

# Protecting Patient Privacy in Genomic Analysis

David Wu

Stanford University

based on joint works with:

Gill Bejerano, Bonnie Berger, Johannes A. Birgmeier,  
Dan Boneh, Hyunghoon Cho, and Karthik A. Jagadeesh

# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

*What gene causes a specific (rare) disease?*



Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes

# Rare Disease Diagnosis

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]

	0	1	0	0	0
<i>A1BG</i>	1	1	1	0	1
	⋮	⋮	⋮	⋮	⋮
<i>ZZZ3</i>	0	0	1	0	0

Each patient has a vector  $v$  where  $v_i = 1$  if patient has a rare variant in gene  $i$



Patients with Kabuki Syndrome

**Goal:** Identify gene with most variants across all patients

Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes

# Rare Disease Diagnosis

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]

Gene	A1BG	0	1	0	0	0
		1	1	1	0	1
		⋮	⋮	⋮	⋮	⋮
	ZZZ3	0	0	1	0	0



Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes

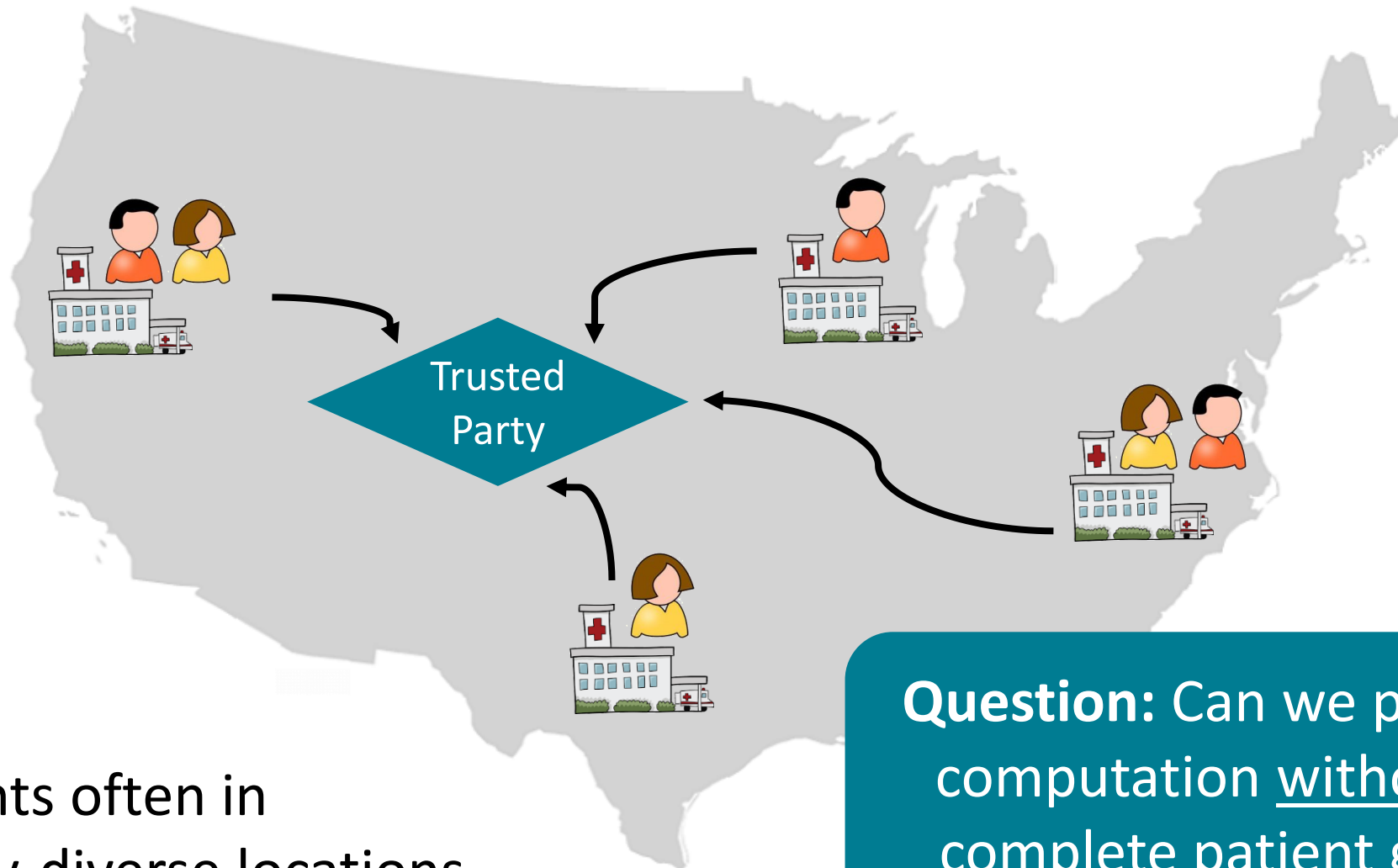
Each patient has a vector  $v$  where  $v_i = 1$  if patient has a rare variant in gene  $i$

**Goal:** Identify gene with most variants across all patients

Works well for Mendelian (monogenic) diseases (estimated to affect  $\approx 10\%$  of individuals)

# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

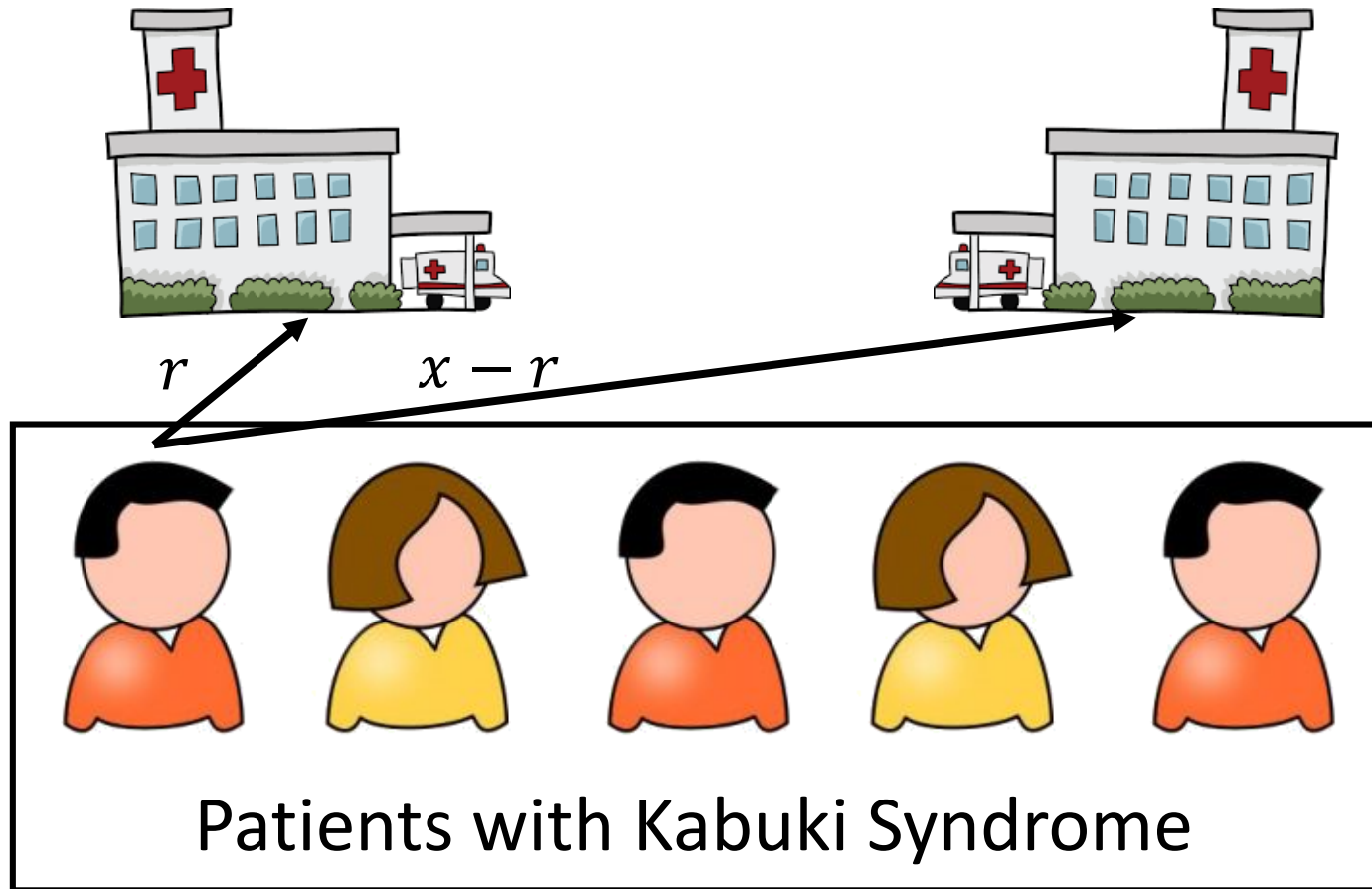


Patients often in  
geographically-diverse locations

**Question:** Can we perform this  
computation without seeing  
complete patient genomes?

# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

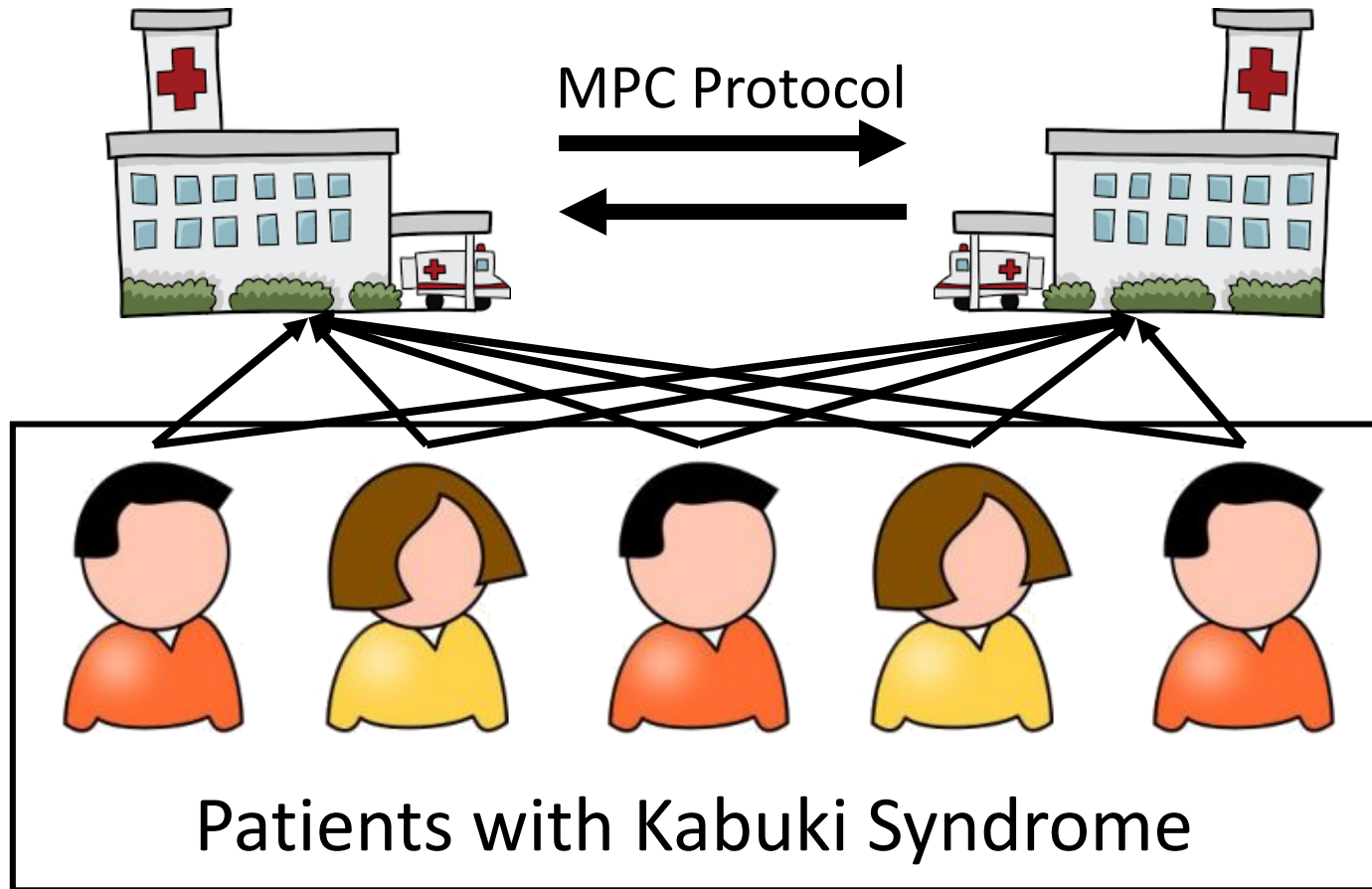


Patients “secret share”  
their data with two  
non-colluding hospitals

Each patient has a list of 200-400  
rare variants over  $\approx 20,000$  genes

# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*



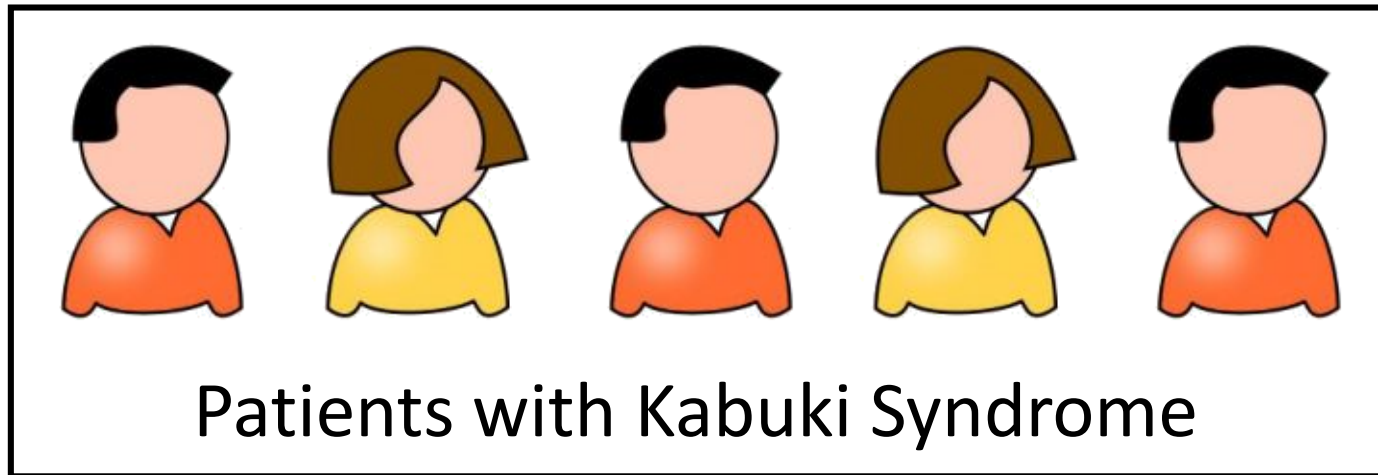
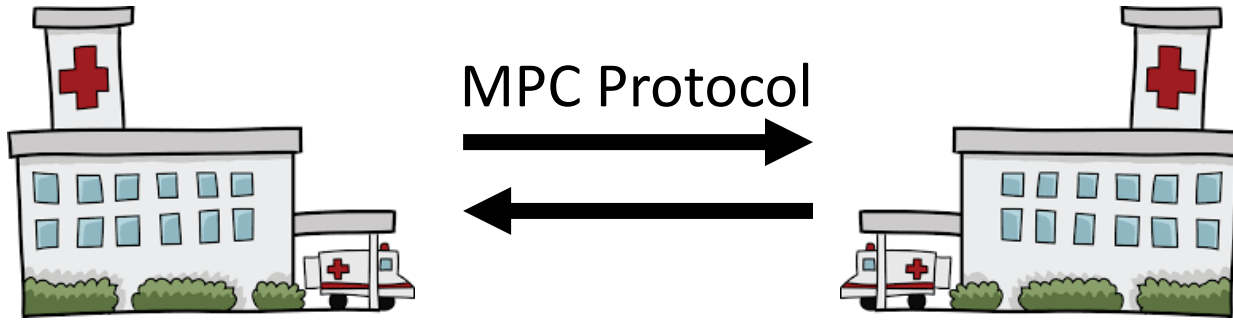
Hospitals run a multiparty computation (MPC) protocol on pooled inputs

Patients “secret share” their data with two non-colluding hospitals

Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes

# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*



Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes



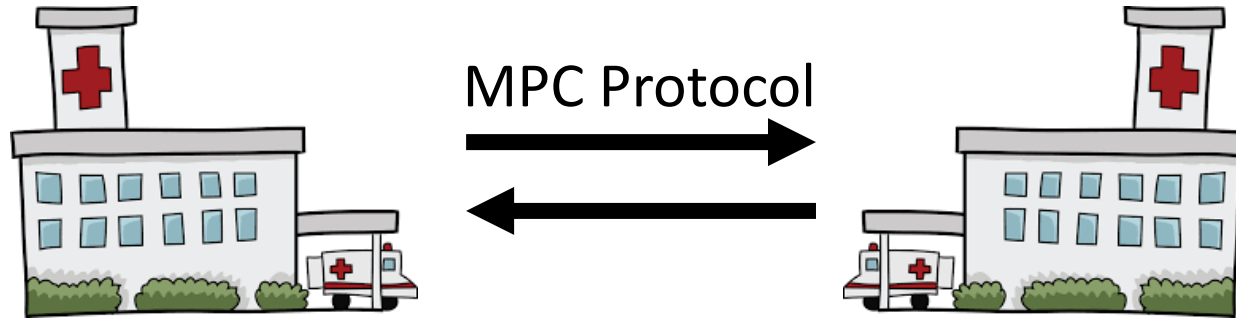
Top variants (sorted):  
**KMT2D, COL6A1, FLNB**

Known cause of disease



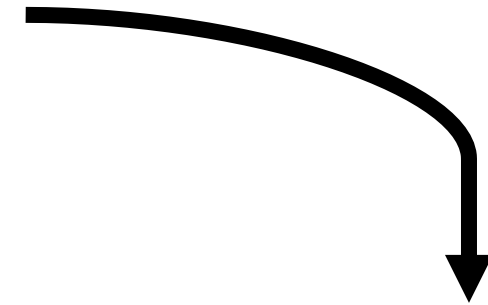
# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*



Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over  $\approx 20,000$  genes



Top variants (sorted):  
**KMT2D, COL6A1, FLNB**

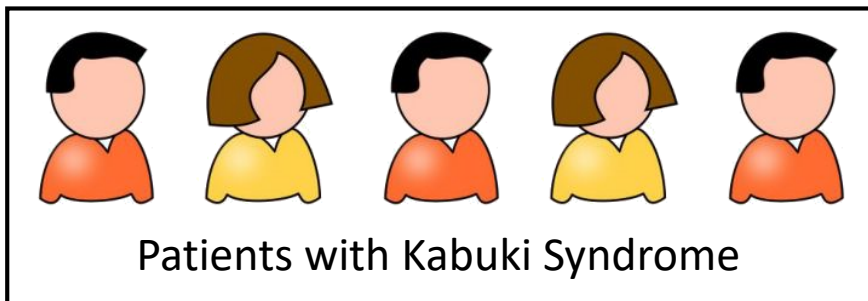
Other variants that the patients possess are kept hidden

# Rare Disease Diagnosis

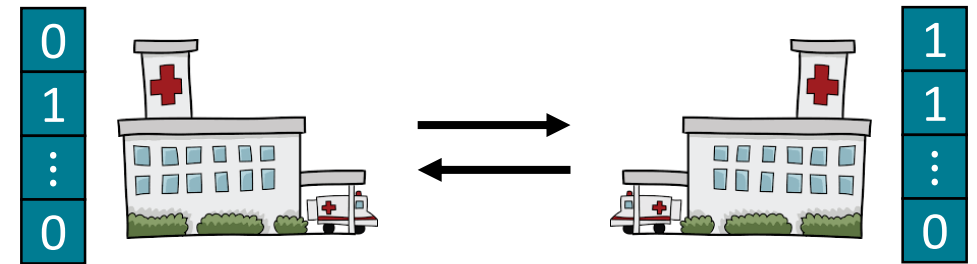
*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

General techniques apply to many different scenarios for diagnosing Mendelian diseases

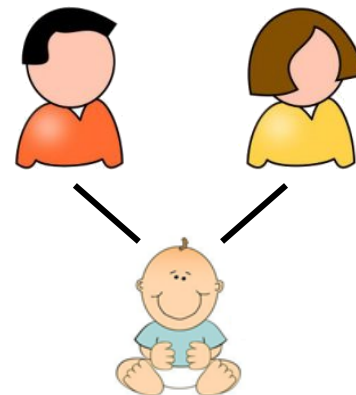
	0	1	0	0	0
<i>A1BG</i>	1	1	1	0	1
	⋮	⋮	⋮	⋮	⋮
<i>ZZZ3</i>	0	0	1	0	0



Identify causal gene for a rare disease given a small patient cohort



Identify patients with the same rare functional mutation at two different hospitals

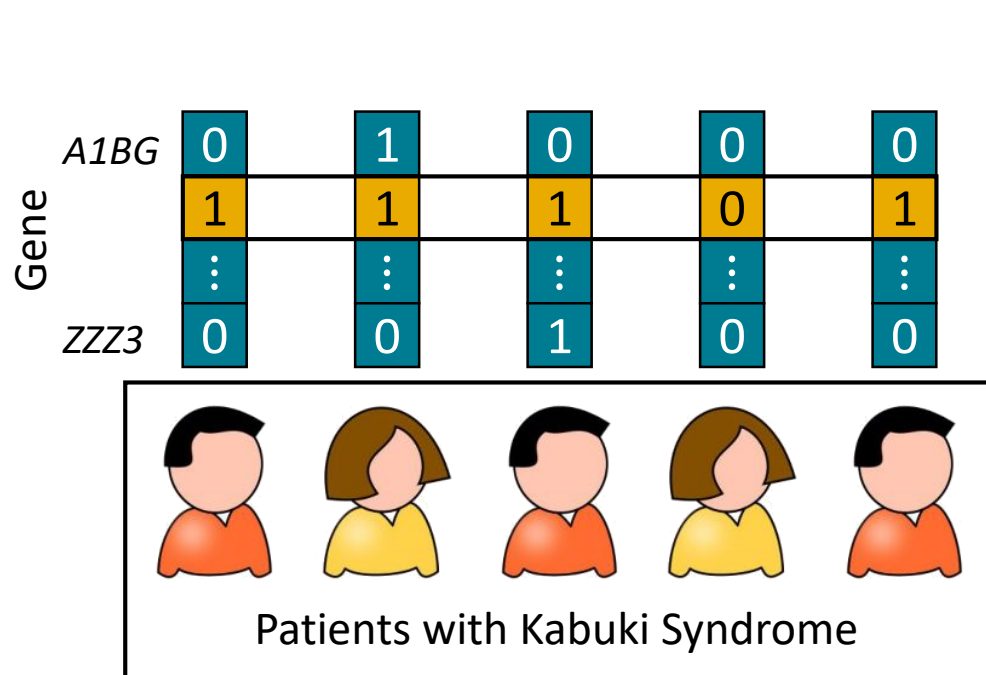


Identify rare functional variants that are present in the child but in neither of the parents

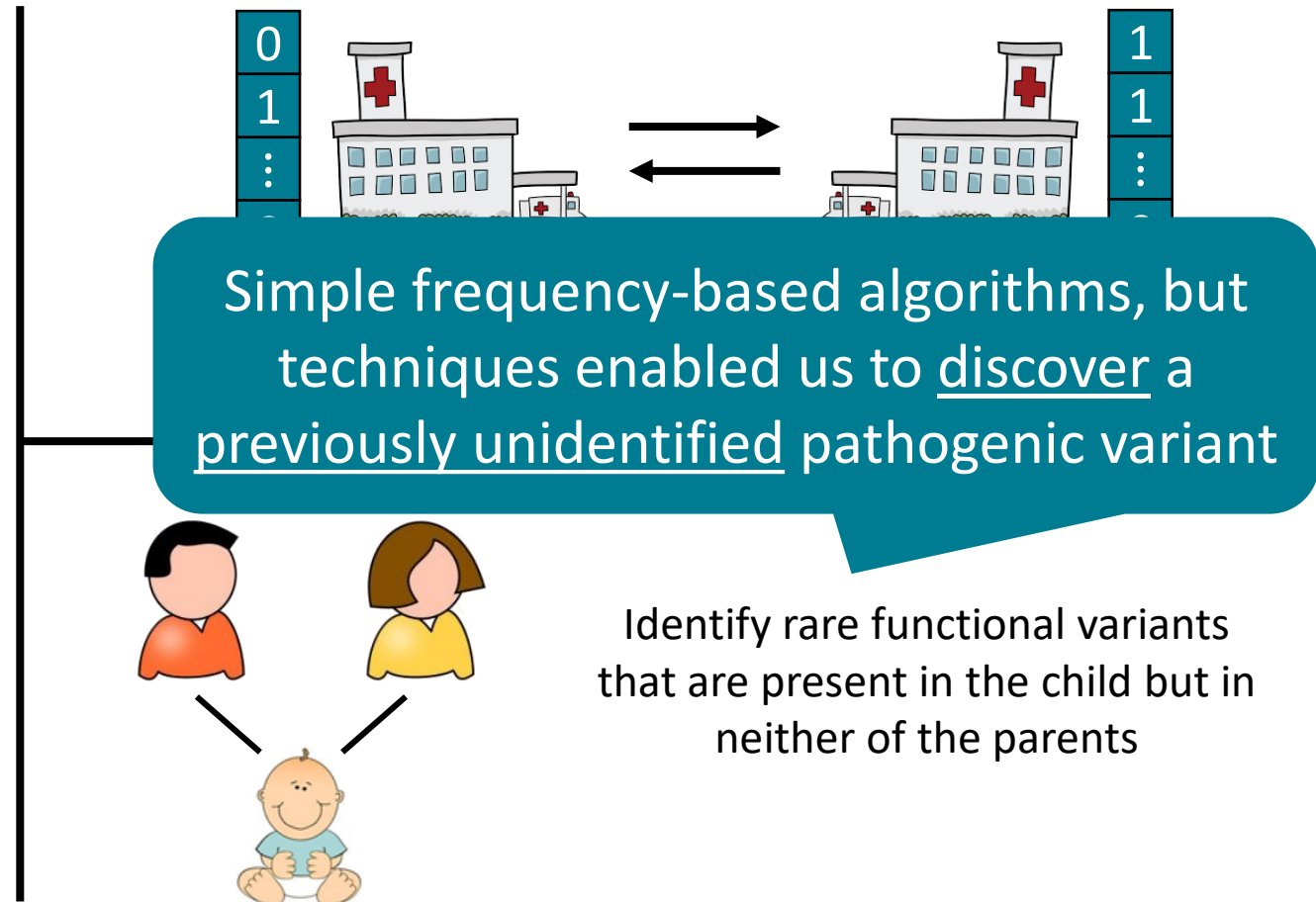
# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

General techniques apply to many different scenarios for diagnosing Mendelian diseases



Identify causal gene for a rare disease given a small patient cohort

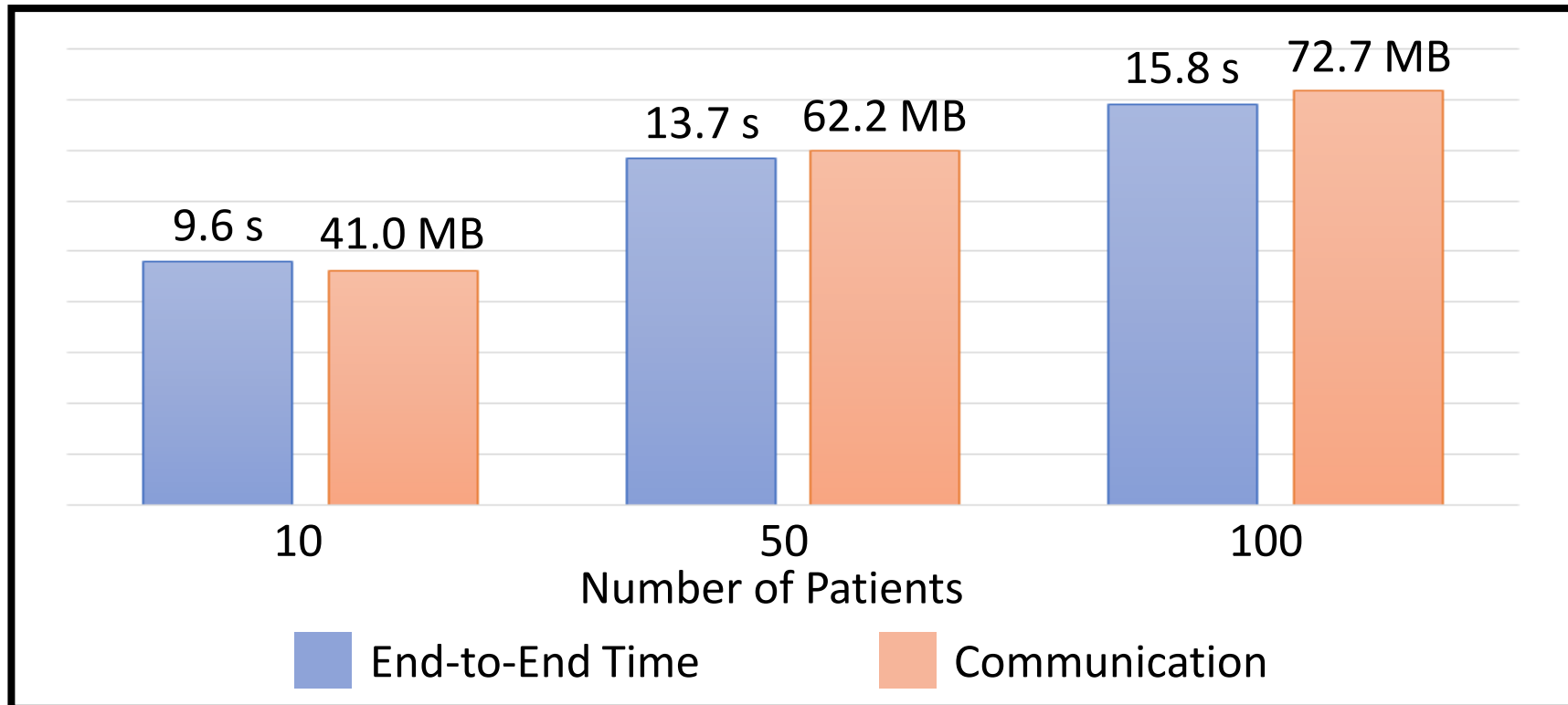


# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

Experimental benchmarks for identifying causal gene in small disease cohort

- Simulated two non-colluding entities with 1 server on East Coast and 1 on West Coast

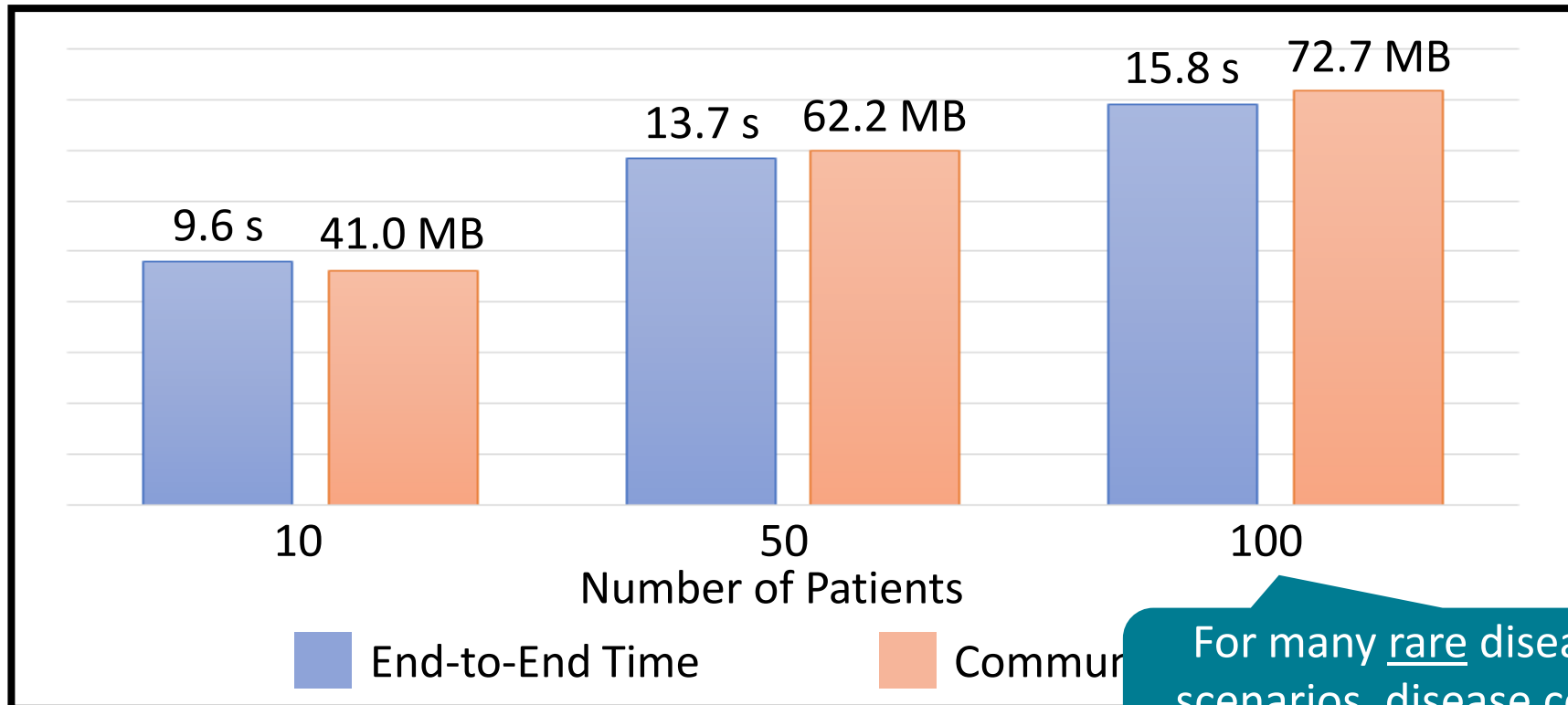


# Rare Disease Diagnosis

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*

Experimental benchmarks for identifying causal gene in small disease cohort

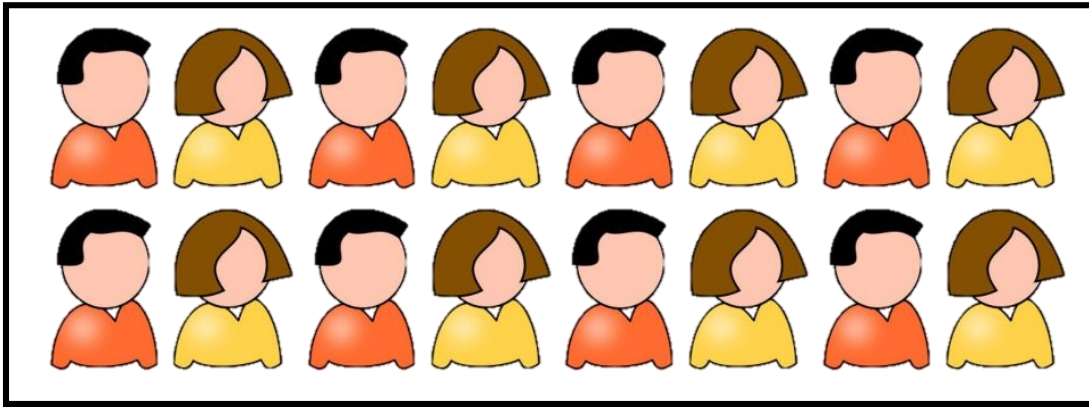
- Simulated two non-colluding entities with 1 server on East Coast and 1 on West Coast



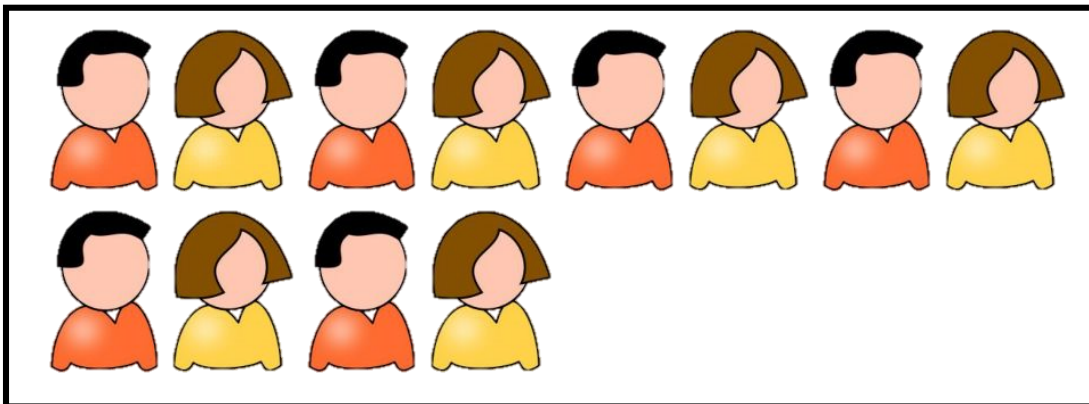
For many rare disease diagnosis scenarios, disease cohort size can be very small (e.g., 5-10 patients)

# What About More Complex Diseases?

*Cho-W-Berger [Nature Biotechnology 2018]*



Control group (healthy)



Case group (affected)

Genome-wide association studies (GWAS):

- Identify genetic variants most correlated with a particular disease (or particular phenotype)
- Oftentimes, focused on identifying complex interactions between many variants

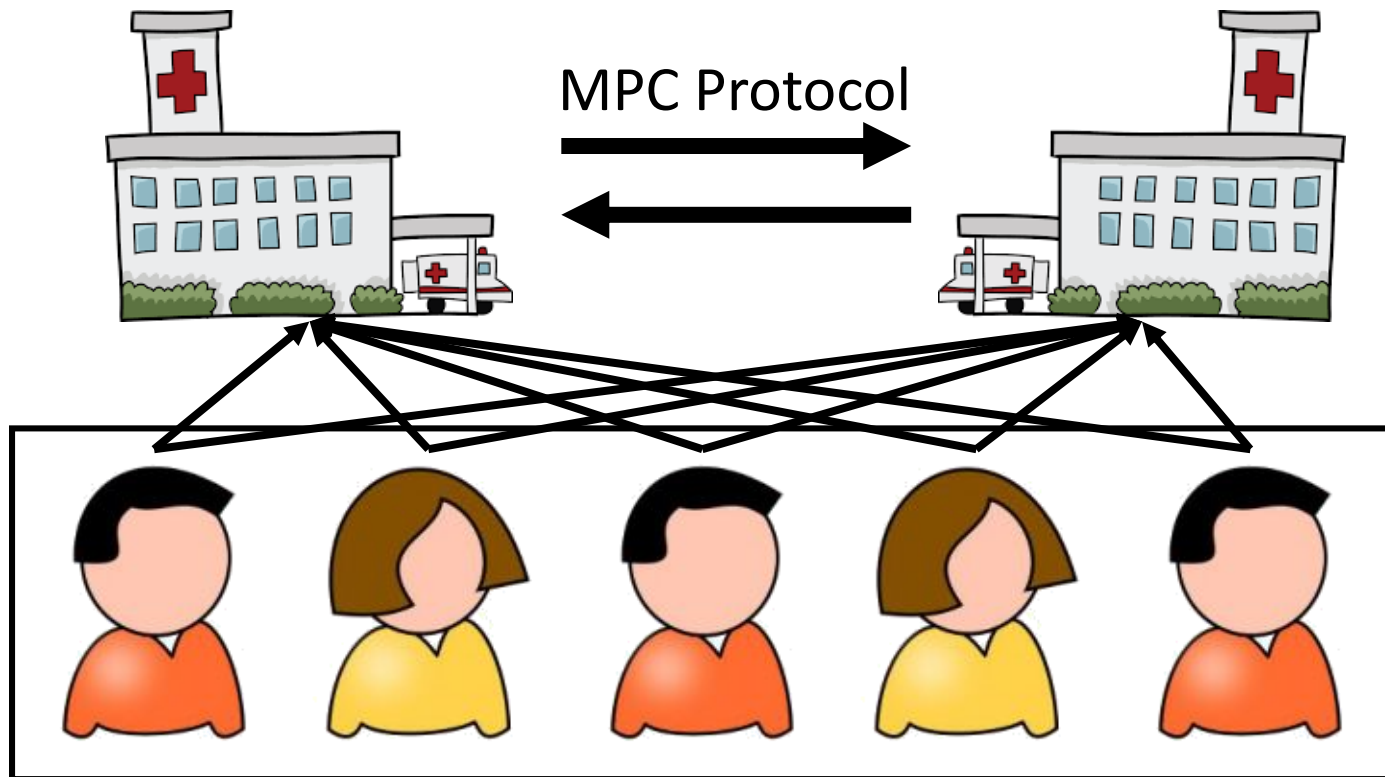
Challenges:

- Need large case/control groups to identify weak genetic signals (e.g., 10K-1M individuals)
- Oftentimes need to correct for population-level effects

# What About More Complex Diseases?

*Cho-W-Berger [Nature Biotechnology 2018]*

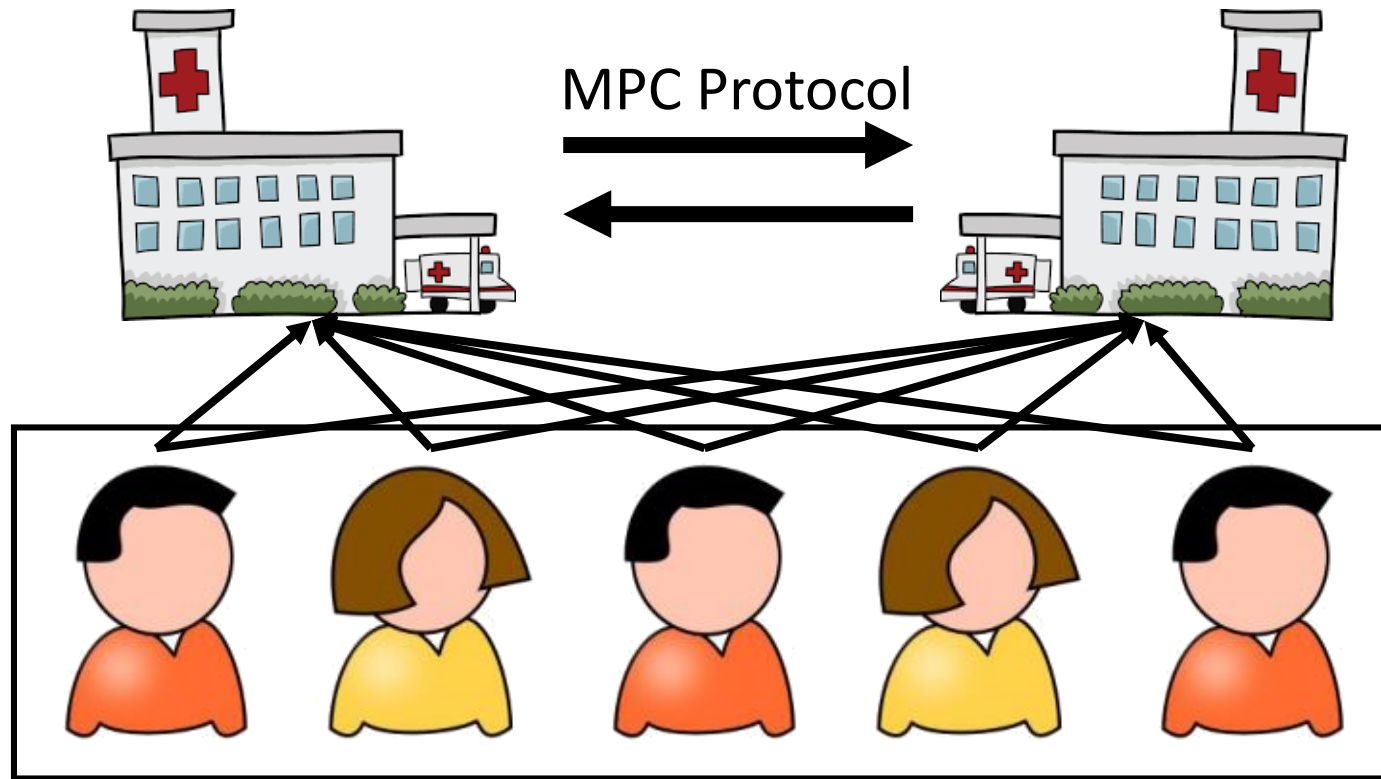
Similar model for genome outsourcing:



**This work:** first end-to-end GWAS protocol (with population correction)

- For 25K individuals, computation completes in about 3 days: feasible for performing large-scale scientific studies

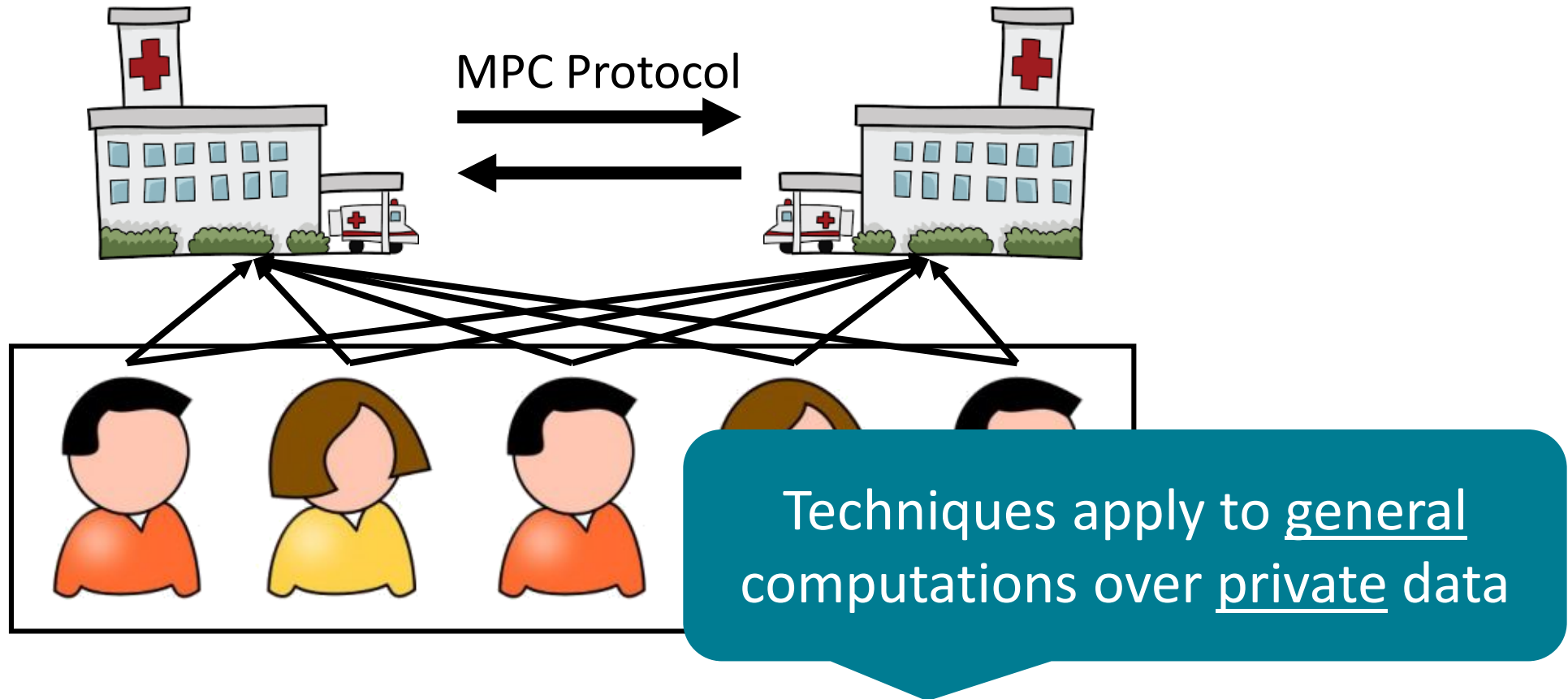
# Secure Genome Computation



Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes



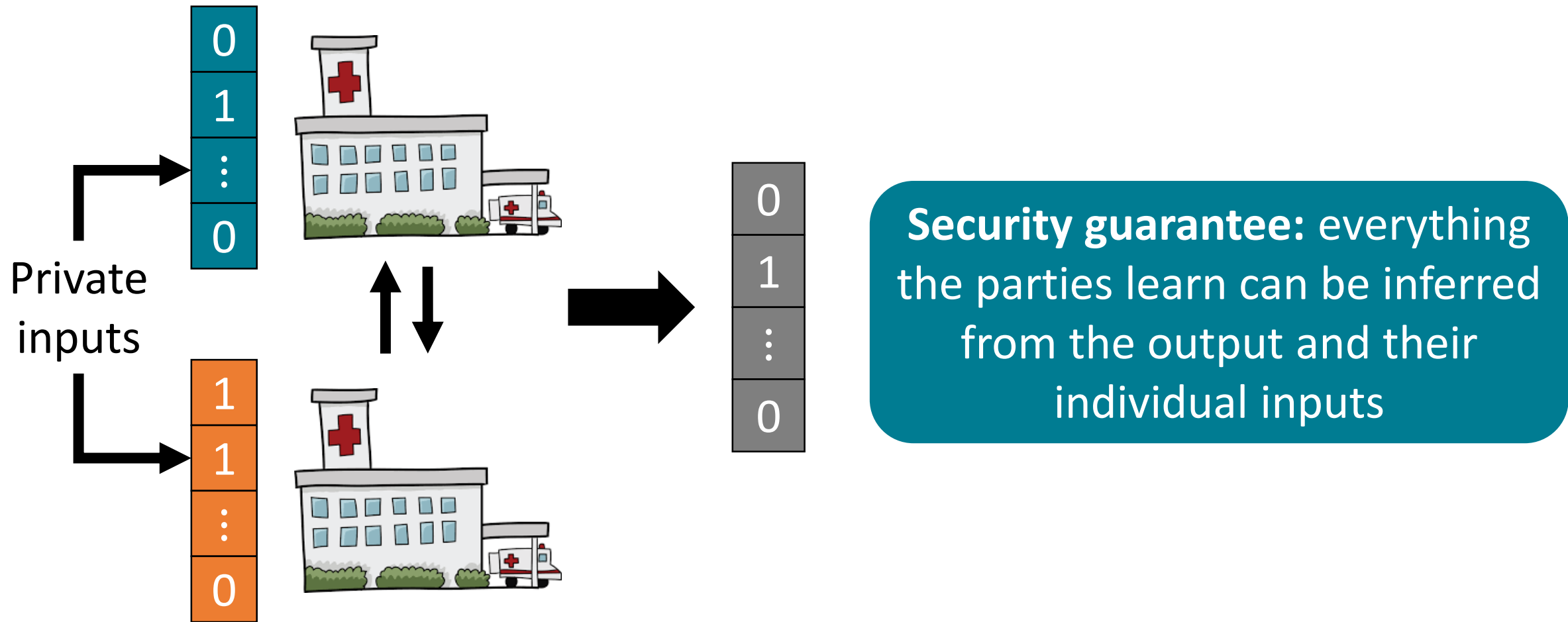
# Secure Genome Computation



Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes

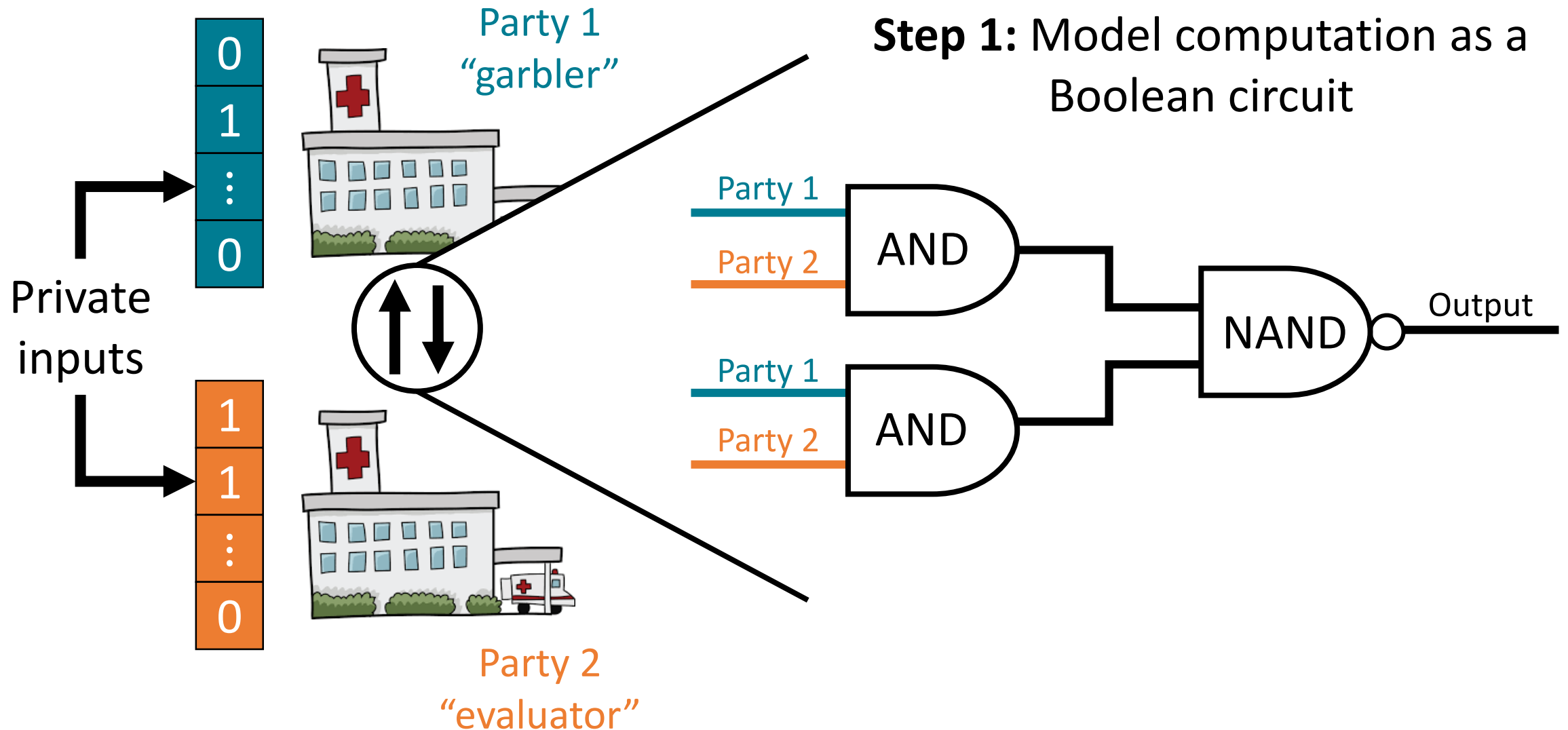
# Yao's Protocol for Two-Party Computation

# Yao's Protocol for Two-Party Computation [Yao82]



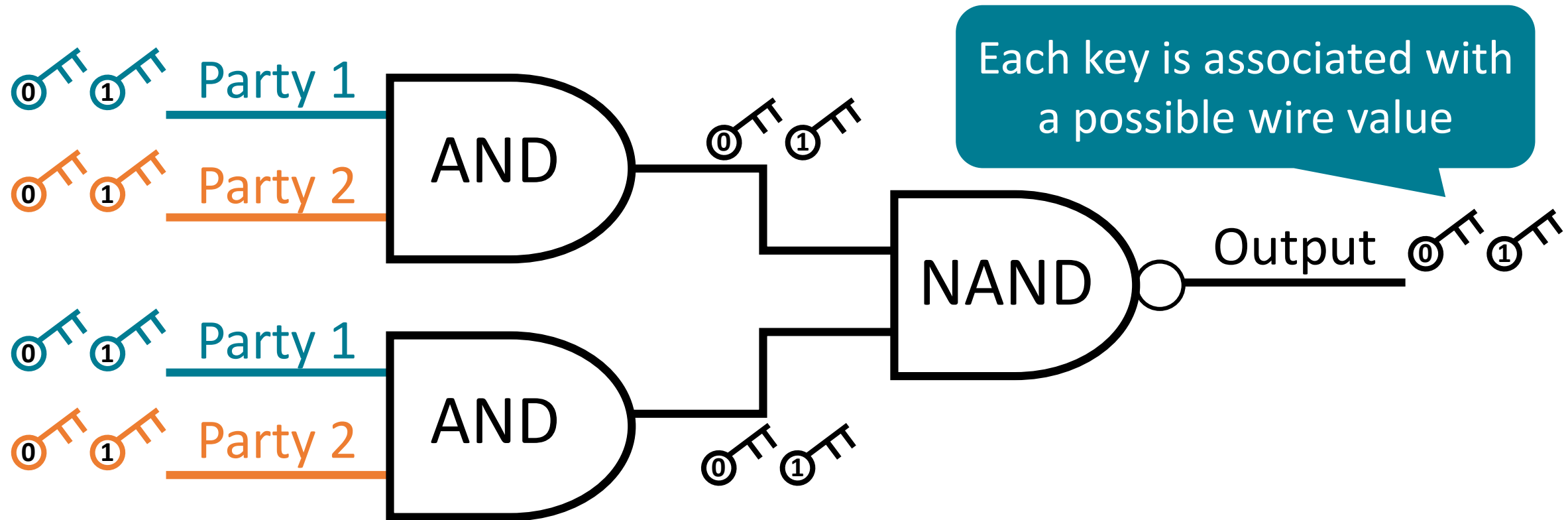
Classic protocol for two-party computation

# Yao's Protocol for Two-Party Computation [Yao82]



# Yao's Protocol for Two-Party Computation [Yao82]

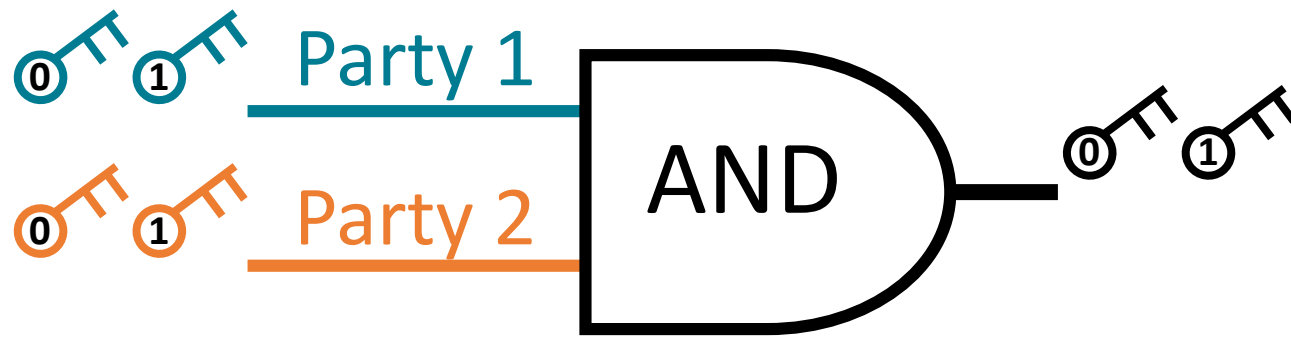
**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



Garbler chooses two different encryption keys for every wire in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



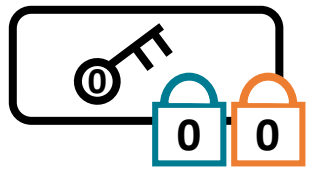
**Idea:** Encrypt the output key (for the output wire) with the two input keys (for the input wires)

	Inputs		Output
	Party 1	Party 2	
	0		0
	0		0
	1		0
	1		1

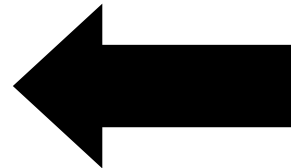
Garbler constructs a garbled truth table for each gate in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



$$\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$$

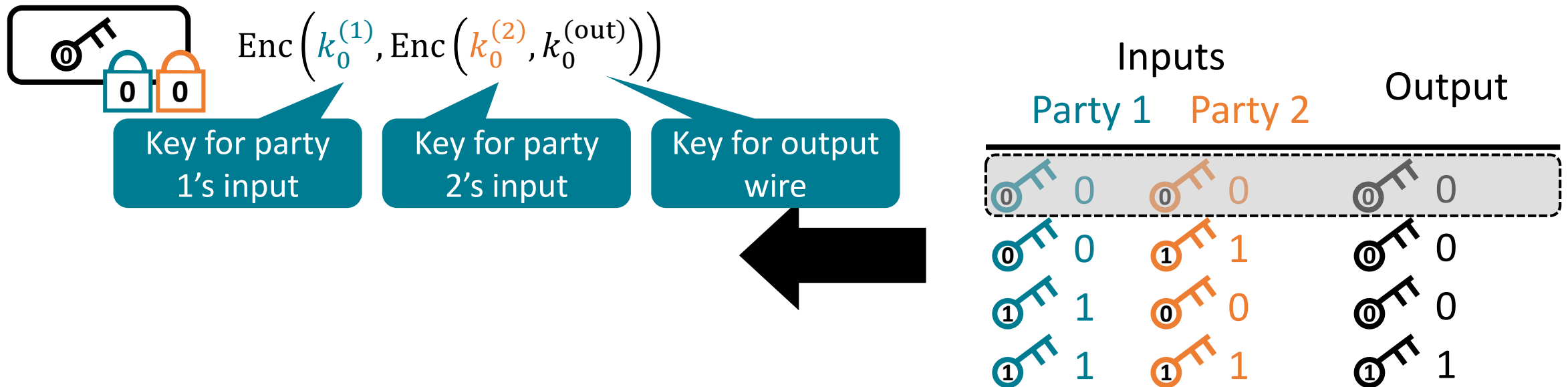


Inputs			Output
Party 1	Party 2		
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Garbler constructs a garbled truth table for each gate in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

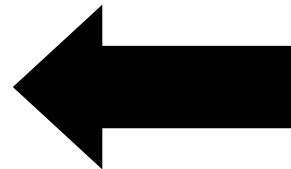
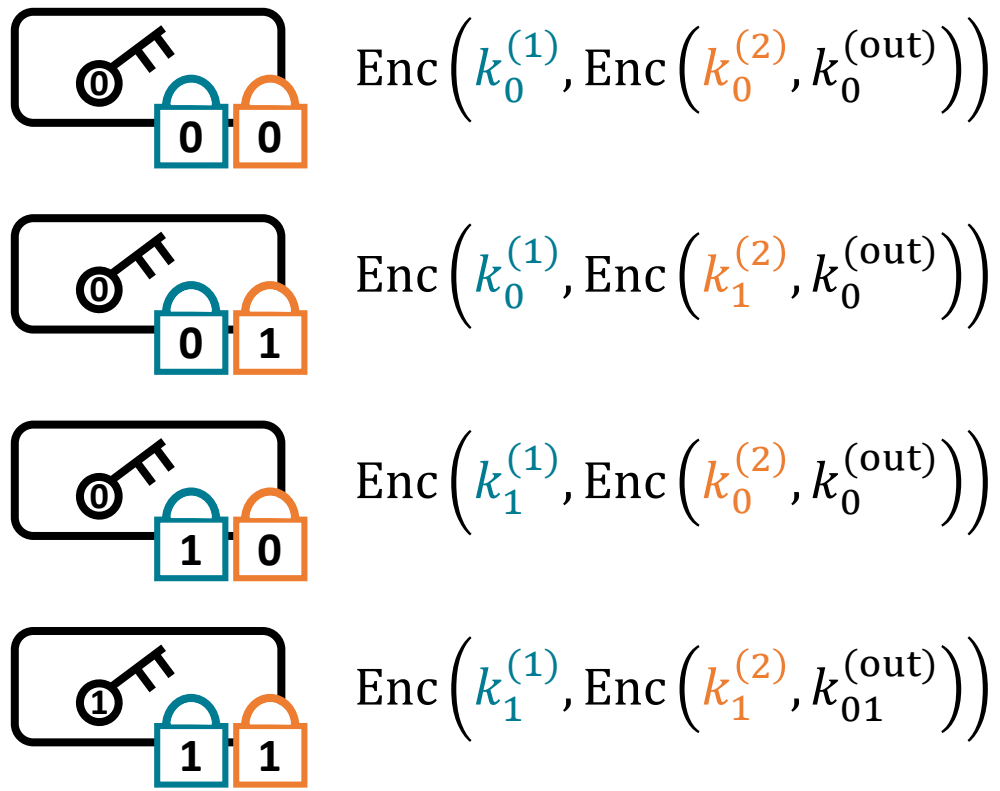


Garbler constructs a garbled truth table for each gate in the circuit



# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

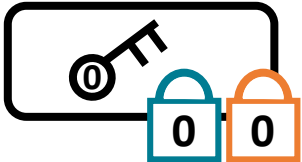


Inputs		Output	
Party 1	Party 2		
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

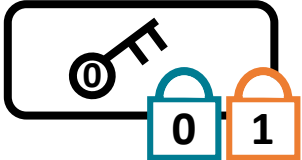
Garbler constructs a garbled truth table for each gate in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]

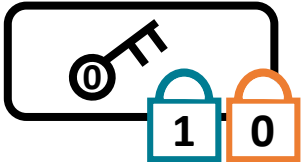
**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



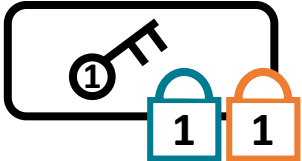
$\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$



$\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_1^{(2)}, k_0^{(\text{out})} \right) \right)$



$\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$




$\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_1^{(2)}, k_{01}^{(\text{out})} \right) \right)$

Garbled truth table  
randomly permuted

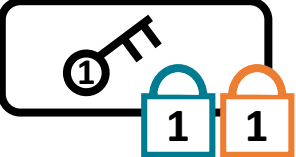
Garbler constructs a garbled truth table for each gate in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]


**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



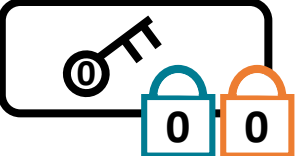
$\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$



$\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_1^{(2)}, k_1^{(\text{out})} \right) \right)$



$\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_1^{(2)}, k_0^{(\text{out})} \right) \right)$



$\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$

Garbled truth table  
randomly permuted

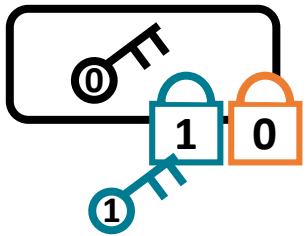
**Invariant:** Given just a single key for each input wire, evaluator can learn a single key for the output wire



Garbler constructs a garbled truth table for each gate in the circuit

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



$$\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right)$$

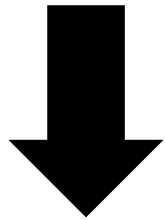
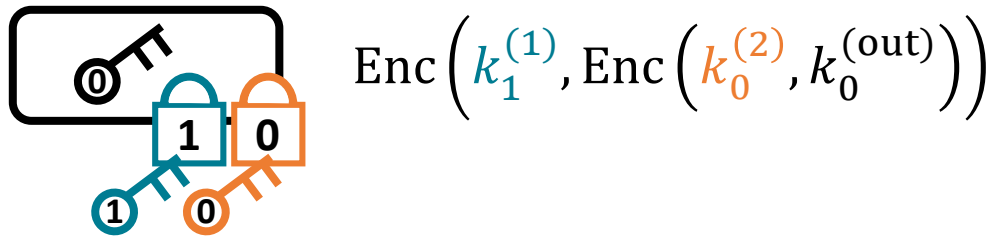
Garbled truth table  
randomly permuted


**Invariant:** Given just a single key for each input wire, evaluator can learn a single key for the output wire

$$k_1^{(1)} \quad \text{key } k_0^{(2)}$$

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



  $k_0^{(\text{out})}$

$k_0^{(\text{out})}$  is just a symmetric key – does not reveal what the output bit is

Garbled truth table  
randomly permuted

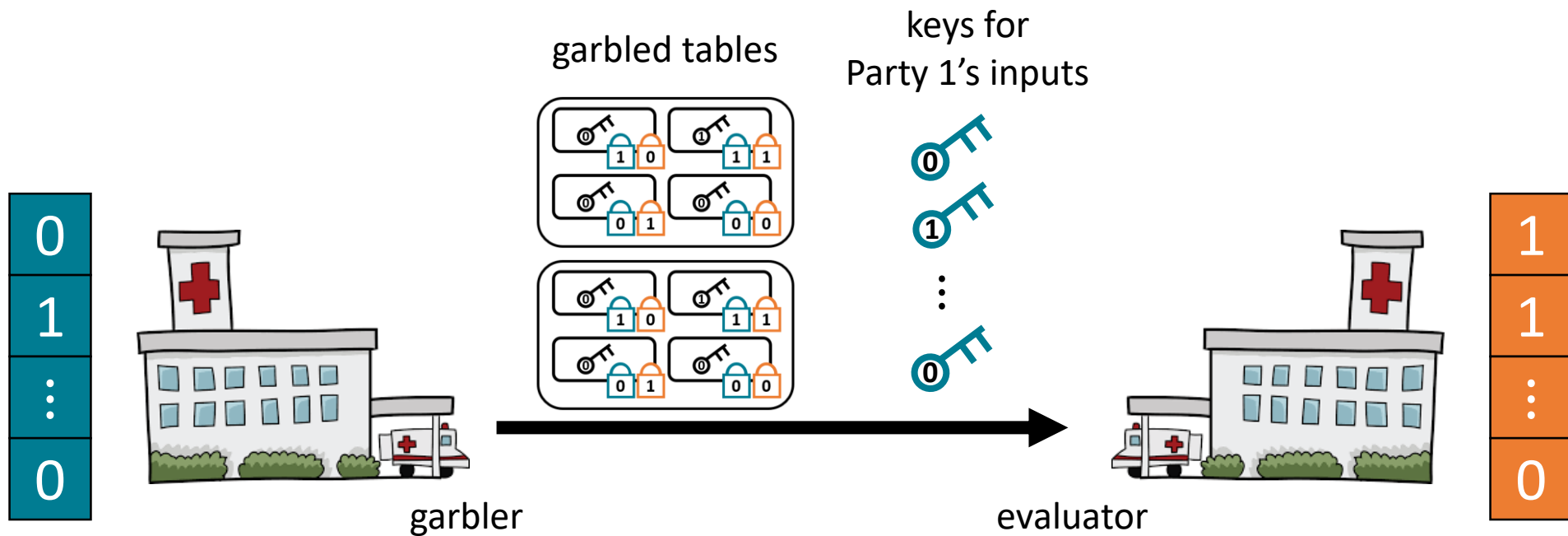
**Invariant:** Given just a single key for each input wire, evaluator can learn a single key for the output wire

$k_1^{(1)}$

$k_0^{(2)}$

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)



**Question:** how does evaluator obtain keys for its input?

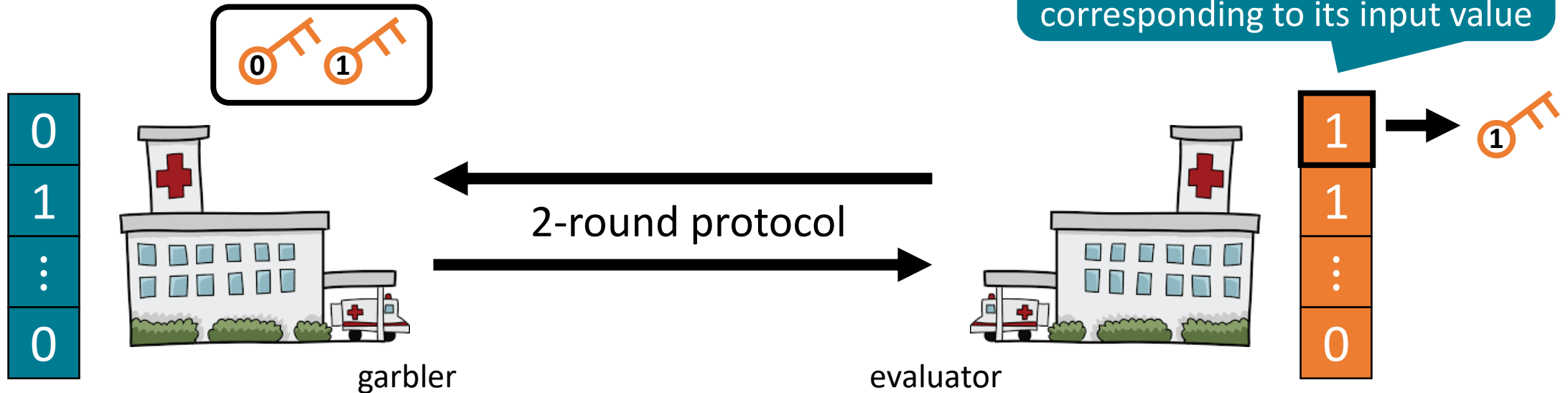
Garbler can send garbled truth tables and keys for its inputs

# Yao's Protocol for Two-Party Computation [Yao82]

**Step 3:** Evaluator uses “oblivious transfer” to obtain keys for its input

For each wire corresponding to evaluator's input, the garbler has two keys

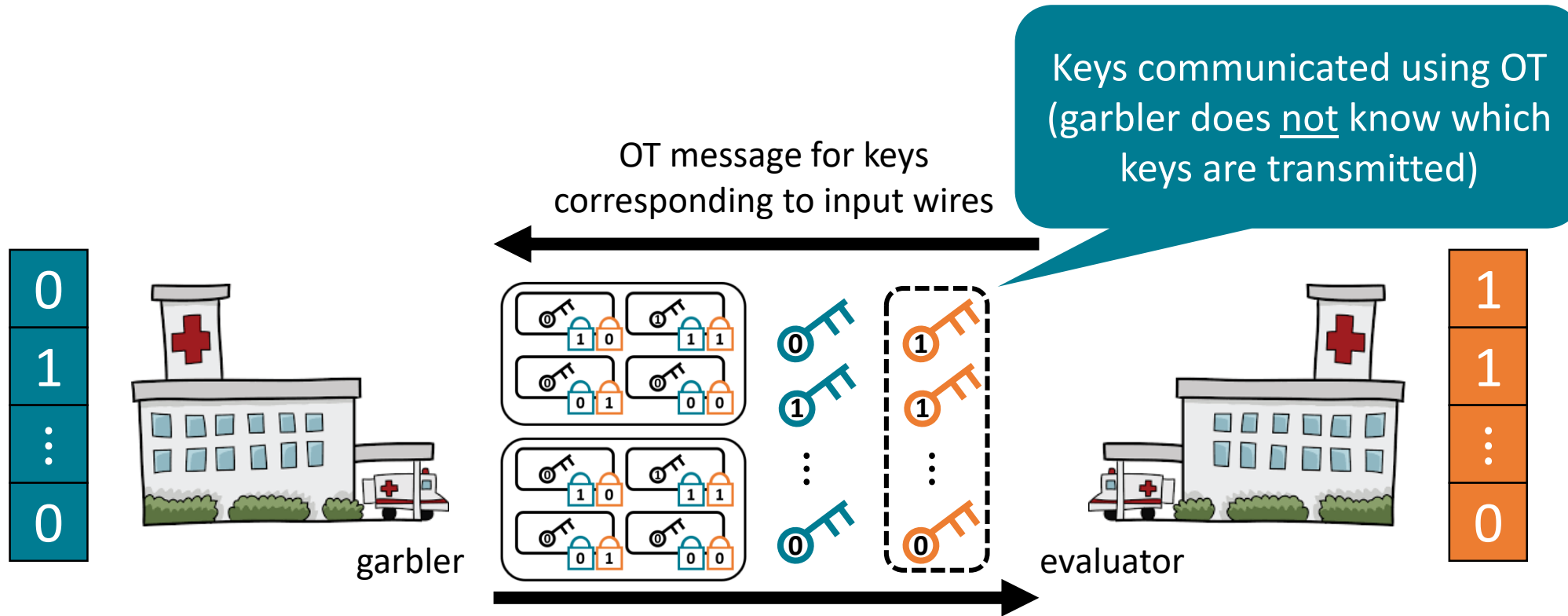
For each input wire, evaluator wants to obtain key corresponding to its input value



At the end of the oblivious transfer protocol, garbler learns nothing about which key evaluator obtains, and evaluator learns exactly one of the two keys

# Yao's Protocol for Two-Party Computation [Yao82]

Two-round protocol for secure two-party communication



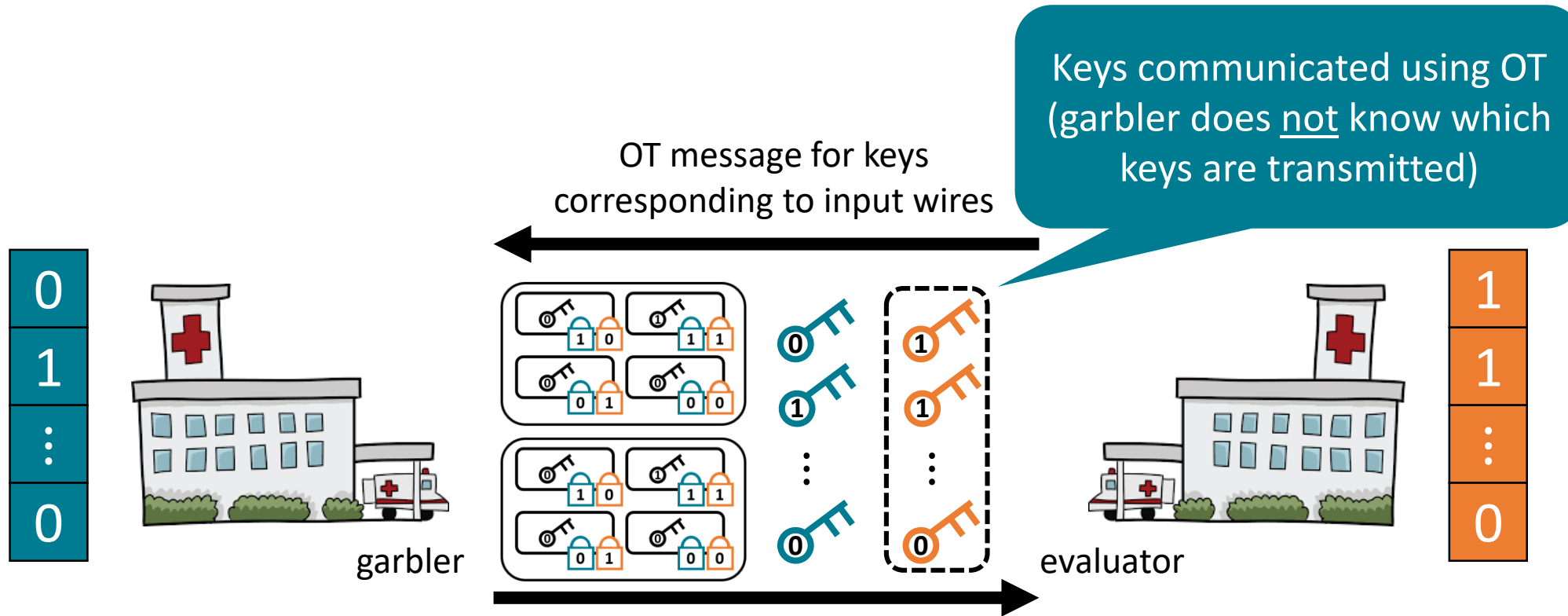
Many improvements are possible to achieve better performance

Evaluator uses keys to evaluate circuit gate-by-gate



# Yao's Protocol for Two-Party Computation [Yao82]

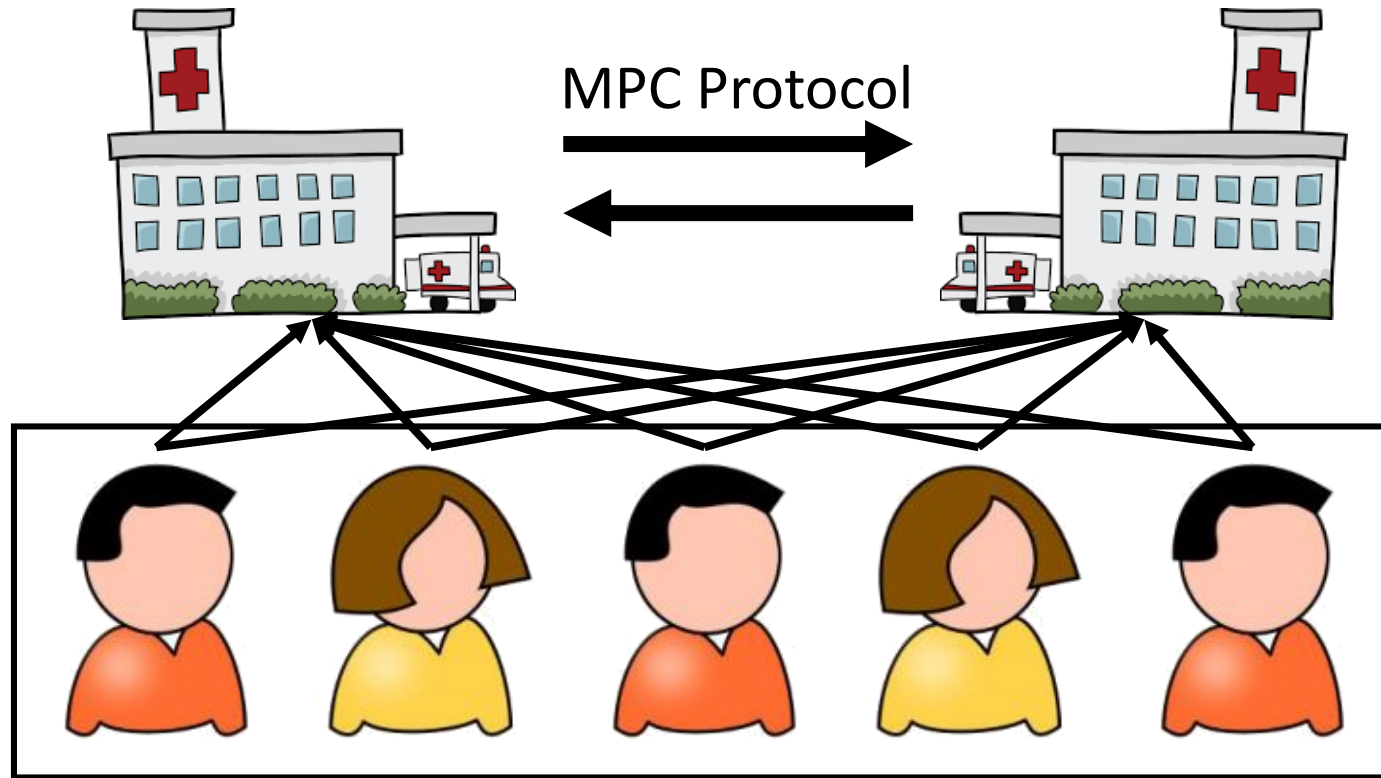
Two-round protocol for secure two-party communication



Many improvements are possible to achieve better performance

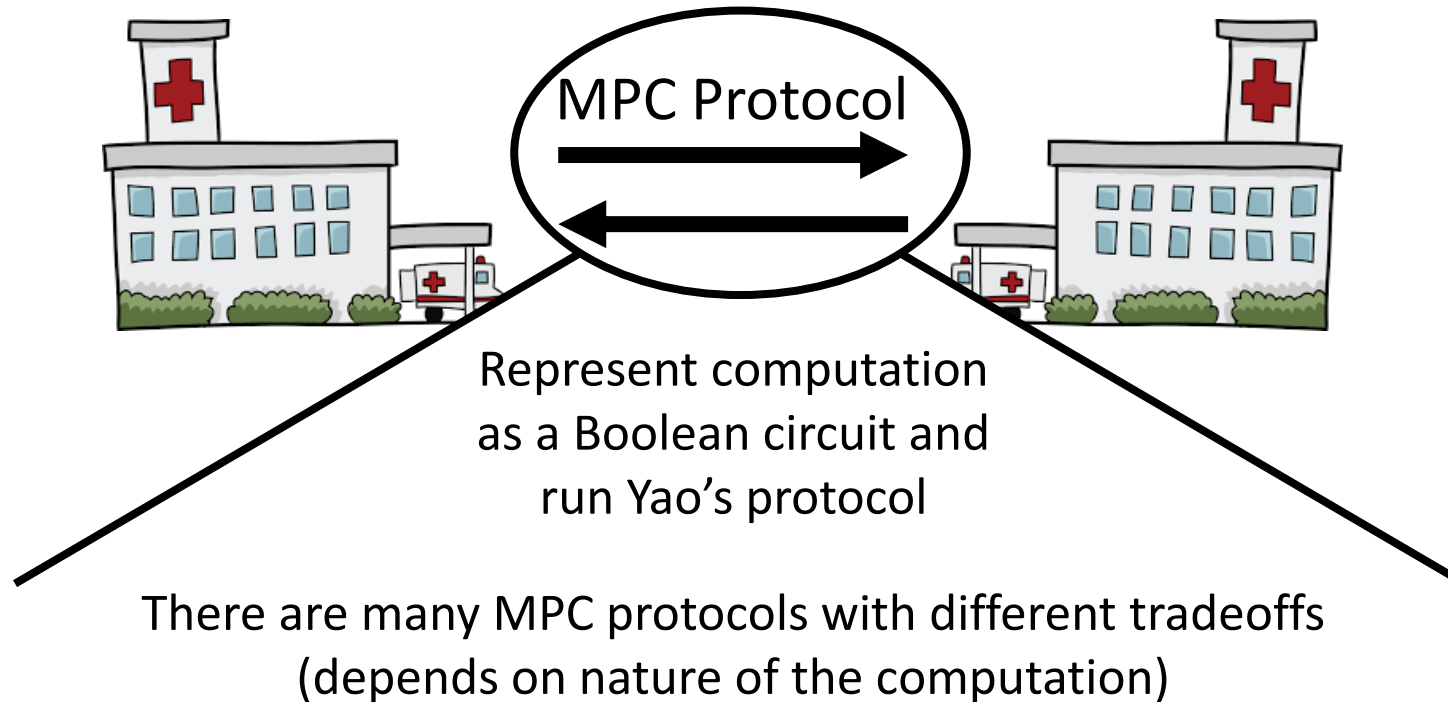
Protocol is very efficient; communication is the bottleneck

# Secure Genome Computation



Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes

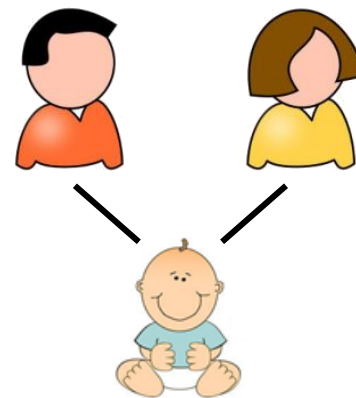
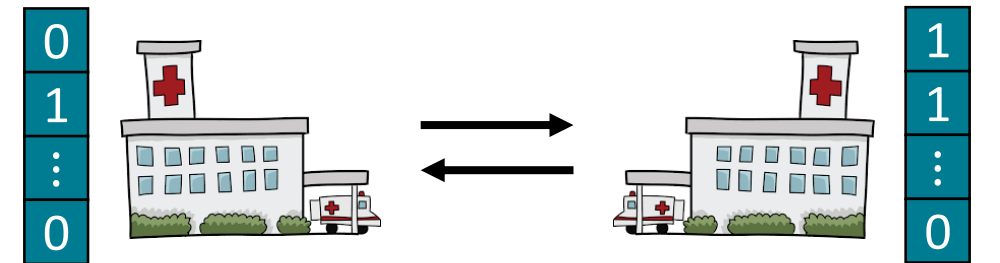
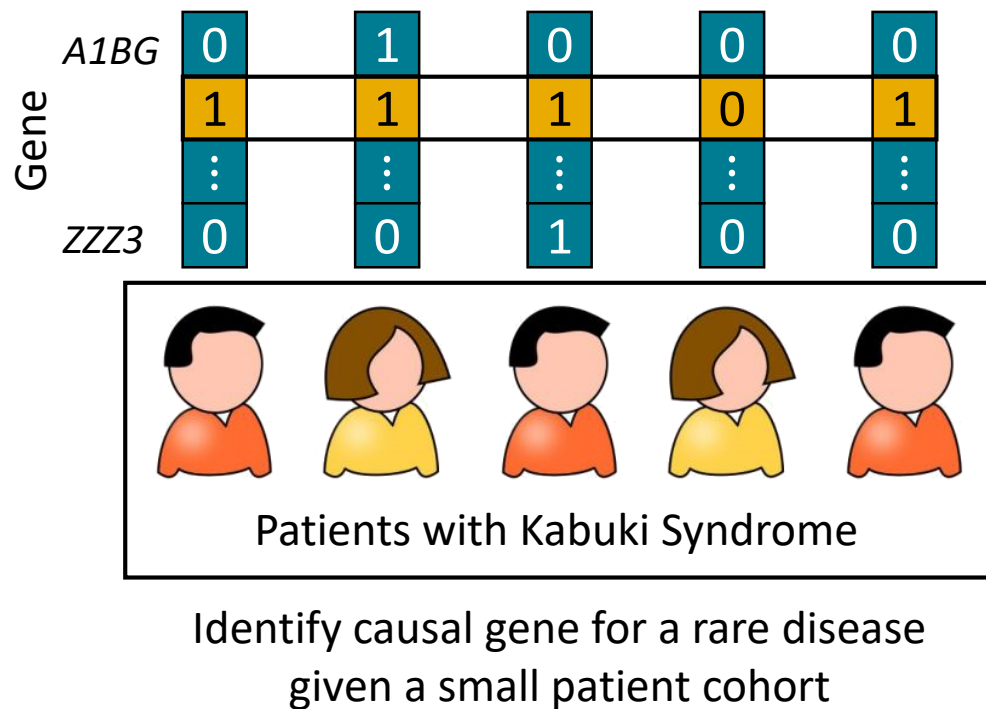
# Secure Genome Computation



Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes

# Secure Genome Computation

Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes



# Secure Genome Computation

Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes

## **Paper:**

<http://science.sciencemag.org/content/357/6352/692>

## **Sample Implementation:**

<https://github.com/dwu4/genome-privacy>

*Nature Biotechnology* paper on privacy-preserving GWAS forthcoming!

**Thank you!**