Advanced Tools from Modern Cryptography

Lecture 4
Secure Multi-Party Computation:
Passive Corruption + Honest-Majority

MPC

- Several dimensions
  - Passive (Semi-Honest) vs. Active corruption
    - Passive: corrupt parties still follow the protocol
  - Honest-Majority vs. Unrestricted corruption
  - Information-theoretic vs. Computational security
  - ...

Security Definition

- Simplest case: Passive corruption, Information-theoretic security
  - Need honest-majority (or similar restriction)
- In passive corruption, only concern will be secrecy
- Perfect secrecy condition similar to secret-sharing

Security Definition

- Multiple parties in a protocol could be corrupt
  - Collusion
  - Modelled using a single adversary who corrupts the parties
    - Its view contains all the corrupt parties' views
- Security guarantee given against an "adversary structure"
  - Set of parties that could be corrupt together

Security Definition

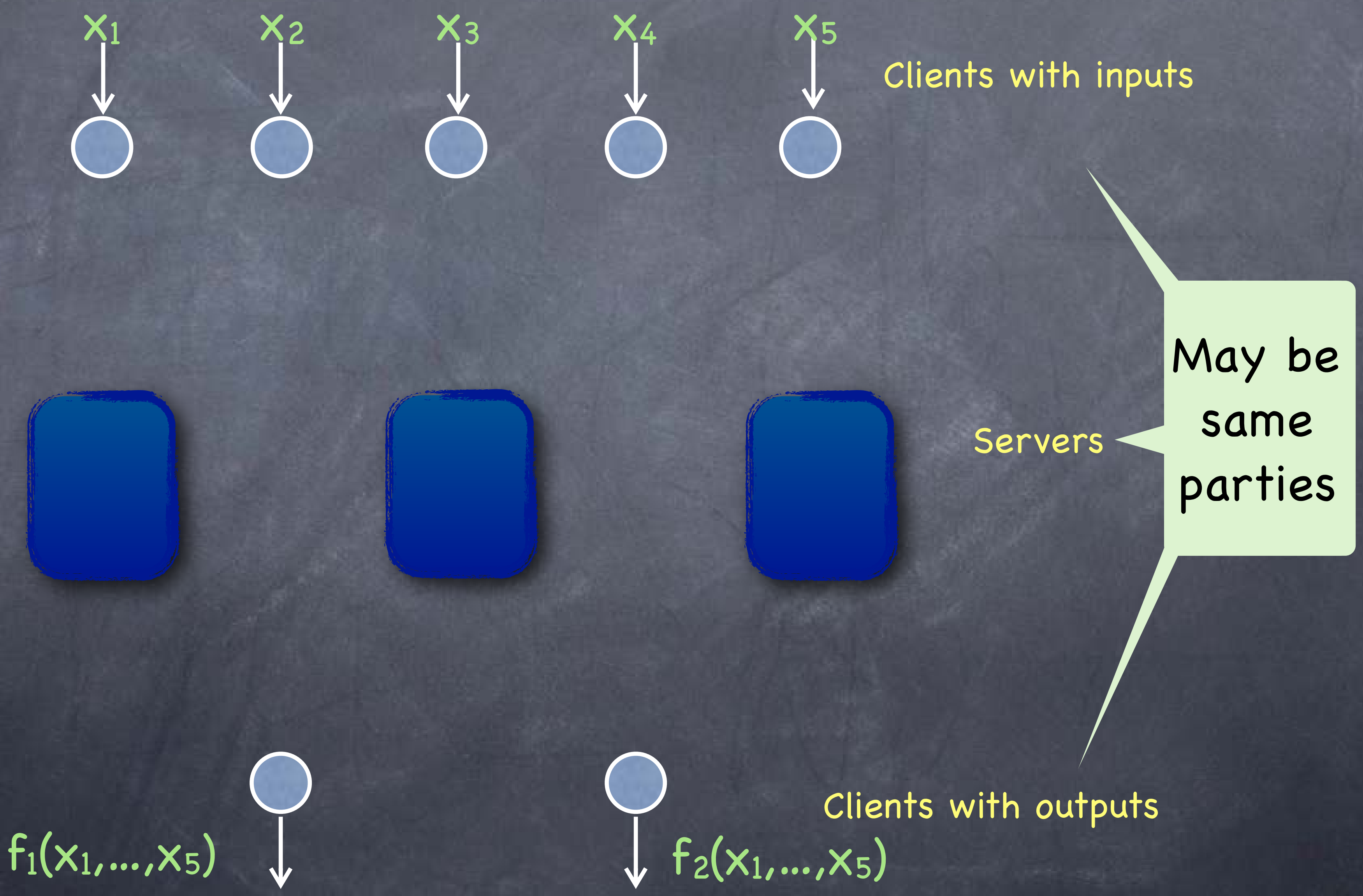
- For secret sharing we needed to formalise "x is secret"
- Now want to say: x is **secret except for f(x)** which is revealed
- $\forall$ x, x′ **s.t. f(x)=f(x′)**, { view | input=x} ≡ { view | input=x′ }

MPC: Outline

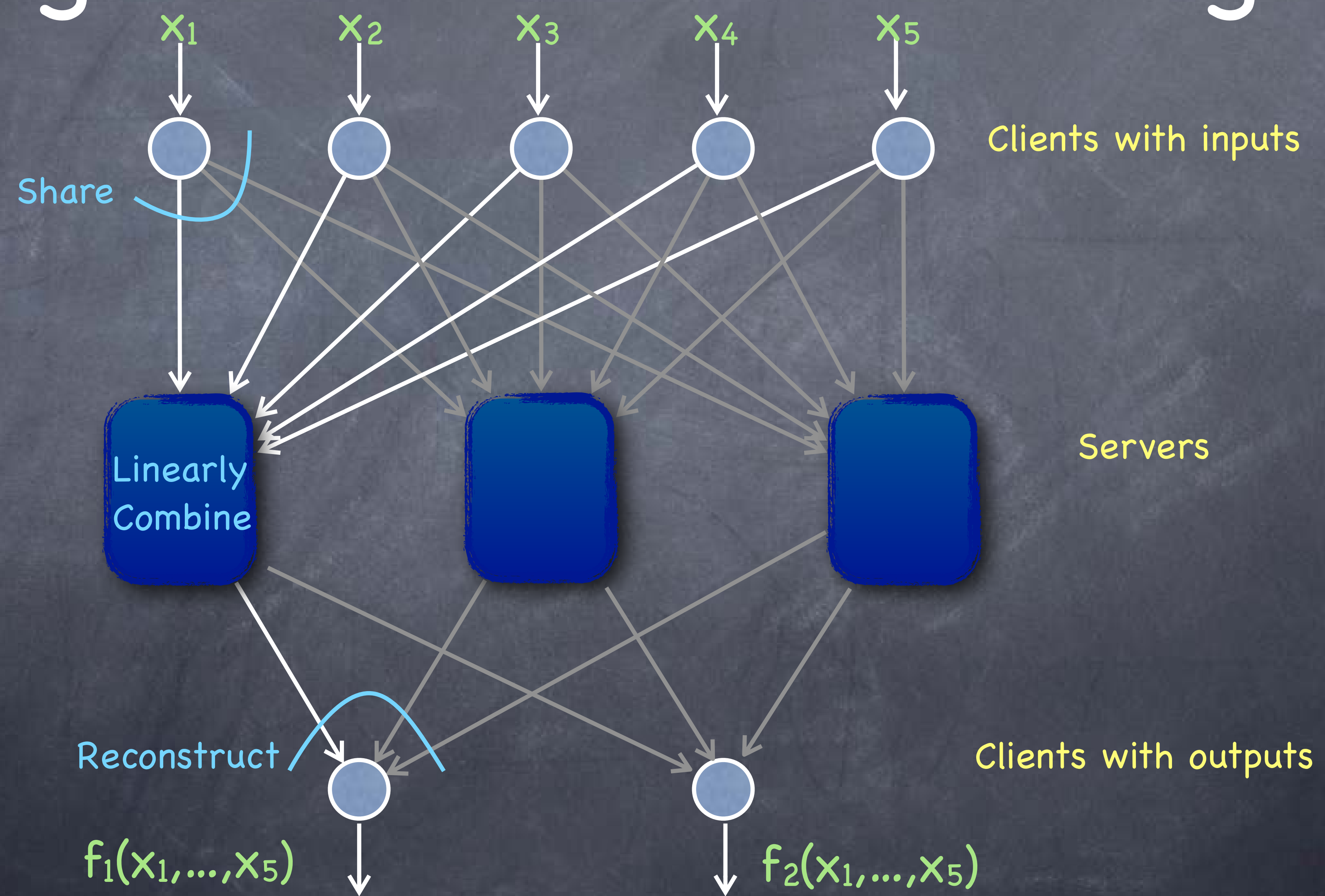
- Today's goal: Perfectly secure MPC against **passive corruption**
- First, MPC for linear functions
  - Arbitrary subset of parties can be corrupt
- MPC for general functions
  - Only with **honest-majority**
    - i.e., adversary structure: subsets with $< N/2$ parties

MPC for Linear Functions

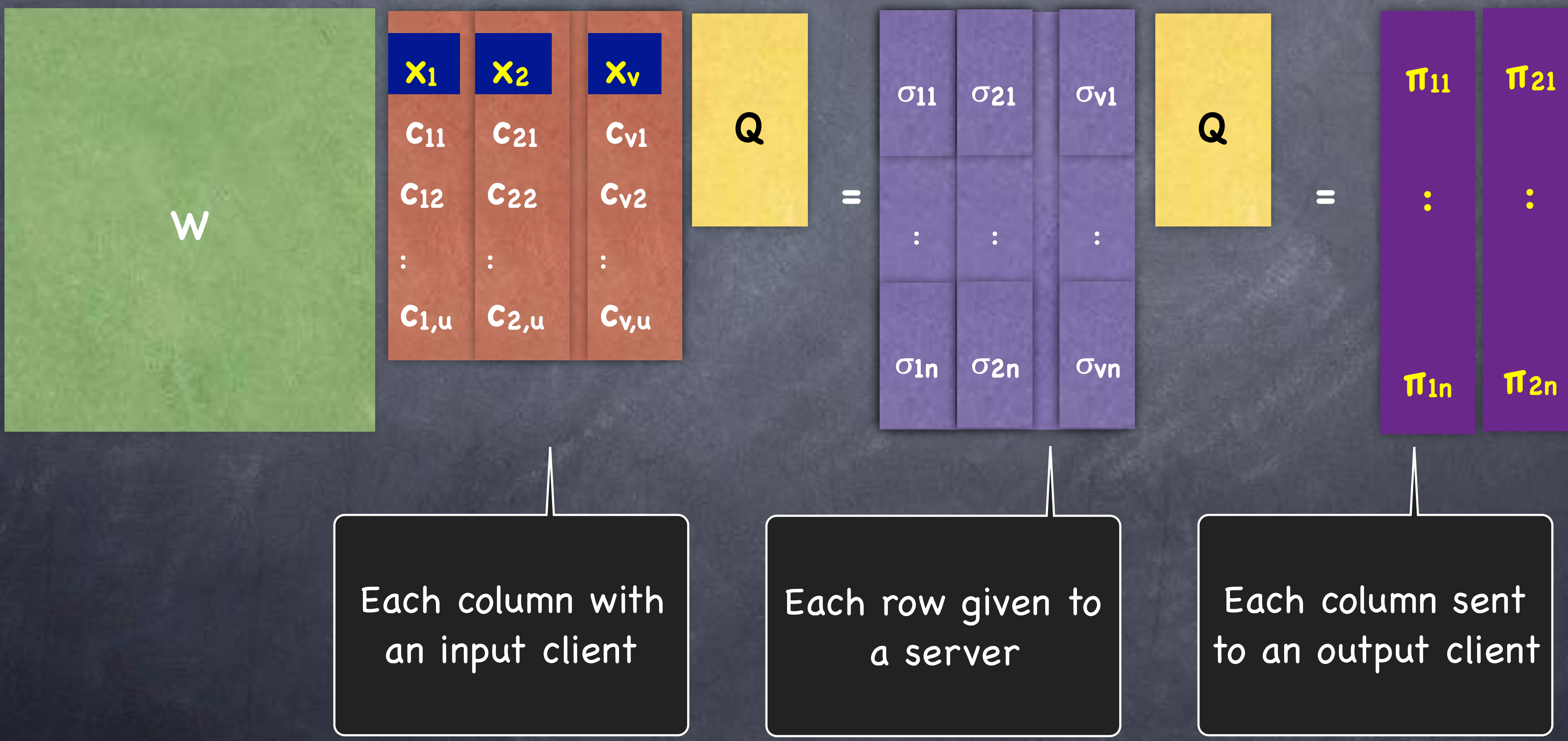
- Client-server setting



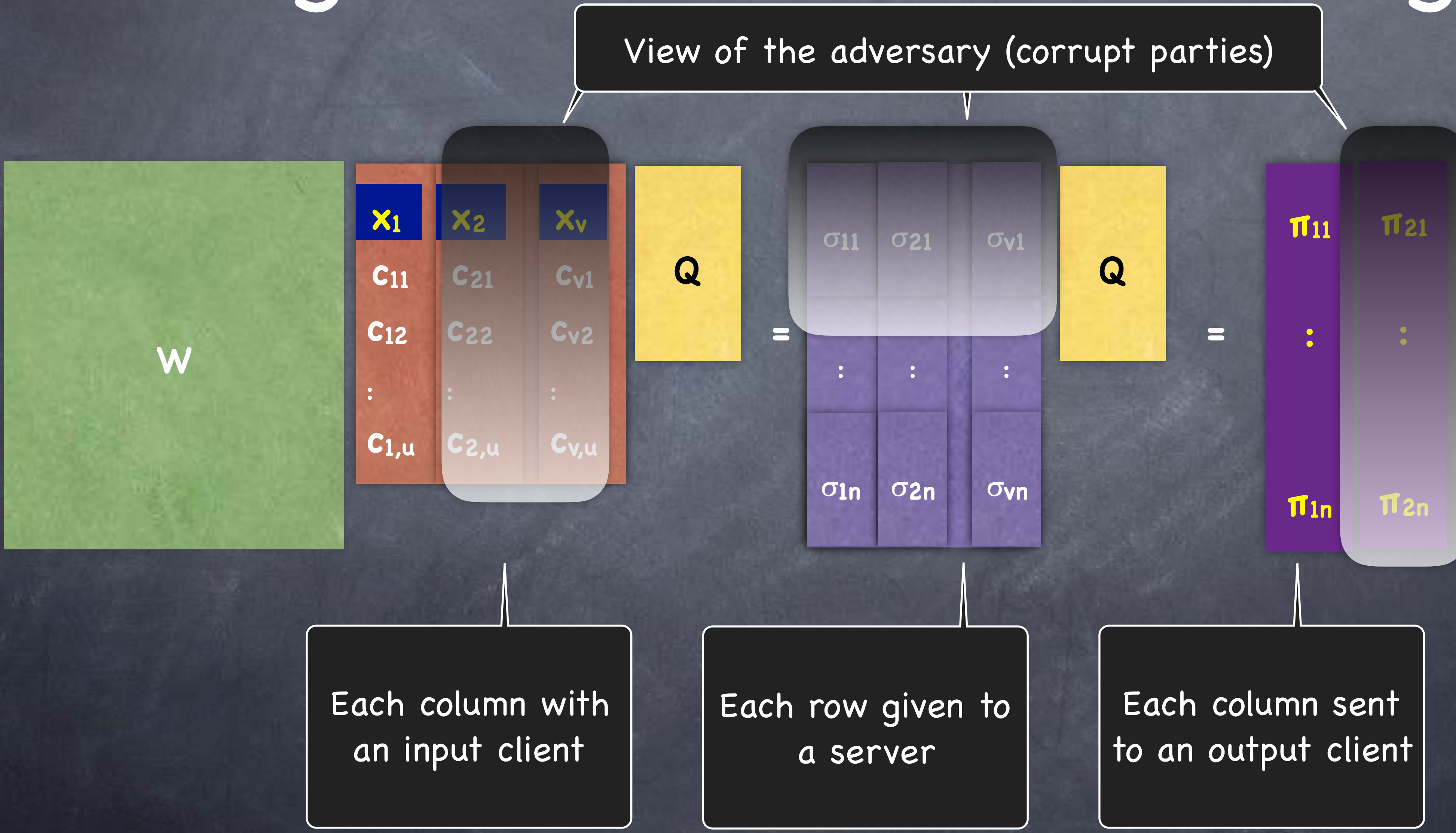
MPC for Linear Functions: Using Linear Secret-Sharing



MPC for Linear Functions: Using Linear Secret-Sharing



MPC for Linear Functions: Using Linear Secret-Sharing



Security

- Adversary allowed to corrupt any set of input and output clients and any subset T which is not a privileged set (i.e., not in the access structure) for the secret-sharing scheme
- View of adversary should reveal nothing beyond the inputs and outputs of the corrupted clients
  - Claim: Consider any input y of corrupt clients. If x, x′ of uncorrupted clients such that for each corrupt output client i $f_i(x,y)=f_i(x',y)$, then the view of the adversary in the two cases are identically distributed
    - Because for any given view of the adversary, the solution space of randomness has the same dimension in the two cases
    - Exercise

MPC for General Functions?

- So far: a 2-round protocol for any linear function
- How about other functions?
- Any function over a finite field can be computed using addition and multiplication
  - Interested in functions which are efficiently computable
  - Arithmetic circuit: representation of the computation using addition and multiplication
- Goal: MPC Protocol for f, which is efficient if we are given an efficient arithmetic circuit for f

MPC for General Functions?

- Plan: Gate-by-gate evaluation
  - Servers maintain shares of the wires at all times
  - Types of gates:
    - Input gate
      - Each input client acts as a dealer ✓
    - A linear function
      - Each server locally computes ✓
    - A binary multiplication
      - How?
    - Output gate
      - Send shares to each output client ✓
  - Question: How to go from shares(x), shares(y) to shares($x \cdot y$) securely?

MPC for General Functions: Using Shamir Secret-Sharing

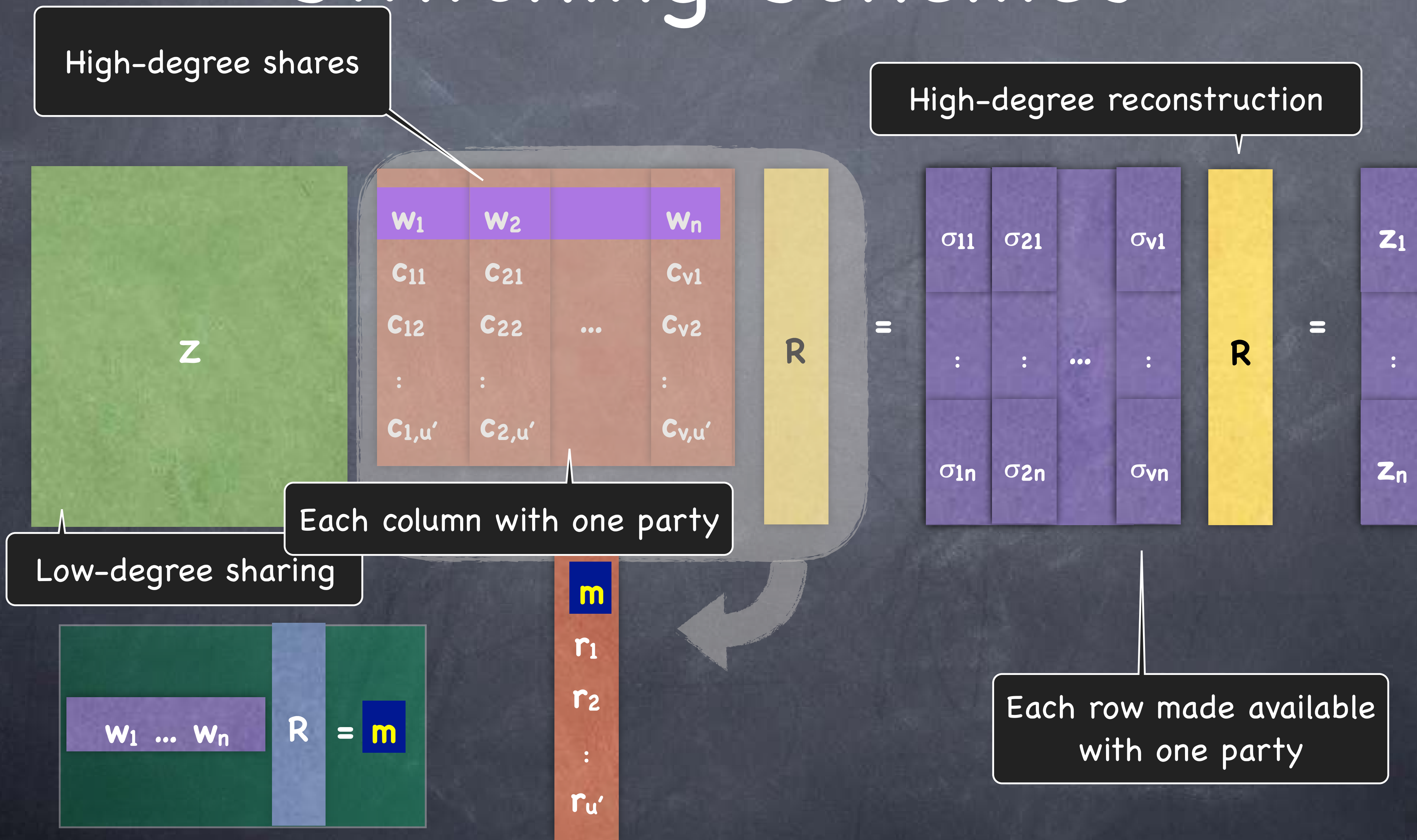
- Question: How to go from shares(x), shares(y) to shares($x \cdot y$) securely?
- Idea: Use Shamir secret-sharing!
  - For polynomials, multiplication commutes with evaluation: $(f \cdot g)(x) = f(x) \cdot g(x)$
  - In particular, to get a polynomial h with $h(0)= f(0) \cdot g(0)$, simply define $h = f \cdot g$. Shares $h(x)$ can be computed as $f(x) \cdot g(x)$
  - But note: h has a higher degree!
    - Problem 1: Can't continue protocol after one multiplication
    - Problem 2: If degree $\geq N$, can't reconstruct the secret even if all parties reveal their shares

MPC for General Functions: Using Shamir Secret-Sharing

- Problem: If x, y shared using a degree d polynomial, $x \cdot y$ is shared using a degree 2d polynomial
- Solution: Bring it back to the original secret-sharing scheme!
  - By "securely" switching shares from degree-2d shares to degree-d shares
    - Note: All N servers together should be able to linearly reconstruct the degree-2d sharing
    - Start with $N \geq 2d+1$
      - Can tolerate only up to d ($\leq (N-1)/2$) corrupt servers

$< N/2$

Linear Secret-Sharing: Switching Schemes



MPC for General Functions?

- Plan: Gate-by-gate evaluation
  - Servers maintain shares of the wires at all times
  - Types of gates:
    - Input gate
      - Each input client acts as a dealer ✓
    - A linear function
      - Each server locally computes ✓
    - A binary multiplication
      - Local mult. & degree reduction
    - Output gate
      - Send shares to each output client ✓