(\epsilon_1) = v_1, \ldots \; p(\epsilon_\ell) = v_\ell$

Compute shares:

$s_1 = p(\alpha_1), \; s_2 = p(\alpha_2), \ldots, \; s_n = p(\alpha_n)$



→ Reconstruct$(s_1, \ldots, s_{t+\ell})$: Lagrange interpolation to find $p(x)$. Then evaluate $v_1 = p(\epsilon_1) \ldots, \; v_\ell = p(\epsilon_\ell)$

(ℓ, t, n) - Packed Secret Sharing

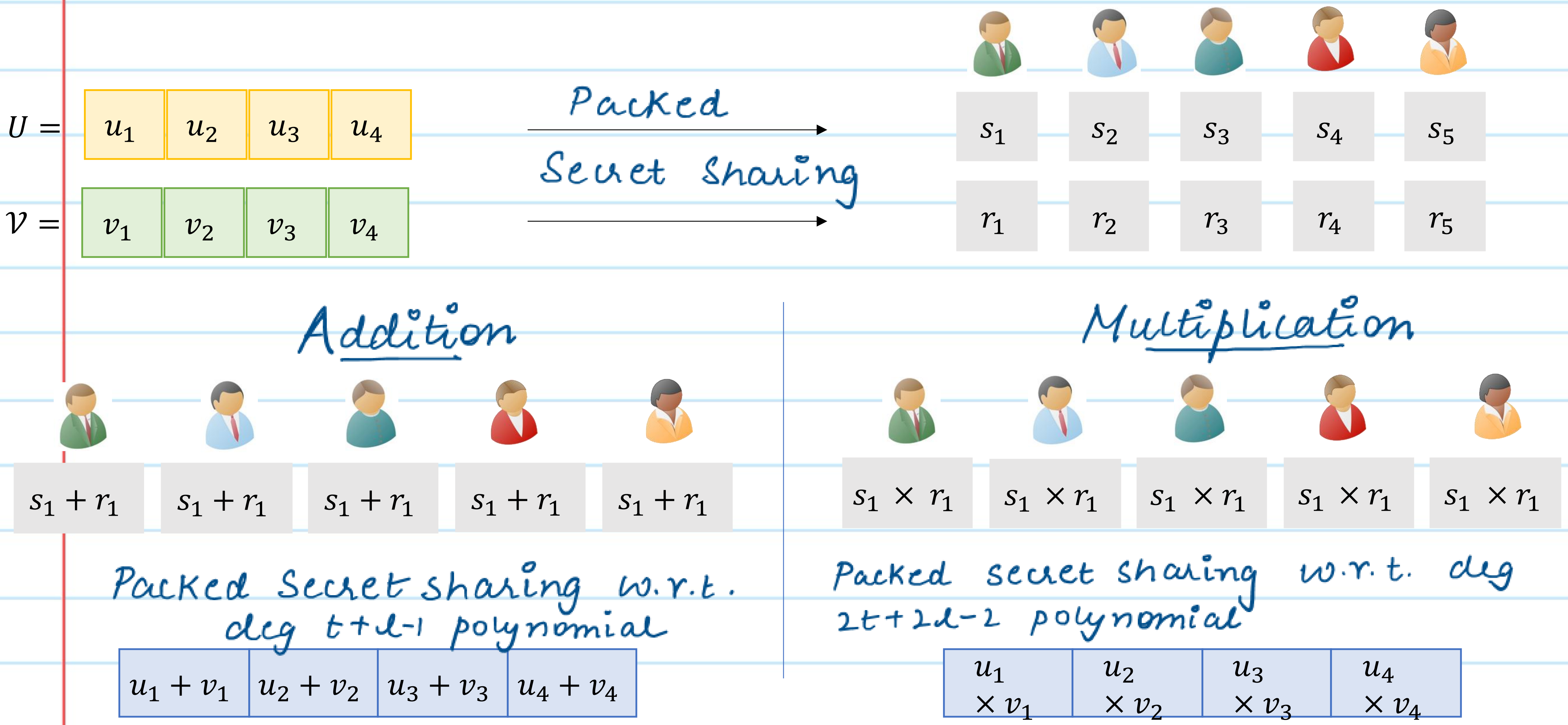
Secrecy: $v_1, \ldots, v_\ell$ remain hidden from any subset of t-parties

Reconstruction: Any subset to $t+\ell$ parties are sufficient for reconstructing $v_1, \ldots, v_\ell$.

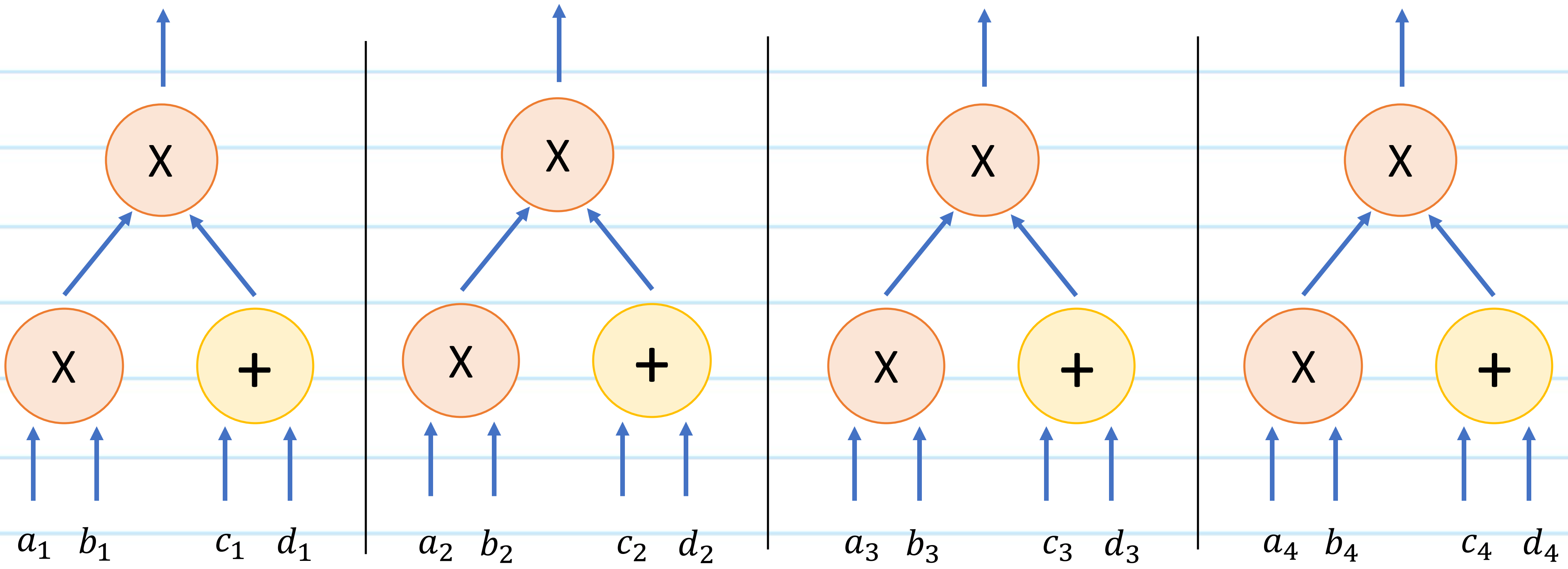
Observe that in contrast in regular secret-sharing, the difference between security & reconstruction threshold is 1.

Here the difference is ℓ : Such schemes are also called ramp secret sharing schemes.

Computing using Packed Secret Sharing

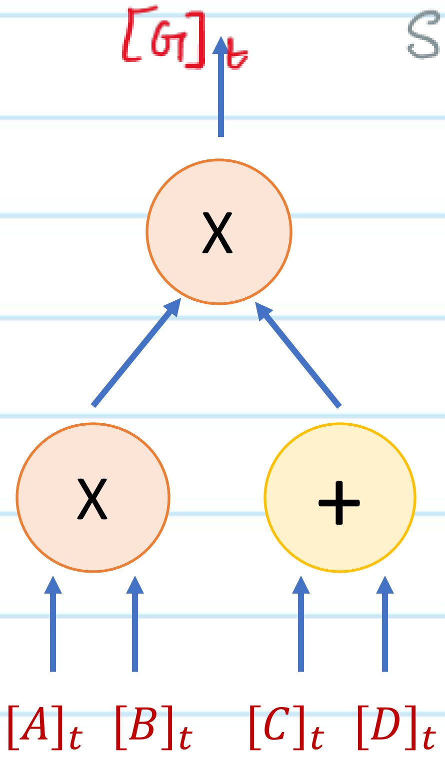


Multiple copies of a Function



Let's assume that the parties want to jointly & securely compute $O(n)$ copies of the same function but on different sets of inputs.

BGW over Packed Secret Shares



Start with PSS of the following vectors in the input sharing Phase

$A = \boxed{a_1 \mid a_2 \mid a_3 \mid a_4}$ $\quad C = \boxed{b_1 \mid b_2 \mid b_3 \mid b_4}$

$B = \boxed{c_1 \mid c_2 \mid c_3 \mid c_4}$ $\quad D = \boxed{d_1 \mid d_2 \mid d_3 \mid d_4}$

Then run a single instance of BGW for computing this function but on packed secret shares.

Reconstruct all output vectors in the output reconstruction phase.

BGW over Packed Secret Shares: Communication Complexity

Communication needed for computing $O(n)$ copies of each multiplication gate: n^2 field elements

Ammortized communication needed for computing a single copy of the multiplication gate: $O(n)$ field elements.

exercise: Think about how you can combine this approach with Beaver triples to get ammortized communication complexity of $O(1)$ field elements per multiplication gate