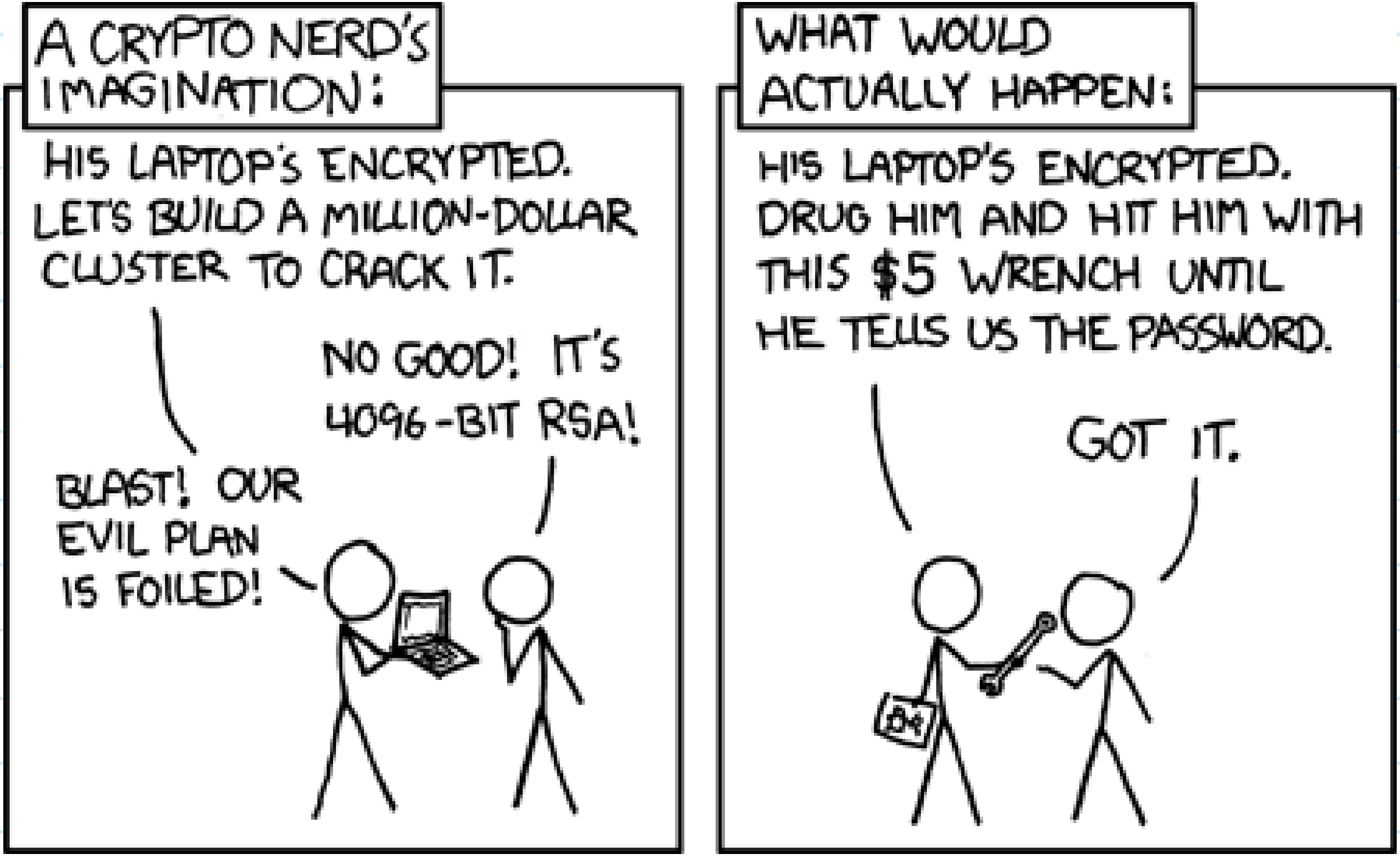
CS 65500
Advanced Cryptography

Lecture 3: Oblivious Transfer

Instructor: Aarushi Goel

Spring 2025


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

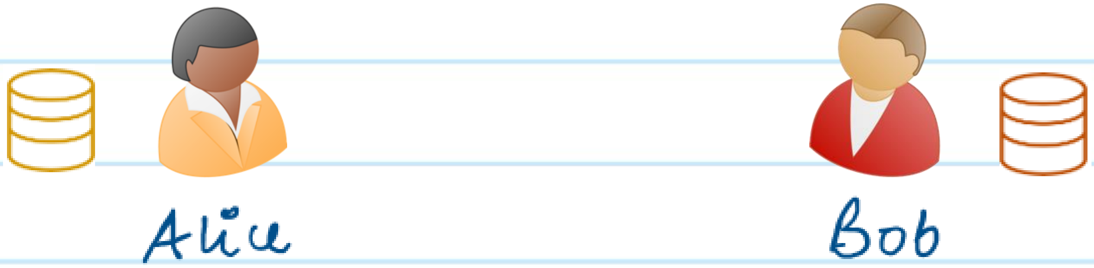
Agenda

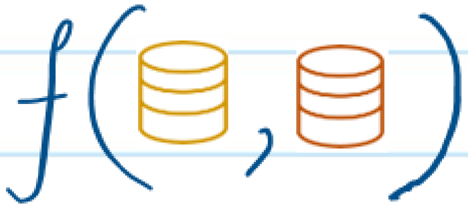
- Secure Two-Party Computation
  - Motivation
  - Definition
- Oblivious Transfer
  - Construction

Two-Party Computation: Setting

→ Alice and Bob have some data

→ They want to compute some function on their combined data

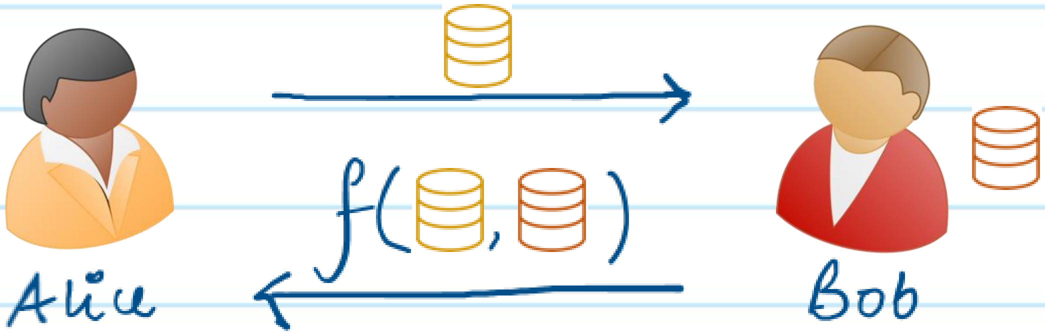




How can they do this computation?

Two-Party Computation : Setting

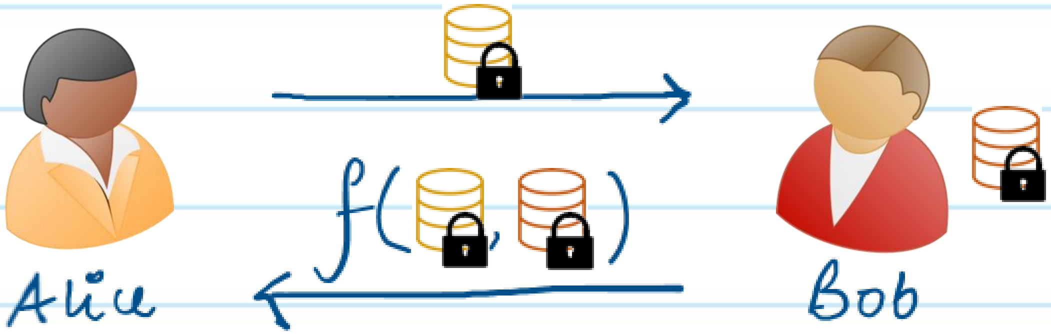
Option 1



- → Alice can send her data to Bob
- → Bob can then compute f using her & his own data.
- → Bob can then send the output to Alice.

Two-Party Computation : Setting

Option 1

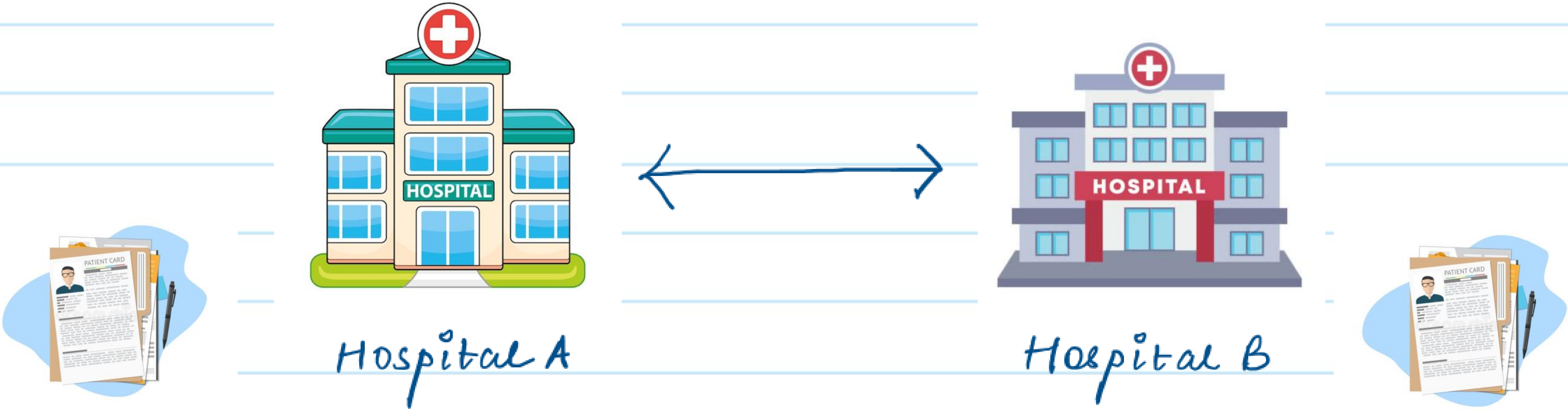


Problems with this approach

1. What if Alice's data contains sensitive information that she doesn't want to reveal to Bob?

2. Can Alice really trust Bob to compute f honestly?

Examples Of Problems Involving Sensitive Data

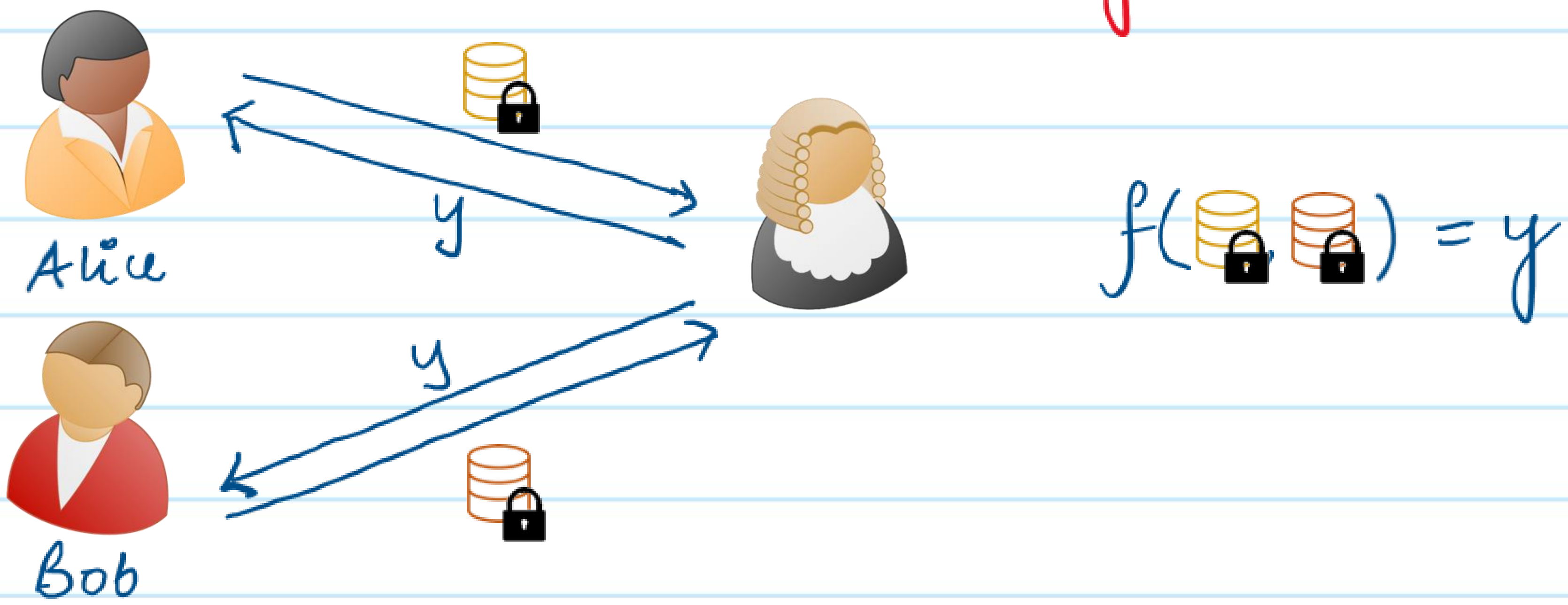


Two hospitals want to use their collective patient records for medical research.

Examples of Problems Involving Sensitive Data

- → Google has a list of leaked passwords.
- → A user wants to check if her password is in the list.

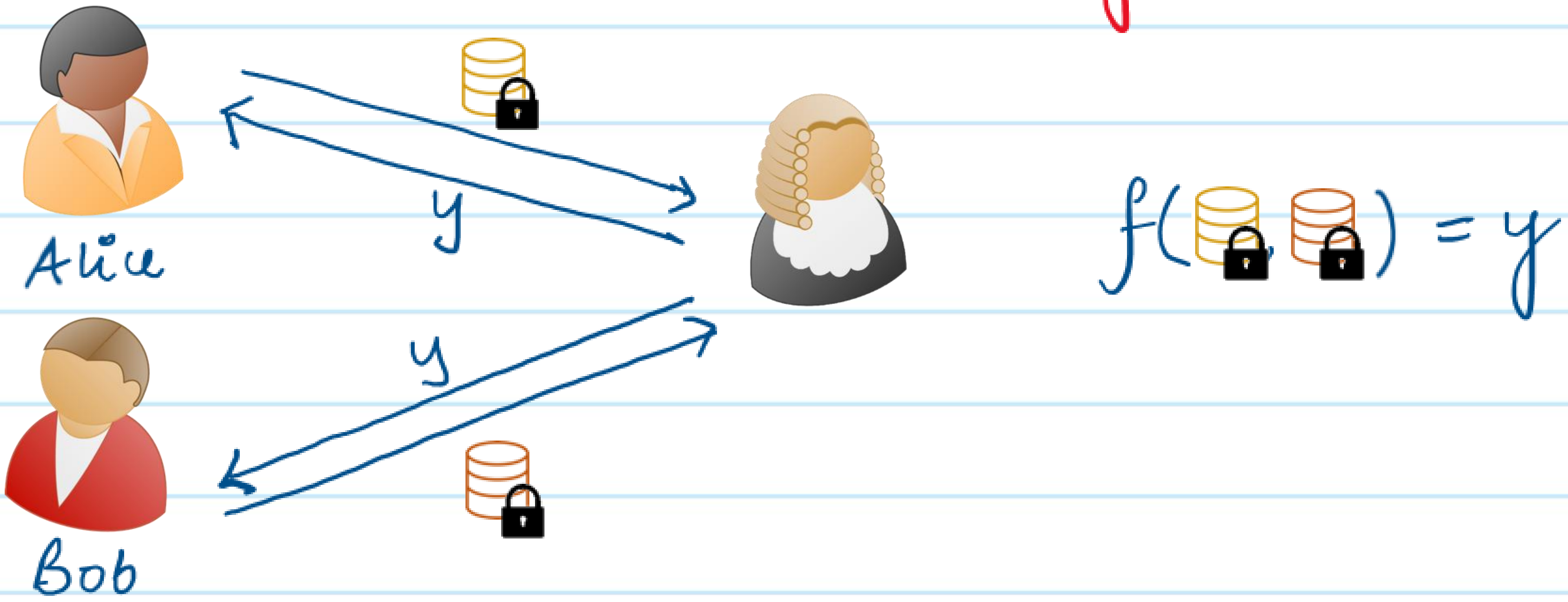
Two-Party Computation : Setting



→ Since Alice and Bob don't want to share their respective private data with each other

→ Can they share it with some other "trusted" entity?

Two-Party Computation : Setting



Problem with this idea

→ Who is this trusted entity?

→ Do such trusted entities really exist?

Secure Two-Party Computation : Setting

Can Alice and Bob still somehow compute f without sending their data to each other or to anyone else?

Secure Two-Party Computation

- Alice has input x, Bob has input y.
- Alice & Bob want to learn $f(x,y)$ by talking to each other

What are the security requirements?

- If Alice is adversarial, she should not learn anything about y
- If Bob is adversarial, he should not learn anything about x.

But $f(x,y)$ already leaks some information about x & y! Are these contradictory requirements?

Secure Two-Party Computation

→ Alice has input x, Bob has input y.

→ Alice & Bob want to learn $f(x,y)$ by talking to each other

What are the security requirements?

→ If Alice is adversarial, she should not learn anything about y beyond $f(x,y)$

→ If Bob is adversarial, he should not learn anything about x beyond $f(x,y)$

What does Adversarial Mean?

We typically consider two-types of adversaries:
(there are many other types!)

Semi-Honest These adversaries honestly follow the instructions of the protocol. But they will later try to analyze the protocol transcript to learn any extra information about the other party's input that is not already implied by $f(x,y)$

Malicious These adversaries will deviate from the protocol instructions and do whatever they want

What does Adversarial Mean?

We typically consider two-types of adversaries:
(there are many other types!)

Today **Semi-Honest** These adversaries honestly follow the instructions of the protocol. But they will later try to analyze the protocol transcript to learn any extra information about the other party's input that is not already implied by $f(x,y)$

Malicious These adversaries will deviate from the protocol instructions and do whatever they want

Formalizing the Security Requirements for Secure Two-Party Computation

→ We don't want the adversary participating in the protocol to learn anything about the other party's input beyond what can be inferred from the output.

→ In other words, whatever the adversary sees in the protocol could have been simulated by him given only his own input & output

Formalizing the Security Requirements for Secure Two-Party Computation

{View of the adversary in the protocol}

is indistinguishable from

{A simulated view that the adversary could have computed himself given his own input and output, without having communicated with the other party.}

- Input of the other party is not involved in the second case
- The adversary must not have learnt anything else about that input.

Semi-Honest Secure Two-Party Computation

Definition: A protocol π securely computes a function f in the semi-honest model, if $\exists$ a pair of two n.u. PPT simulator algorithms S_A and S_B, such that for every security parameter k, and $\forall$ inputs $x, y \in \{0,1\}^k$, it holds that:

$$\{S_A(x, f(x,y)), f(x,y)\} \approx \{\text{view}_A(e), \text{out}_B(e)\}$$

$$\{S_B(y, f(x,y)), f(x,y)\} \approx \{\text{view}_B(e), \text{out}_A(e)\}$$

where $e \leftarrow [A(x) \leftrightarrow B(y)]\}$

Semi-Honest Secure Two-Party Computation

Definition: A protocol π securely computes a function f in the semi-honest model, if $\exists$ a pair of two n.u. PPT simulator algorithms S_A and S_B, such that for every security parameter k, and $\forall$ inputs $x, y \in \{0,1\}^k$, it holds that:

$$\{S_A(x, f(x,y)), f(x,y)\} \approx \{\text{view}_A(e), \text{out}_B(e)\}$$

Why is this here?

$$\{S_B(y, f(x,y)), f(x,y)\} \approx \{\text{view}_B(e), \text{out}_A(e)\}$$

$$\text{where } e \leftarrow [A(x) \leftrightarrow B(y)]\}$$

Oblivious Transfer (OT)

Consider the following functionality:

	Alice	Bob
Input:	(a_0, a_1)	b
Output:	$\perp$	a_b

Security: Alice doesn't learn b.
Bob doesn't learn a_{1-b}

Why Should we care About OT?

→ OT is compute: Given a secure protocol for OT, it is possible to compute any function securely without relying on any other computational assumptions.

→ OT is necessary: OT is the minimal assumption for secure computation.
(unless we assume some restrictions)

Constructing Oblivious Transfer

Building Block I

Hardcore Predicate: Hard core bit cannot be predicted with probability $> 1/2 +$ negl(K), even given the output of a one-way function.

Definition: A predicate $h:\{0,1\}^* \to \{0,1\}$ is a hardcore predicate for a OWF $f(.)$, if it is efficiently computable given x, $\exists$ a negligible function $\nu(.)$, s.t., $\forall$ n.u. PPT adv A, & $\forall$ security parameters K,

$$\Pr\left[A(1^K, f(x)) = h(x) ;\ x \xleftarrow{\$} \{0,1\}^K\right] \leq \frac{1}{2} + \nu(K)$$

Constructing Oblivious Transfer: Building Block II

Trapdoor One-way Permutations: A collection of permutations is a family of permutations $F = \{f_i : D_i \to R_i\}_{i \in I}$ satisfying the following properties:

- Sampling Function: $\exists$ a PPT Gen, s.t. $Gen(1^K) \to (i \in I, t)$
- Sampling from Domain: $\exists$ a PPT algorithm that on input i outputs a uniformly random element of D_i
- Evaluation: $\exists$ a PPT algorithm that on input $i, x \in D_i$, outputs $f_i(x)$.
- Inversion with trapdoor: $\exists$ a PPT algorithm Inv s.t.

$$Inv(i, t, y) \longrightarrow f_i^{-1}(y)$$

- Hard to invert: $\forall$ n.u PPT Adv A, $\exists$ negl func $\nu(.)$ st.

$$Pr[f_i(A(1^K, i, y)) = x] ; i \leftarrow Gen(1^K), x \leftarrow D_i, y \leftarrow f_i(y)] \leq \nu(K)$$

Construction of Oblivious Transfer.

Alice	Bob
Input: (a_0, a_1)	b

Protocol: $(f_i, f_i^{-1}) \leftarrow Gen(1^k)$ $\xrightarrow{f_i}$ $x \xleftarrow{\$} \{0,1\}^k$, $y_{1-b} \xleftarrow{\$} \{0,1\}^k$

$\xleftarrow{(y_0, y_1)}$ $y_b = f_i(x)$

$\forall j \in \{0,1\}$

$z_j = h(f_i^{-1}(y_j)) \oplus a_j$ $\xrightarrow{(z_0, z_1)}$

Output $h(x) \oplus z_b$