Advanced Tools from Modern Cryptography

Lecture 3
Secret-Sharing (ctd.)

Secret-Sharing

- Last time
  - (n,t) secret-sharing
    - (n,n) via additive secret-sharing
    - Shamir secret-sharing for general (n,t)
    - Shamir secret-sharing is a linear secret-sharing scheme

Shamir Secret-Sharing

- Share(m): Pick a random degree t-1 polynomial $f(X) = \Sigma_{i \in \{0..t-1\}} c_i X^i$, such that $f(0)=m$ (i.e., $c_0 = m$). Shares are $s_i = f(a_i)$, where a_i are distinct and non-zero.

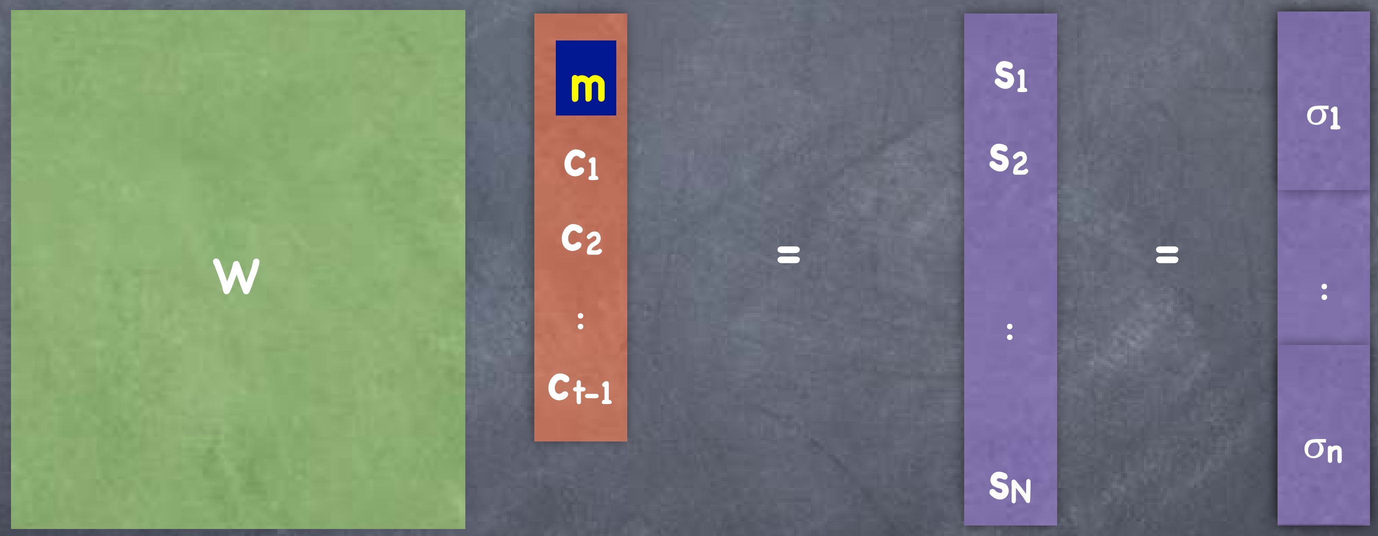
$$\begin{pmatrix} 1 & a_1 & a_1^2 & \dots & a_1^{t-1} \\ 1 & a_2 & a_2^2 & \dots & a_2^{t-1} \\ & & \vdots & & \\ 1 & a_n & a_n^2 & \dots & a_n^{t-1} \end{pmatrix} \begin{pmatrix} \boxed{m} \\ c_1 \\ c_2 \\ \vdots \\ c_{t-1} \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}$$

- Reconstruct($s_{i_1},\dots,s_{i_t}$): Lagrange interpolation to find $m=c_0$
  - i.e., solve for $(m\ c_1\ \dots\ c_{t-1})$ from t rows of the above system

Linear Secret-Sharing

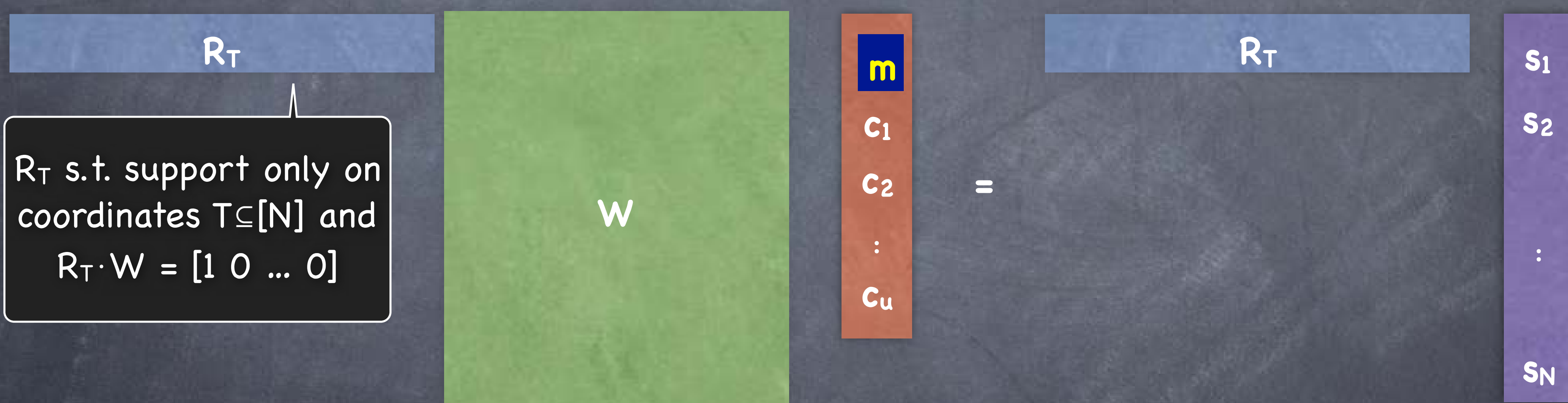
- Share(M): For some fixed n×t matrix W, let **s** = W.**c** where c_0 = M and other t-1 coordinates are random

  Shares are "sub-vectors" of **s**.



Linear Secret-Sharing

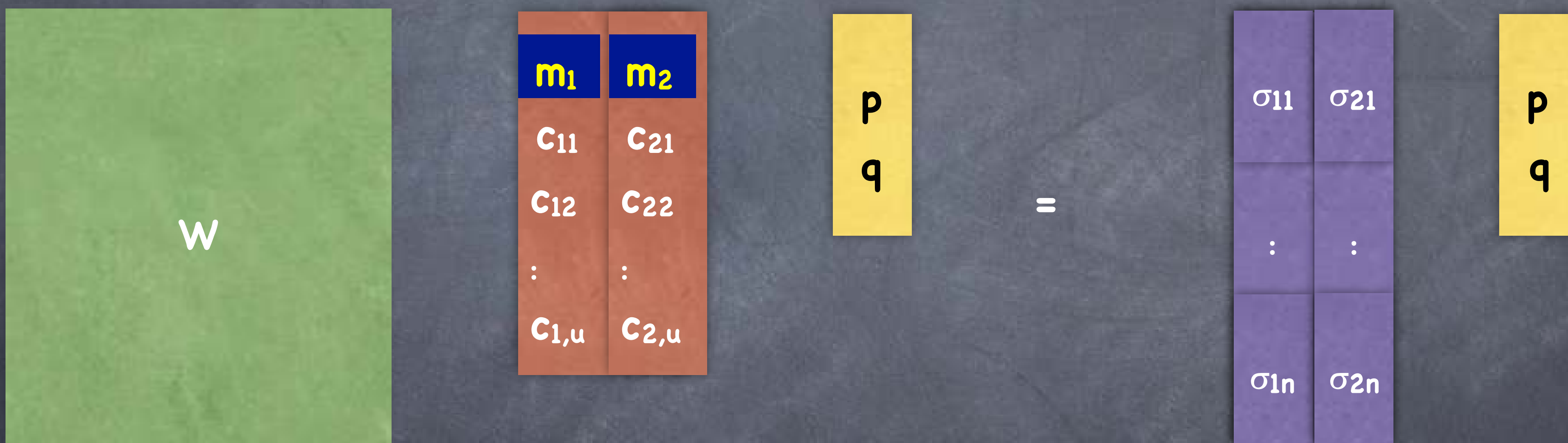
- Reconstruct($\sigma_{i_1}, \ldots, \sigma_{i_t}$): pool together available coordinates $T \subseteq [N]$. Can reconstruct if there are enough equations to solve for m.



- Claim: $\forall T \subseteq [n]$, $\underline{s}_T$ either fully determines m, or is independent of m
  - If $T \subseteq [N]$ s.t. $[1\ 0\ \ldots\ 0]$ not in the row span of W_T, for any $\gamma \in F$, we can add an equation $m=\gamma$ to the system $W_T \cdot \underline{c} = \underline{s}_T$. Number of solutions for $\underline{c}$ in this system is independent of γ.

Linear Secret-Sharing: Computing on Shares

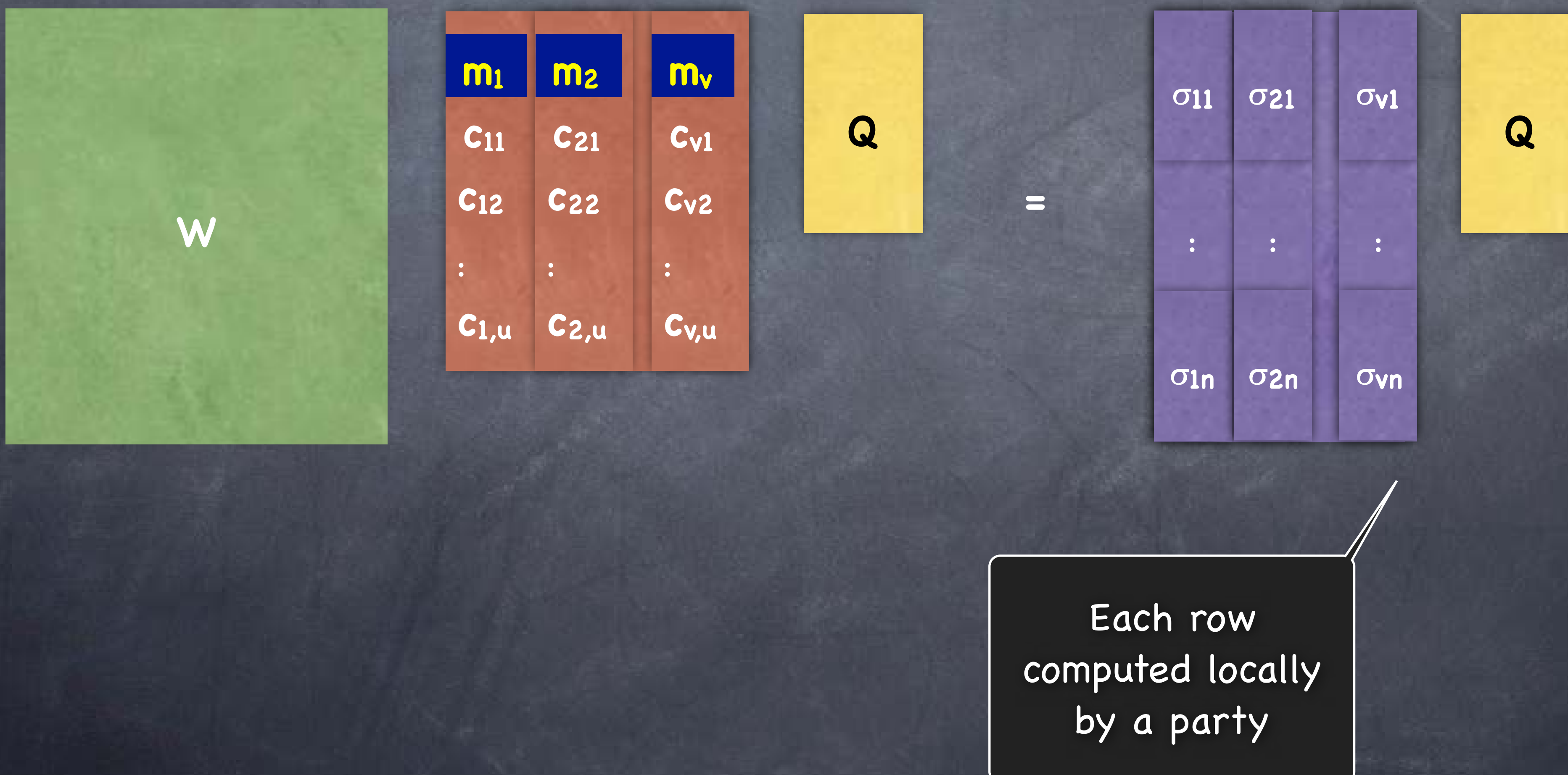
- Suppose two secrets m_1 and m_2 shared using the same secret-sharing scheme



- Then for any $p,q \in F$, shares of $p \cdot m_1 + q \cdot m_2$ can be computed <u>locally</u> by each party i as $\sigma_i = p \cdot \sigma_{1i} + q \cdot \sigma_{2i}$

Linear Secret-Sharing: Computing on Shares

- More generally, can compute shares of any linear transformation

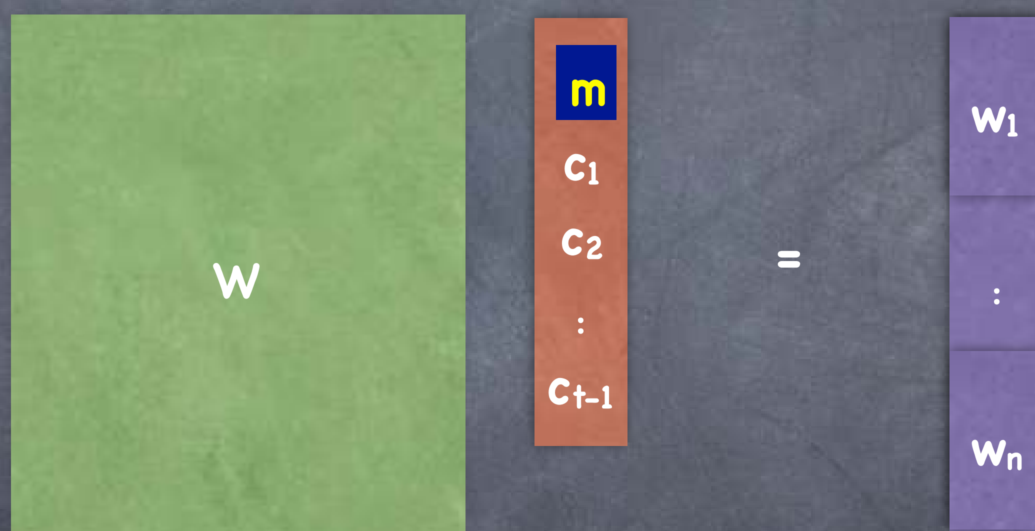


Switching Schemes

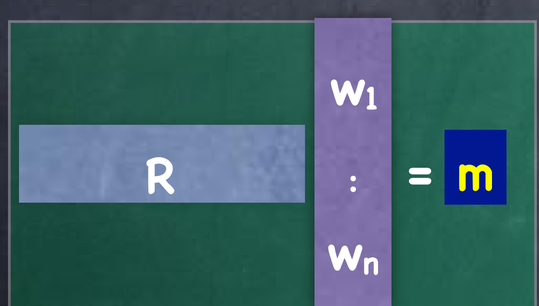
- Can move from any linear secret-sharing scheme W to any other linear secret-sharing scheme Z "securely"
- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)
- Locally each party j reconstructs using scheme W: $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$
- Claim: $(z_1, \ldots, z_n)$ is a valid Z-sharing of m

Linear Secret-Sharing: Switching Schemes

- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)

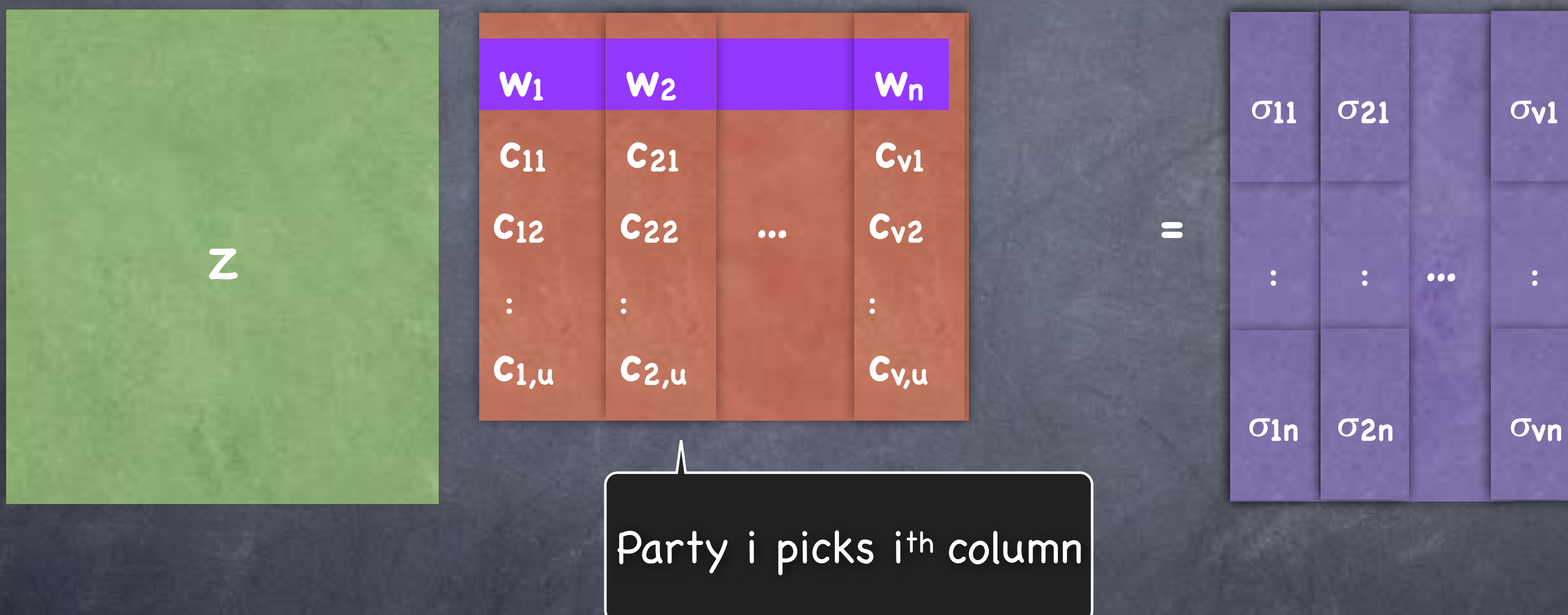


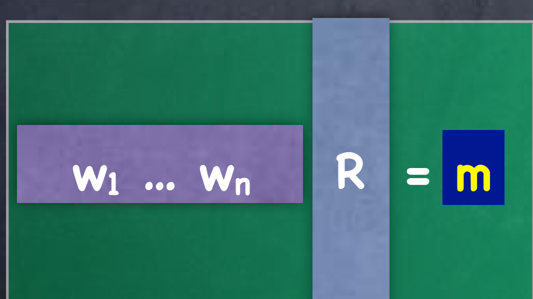
- Recall reconstruction in W:



Linear Secret-Sharing: Switching Schemes

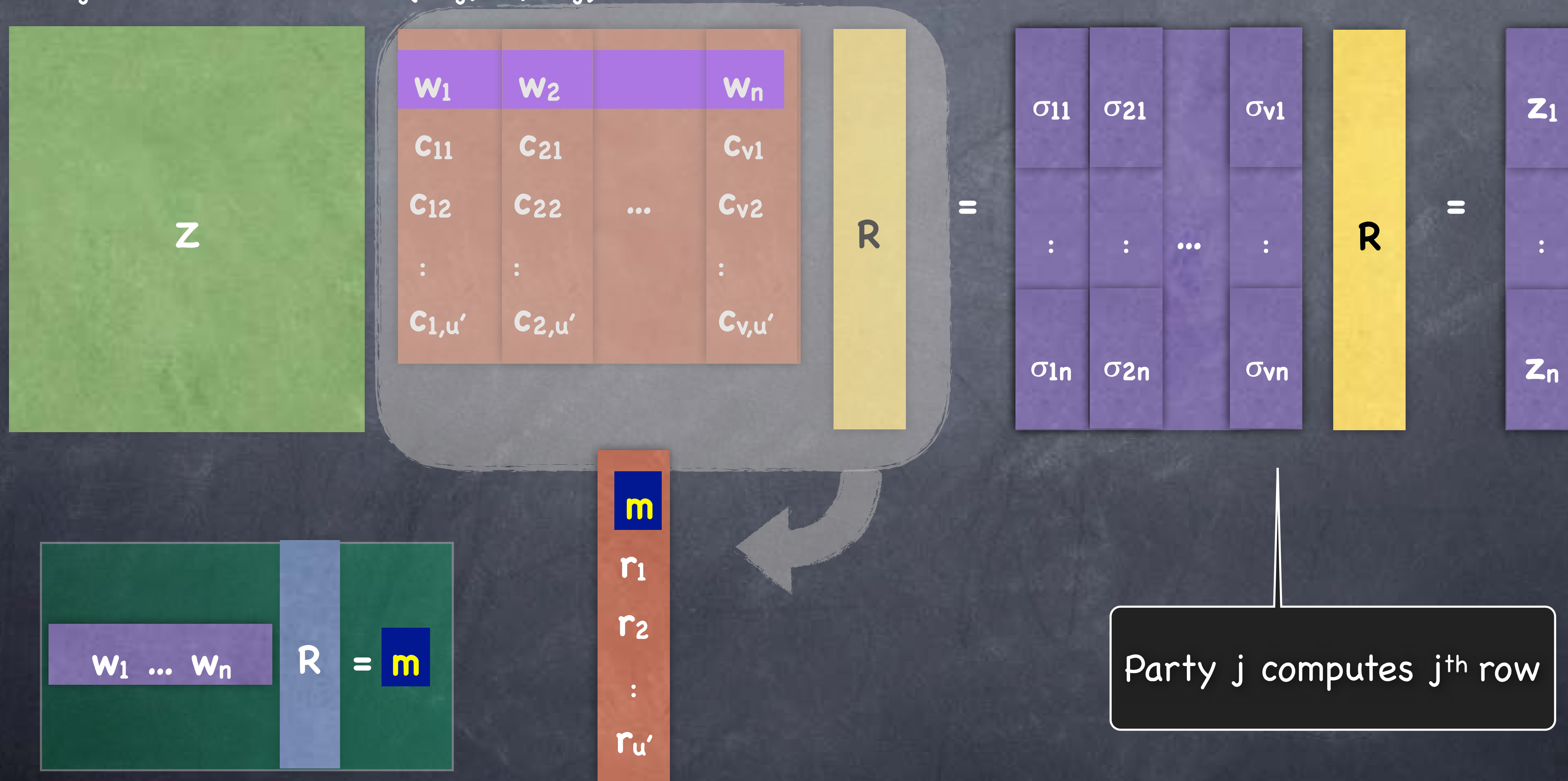
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)





Linear Secret-Sharing: Switching Schemes

- Locally each party j reconstructs using scheme W: $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$



Switching Schemes

- Can move from any linear secret-sharing scheme W to any other linear secret-sharing scheme Z "securely"
- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)
- Locally each party j reconstructs using scheme W: $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$
- Claim: $(z_1, \ldots, z_n)$ is a valid Z-sharing of m ✓
- Claim: If a party-set T⊆[n] is not allowed to learn the secret by both W and Z, then T learns nothing about m from this process
  - Exercise

More General Access Structures

- (n,t)-secret-sharing allowed any t (or more) parties to reconstruct the secret
  - i.e., "access structure" $\mathcal{A} = \{S: |S| \geq t\}$, is the set of all subsets of parties who can reconstruct the secret
  - In general access structure could be any monotonic set of subsets

If $S^* \in \mathcal{A}$, then for all $S \supseteq S^*$, $S \in \mathcal{A}$.

- Shamir's secret-sharing solves threshold secret-sharing. How about the others?

More General Access Structures

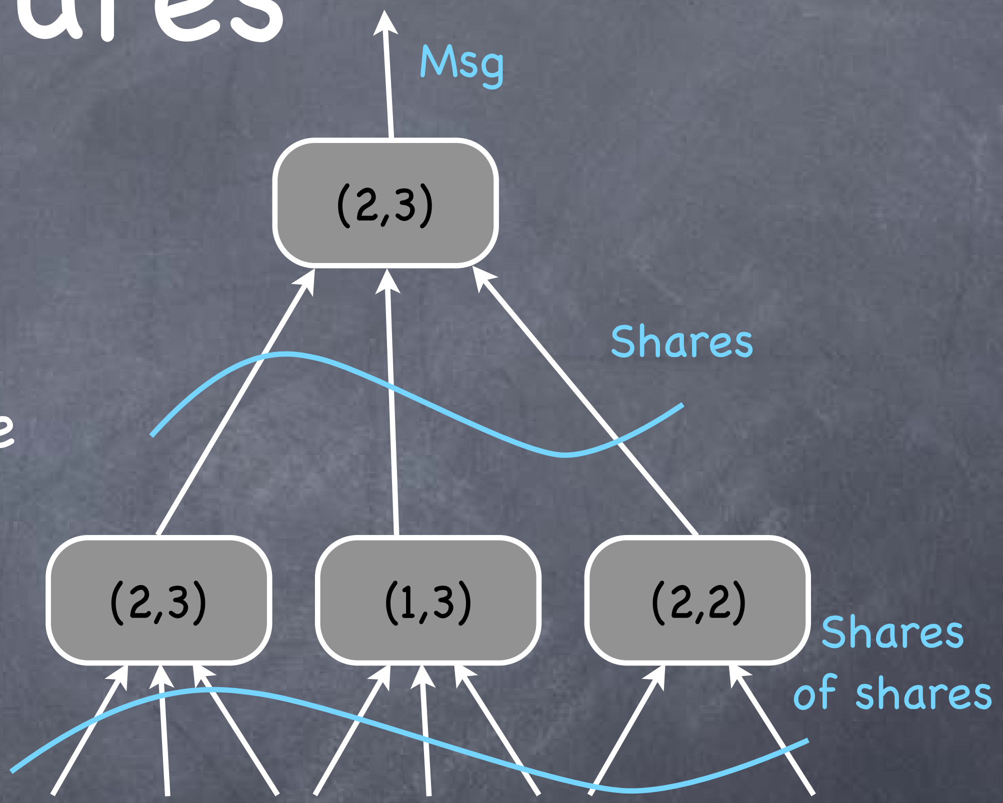
- Idea: For arbitrary monotonic access structure $\mathcal{A}$, there is a "basis" $\mathcal{B}$ of minimal sets in $\mathcal{A}$. For each S in $\mathcal{B}$ generate an (|S|,|S|) sharing, and distribute them to the members of S.
  - Works, but very "inefficient"
    - How big is $\mathcal{B}$? (Say when $\mathcal{A}$ is a threshold access structure)

      $|\mathcal{B}|$ = (n choose t)
    - Total share complexity = $\Sigma_{S\in\mathcal{B}}$ |S| field elements. (Compare with Shamir's scheme: n field elements in all.)

      $t\cdot$(n choose t)
  - More efficient schemes known for large classes of access structures

More General Access Structures

- A simple generalization of threshold access structures
  - A threshold tree to specify the access structure
  - Can realize by recursively threshold secret-sharing the shares
- Note: linear secret-sharing
- Fact: Access structures that admit linear secret-sharing are those which can be specified using "monotone span programs"



Efficiency

- Main measure: size of the shares (say, total of all shares)
  - Shamir's: each share is as as big as the secret (a single field element)
  - Naïve scheme for arbitrary monotonic access structure: if a party is in N sets in $\mathcal{B}$, N basic shares
    - N can be exponential in n (as $\mathcal{B}$ can have exponentially many sets)
  - Share size must be at least as big as the secret: "last share" in a minimal authorized set should contain all the information about the secret
    - Ideal: if all shares are only this big (e.g. Shamir's scheme)
    - Not all access structures have ideal schemes
  - Non-linear schemes can be more efficient than linear schemes