Secure Computation: *Why, How and When*

Mariana Raykova

Yale University

Predictive Model

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Given samples $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)$
  - $\mathbf{x_i} \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- Learn a function f such that $f(\mathbf{x_i}) = y_i$

Linear Regression

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhythmia	…	Inflammation	Dysphagia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Given samples $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)$
  - $\mathbf{x_i} \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- Learn a function f such that $f(\mathbf{x_i}) = y_i$

f is well approximated by a linear map
$y_i \approx \theta^T x_i$

Distributed Data

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhythmia	…	Inflammation	Dysphagia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **Shared database** - $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ do not belong to the same party

Horizontally Partitioned Database

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **Different rows belong to different parties**
  - E.g., each patient has their own information

Vertically Partitioned Database

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **Different columns belong to different parties**
  - E.g., different specialized hospitals have different parts of the information for all patients

Can the parties holding the distributed data construct the predictive model on the whole database **while protecting the privacy of their inputs?**

~~**Without sending their own data to other parties**~~

Without revealing more about their own inputs

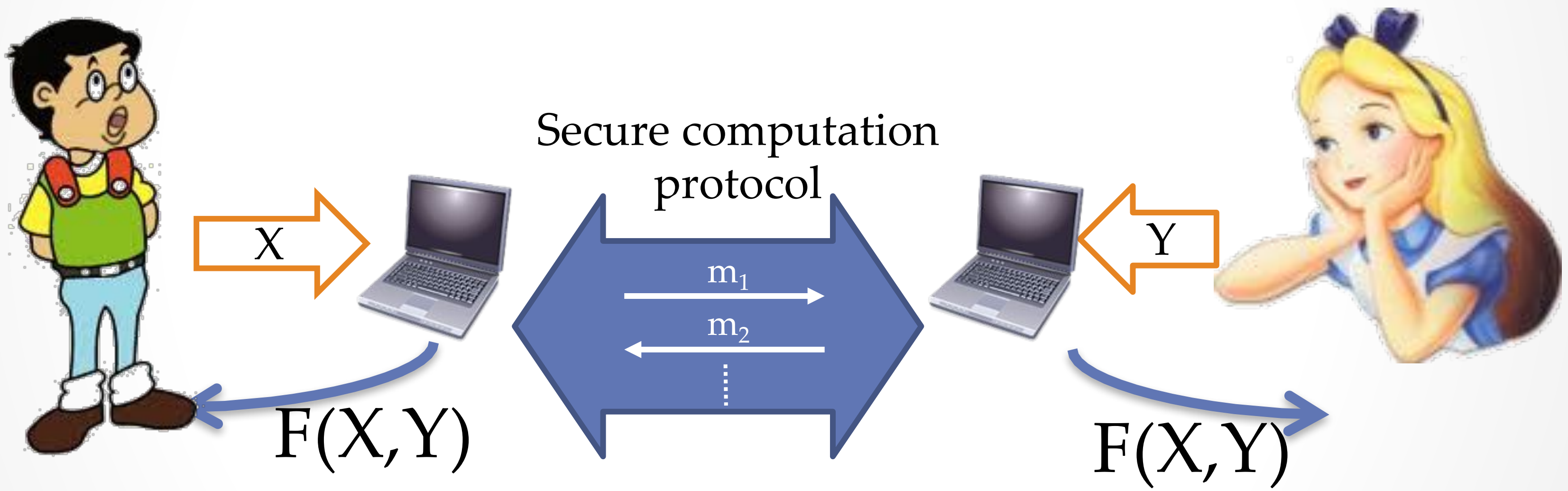
Secure Computation

Alice and Bob want to compute F(X,Y)
without revealing their inputs

X

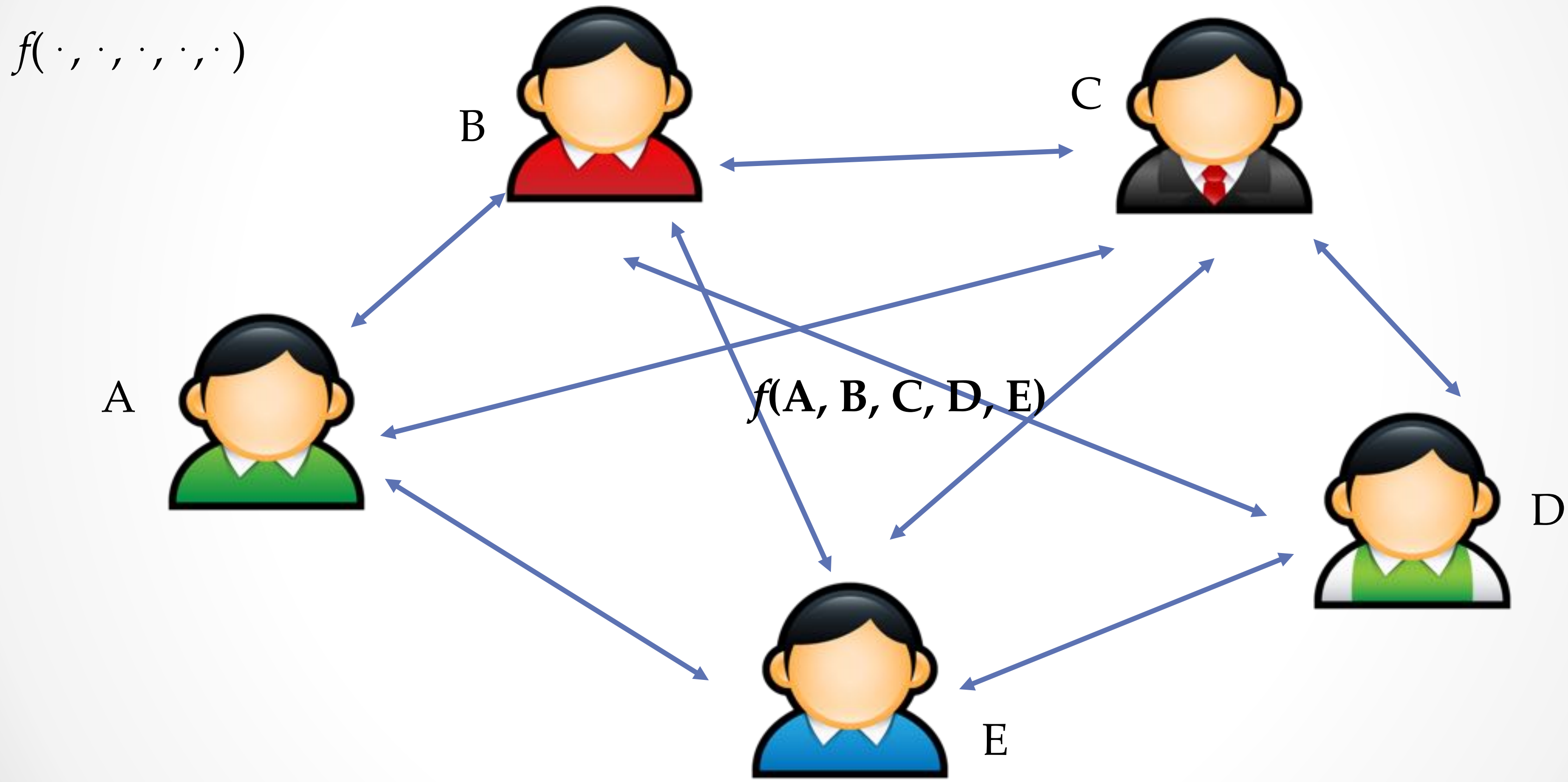
Y

Secure Computation



Security: **the parties cannot learn more than what is revealed by the result**

Secure Multiparty Computation (MPC)



Security: **the parties cannot learn more than what is revealed by the result**

Applications

- Auctions:
  - inputs: bids; output: winner, price to pay

  - Sugar beet auction in Denmark, 2008
  - Energy trade auctions

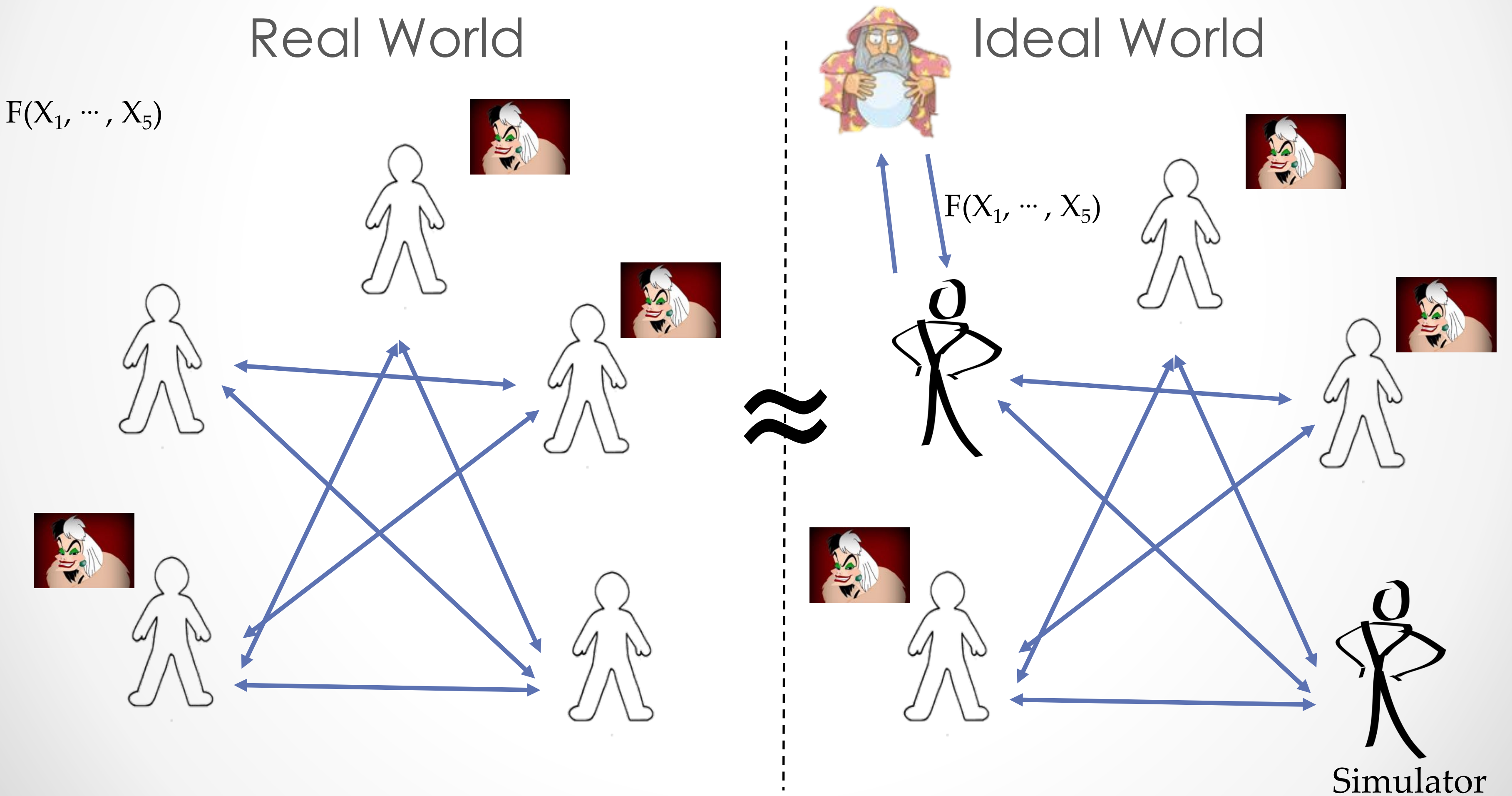
What Does and Does Not MPC Guarantee?

Guarantee: The computation does not reveal more than what the output reveals.

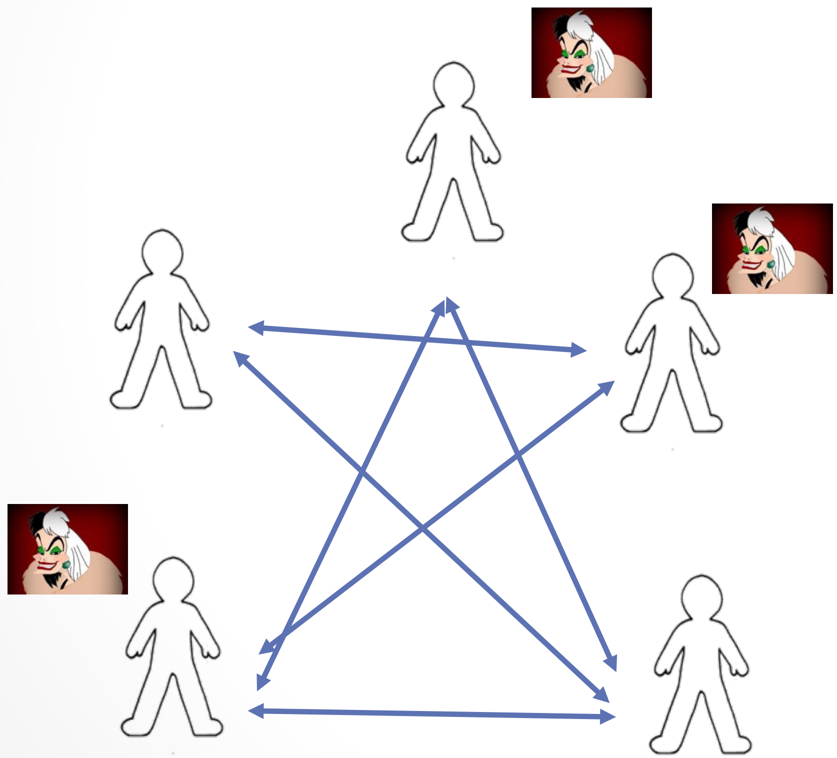
No Guarantee:
How much does the output reveal.

Differential Privacy

Security



Adversarial Models



Adversary behavior:

- **Semi-honest** – corrupt parties follow the MPC protocol
- **Malicious** – corrupt parties deviate arbitrarily from the MPC protocol

Party corruption:

- **Static** – corrupted parties are chosen before the start of the MPC protocol execution
- **Adaptive** – parties can be corrupted during the execution

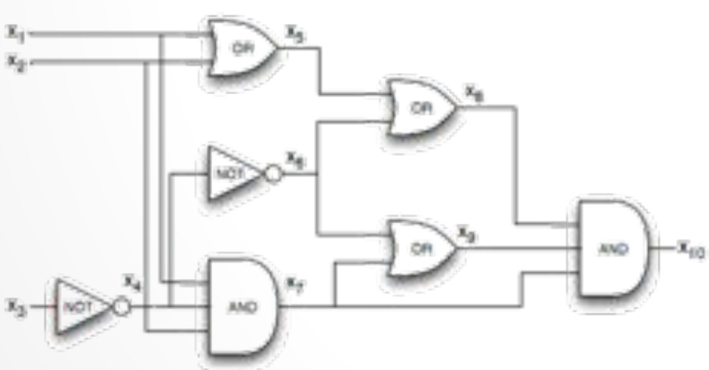
What Can We Compute Securely?

- **We can compute securely any function!**
  - [Yao82, GMW87, CDv88, BG89, BG90, Cha90, Bea92,CvT95, CFGN96, Gol97, HM97, CDM97, FHM98, BW98,KOR98, GRR98, FvHM99, CDD+99, HMP00, CDM00, SR00,CDD00, HM00, Kil00, FGMO01, HM01, CDN01, Lin01,FGMv02, Mau02, GIKR02, PSR02, NNP03, FHHW03, KOS03,CFIK03, Lin03c, DN03, MOR03, CKL03, Pin03, PR03, NMQO+03,Lin03b, Lin03a, Lin03d, FWW04, FHW04, Pas04, IK04,HT04, ST04, KO04, MP04, ZLX05, CDG+05, HNP05, FGMO05, GL05, HN05, DI05, JL05, Kol05, WW05, vAHL05, LT06,CC06, DFK+06, BTH06, HN06, IKLP06, DI06, FFP+06,ADGH06, Dam06, MF06, CKL06, DPSW07, Kat07b, CGOS07,HIK07, DN07, Pen07, NO07, Kat07a, IKOS07, BMQU07,HK07, LP07, Woo07, BDNP08, QT08, PR08, HNP08, GK08,GMS08, SYT08, DIK+08, PCR08, KS08, Lin08, LPS08,GHKL08, CEMY09, GP09, GK09, MPR09, ZHM09, AKL+09,Tof09, BCD+09, DGKN09, DNW09, Lin09b, PSSW09, Lin09a,CLS09, LP09, Unr10, DO10, IKP10, DIK10,GK10,……..]

Computation Over Circuits

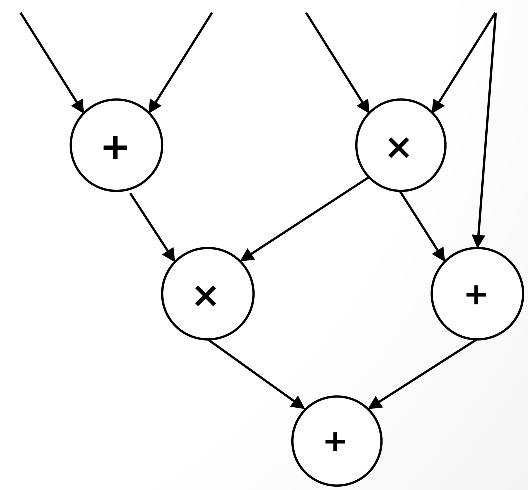
Boolean Circuits

- **Yao Gabled Circuits**



Arithmetic Circuits

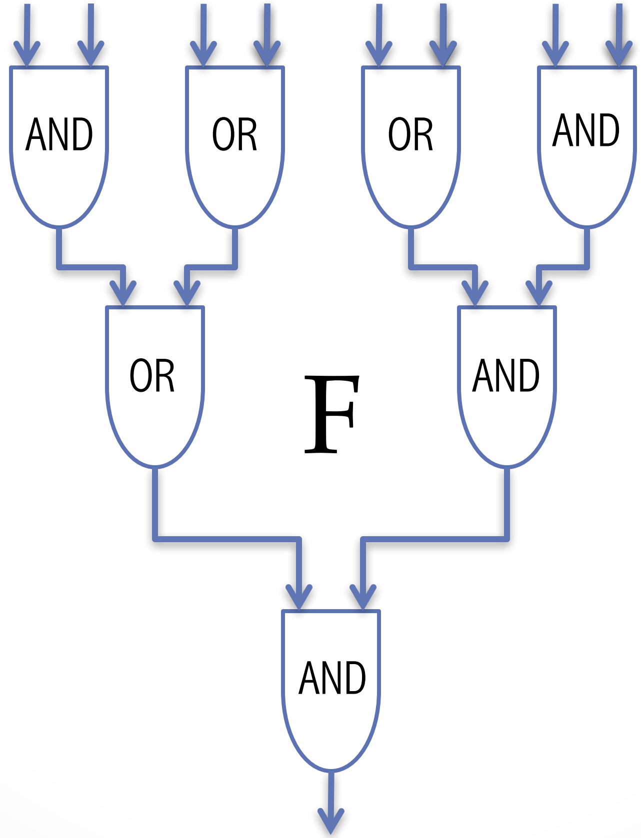
- **BGW Construction**
  - Ben-Or, Goldwasser, Widgerson



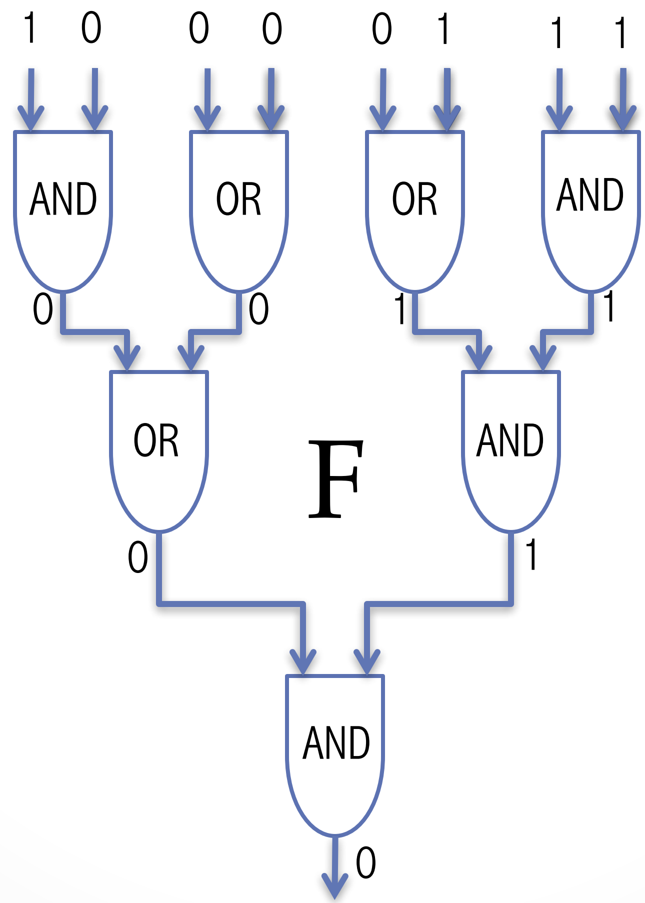
Yao Garbled Circuits

Two Party Computation

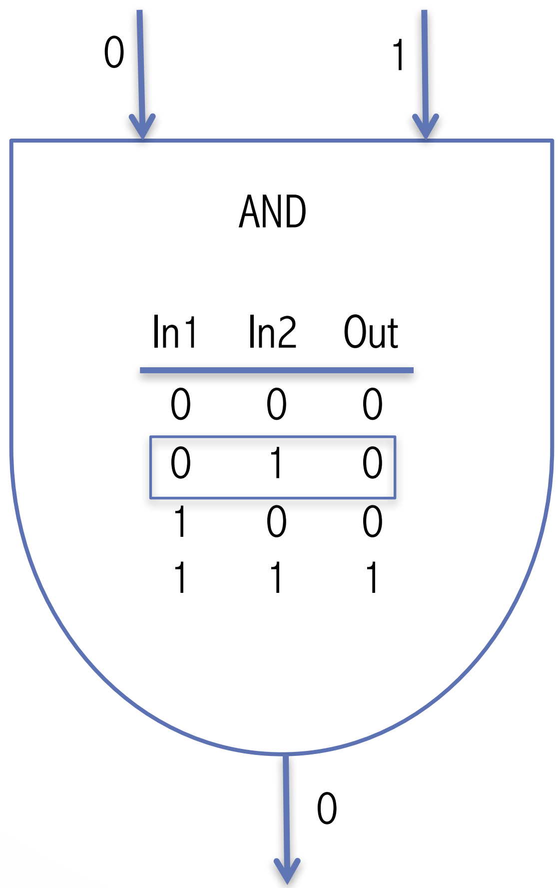
Circuit Evaluation



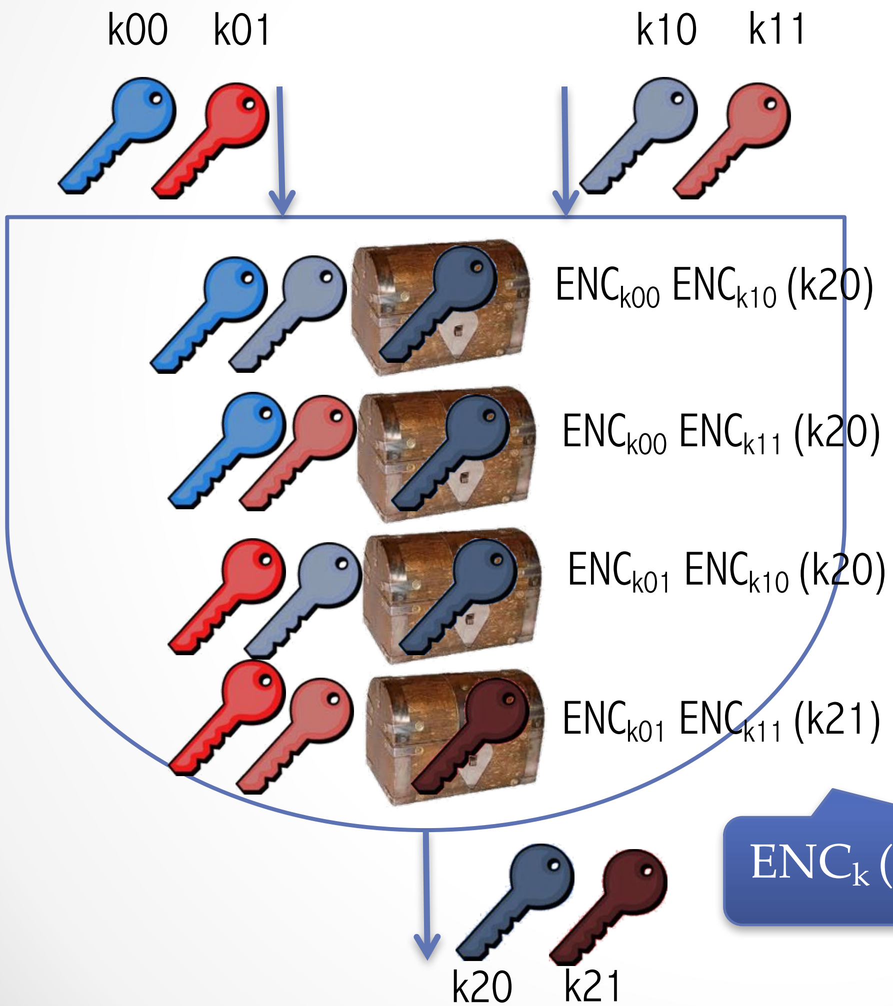
Circuit Evaluation

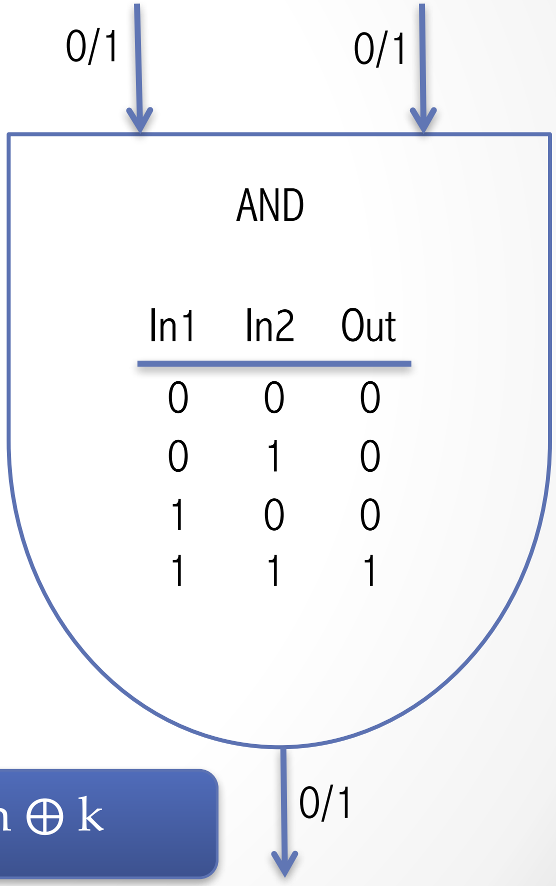


Evaluation



Yao Garbled Evaluation

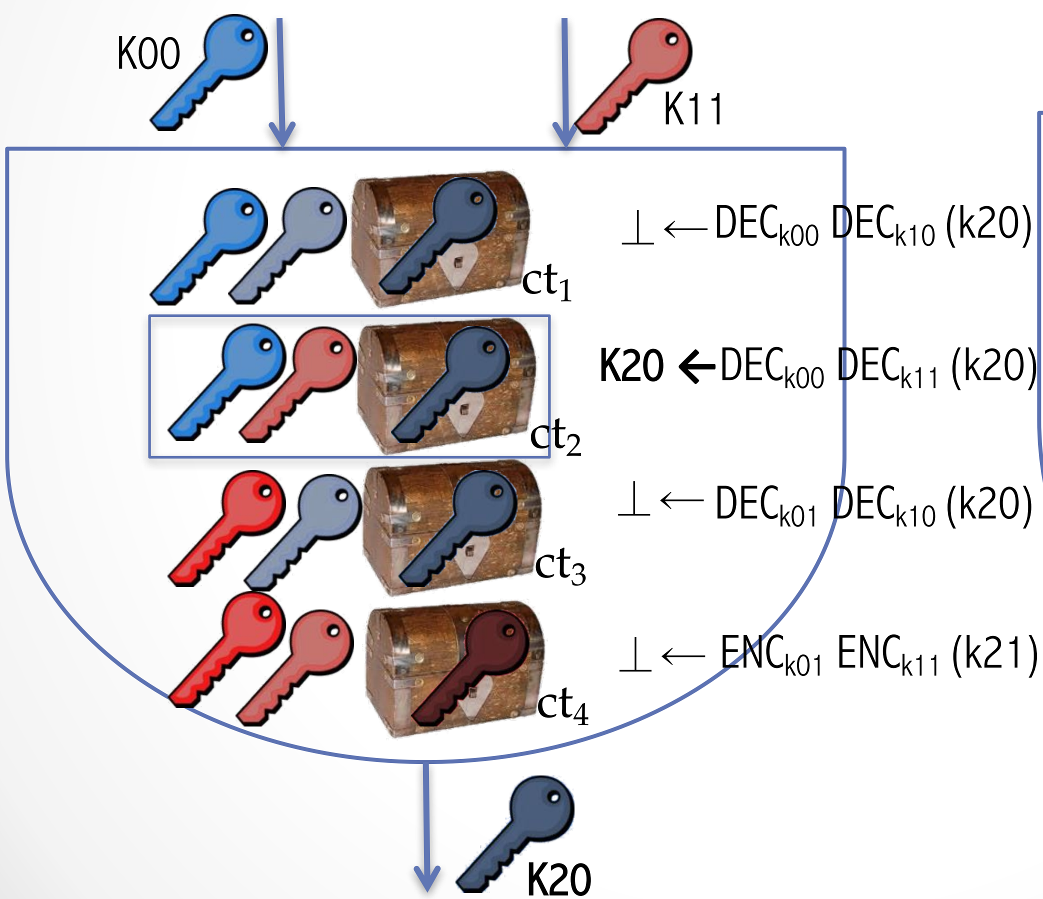


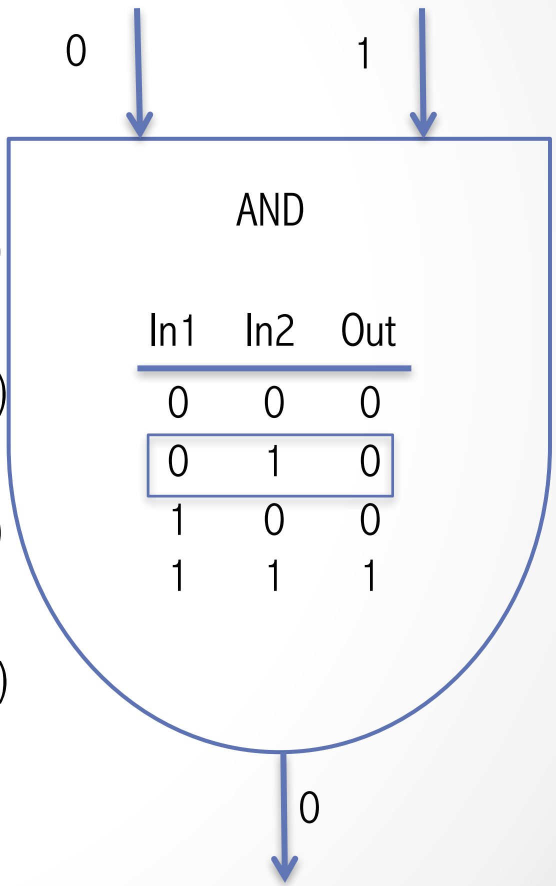


In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

$ENC_k(m) = m \oplus k$

Garbled Evaluation

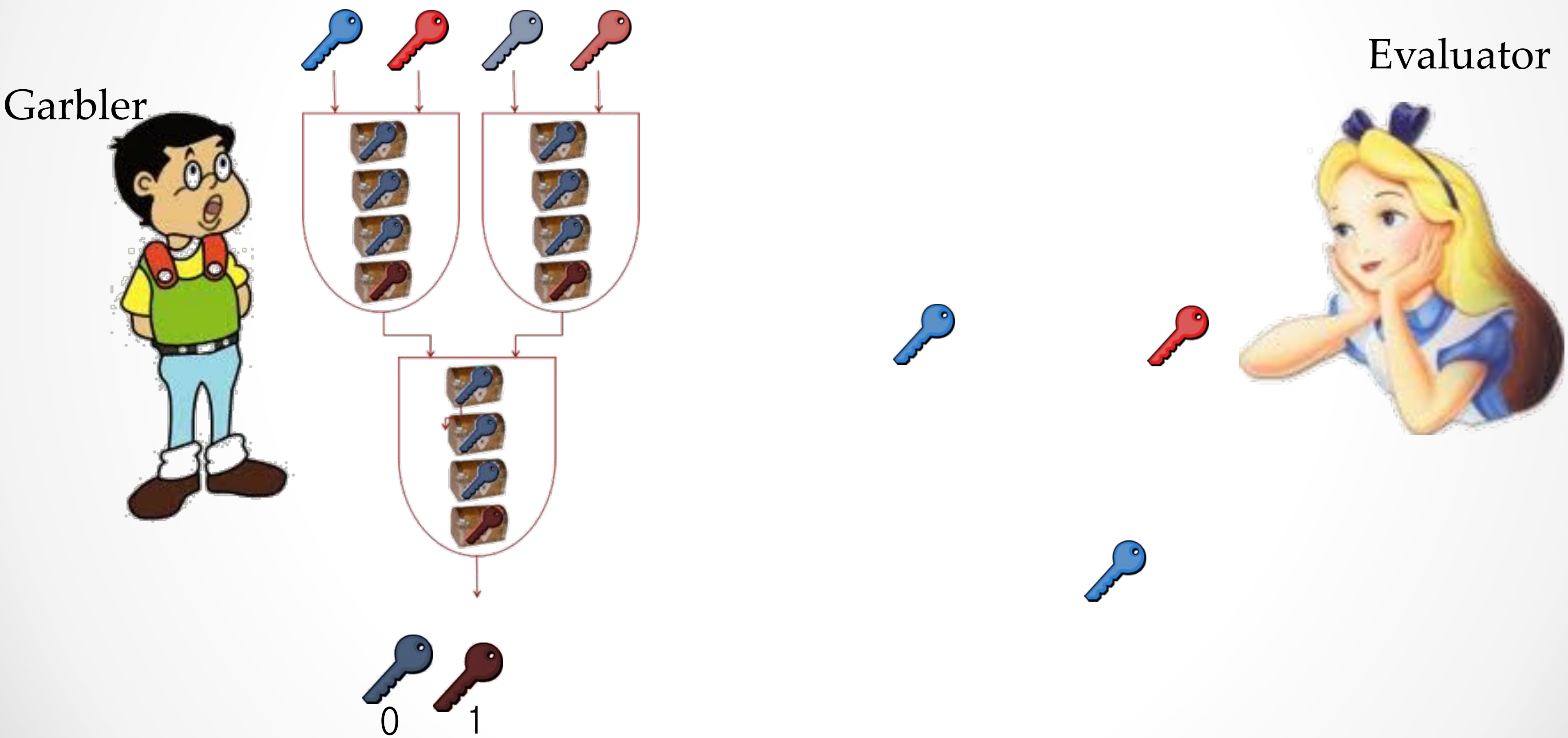




In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Secure Computation

F (X_{alice}, Y_{bob})



Oblivious Transfer (OT)

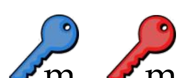
Sender
Inputs: m_0, m_1

Receiver
Inputs: b

Output: $\perp$

Output: m_b

For each inputs wire corresponding to evaluator's input execute OT



m_0 m_1

b

Output: m_b

The Evolution Of Garbled Circuits

	Size (x sec.param) AND	XOR	Garble cost AND	XOR	Eval cost AND	XOR	Assumption
Classical [Yao86]	large		8		5		PKE
P&P [BMR90]	4	4	4/8	4/8	1/2	1/2	hash/PRF
GRR3 [NPS99]	3	3	4/8	4/8	1/2	1/2	PRF/hash
Free XOR [KS08]	3	0	4	0	1	0	circ. hash
GRR2 [PSSW09]	2	2	4/8	4/8	1/2	1/2	PRF/hash
FlexOR [KMR14]	2	{0,1,2}	4	{0,1,2}	1	{0,1,2}	circ. symm
HalfGates [ZRE15]	2	0	4	0	2	0	circ. hash

Threshold gates, garbling arithmetic operations [BMR16]

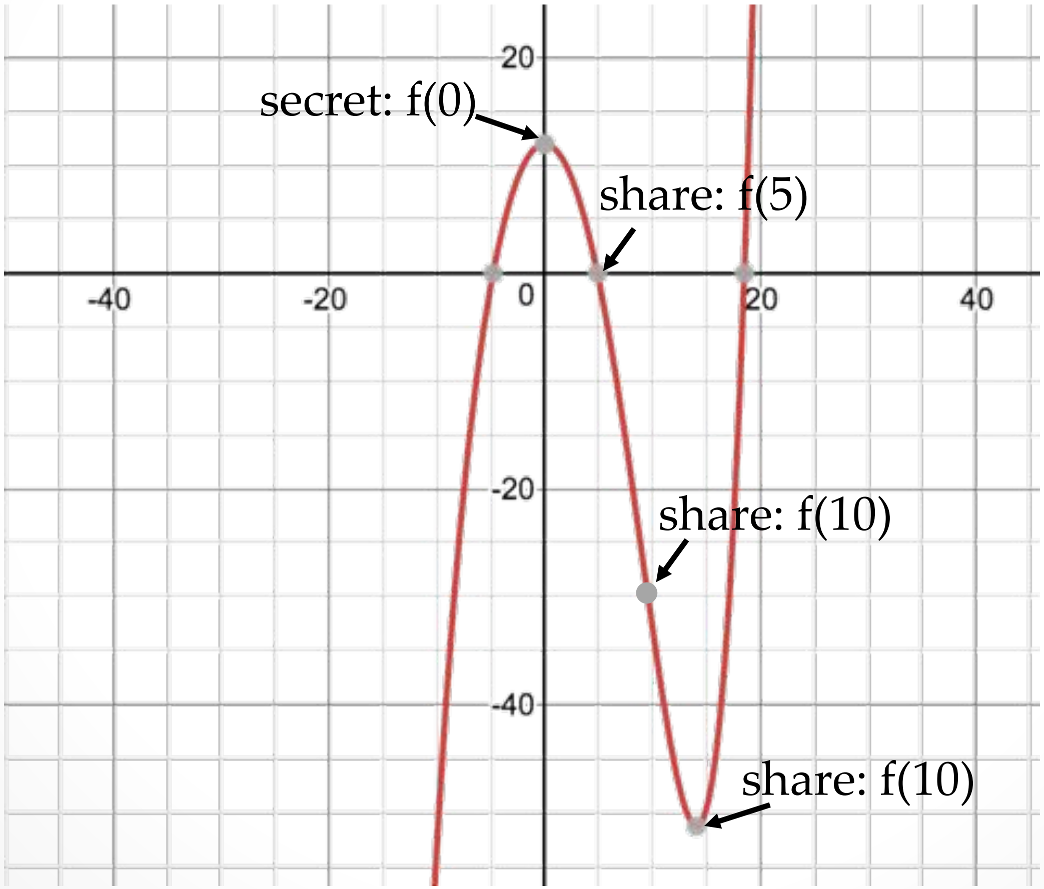
- Asymptotic and concrete improvements

* Comparison table, thanks Mike Rosulek

BGW Protocol

Multi Party Computation for Arithmetic Circuits

Shamir's Secret Sharing

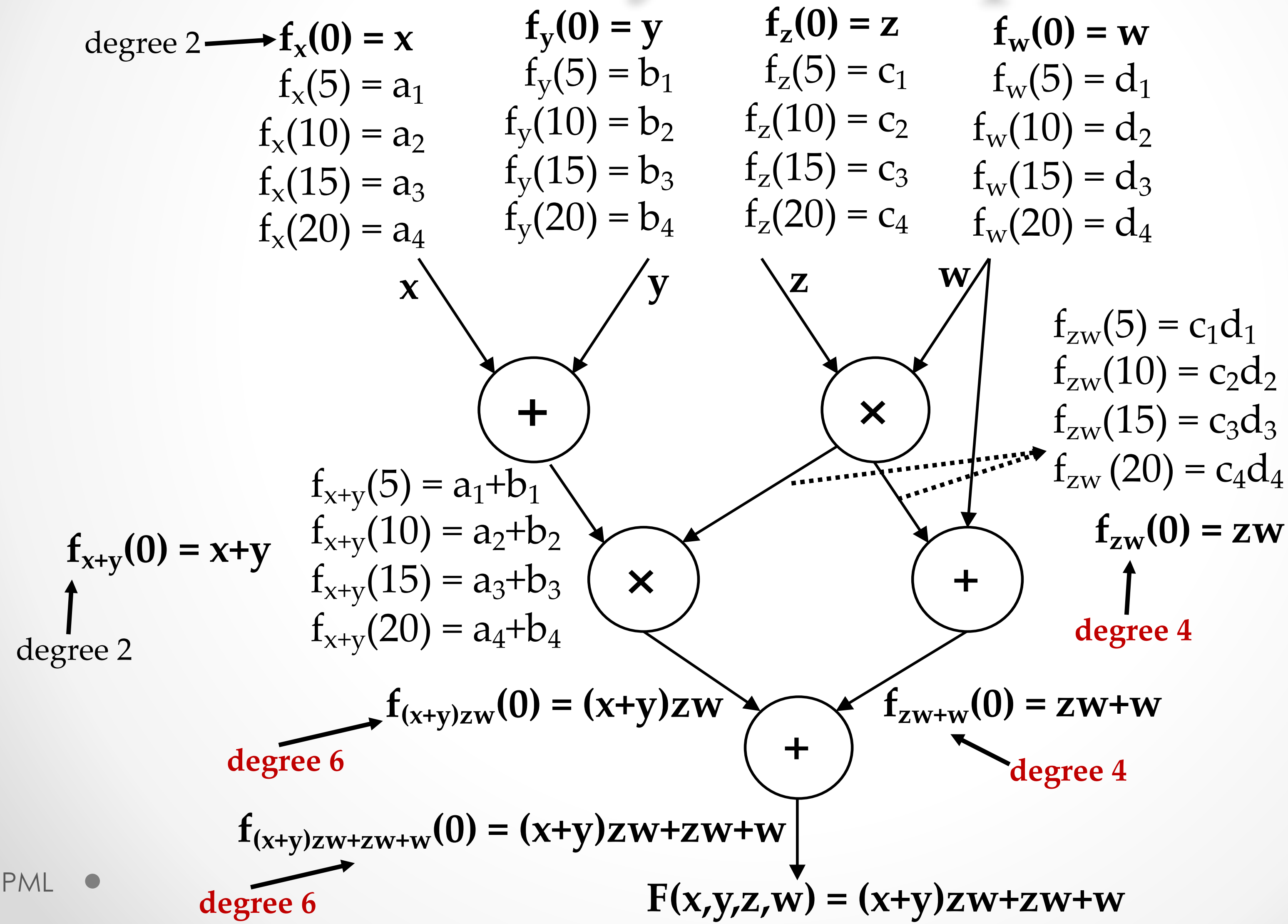


t-out-of-n sharing: random degree t polynomial

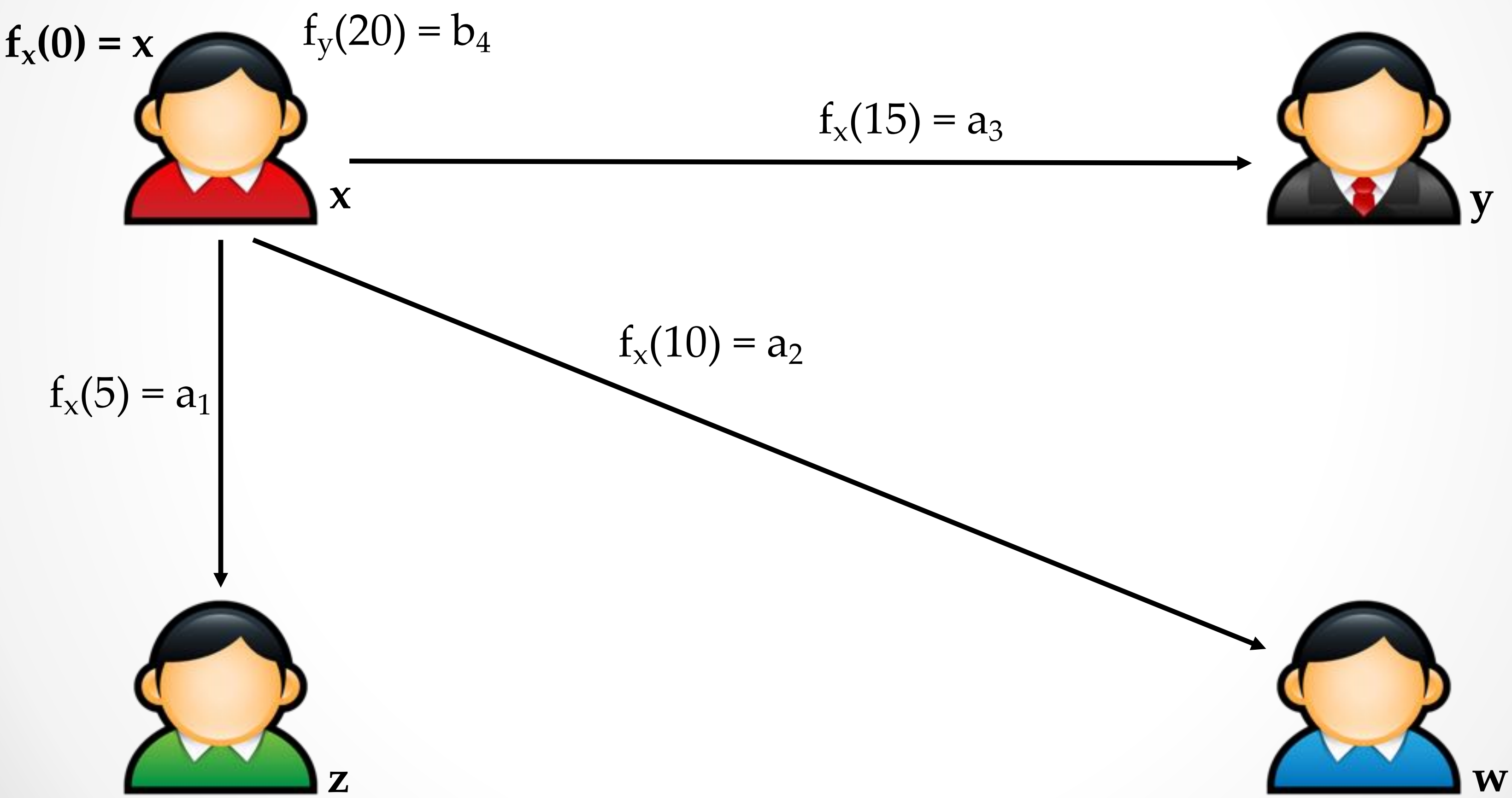
t shares reveal nothing about the secret

t+1 shares interpolate the secret

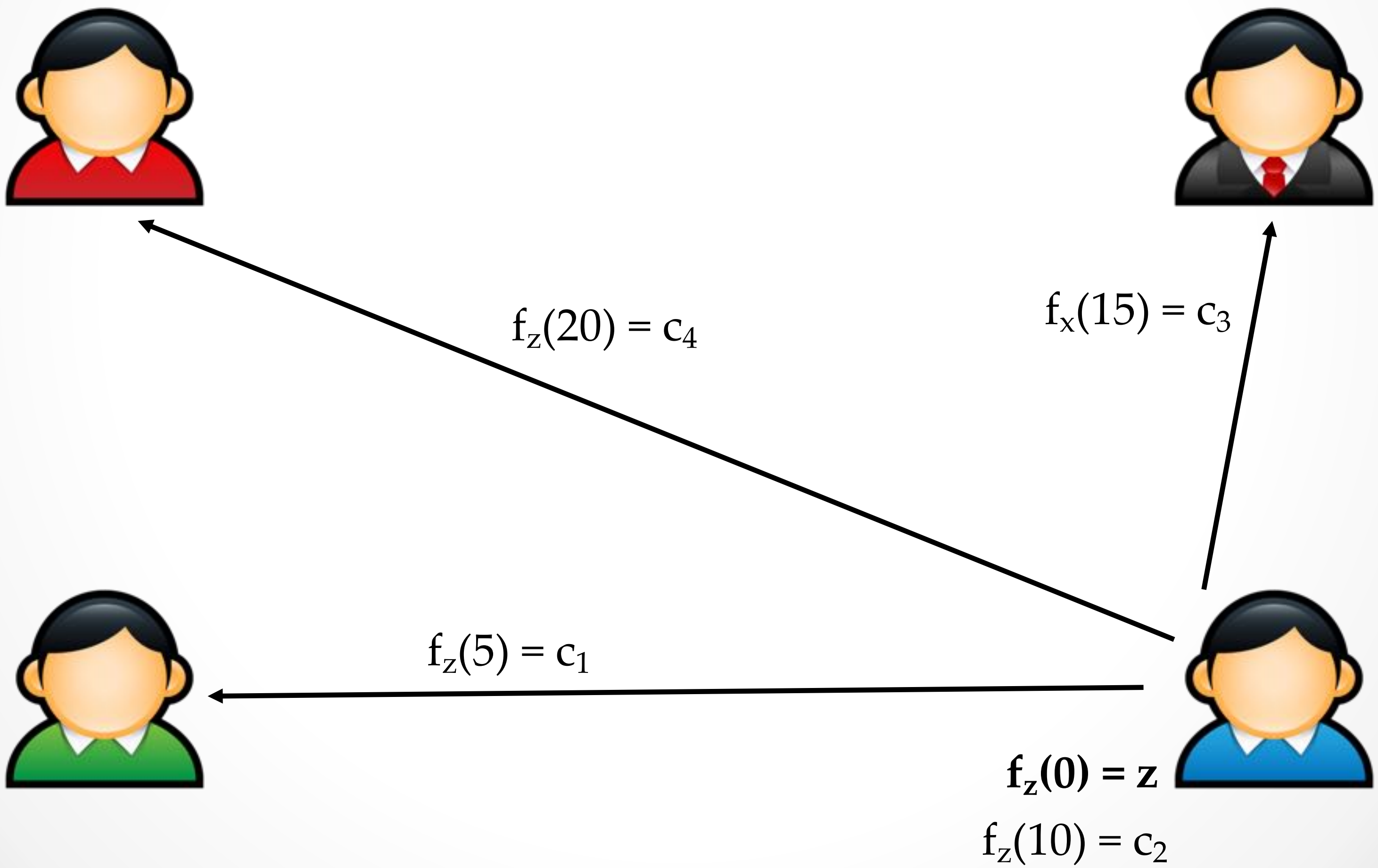
Multi-Party Computation



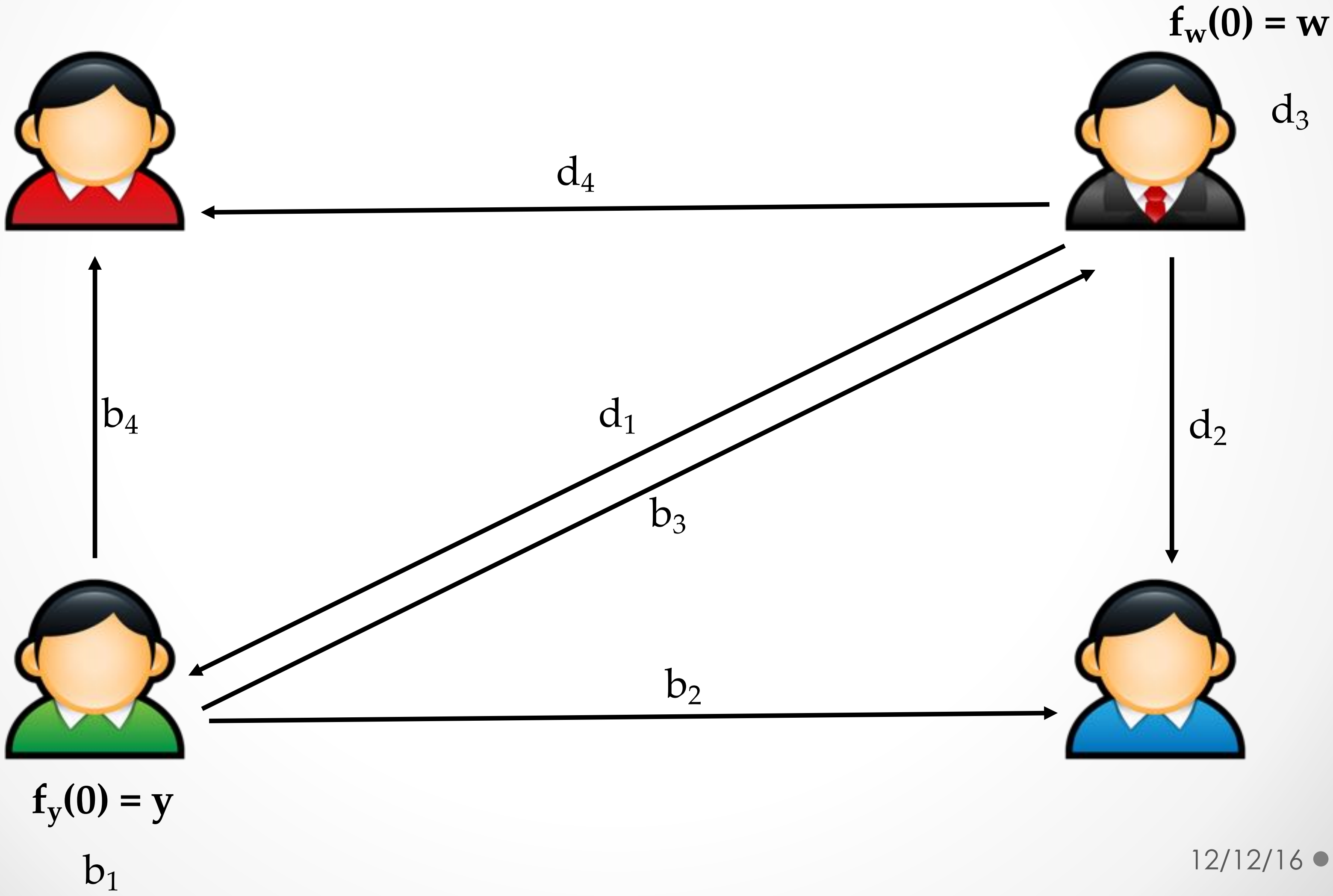
Multi-Party Computation



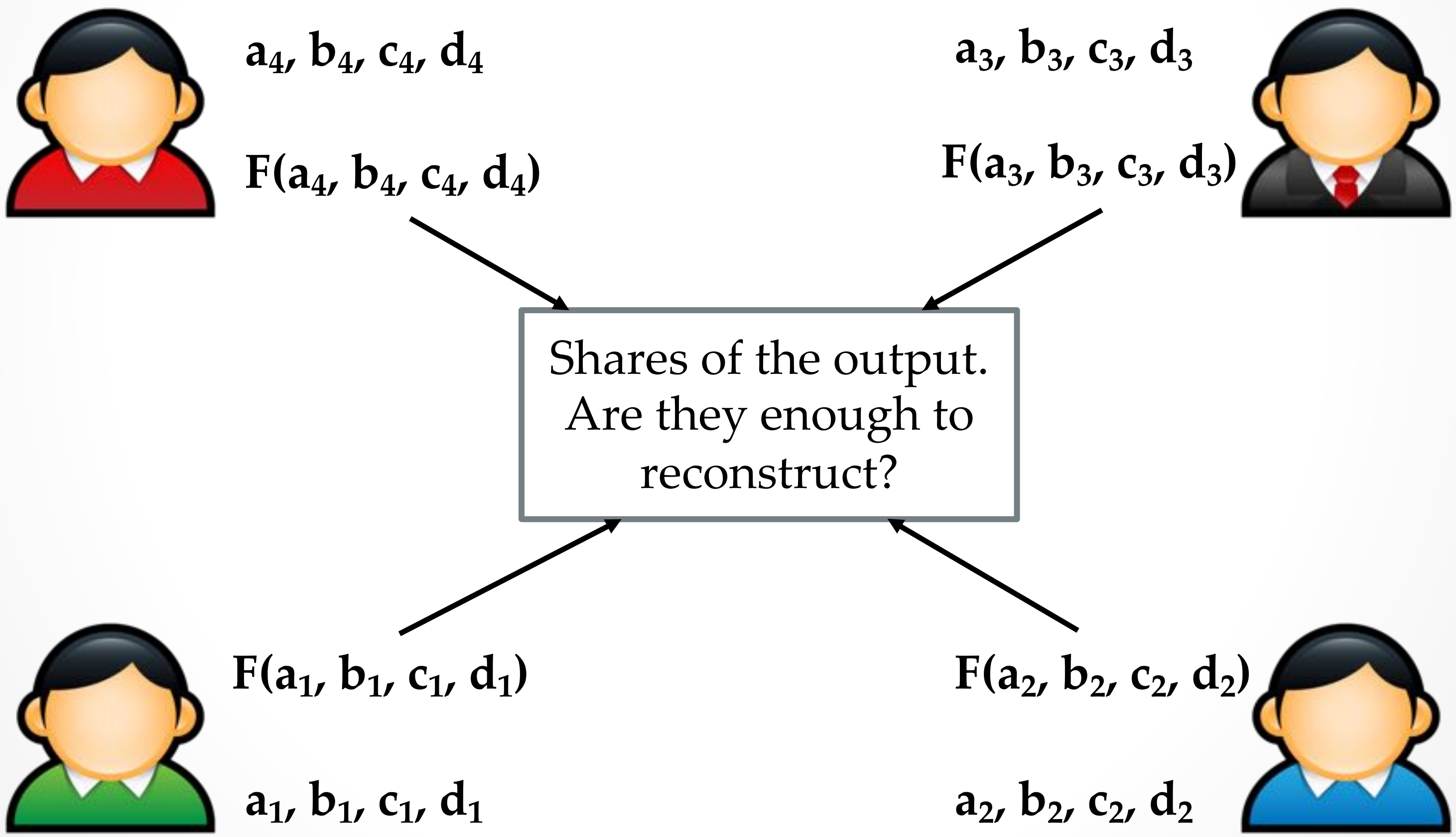
Multi-Party Computation



Multi-Party Computation



Multi-Party Computation



How Many Shares?

- If we allow **t corrupt parties**, we need polynomials of **degree t**
  - The secret can be reconstructed by at least t+1 parties

- *Addition gates:*
  - Output shares lie on a polynomial of **degree t**
- *Multiplication gates:*
  - Output shares lie on a polynomial of **degree 2t**
  - We need **at least 2t+1 parties** to reconstruct the secret

- *Does the degree increase exponentially with the multiplicative depth of the circuit?*
  - **"Luckily" not – we can reduce the degree after each multiplication gate**
  - For any n>2t+1 and points $\alpha_1, \ldots, \alpha_n$, there exists an n×n matrix A such that for all polynomial **p(x) of degree 2t**

    A $(p(\alpha_1), \ldots, p(\alpha_n)) = (p'(\alpha_1), \ldots, p'(\alpha_n))$ where
    - **p'(x) is of degree t**
    - **p'(x)=p(x)**

How to Reduce the Degree?

$$\begin{bmatrix} p(\alpha_1) \\ \vdots \\ \vdots \\ p(\alpha_n) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{n-1} \\ & & \vdots & \\ 1 & \alpha_n & \cdots & \alpha_n^{n-1} \end{bmatrix}}_{\text{Vandermond matrix}} \times \begin{bmatrix} p_0 \\ \vdots \\ p_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{n-1} \\ & & \vdots & \\ 1 & \alpha_n & \cdots & \alpha_n^{n-1} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 0 & 1 & \cdots & & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots 1 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & & 0 \end{bmatrix} \times \begin{bmatrix} p_0 \\ \vdots \\ p_{2t} \\ 0 \\ \vdots \\ 0 \end{bmatrix} =$$

$$= \underbrace{\begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{n-1} \\ & & \vdots & \\ 1 & \alpha_n & \cdots & \alpha_n^{n-1} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 0 & 1 & \cdots & & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots 1 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & & 0 \end{bmatrix} \times \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{n-1} \\ & & \vdots & \\ 1 & \alpha_n & \cdots & \alpha_n^{n-1} \end{bmatrix}^{-1}}_{\mathbf{A}} \times \begin{bmatrix} p'(\alpha_1) \\ \vdots \\ \vdots \\ p'(\alpha_n) \end{bmatrix}$$

Multi-Party Computation

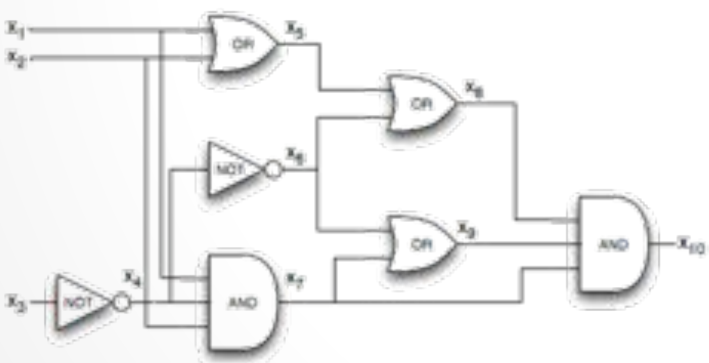
- BGW security guarantees for n party computation
  - Semi-honest model: up to **n/2 corrupt parties**
  - Malicious model: up to **n/3 corrupt parties**
  - Information theoretic/perfect security

- Security against **arbitrary number (up to n-1)** of corrupt parties
  - Computational security (relies on computational assumptions)
  - Constructions:
    - GMW Protocol [GMW87] (Goldreich-Micali-Wigderson)
    - Preprocessing model: SPDZ [DPSZ12], SPDZ-BMR [LPSY15], BMR-SHE [LSS16], Mascot [KOS16]

Computation Over Circuits

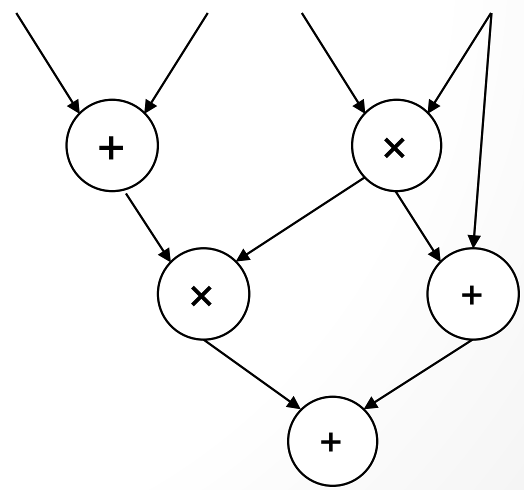
Boolean Circuits

- **Yao Gabled Circuits**



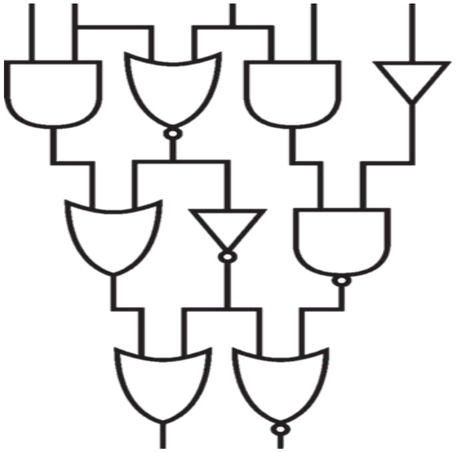
Arithmetic Circuits

- **BGW Construction**
  - Ben-Or, Goldwasser, Widgerson



How Efficient is Computation with Circuits?

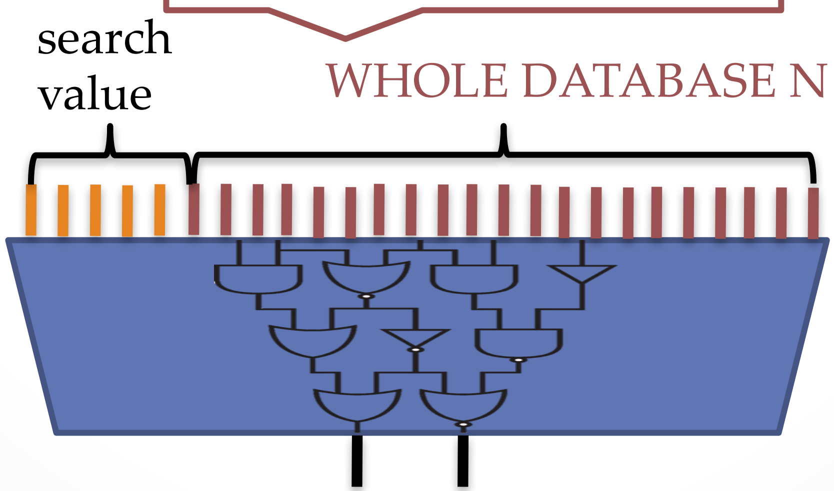
- **Linear in the circuit size!**



Binary Search

Query x

Binary search has logarithmic complexity in plaintext computation



Yes, if you do not touch some part of the data, you reveal it is not used in the computation

Is MPC inherently linear?

No, in the amortized setting

Random Access Machine (RAM)

LOAD #5
STORE 15
LOAD #0
EQUAL 15
JUMP #6
HALT
ADD #1
JUMP #3

RAM Computation

While

state ≠ stop

Computation :

- update **state**
- compute next memory instruction

Access memory

- fetch next program instruction
- read/write data

Binary Search RAM

While

item not found and non-empty search range

Computation

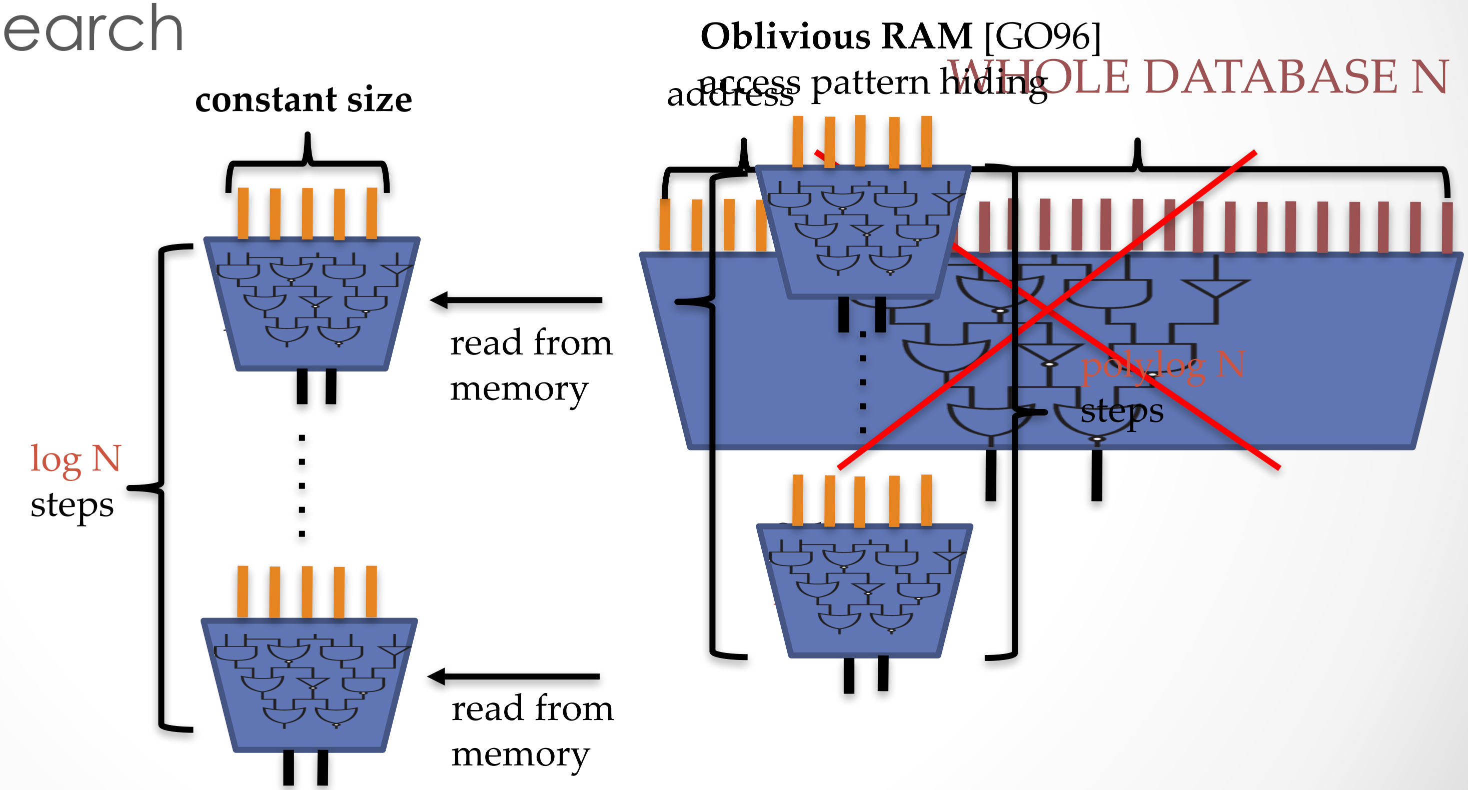
check for match, compute next address p to access

Access memory

Read data from address p

Secure Computation for RAMs

- Binary Search



ORAM Properties

- **Access pattern hiding**
  - The physical accesses in memory for any two query sequences of equal length are indistinguishable

Example:
read 1, read 1, read 1
write 3, read 1, read 5

- **Efficiency** - random access (logarithmic)
  - Note: trivial solution is to read the whole memory at each access. Very expensive!

Subquery$_1$
Subquery$_2$
…
Subquery$_n$

Logarithmic number of subqueries for memory part of constant size

- **ORAM Initialization** – one time linear computation
- Constructions:
  - **Hierarchical-based:** [GO96], [KLO12]
  - **Tree-based:** Tree ORAM [SCSL11], Path ORAM [SDSCFRYD13], Circuit ORAM [WCS15]

MPC for RAMs enables secure computation with sublinear complexity in the amortized setting!

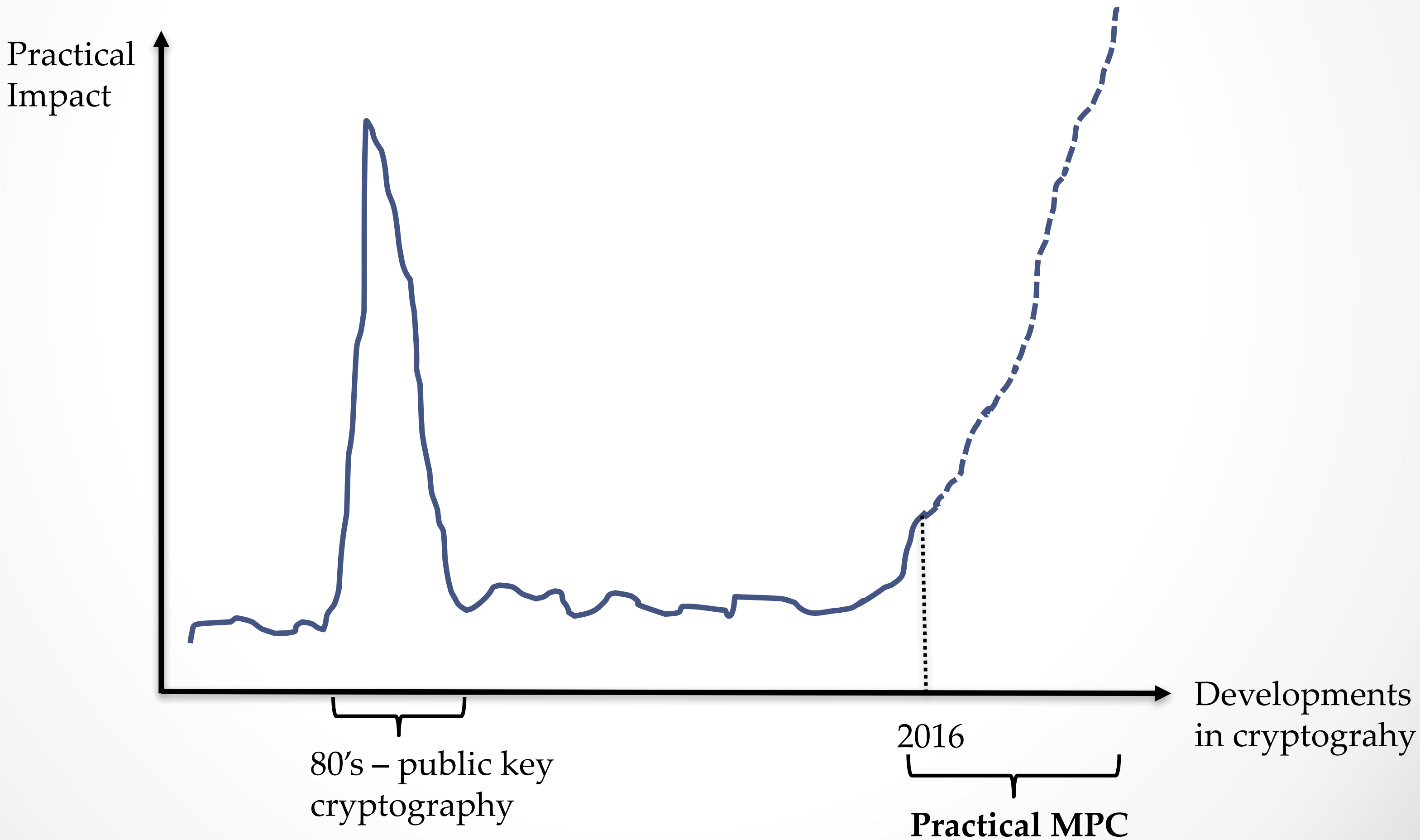
What Does and Does Not MPC Guarantee?

Guarantee: The computation does not reveal more than what the output reveals.

Secure Computation for Approximations:
An approximation may reveal more than the exact output of the computation. One needs to argue that such leakage does not exist. [FIMNSW06]

No Guarantee:
How much does the output reveal.

The Impact of Cryptography




THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
GRAZIE
BOLZİN
MERCI
BIYAN