Some Secure Computation Concepts

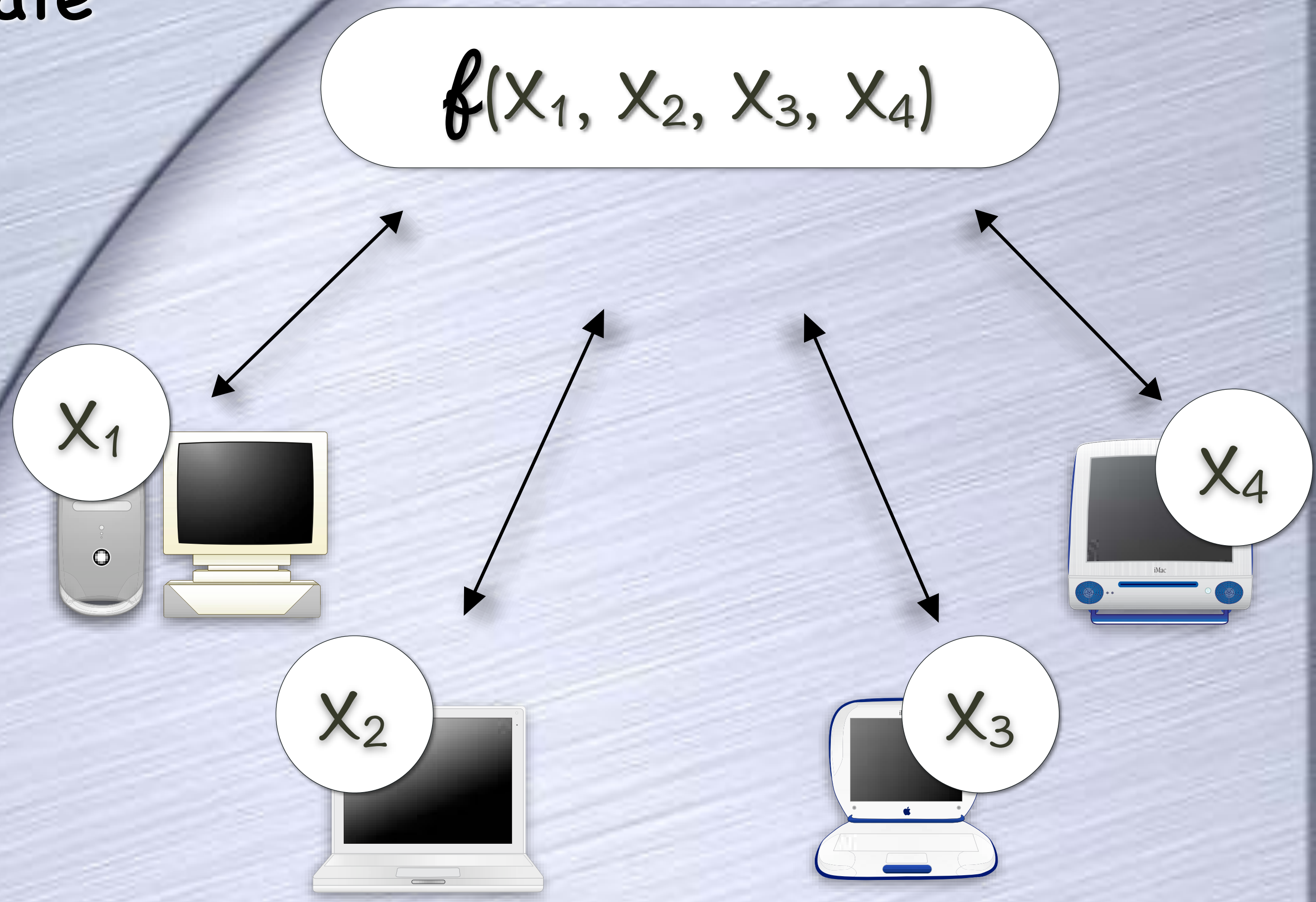
Manoj Prabhakaran

Keywords

- Secure multi-party computation
  - Linear Secret-Sharing
  - Passive secure BGW protocol
  - Omitted: Yao's Garbled Circuit, Randomized Encoding, Conditional Disclosure of Secrets, UC Security, …
- Private Information Retrieval, Oblivious RAM, Searchable Encryption
- Homomorphic Encryption & Fully Homomorphic Encryption
- More tools in the horizon
  - Obfuscation, Functional Encryption, …

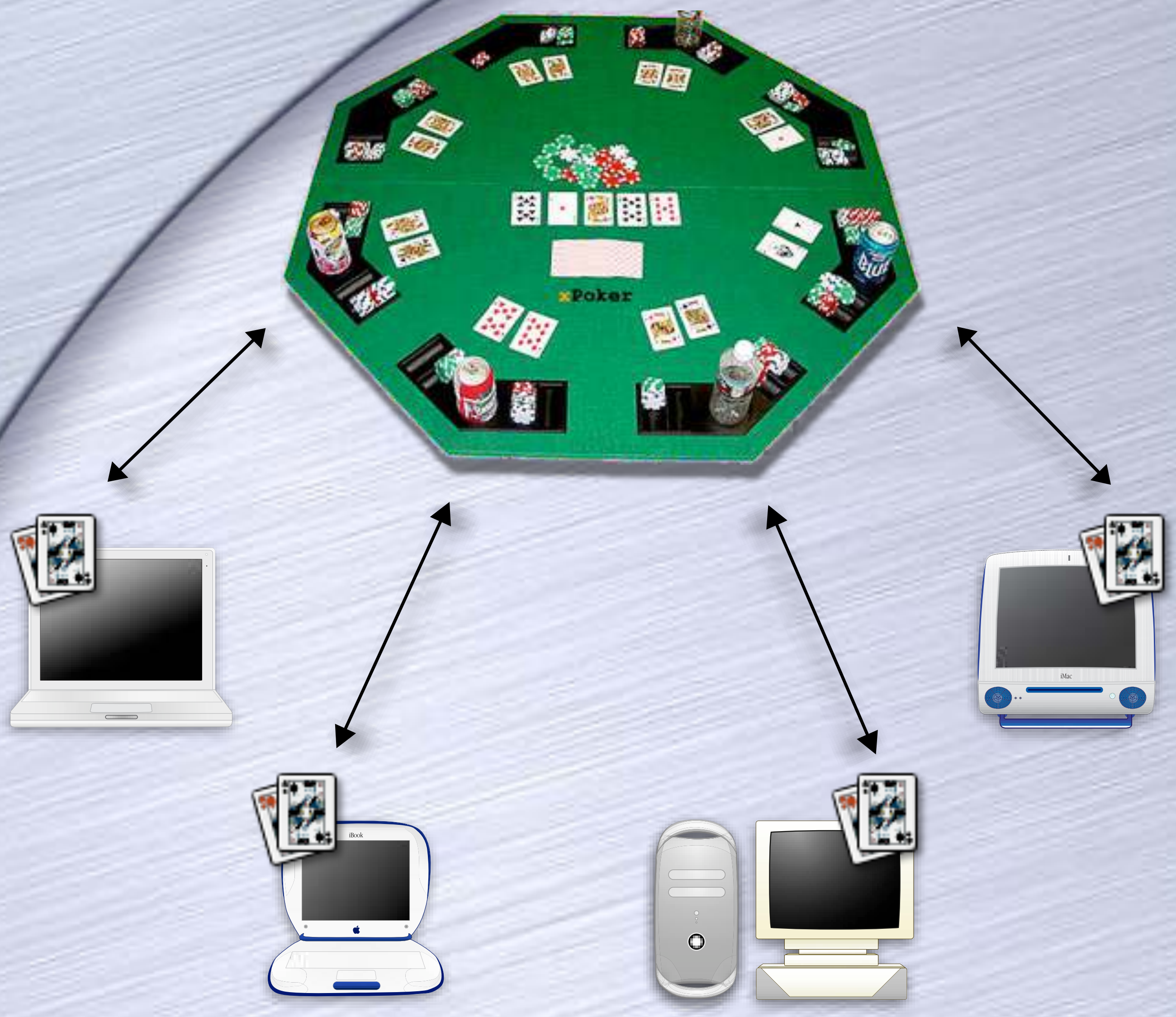
Secure Function Evaluation

- A general problem
- To compute a function of private inputs without revealing information about the inputs
  - Beyond what is revealed by the function



Poker With No Dealer?

- Need to ensure
  - Cards are shuffled and dealt correctly
  - Complete secrecy
  - No “cheating” by players, even if they collude
- No universally trusted dealer



Mental Poker

Adi Shamir, Ronald L. Rivest and Leonard M. Adleman

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ABSTRACT

Can two potentially dishonest players play a fair game of poker without using any cards—for example, over the phone? This paper provides the following answers:

1. No. (Rigorous mathematical proof supplied.)
2. Yes. (Correct and complete protocol given.)

Emulating Trusted Computation

- MPC: to emulate a source of trusted computation
  - Trusted means it will not “leak” a party’s information to others
  - And it will not cheat in the computation
- A tool for mutually distrusting parties to collaborate

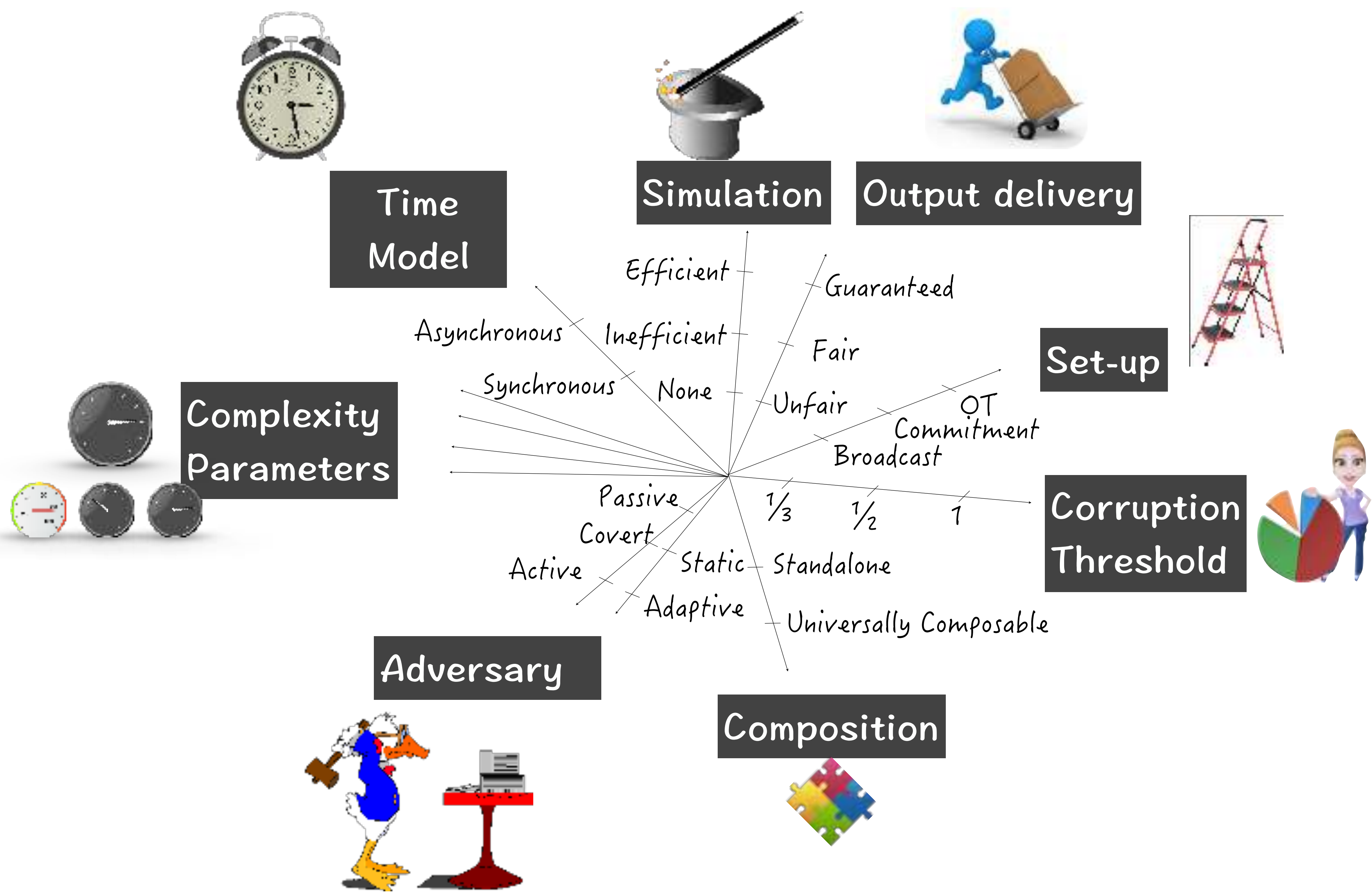
Is it for Real?

- Getting there! Many implementations/platforms
  - Fairplay, VIFF
  - Sharemind
  - SCAPI
  - Obliv-C
  - JustGarble
  - SPDZ/MASCOT
  - ObliVM
  - ...
  - www.multipartycomputation.com/mpc-software

Is it for Real?

- And many practical systems using some form of MPC
  - Danish company Partisia with real-life deployments (since 2008)
    - sugar beet auction, electricity auction, spectrum auction, key management
  - A prototype for credit rating, supported by Danish banks
  - A proposal to the Estonian Tax & Customs Board
  - A proposal for Satellite Collision Analysis
  - Proposed legislation in the US to use MPC for an application ("higher education data system")
  - ...

MPC Dimensions



Secret-Sharing

- Dealer encodes a message into n shares for n parties
  - Privileged subsets of parties should be able to reconstruct the secret
  - View of an unprivileged subset should be independent of the secret
- Very useful
  - Direct applications (distributed storage of data or keys)
  - Important component in other cryptographic constructions
    - Secure multi-party computation
    - Attribute-Based Encryption
    - Leakage resilience ...

Threshold Secret-Sharing

- (n,t)-secret-sharing
  - Divide a message m into n shares $s_1,\ldots,s_n$, such that
    - any t shares are enough to reconstruct the secret
    - up to t-1 shares should have no information about the secret

e.g., $(s_1,\ldots,s_{t-1})$ has the same distribution for every m in the message space

- (2,2) secret-sharing
  - One share is a key, and the other an encryption of the message using the key
  - One-time encryption suffices: One share is K and the other is $m \oplus K$

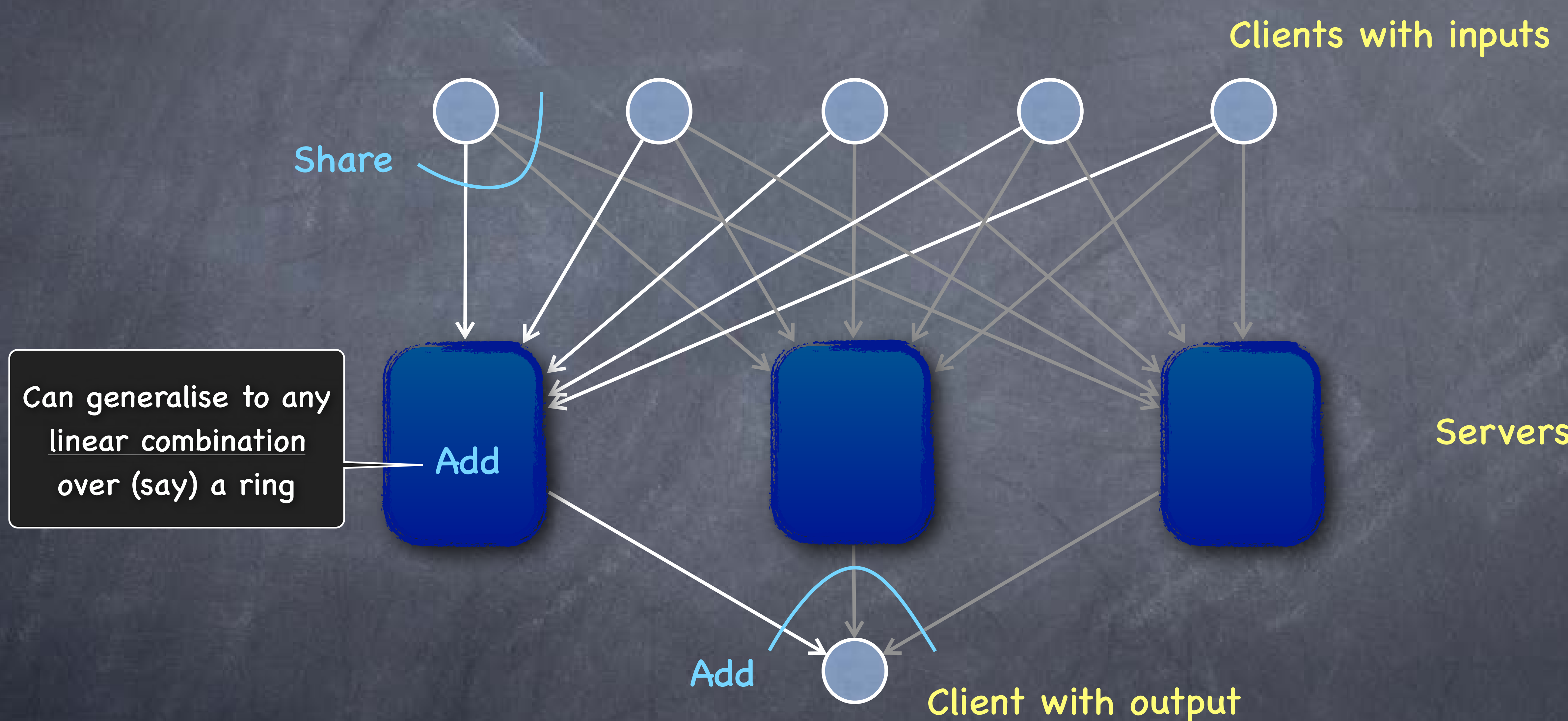
Threshold Secret-Sharing

Additive Secret-Sharing

- Construction: (n,n) secret-sharing
  - Message-space = share-space = G, a finite group
    - e.g. $G = \mathbb{Z}_2^d$ (group of d-bit strings, with $\oplus$ as the group operation)
  - Share(M):
    - Pick $s_1,\ldots,s_{n-1}$ uniformly at random from G
    - Let $s_n = -(s_1 + \ldots + s_{n-1}) + M$
  - Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$

An Application

- Gives a "private summation" protocol (for commutative groups)



- "Secure against passive corruption" (i.e., no colluding set of servers/clients learn more than what they must) if at least one server stays out of the collusion

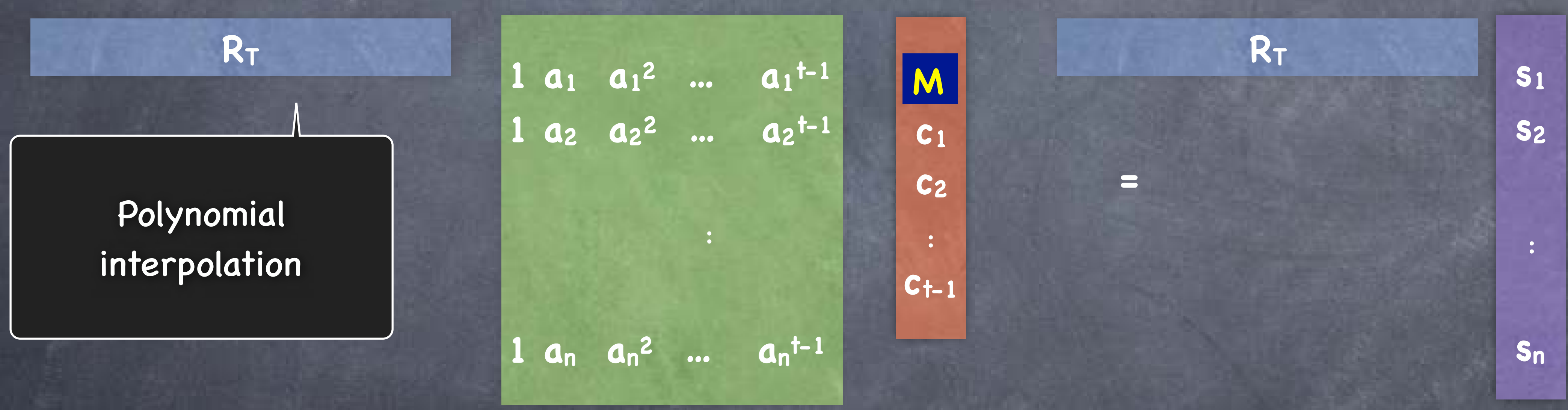
Linear Secret-Sharing

- Recall (n,n) secret-sharing
  - Share(M): $s_1,\ldots,s_{n-1}$ uniform, and $s_n = -(s_1 + \ldots + s_{n-1}) + M$
  - Reconstruct($s_1,\ldots,s_n$): $M = s_1 + \ldots + s_n$

- (n,t) secret-sharing: for each privileged set $T \subseteq n$ (i.e., T with $|T| \geq t$) $\exists R_T$ with support only on positions in T, s.t. $R_T \cdot W = [1\ 0\ \ldots\ 0]$
  - Linearity guarantees that for unprivileged sets, the view is equally likely for all messages

Linear Secret-Sharing

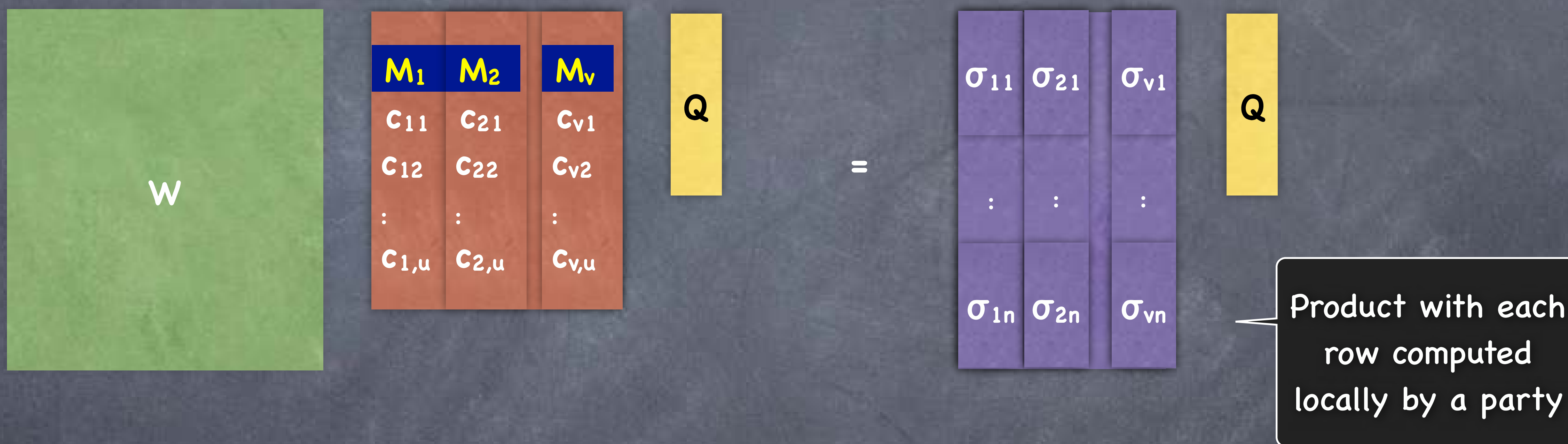
- (n,t) secret-sharing: for each privileged set T ⊆ n (i.e., T with $|T| \geq t$) $\exists R_T$ with support only on positions in T, s.t. $R_T \cdot W = [1\ 0\ \ldots\ 0]$
  - Linearity guarantees that for unprivileged sets, the view is equally likely for all messages



- Shamir secret-sharing (all operations over a field):
  - Sharing: Using the Vandermonde matrix. n shares are evaluations of a polynomial $f(X) = M + c_1X + \ldots + c_{t-1}X^{t-1}$ at points $a_1,\ldots,a_n$
  - Reconstruct($\{s_i\}_{i\in T}$): Langrange interpolation to obtain $M = f(0)$

Linear Secret-Sharing

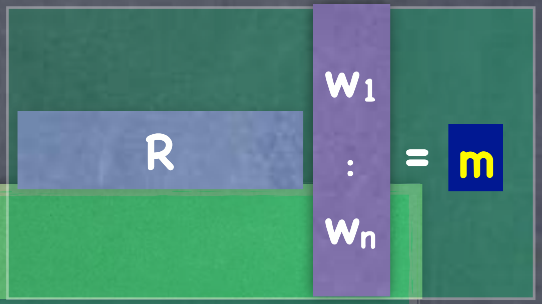
- Allows computing on shares!



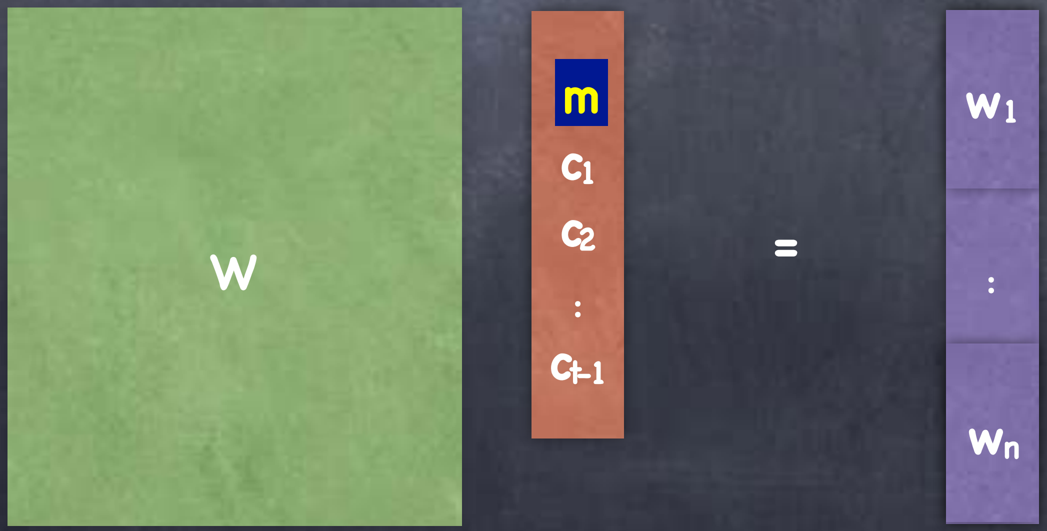
- Associativity of matrix multiplication: Can compute the shares of a linear combination of the messages as a linear combination of the shares of the messages

Switching Schemes

- Can move from any linear secret-sharing scheme W to any other linear secret-sharing scheme Z "securely"

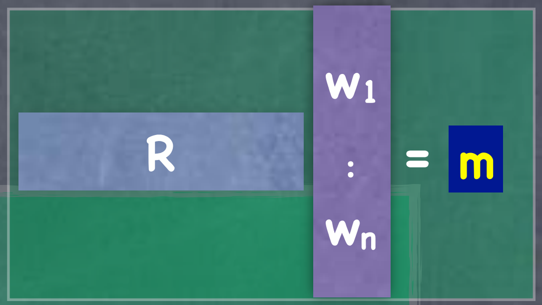


- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)
- Locally each party j reconstructs using scheme W: $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$

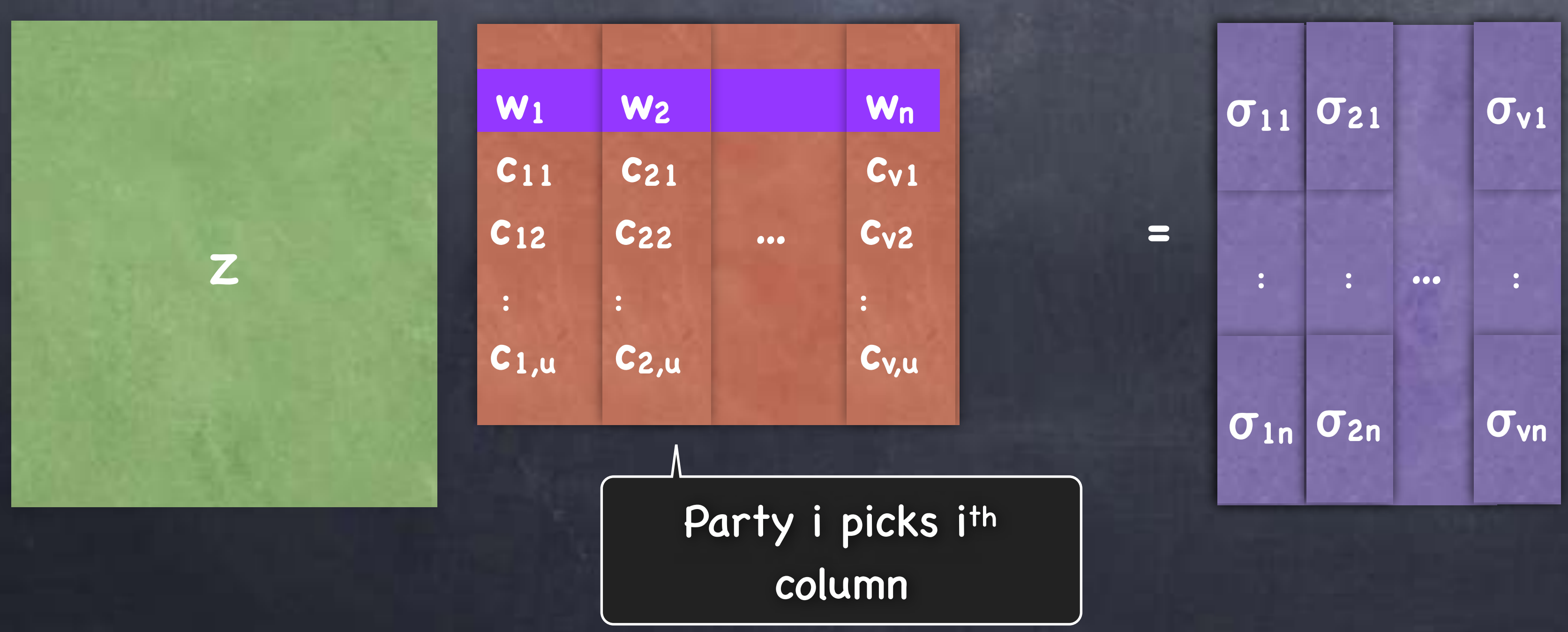


Switching Schemes

- Can move from any linear secret-sharing scheme W to any other linear secret-sharing scheme Z "securely"

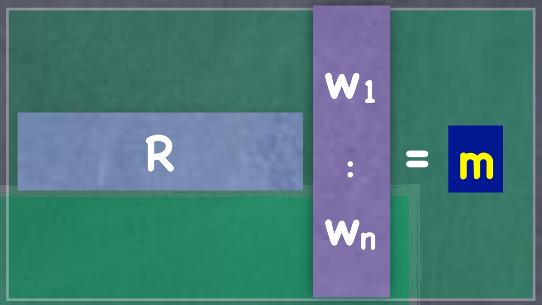


- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)
- Locally each party j reconstructs using scheme W:
  $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$

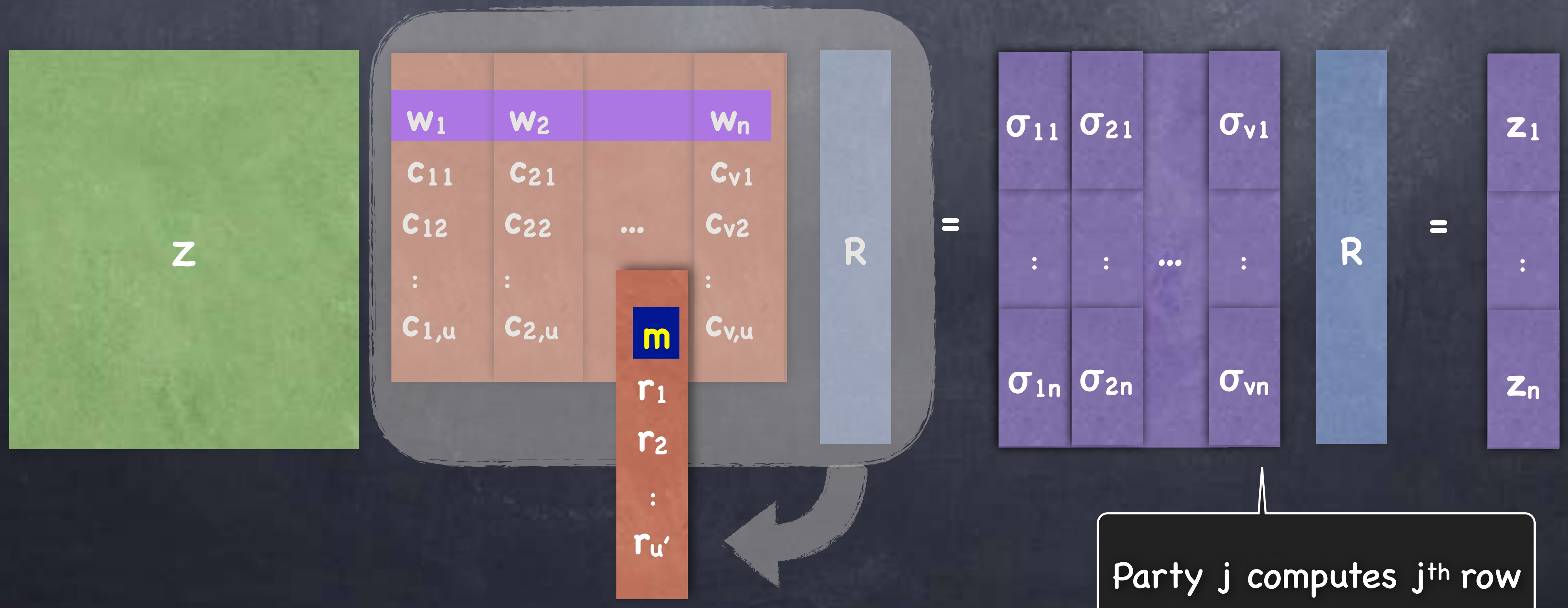


Switching Schemes

- Can move from any linear secret-sharing scheme W to any other linear secret-sharing scheme Z "securely"

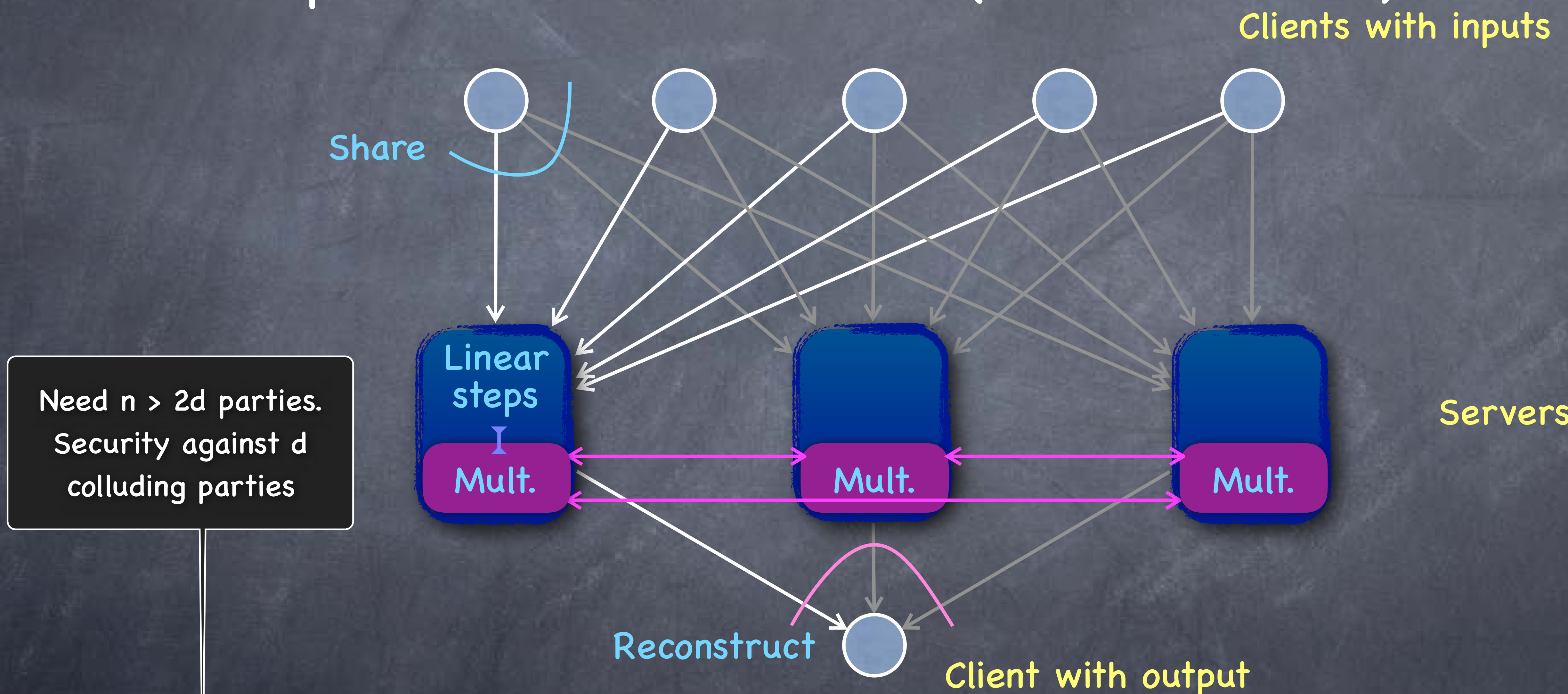


- Given shares $(w_1, \ldots, w_n) \leftarrow$ W.Share(m)
- Share each w_i using scheme Z: $(\sigma_{i1},\ldots,\sigma_{in}) \leftarrow$ Z.Share(w_i)
- Locally each party j reconstructs using scheme W: $z_j \leftarrow$ W.Recon $(\sigma_{1j},\ldots,\sigma_{nj})$



MPC from Shamir Secret-Sharing

- A function f given as a program with linear steps and multiplications: arithmetic circuit (over a finite field)



- Locally multiplying degree d shares of M_1 and M_2 gives a degree 2d share of $M_1 \cdot M_2$. Then switch back to a degree d share (involves communicating deg. d shares of deg. 2d shares)

MPC Protocols

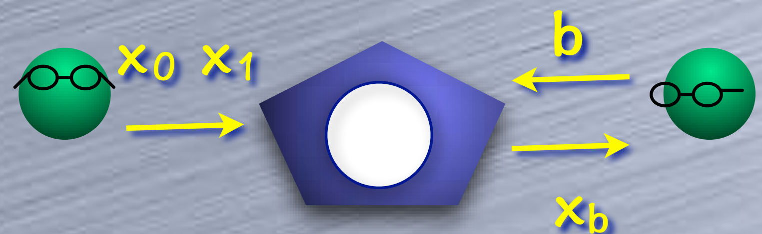
- For every function, if there is an honest majority (or #honest parties > n/3, when colluding parties can be actively corrupt)
  - Based on [Secret-Sharing]() and broadcast (e.g., BGW scheme)
  - No computational hardness required
  - Not applicable in the 2-party setting
- Also possible for every function when there is no honest majority (e.g., in the 2-party setting), using [Oblivious Transfer]()
  - e.g., scheme based on Yao's Garbled Circuit (a special case of "[Randomized Encoding]()") or "[MPC in the head]()" techniques
  - Oblivious Transfer requires computational hardness
- Also possible for a few functions without honest majority and without computational hardness

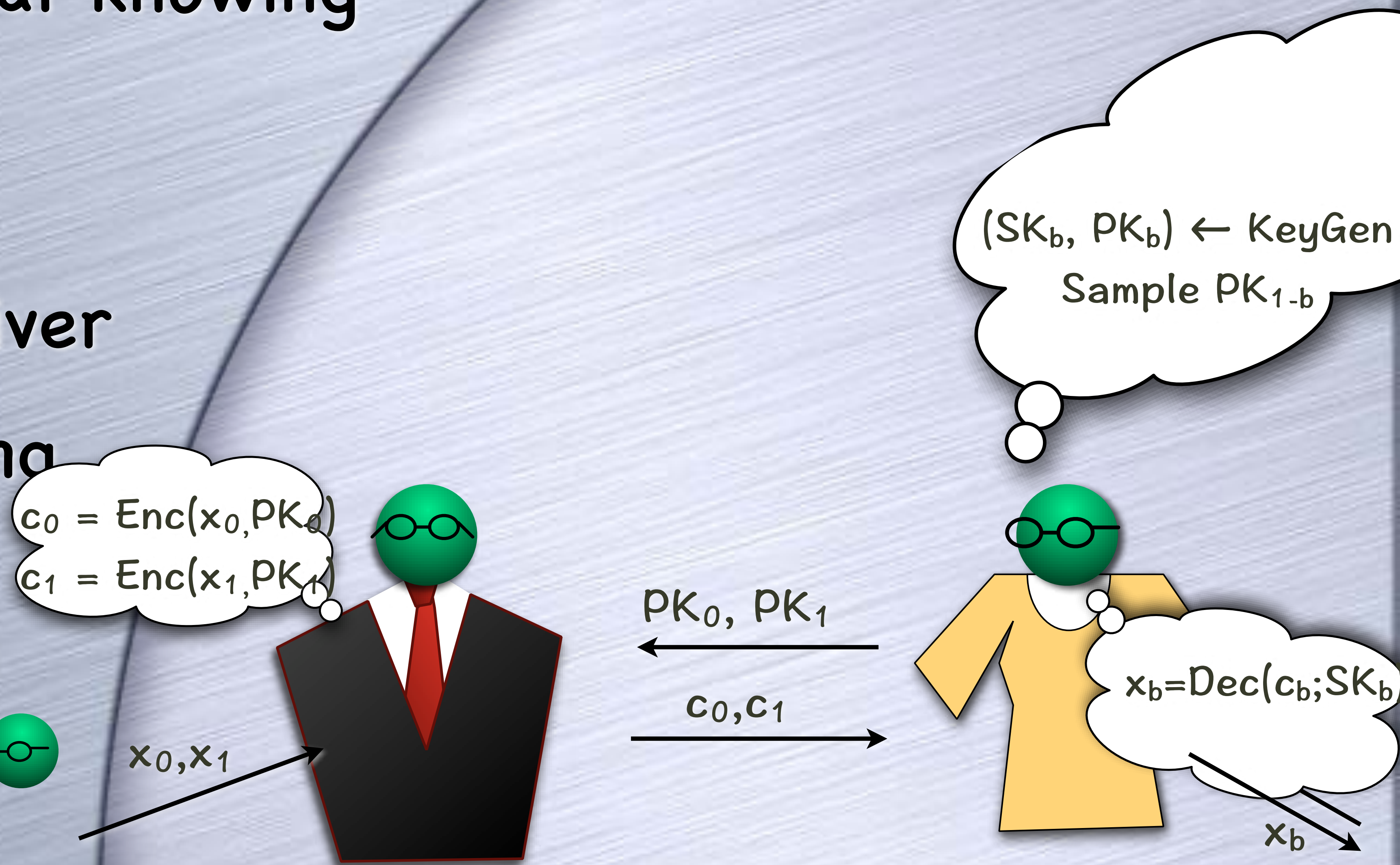
Or similar

Exactly which ones remains open!

An OT Protocol (passive corruption)

- Using a (special) encryption
  - PKE in which one can sample a public-key without knowing secret-key
- c_{1-b} inscrutable to a passive corrupt receiver
- Sender learns nothing about b





Additional Aspects

- Efficiency/Scalability
- What is OK to compute
  - cf. Differential Privacy
  - Effect of "leakages"?
- Network model
  - Incomplete network
  - Variable set of participants
  - Not everyone online at once
  - Asynchronous communication vs. need for good throughput
- ...

Securing Cloud Storage

- **Private Information Retrieval**
  - Retrieve D[i] from a server without revealing i
- **Oblivious RAM**
  - Allow read and write operations on data stored on the server, and do not reveal access pattern
- **Searchable Encryption**
  - Allow search operations on data stored encrypted on the server (OK to reveal the access pattern)

Homomorphic Encryption

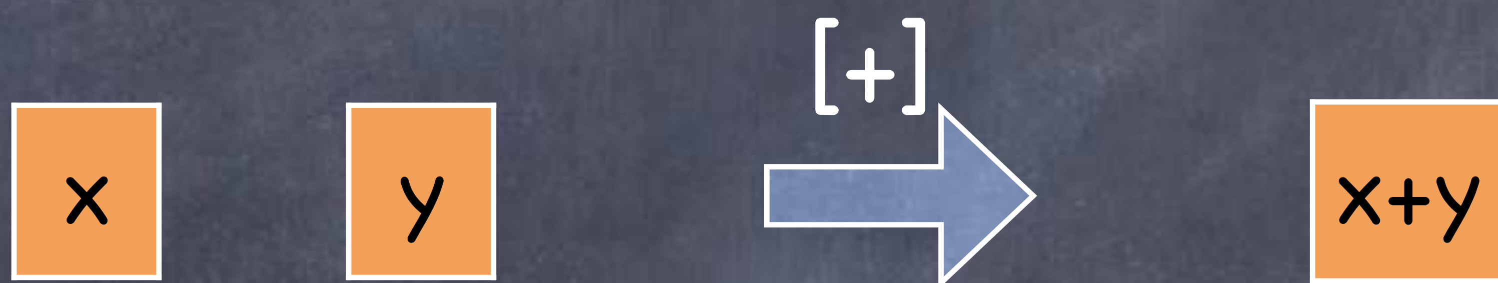
- Group Homomorphism: Two groups G and G′ are homomorphic if there exists a function (homomorphism) $f:G \rightarrow G'$ such that for all $x,y \in G$, $f(x) +_{G'} f(y) = f(x +_G y)$
- Homomorphic Encryption: A CPA secure (public-key) encryption s.t. $\mathrm{Dec}(C) +_M \mathrm{Dec}(D) = \mathrm{Dec}(C +_C D)$ for ciphertexts C, D
  - i.e. $\mathrm{Enc}(x) +_C \mathrm{Enc}(y)$ is like $\mathrm{Enc}(x +_M y)$
  - Interesting when $+_C$ doesn't require the decryption key
- e.g. El Gamal Encryption:
  Given PK, $Y=g^y$, $\mathrm{Enc}_Y(m;x) = (g^x, mY^x)$ (x being random).
  So, $(g^{x1}, m_1Y^{x1}) \times (g^{x2}, m_2Y^{x2}) = (g^{x3}, m_1m_2Y^{x3})$

Homomorphic Encryption: Examples

- Group structure of the message space determined by the construction of the encryption scheme
  - ElGamal: Message space is a group where the Decisional Diffie-Hellman assumption is made (e.g., quadratic residues modulo a "safe prime," with group operation being modular multiplication)
  - Goldwasser-Micali: Message space is $\mathbb{Z}_2$ (i.e., bits with XOR as the group operation)
  - Paillier: Message space is $\mathbb{Z}_n$ for specially chosen n (group operation is modular addition)
  - Damgård-Jurik: Message space $\mathbb{Z}_{n^s}$ yields ciphertext space within $\mathbb{Z}_{n^{s+1}}$. Enables encrypt-compute-encrypt-compute-...

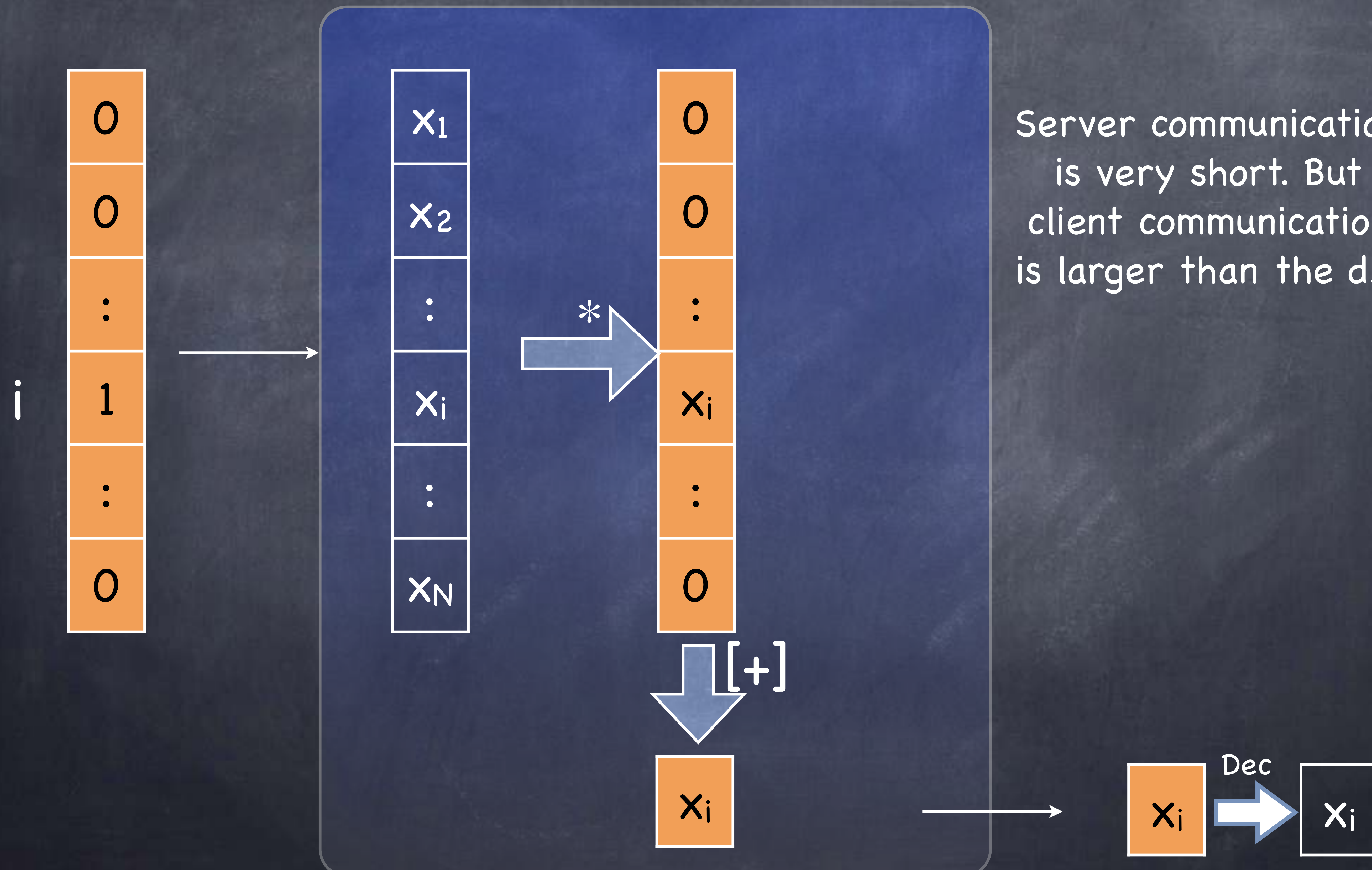
Computational PIR from HE

- In the following: database values are <u>integers</u> in [0,m), and we can use any homomorphic encryption scheme with a message space isomorphic with $\mathbb{Z}_n$ with $n \geq m$
  - e.g., Paillier encryption with message space $\mathbb{Z}_n$ ($n \geq m$)



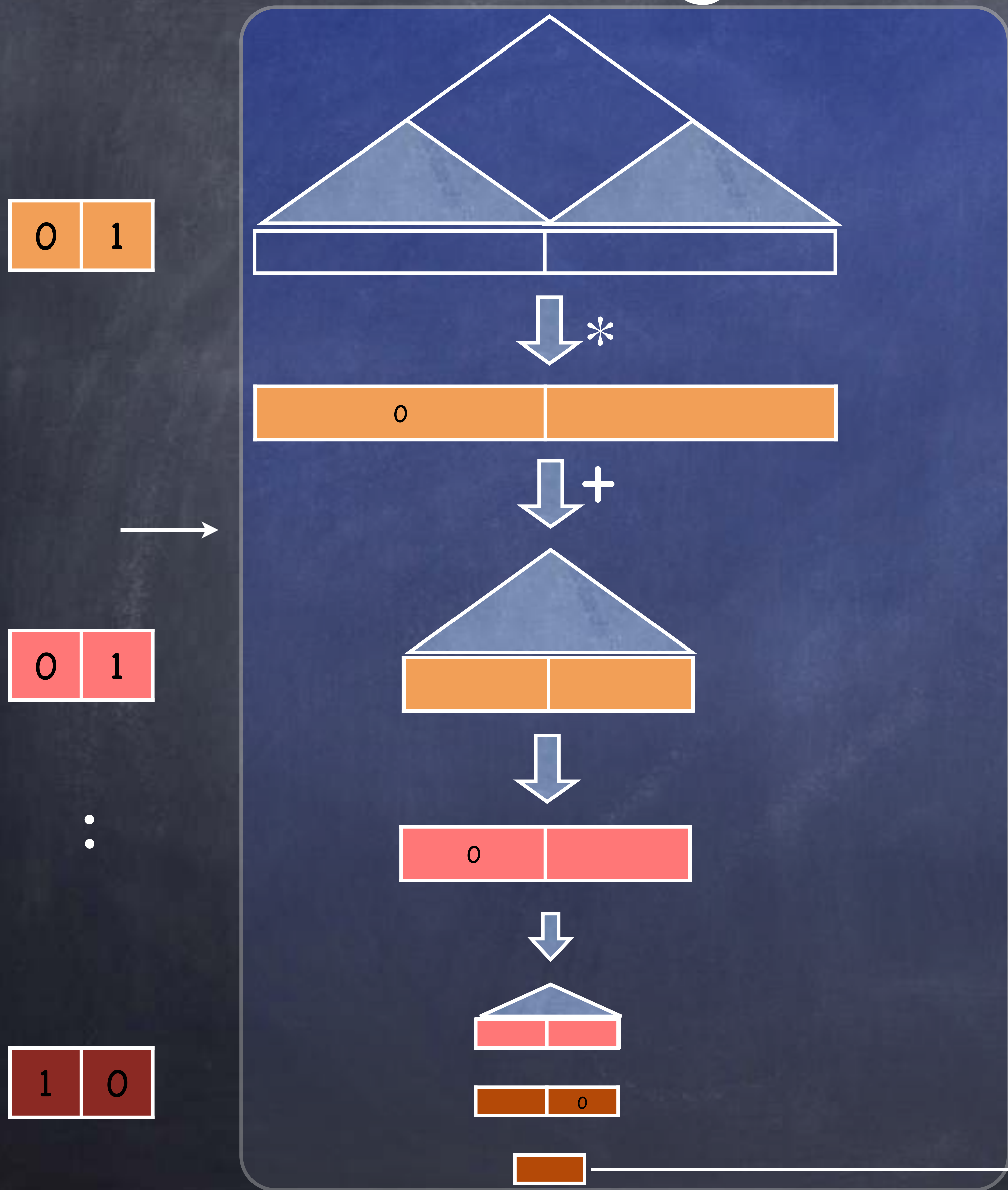
- For integer a and ciphertext $\underline{c}$, can define $a*\underline{c}$ recursively: $0*\underline{c} = E(0)$; $1*\underline{c} = \underline{c}$; $(a+b)*\underline{c} = a*\underline{c}$ [+] $b*\underline{c}$.

Private Information Retrieval



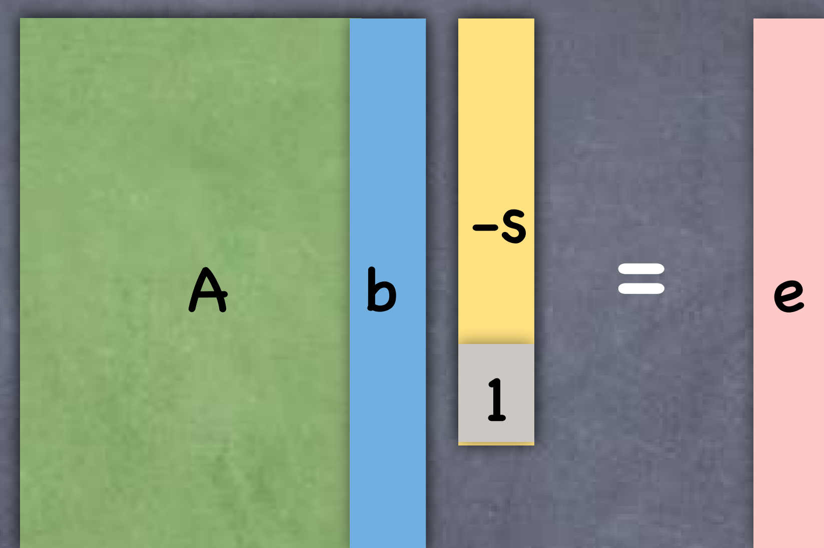
Server communication is very short. But client communication is larger than the db!

Full-Fledged PIR protocol



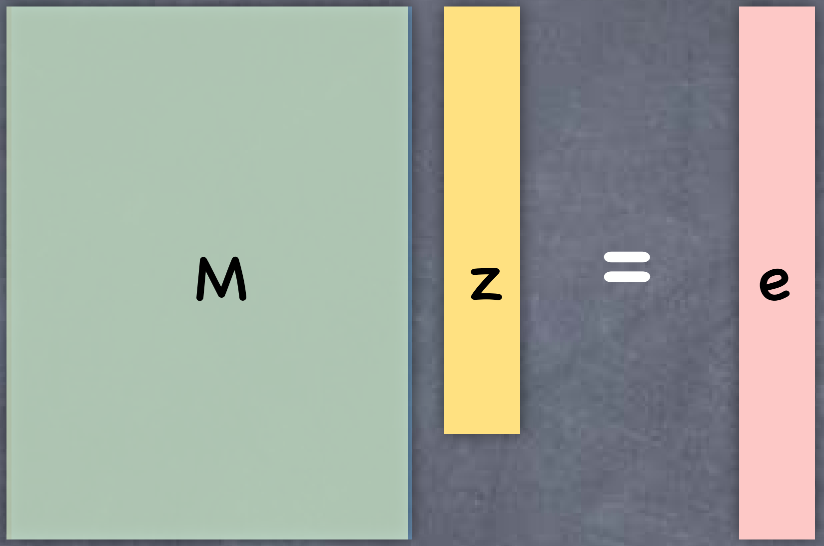
- Uses Damgård-Jurik to treat ciphertext at one level as plaintext at the next level
- Total communication from server = O(log N log m)

Learning With Errors



- A consequence of LWE: Can generate a pseudorandom matrix $M \in \mathbb{Z}_q^{m \times n'}$ and $\underline{\mathbf{z}} \in \mathbb{Z}_q^{n'}$ s.t. entries of $M\underline{\mathbf{z}}$ are all "small"

FHE from LWE



- A scheme to encrypt bits, supporting homomorphic NAND
- $\mathrm{Enc}(\mu) = M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m \times km}$ and $G \in \mathbb{Z}_q{}^{n \times km}$ to be defined
- $\mathrm{Dec}_z(C)$: $\underline{z}^T C = \underline{\delta}^T + \mu \underline{z}^T G$ where $\underline{\delta}^T = e^T R$. Check if it is "small."
- $\mathrm{NAND}(C_1,C_2)$: $G - C_1 \cdot B(C_2)$, where B carries out "bit decomposition"
  - G is a ciphertext of 1, and $C_1 \cdot B(C_2)$ works as $\mathrm{AND}(C_1,C_2)$

FHE from LWE

- A scheme to encrypt bits, supporting homomorphic NAND
- $\mathrm{Enc}(\mu) = M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m \times km}$ and $G \in \mathbb{Z}_q^{n \times km}$ to be defined
- $\mathrm{Dec}_z(C)$: $\underline{\mathbf{z}}^T C = \underline{\delta}^T + \mu \underline{\mathbf{z}}^T G$ where $\underline{\delta}^T = e^T R$. Check if it is "small."
- $\mathrm{NAND}(C_1,C_2)$: $G - C_1 \cdot B(C_2)$, where B carries out "bit decomposition"
  - G is a ciphertext of 1, and $C_1 \cdot B(C_2)$ works as $\mathrm{AND}(C_1,C_2)$
  - G is the matrix that inverts bit decomposition: $G \cdot B(X) = X$
  - $\underline{\mathbf{z}}^T C_1 \cdot B(C_2) = \underline{\mathbf{z}}^T C_1 \cdot B(C_2) = (\underline{\delta}_1^T + \mu_1 \underline{\mathbf{z}}^T G)\, B(C_2)$

    $= \underline{\delta}_1^T B(C_2) + \mu_1 \underline{\mathbf{z}}^T C_2 = \underline{\delta}^T + \mu_1 \mu_2 \underline{\mathbf{z}}^T G$

    where $\underline{\delta}^T = \underline{\delta}_1^T B(C_2) + \mu_1 \underline{\delta}_2^T$ has "small" entries
  - In general, error gets multiplied by km. Allows depth $\approx \log_{km} q$

Keywords

- Secure multi-party computation
  - Linear Secret-Sharing
  - Passive secure BGW protocol
  - Omitted: Yao's Garbled Circuit, Randomized Encoding, Conditional Disclosure of Secrets, Composition (e.g., UC security), ...
- Private Information Retrieval, Oblivious RAM, Searchable Encryption
- Homomorphic Encryption & Fully Homomorphic Encryption
- More tools in the horizon
  - Obfuscation, Functional Encryption, ...