Bar-Ilan University
Dept. of Computer Science

Session 1: Background and Definitions

Yehuda Lindell
Bar-Ilan University

Secure Computation in Practice

- A request from 1 month ago:
  - A nonprofit organization in New York, under contract from the US government is doing research on criminal justice
  - The organization asked the US immigration authorities for the list of "Alien Registration Numbers" of aliens arrested in New York City
    - To see which of them are on their list
  - Neither party can hand over their list due to privacy concerns
- This is secure set intersection

Secure Multiparty Computation

Bar-Ilan University
Dept. of Computer Science

- A set of parties with private inputs
- Parties wish to jointly compute a function of their inputs so that certain security properties are preserved
- Properties must be ensured even if some of the parties maliciously attack the protocol
- Can model any cryptographic task

Security Requirements

- Consider a secure auction (with secret bids):
  - An adversary may wish to learn the bids of all parties – to prevent this, require PRIVACY
  - An adversary may wish to win with a lower bid than the highest – to prevent this, require CORRECTNESS
  - But, the adversary may also wish to ensure that it always gives the highest bid – to prevent this, require INDEPENDENCE OF INPUTS
  - An adversary may try to abort the execution if its bid is not the highest – require FAIRNESS

General Security Properties

Bar-Ilan University
Dept. of Computer Science

- **Privacy**: only the output is revealed
- **Correctness**: the function is computed correctly
- **Independence of inputs**: parties cannot choose inputs based on others' inputs
- **Fairness**: if one party receives output, all receive output
- **Guaranteed output delivery**

Defining Security

Bar-Ilan University
Dept. of Computer Science

- Option 1: analyze security concerns for each specific problem
  - Auctions: as in previous slide
  - Elections: privacy, correctness and fairness only (?)

- Problems:
  - How do we know that all concerns are covered?
  - Definitions are application dependent and need to be redefined from scratch for each task

Defining Security

Bar-Ilan University
Dept. of Computer Science

- Option 2: general definition that captures all (most) secure computation tasks

- Properties of any such definition
  - Well-defined adversary model
  - Well-defined execution setting
  - Security guarantees are clear and simple to understand

Modeling Adversaries

Bar-Ilan University
Dept. of Computer Science

- Adversarial behavior
  - Semi-honest: follows the protocol specification
    - Tries to learn more than allowed by inspecting transcript
  - Malicious: follows any arbitrary strategy
  - Covert: follows any arbitrary strategy, but is averse to being caught…
- Adversarial power
  - Polynomial-time
  - Computationally unbounded: information-theoretic security

Modeling Adversaries

- **Corruption strategy**
  - Static: the set of corrupted parties is fixed before the execution begins
  - Adaptive: the adversary can corrupt parties during the execution, based on what has happened
    - Models modern "hacking"
    - Cannot use strategies that choose a small set of representatives to compute for all
    - In general, **much harder!**

Execution Setting

Bar-Ilan University
Dept. of Computer Science

- Stand-alone
  - Consider a single protocol execution only (or that only a single execution is under attack)
- Concurrent general composition
  - Arbitrary protocols executed concurrently
  - Realistic setting, very important model
- Stand-alone vs composition
  - **Stand-alone**: a good place to start studying secure computation, techniques and tools are helpful
  - **Composition**: true goal for constructions

Feasibility of Secure Computation

Bar-Ilan University
Dept. of Computer Science

- Assuming an honest majority, any functionality can be securely computed
  - Even information theoretically, and with adaptive security
- Without an honest majority, it is impossible to achieve fairness in general
  - Intuition behind proof of impossibility – later
  - Current understanding of fairness
- Without an honest majority, any funct. can be securely computed without fairness

Preliminaries

Bar-Ilan University
Dept. of Computer Science

- **Notations:**
  - Security parameter **n**
  - We wish security to hold for all inputs of all lengths, as long as **n** is large enough

- **Function μ is negligible:** if for every polynomial $p(\cdot)$ there exists an N such that for all $n>N$ we have $\mu(n) < 1/p(n)$

Preliminaries

Bar-Ilan University
Dept. of Computer Science

- **Probability ensemble X={X(a,n)}**
  - Infinite series, indexed by a string **a** and natural **n**
  - Each **X(a,n)** is a random variable
    - In our context: output of protocol execution with input **a** and security parameter **n**
    - Probability space: randomness of parties

Preliminaries

Bar-Ilan University
Dept. of Computer Science

- **Computational indistinguishability $X \approx Y$**
  - For every (non-uniform) polynomial-time distinguisher D there exists a negligible function μ such that for every **a** and all large enough **n**'s:

$$|\Pr[D(X(a,n)=1]-\Pr[D(Y(a,n)=1]| < \mu(n)$$

- **Statistical closeness**
  - The same but D is unbounded in running time

Notation

Bar-Ilan University
Dept. of Computer Science

- Functionality
  - $f=(f_1,\ldots,f_m)$: for input vector $\mathbf{x}$, each $f_i(\mathbf{x})$ is a random variable (for probabilistic functionalities)
  - Party P_i receives f_i
  - We denote $(x,y) \rightarrow (f_1(x,y),f_2(x,y))$

Semi-Honest Adversaries

Bar-Ilan University
Dept. of Computer Science

- **Simulation:**
  - Given input and output, can generate the adversary's view of a protocol execution
  - Important: since parties follow protocol, the inputs are **well defined**

Semi-Honest Adversaries

Bar-Ilan University
Dept. of Computer Science

- For every semi-honest A, there exists a simulator S such that for every set of corrupted parties I and every vector of inputs x, the following are *close*
  - The output of A, and the outputs of all parties after a protocol execution
  - The output of S given x_i and $f_i(x)$ for all $i \in I$), and all the values $f_1(x),\ldots,f_m(x)$

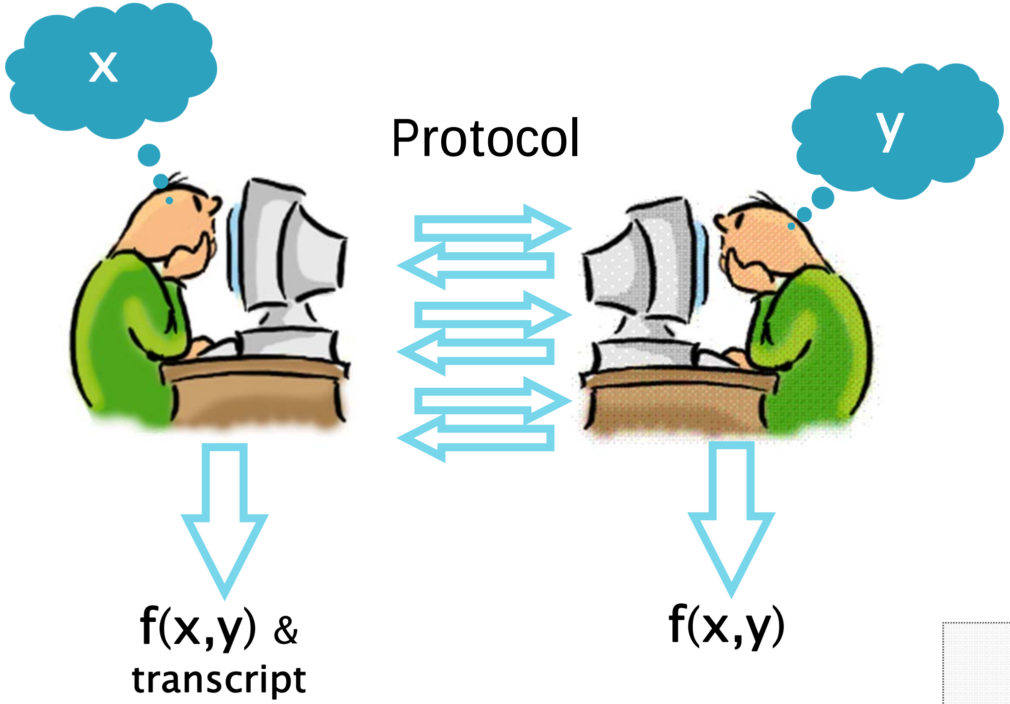
Security Levels

Bar-Ilan University
Dept. of Computer Science

- Defining "close"
  - Computational security = computational indistinguishability
  - Statistical security = statistical closeness
  - Perfect security = identical distributions

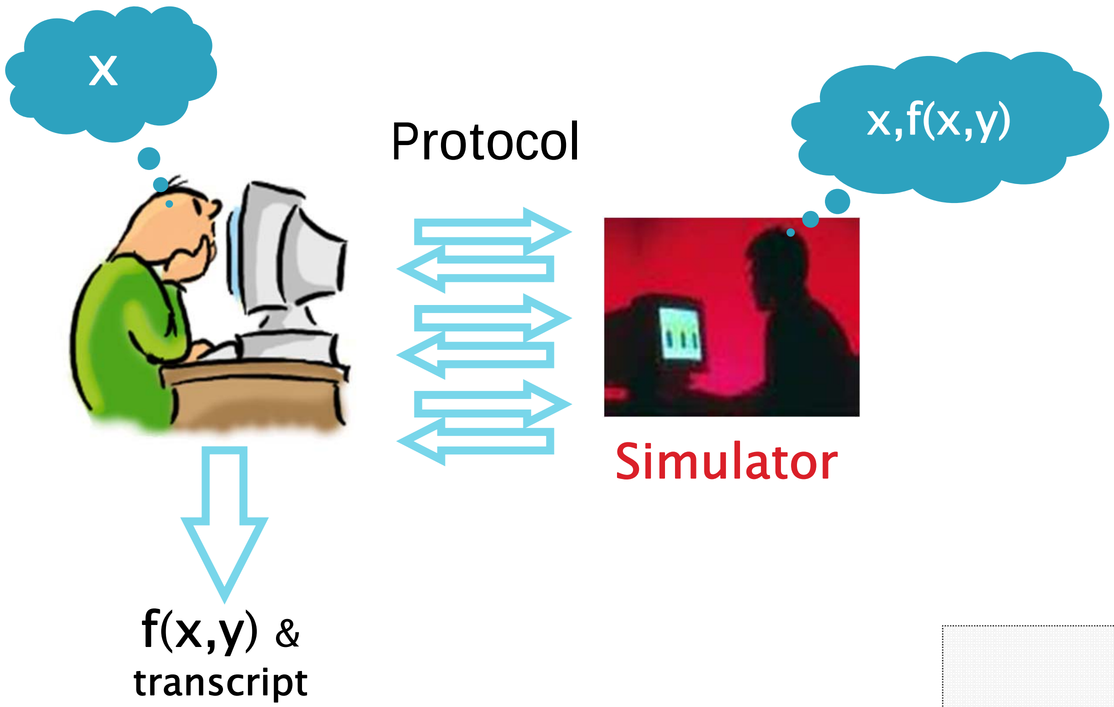
Semi-Honest Adversaries

Bar-Ilan University
Dept. of Computer Science



Semi-Honest Adversaries

Bar-Ilan University
Dept. of Computer Science



Properties

Bar-Ilan University
Dept. of Computer Science

- Correctness, independence of inputs, fairness are all non-issues in the semi-honest model
- Why is privacy guaranteed by this definition?
  - The adversary's view in an execution can be generated from the input and output only
  - If the adversary can compute something after a real protocol execution, it can compute it just from the input/output
  - Very similar to zero-knowledge

Joint Distribution

Bar-Ilan University
Dept. of Computer Science

- A crucial point: need to consider the joint distribution of adversary's output and honest parties' output
- In the definition:
  - We compare the distribution of all inputs and outputs together with the adversary's output

Joint Distribution

Bar-Ilan University
Dept. of Computer Science

- Example:
  - **Functionality**: **A** outputs random bit, **B** outputs nothing
    - **B** should clearly not learn **A**'s output bit
  - **Protocol**: **A** chooses a random bit, outputs it, and sends the bit to **B** (who ignores it)

- **This is simulatable when separately looking at distribution of B's view and actual outputs**

Deterministic Functionalities

Bar-Ilan University
Dept. of Computer Science

- In the case of deterministic functionalities, the outputs are fully determined by the inputs
- It suffices to separately prove
  - Correctness
  - Simulation: can generate view of semi-honest adversary (corrupted parties' view), given inputs and outputs only
    - This is significantly easier!

Malicious Adversaries

- First attempt: require the existence of a simulator that generates the adversary's view given the inputs/outputs of corrupted
- Problem: what are the inputs used by the adversary?
  - They are not necessarily those written on the input tape
  - They are not explicit: the adversary doesn't run the protocol but arbitrary code

Malicious Adversaries

Bar-Ilan University
Dept. of Computer Science

- We also need to require independence of inputs, correctness, fairness etc.
  - These properties are not captured by “view simulation” alone
- Can we separate correctness and privacy?
  - Instead of computing f, compute a function that reveals first input bit of other party
  - Correctness or privacy???
- What about independence of inputs and privacy?

The Ideal/Real Paradigm

Bar-Ilan University
Dept. of Computer Science

- **What is the best we could hope for?**
  - An incorruptible trusted party
  - All parties send inputs to trusted party (over perfectly secure communication lines)
  - Trusted party computes output
  - Trusted party sends each party its output (over perfectly secure communication lines)
  - This is an ideal world
- **What can an adversary do?**
  - Just choose its input…

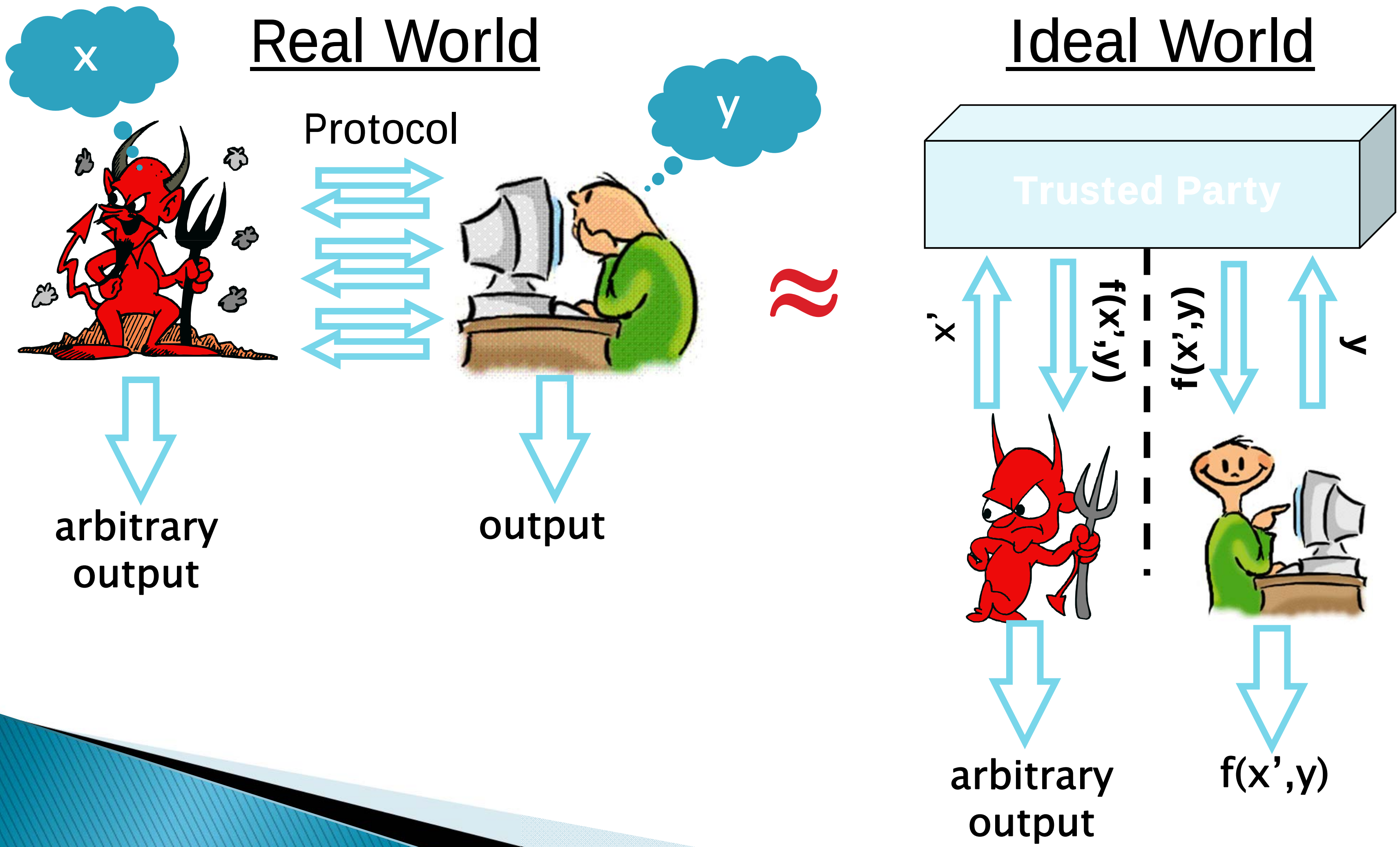
The Ideal/Real Paradigm

Bar-Ilan University
Dept. of Computer Science

- The real protocol must be like the ideal world
- Formalizing this notion:
  - For *every adversary A* attacking the real protocol, *there exists an adversary S* in the ideal model such that the output distributions (of all) are <u>close</u>
    - **Computational indistinguishability, statistical closeness or identical distributions…**
  - S simulates a real protocol execution while interacting in the ideal world
  - Here we always look at the joint output distribution

The Ideal/Real Paradigm

Bar-Ilan University
Dept. of Computer Science



“Formal” Security Definition

Bar-Ilan University
Dept. of Computer Science

- Protocol π securely computes a function f if:
  - For every non-uniform polynomial-time real-model adversary A, there exists a non-uniform polynomial-time ideal-model adversary S, such that for all input vectors and auxiliary inputs:
  - the joint outputs of A and the honest parties in a real execution of π is <u>indistinguishable</u>* from the joint outputs of S and the honest parties in an ideal execution where the trusted party computes f

* Computationally indistinguishable, statistically close or identical distributions for computational, statistical and perfect security

Properties

Bar-Ilan University
Dept. of Computer Science

- The following properties hold
  - Privacy: from adversary's outputs
  - Correctness: from honest parties' outputs
  - Independence of inputs: from ideal execution
  - Fairness and guaranteed output delivery: from ideal execution

  - More?

Relaxing the Ideal Model

Bar-Ilan University
Dept. of Computer Science

- In some cases, this ideal model is too strong and cannot be achieved
- Fairness cannot be achieved in general without an honest majority
  - Consider two parties and consider removing the last message of the protocol execution
    - Works for coin tossing…

Relaxing the Ideal Model

Bar-Ilan University
Dept. of Computer Science

- Change the instructions of the trusted party
  - Trusted party receives input from all parties
  - Trusted party sends corrupted parties' outputs to adversary
  - Adversary says "continue" or "halt"
  - If "continue", trusted party sends output to honest parties; else, it sends "abort"

Reactive Functionalities

- Functionalities that obtain inputs and provide outputs in stages
- Examples:
  - Mental poker
  - Commitment schemes
- This is also useful for relaxing ideal functionalities (give side information to S)
- The definition extends naturally to this as well

Advantages of This Approach

Bar-Ilan University
Dept. of Computer Science

- General – it captures ALL applications
- The specifics of an application are defined by its functionality, security is defined as above
- The security guarantees achieved are easily understood (because the ideal model is easily understood)
  - We can be confident that we did not "miss" any security requirements

Restricted vs General Functionalities

Bar-Ilan University
Dept. of Computer Science

- When constructing protocol for general secure computation, it suffices to consider
  - **Deterministic functionalities**: to compute a probabilistic functionality f, define $g((x,r),(y,s))=f(x,y;r\oplus s)$
  - **Single-output functionalities**: encrypt and MAC the output of the other party
  - **Non-reactive functionalities**: to compute a reactive functionality, define a series of functions that input/output shared state information (with a MAC)

Sequential Modular Composition

Bar-Ilan University
Dept. of Computer Science

- **Sequential modular composition:**
  - Secure protocols are run sequentially, with arbitrary messages sent in between them
- **Why consider this?**
  - An important security goal within itself
  - Very helpful (if not crucial) tool for analyzing the security of protocols
- **Formalization – Hybrid Model**
  - A trusted party helps to compute a sub-functionality
  - REAL messages & IDEAL messages

Sequential Modular Composition

Bar-Ilan University
Dept. of Computer Science

- Subprotocols ρ_i securely compute functionalities f_i
- Protocol π securely computes **g** in a hybrid model where a trusted party is used to compute every f_i
  - This is much easier to analyze since each f_i is effectively "perfectly secure"
- **Theorem**: assuming the above, the real protocol π^ρ that uses real calls to each ρ_i instead of a trusted party for f_i, securely computes **g**.

Sequential Modular Composition

Bar-Ilan University
Dept. of Computer Science

- Proof Sketch
  - Assume that a protocol π with a single call to **f** securely computes **g**
  - Assume that π^{ρ} is not secure; an adversary **A** breaks the protocol (with **D** that distinguishes real from ideal)
  - We construct an adversary **A′** and distinguisher **D′** to attack ρ
  - **A′** receives as auxiliary input the execution prefix of π until ρ begins, that matches the inputs given in ρ
  - After the execution, **D′** receives the outputs of all, and uses the auxiliary input to complete the execution of π
  - **D′** runs **D** and outputs whatever it does

Sequential Modular Composition

Bar-Ilan University
Dept. of Computer Science

- Proof Sketch
  - If D′ received the output of an ideal execution of f, then the output is the same as D after an ideal execution of g
    - This is by the proof of security of π in the hybrid model
  - If D′ received the output of a real execution of ρ, then the output is the same as D after a real execution of π^{ρ}
  - Since D distinguishes between ideal-g and real-π^{ρ} it follows that D′ distinguishes between ideal-f and real-ρ

Concurrent Composition

- **We have considered the stand-alone model**
  - This implies sequential composition
- **What about concurrent composition?**
  - An Internet-like setting where many (arbitrary, secure and insecure) protocols are run concurrently, with the adversary controlling the scheduling
- **This models the real-world setting more accurately**
  - We don't know what the result is of running stand-alone protocols concurrently with related inputs

Concurrent Composition

Bar-Ilan University
Dept. of Computer Science

- Concurrent general composition
  - Strictly harder than the stand-alone model
  - *Impossible* without some trusted set-up assumption (like a common reference string)
- The UC definition (universal composability) guarantees security in this setting
  - Efficient UC security is a special challenge…
- Recommended to study UC next, after studying the stand-alone setting

Relaxed Definitions

- **In order to achieve high efficiency, sometimes can consider weaker definitions**
  - Semi-honest (but this is very weak)
  - Covert adversaries: adversary may be malicious but is guaranteed to be caught cheating with good probability
    - Suitable where adversaries can be penalized for being caught cheating (e.g., business loss)
  - Privacy only (malicious)
    - Problematic…

Defining Privacy Only

- **Defining privacy only is very difficult**
  - No correctness and independence of inputs, but as we have seen it is hard to separate these properties
  - Composition is not guaranteed
- **Example:**
  - Function **f** with the property that for every **x**, there exists a **y** (denoted y_x) such that $f(x,y_x)=x$
  - If P_2 can input y_x implicitly, then it can learn **x**

Private OT

Bar-Ilan University
Dept. of Computer Science

- Oblivious transfer
  - Sender: has two strings x_0,x_1
  - Receiver: has a choice bit b
  - Outputs: sender learns nothing about b, receiver learns only of x_0,x_1
- For oblivious transfer, we know how to define privacy only, for two-round protocols
  - Fortunately we also have such protocols

Private OT

Bar-Ilan University
Dept. of Computer Science

- Why do 2 rounds help?
  - Receiver sends one message
  - Sender replies with one message
- Privacy for a malicious sender
  - Just need to prove indistinguishability of receiver's first message when **b=0** and when **b=1**
  - This can be extended to many messages
- Privacy for a malicious receiver
  - First message is generated before seeing anything
  - Require that for every first message, there exists a bit **b′** such that receiver learns nothing about $x_{b'}$

Semi-Honest vs Malicious

- Now to confuse you all…
- It is clear that any protocol that is secure in the presence of malicious adversaries is secure in the presence of semi-honest adversaries
  - A malicious adversary is stronger, and can always behave semi-honestly…
- But, the simulator in the ideal model is also stronger
  - It can change its input
- Does this make a difference?

Semi-Honest vs Malicious

Bar-Ilan University
Dept. of Computer Science

- Consider the AND function where only P_2 receives output
- Consider the following protocol:
  - P_1 sends its input directly to P_2
- Is the protocol secure?
  - Corrupted P_1 learns nothing and gives its input directly, so clearly secure
  - **Semi-honest P_2** learns P_1's input which doesn't happen if P_2's input is 0 $\Rightarrow$ not secure!
  - **Malicious P_2**: in the ideal model, simulator can always give input 1 and simulate $\Rightarrow$ secure!

Semi-Honest vs Malicious

Bar-Ilan University
Dept. of Computer Science

- Fixing this absurdity
  - Allow a semi-honest adversary to also change its input
  - Arguably, this is legitimate (to choose input)
  - This is called **augmented semi-honest**
    - Note: this stronger notion is also needed for the GMW compilation (this afternoon)
- Theorem:
  - Security for malicious adversaries implies security for augmented semi-honest adversaries

Summary

Bar-Ilan University
Dept. of Computer Science

- Semi-honest: simulator given input/output generates the adversary's view
  - Probabilistic functionalities – must consider joint distribution of view and outputs
  - Deterministic functionalities: easier, suffices to separately consider correctness and view simulation
- Malicious: ideal-real simulation
- Sequential composition
- Advanced topics
  - Concurrent composition
  - Relaxed definition
  - Semi-honest vs malicious