Secure Multi-Party Computation

Lecture 13

Must We Trust eBay?

Must We Trust eBay?

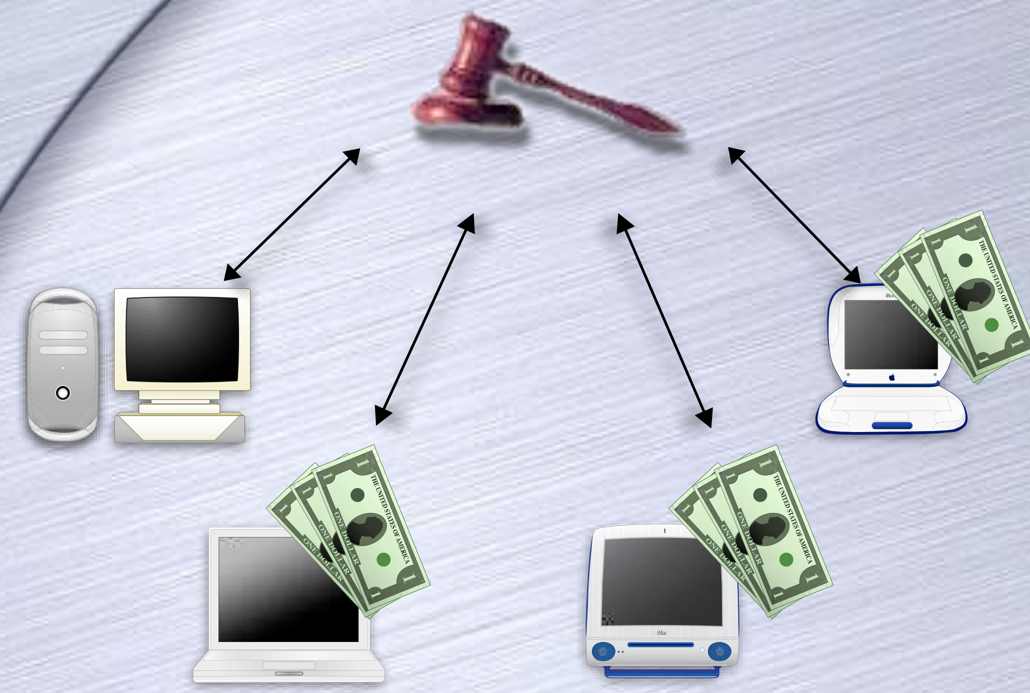
- Can we have an auction without an auctioneer?!

Must We Trust ebay?

- Can we have an auction without an auctioneer?!

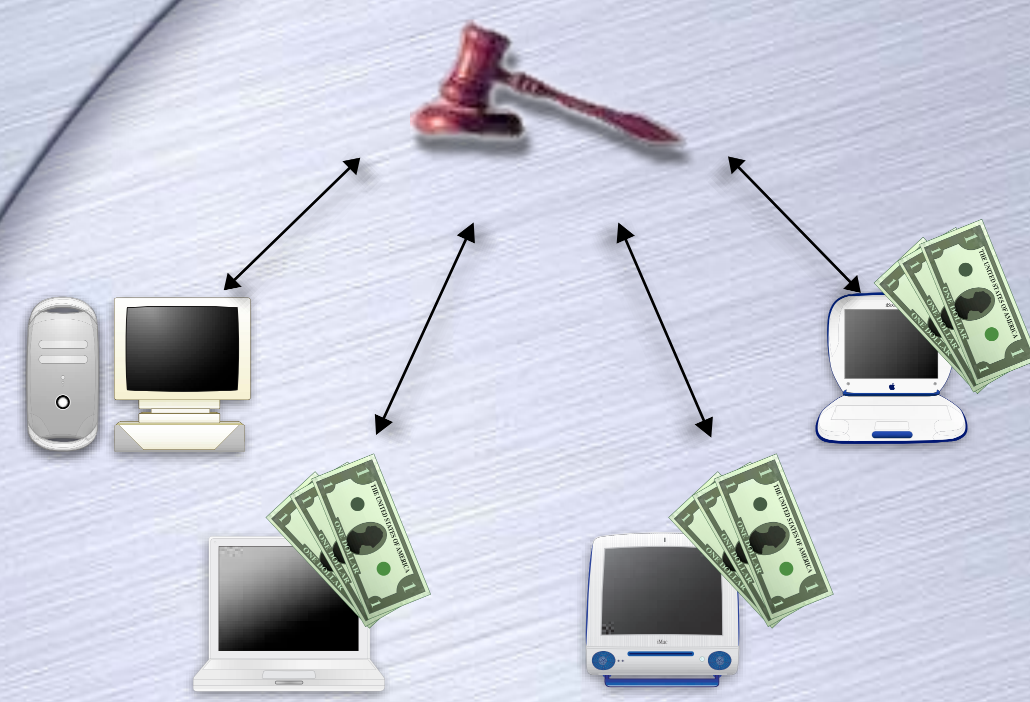
Must We Trust ebay?

- Can we have an auction without an auctioneer?!



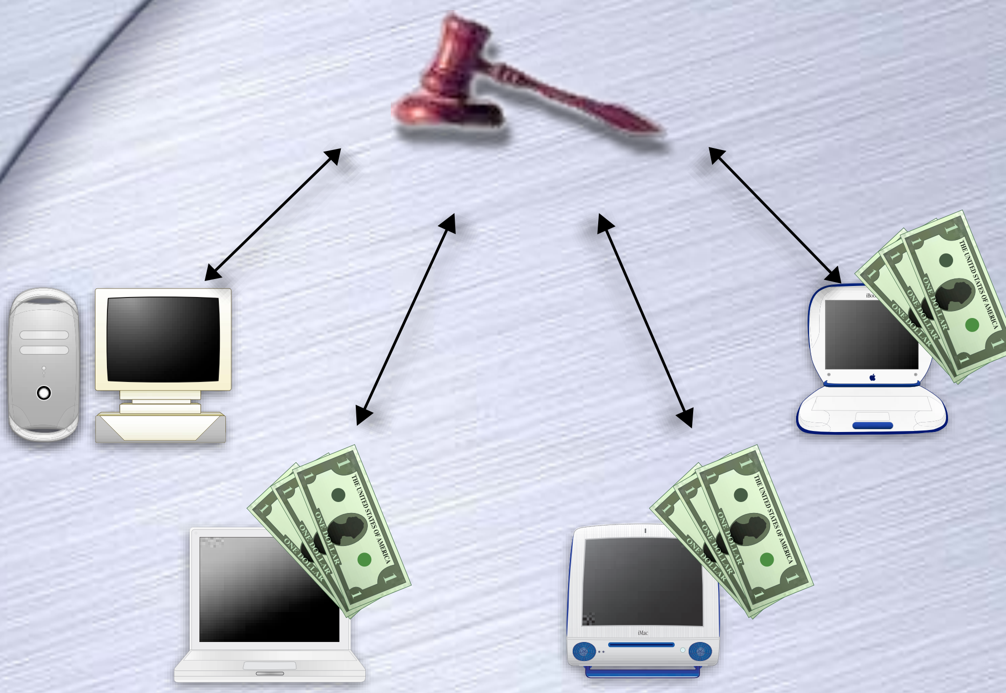
Must We Trust eBay?

- Can we have an auction without an auctioneer?!
  - Declared winning bid should be correct

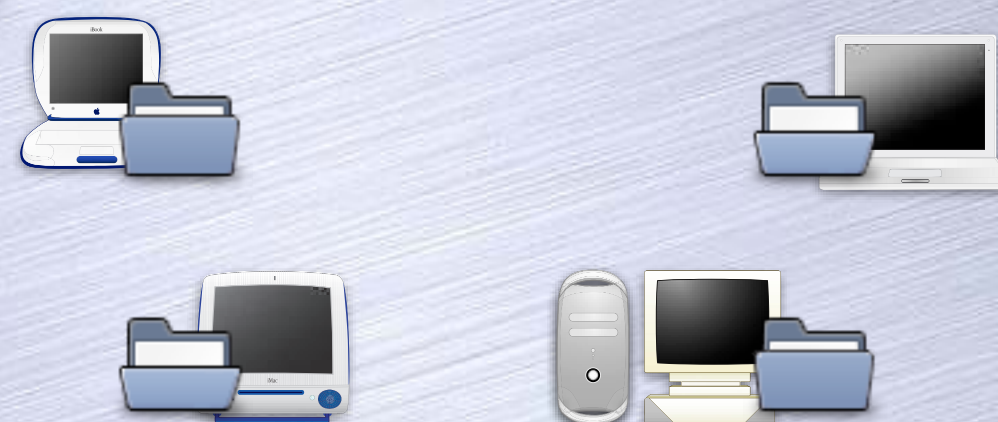


Must We Trust eBay?

- Can we have an auction without an auctioneer?!
  - Declared winning bid should be correct
  - Only the winner and winning bid should be revealed



Using data without sharing?

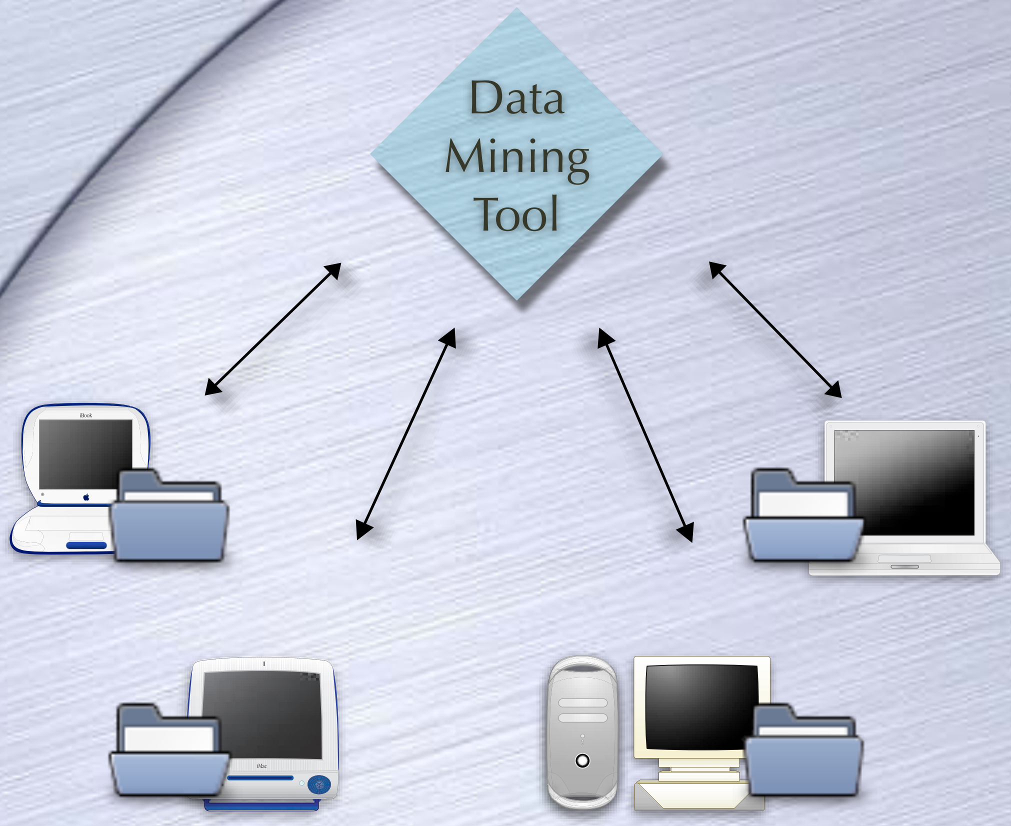


Using data without sharing?

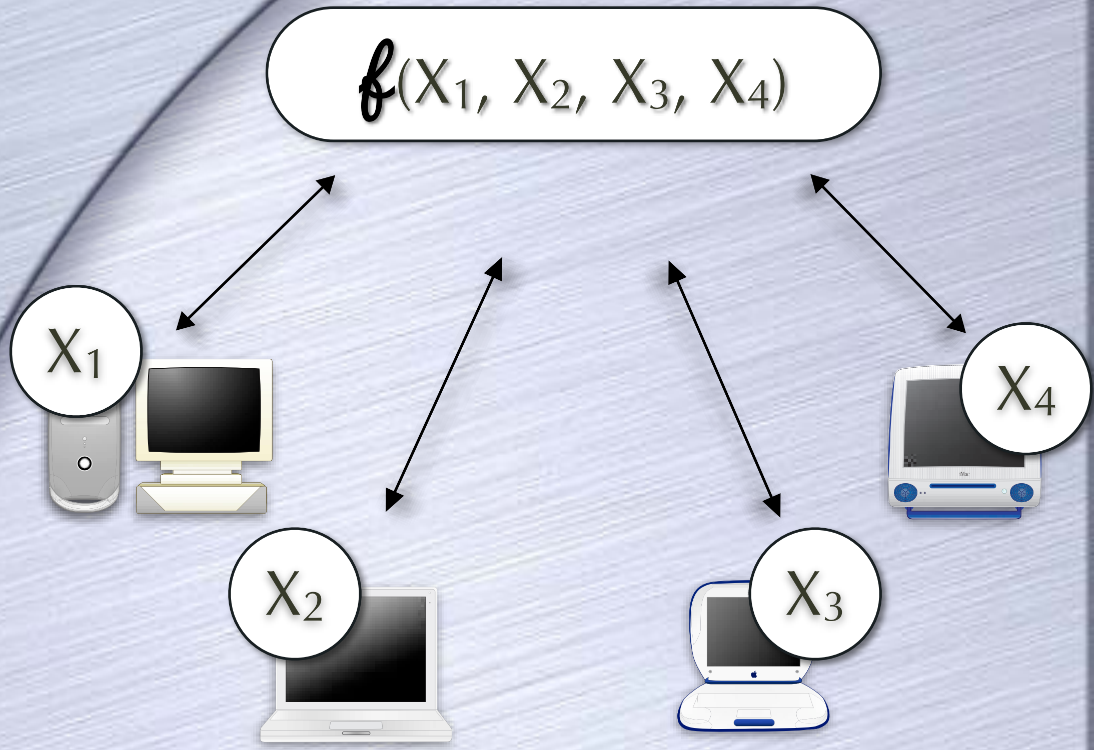
- Hospitals which can't share their patient records with anyone

Using data without sharing?

- Hospitals which can't share their patient records with anyone
  - But want to data-mine on combined data

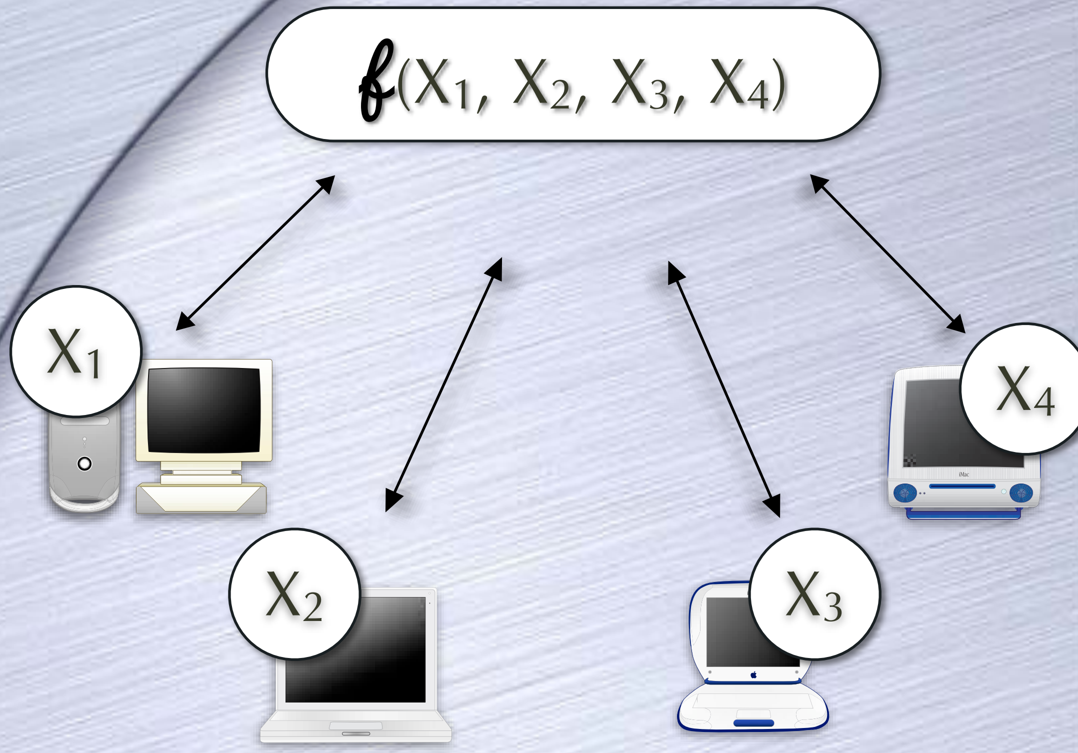


Secure Function Evaluation



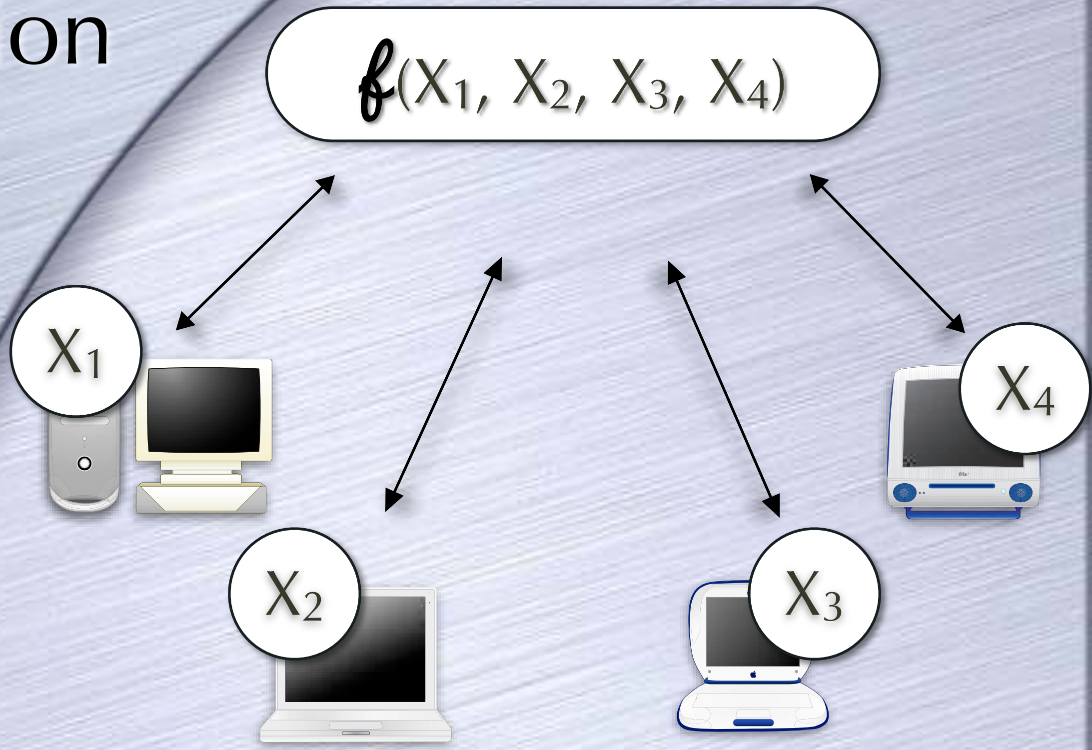
Secure Function Evaluation

- A general problem



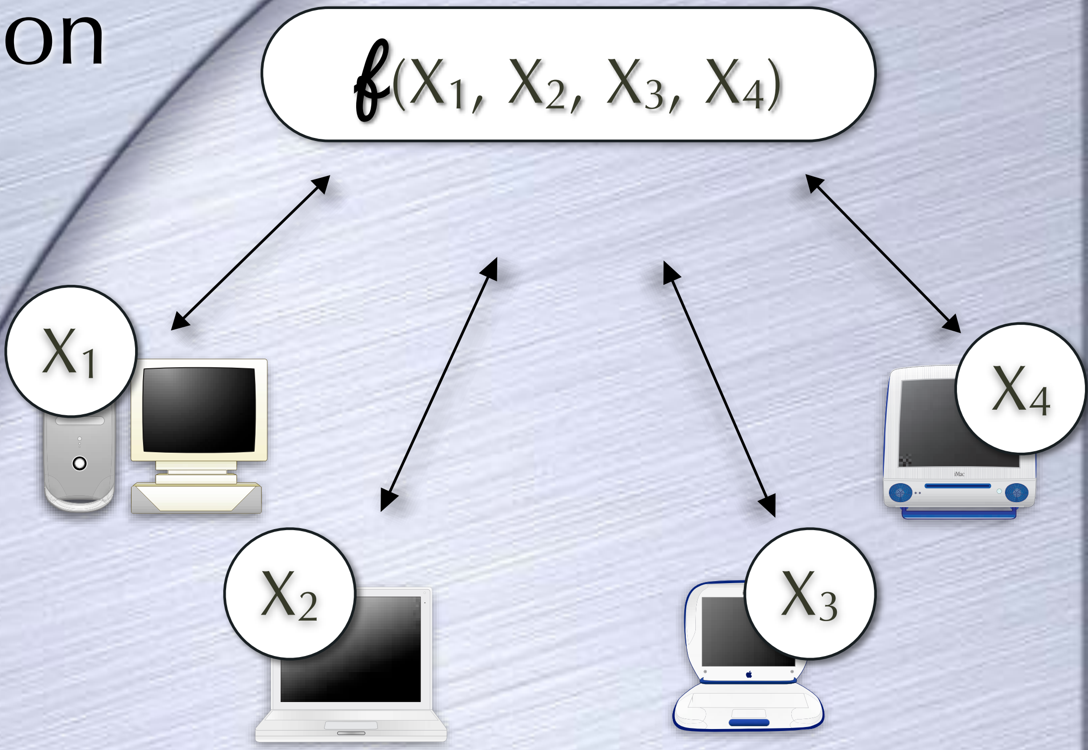
Secure Function Evaluation

- A general problem
- To compute a function of private inputs without revealing information about the inputs



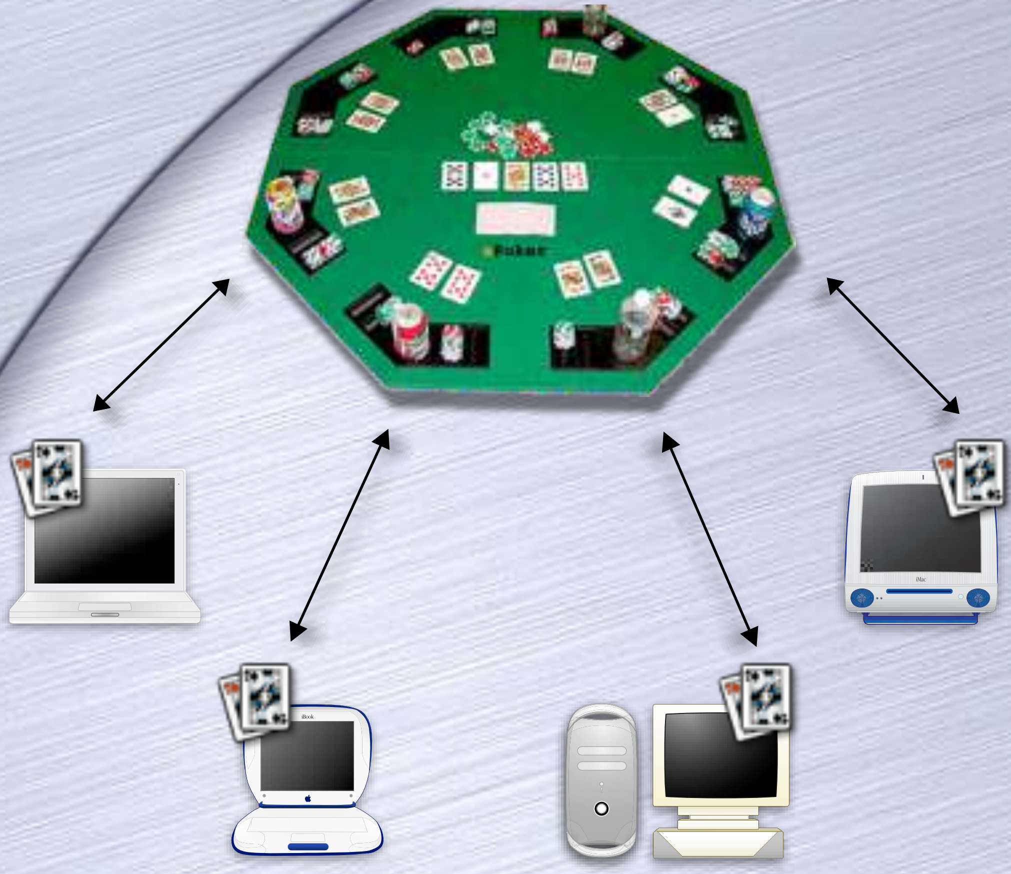
Secure Function Evaluation

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



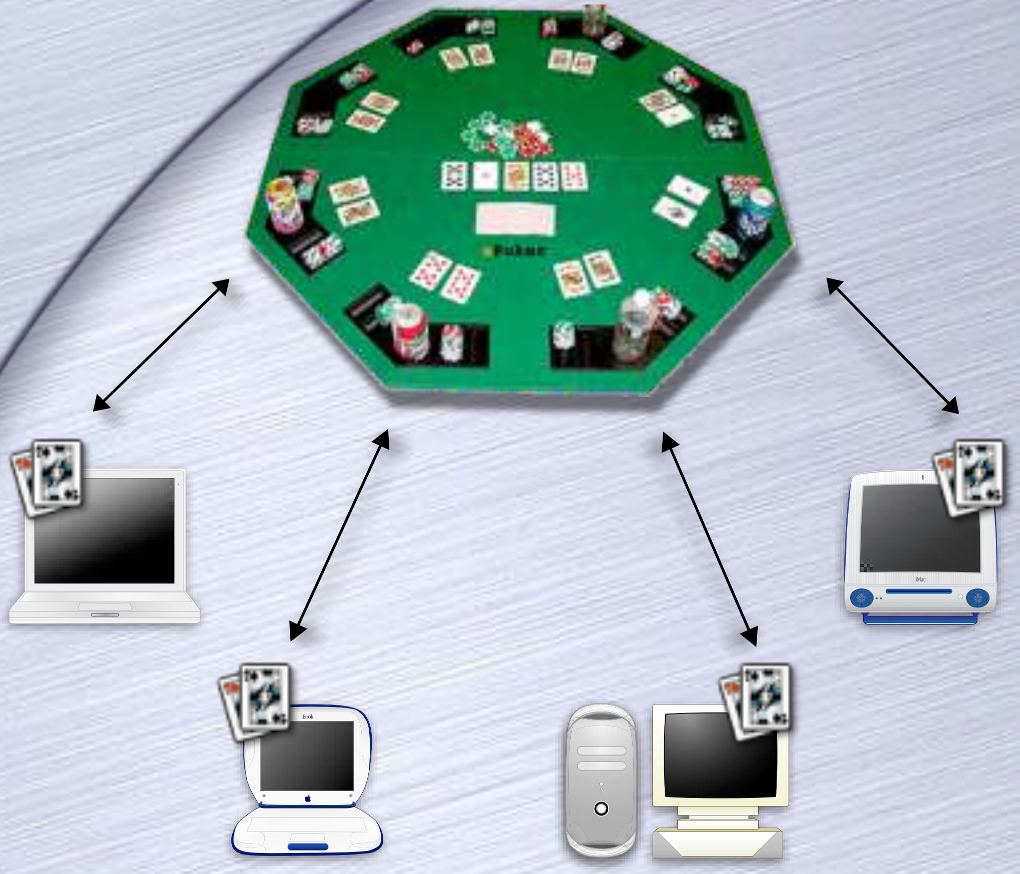
Poker With No Dealer?

Poker With No Dealer?



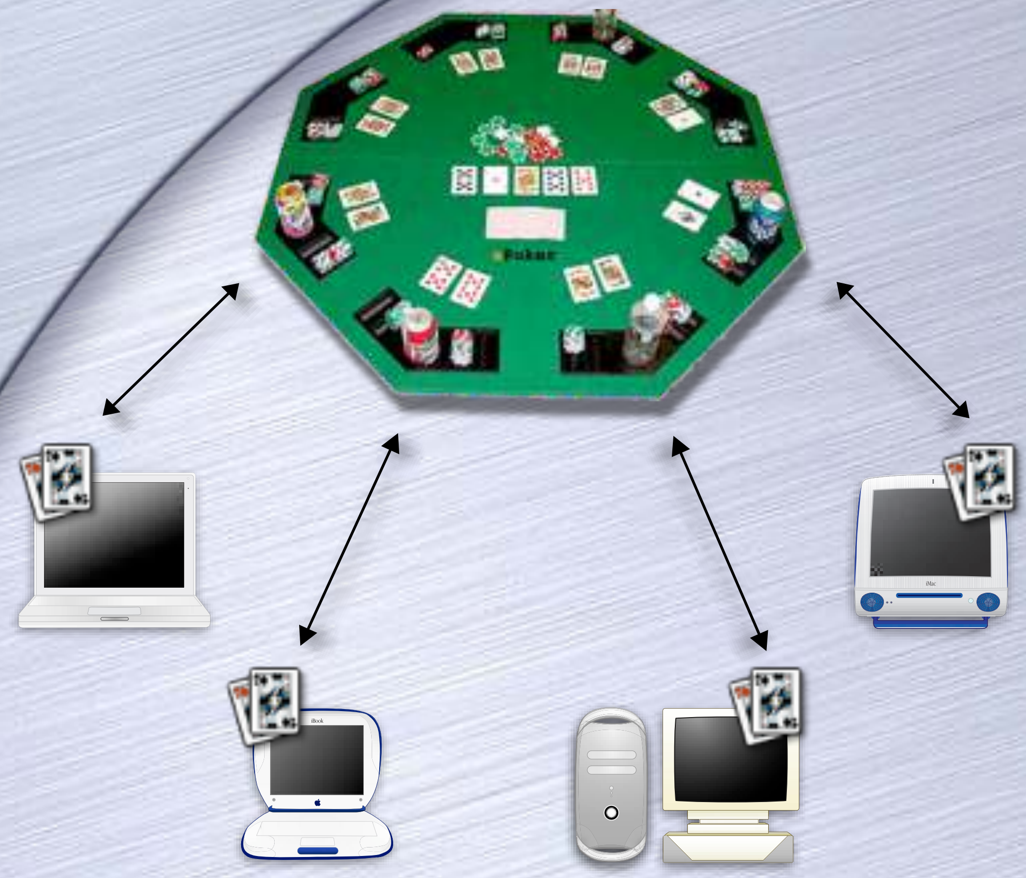
Poker With No Dealer?

- Need to ensure



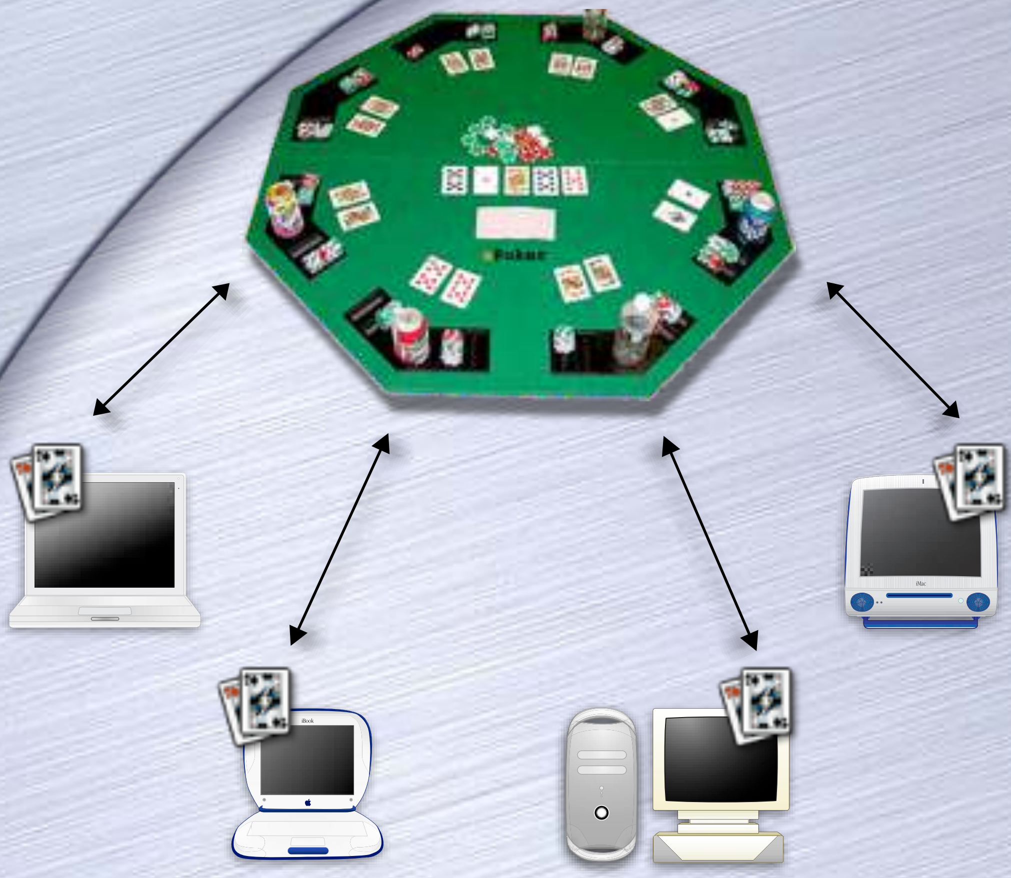
Poker With No Dealer?

- Need to ensure
  - Cards are shuffled and dealt correctly



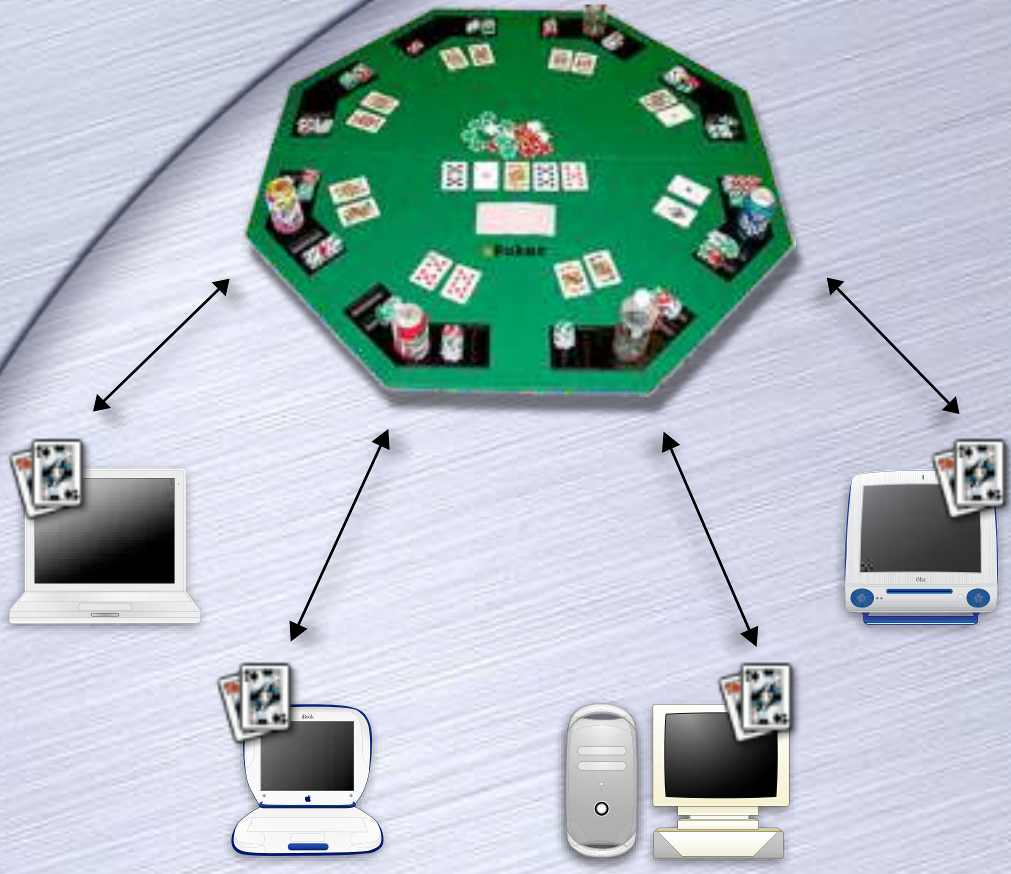
Poker With No Dealer?

- Need to ensure
  - Cards are shuffled and dealt correctly
  - Complete secrecy



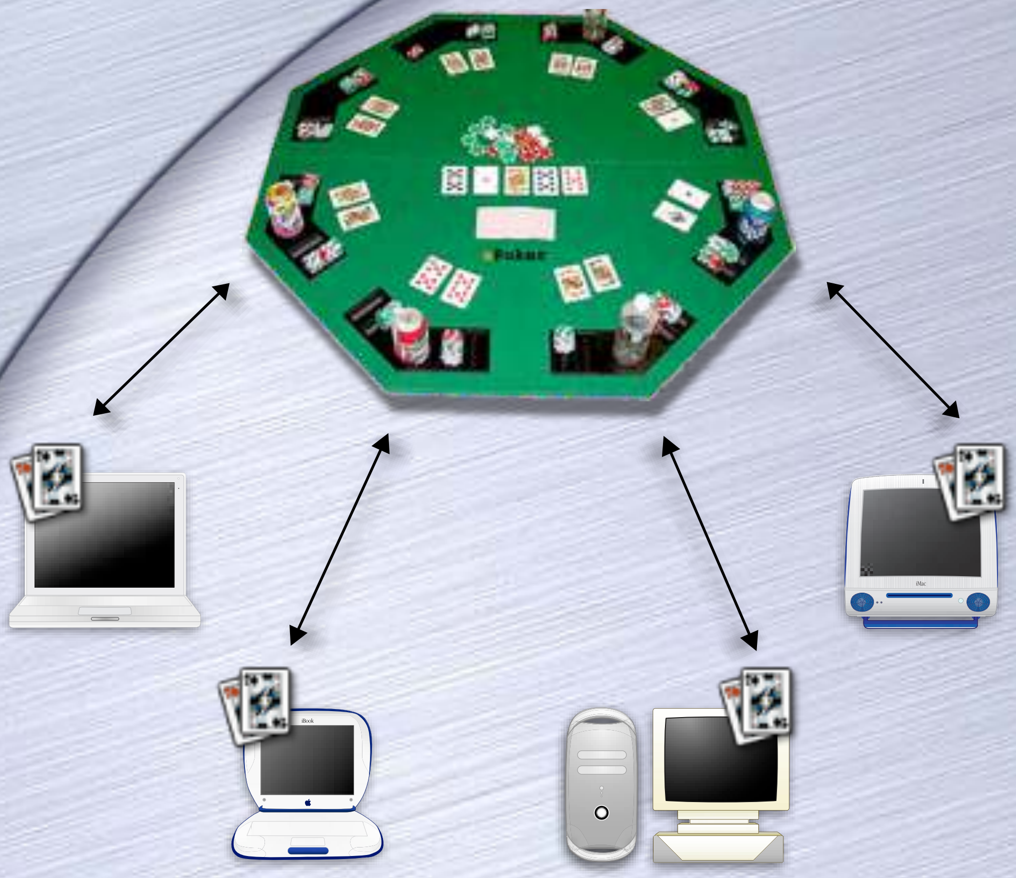
Poker With No Dealer?

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



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



The Ambitious Goal

The Ambitious Goal

- Without any trusted party, securely do
  - Distributed Data mining
  - E-commerce
  - Network Games
  - E-voting
  - Secure function evaluation
  - ....

The Ambitious Goal

- Without any trusted party, securely do
  - Distributed Data mining
  - E-commerce
  - Network Games
  - E-voting
  - Secure function evaluation
  - ....

Emulating Trusted Computation

Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel

Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel
- Secure MPC: to emulate a source of trusted computation

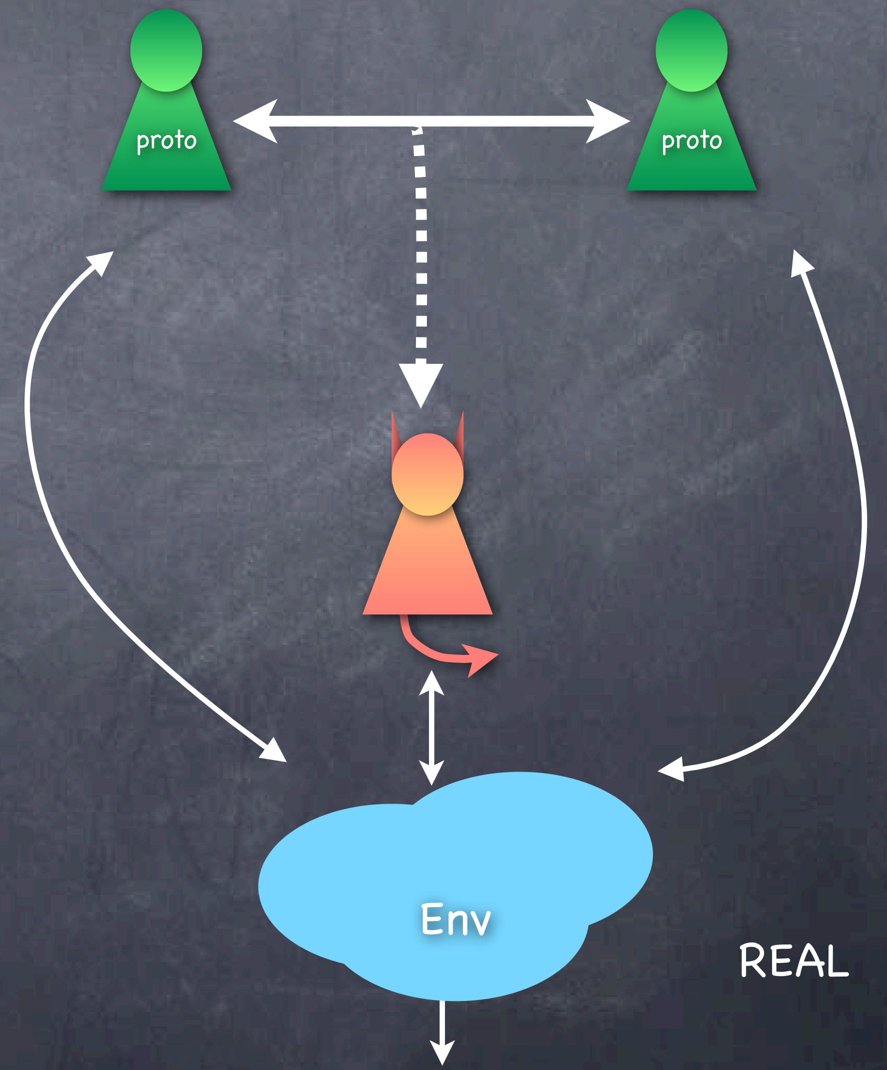
Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel
- Secure MPC: to emulate a source of trusted computation
  - Trusted means it will not "leak" a party's information to others

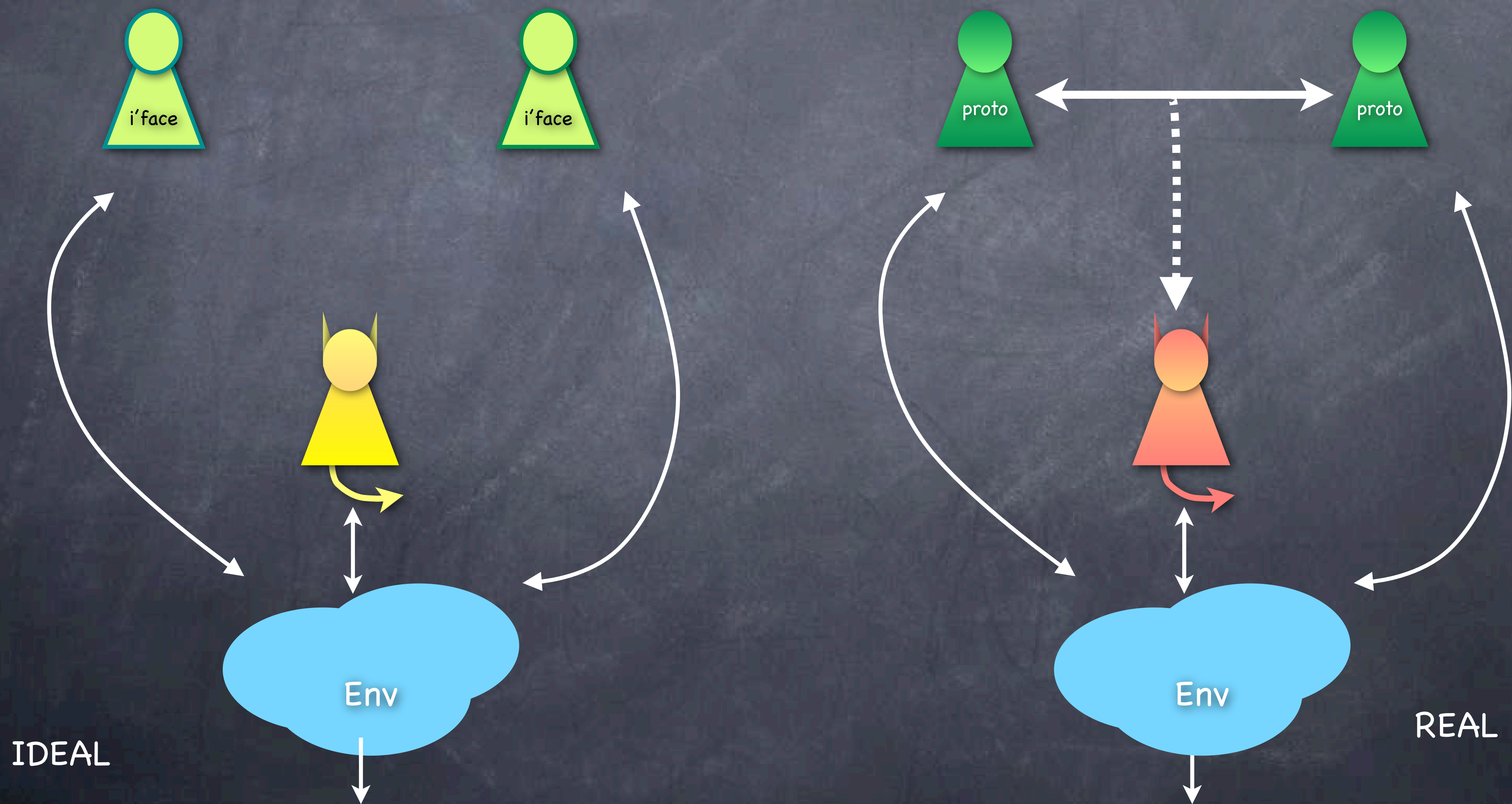
Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel
- Secure 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

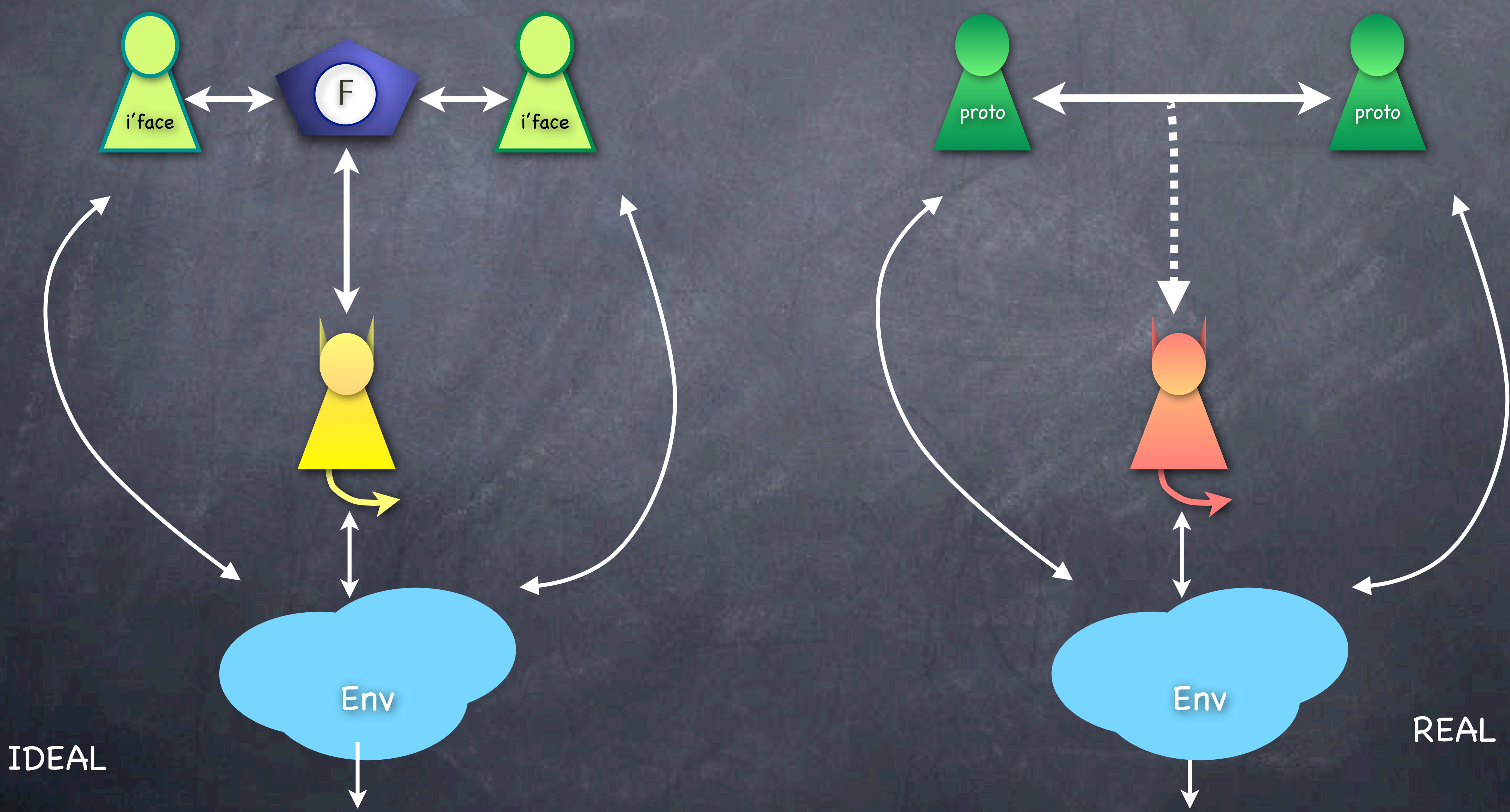
SIM-Secure MPC



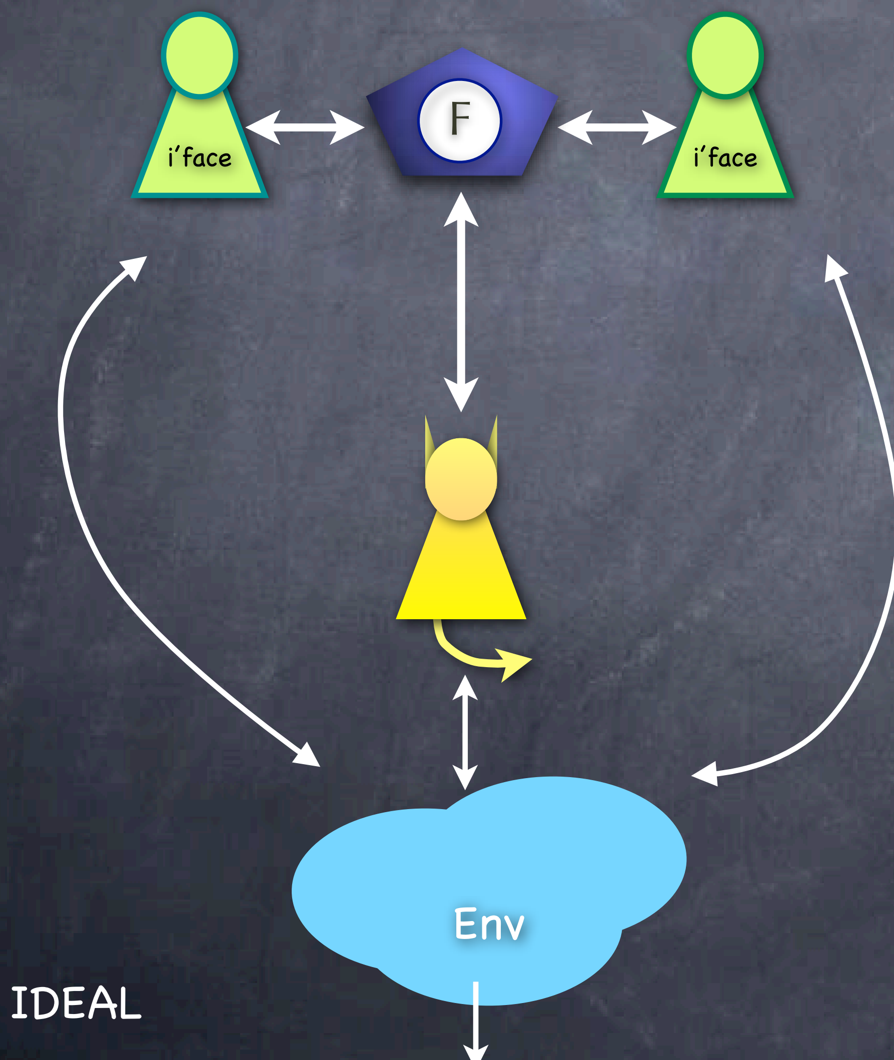
SIM-Secure MPC



SIM-Secure MPC



SIM-Secure MPC



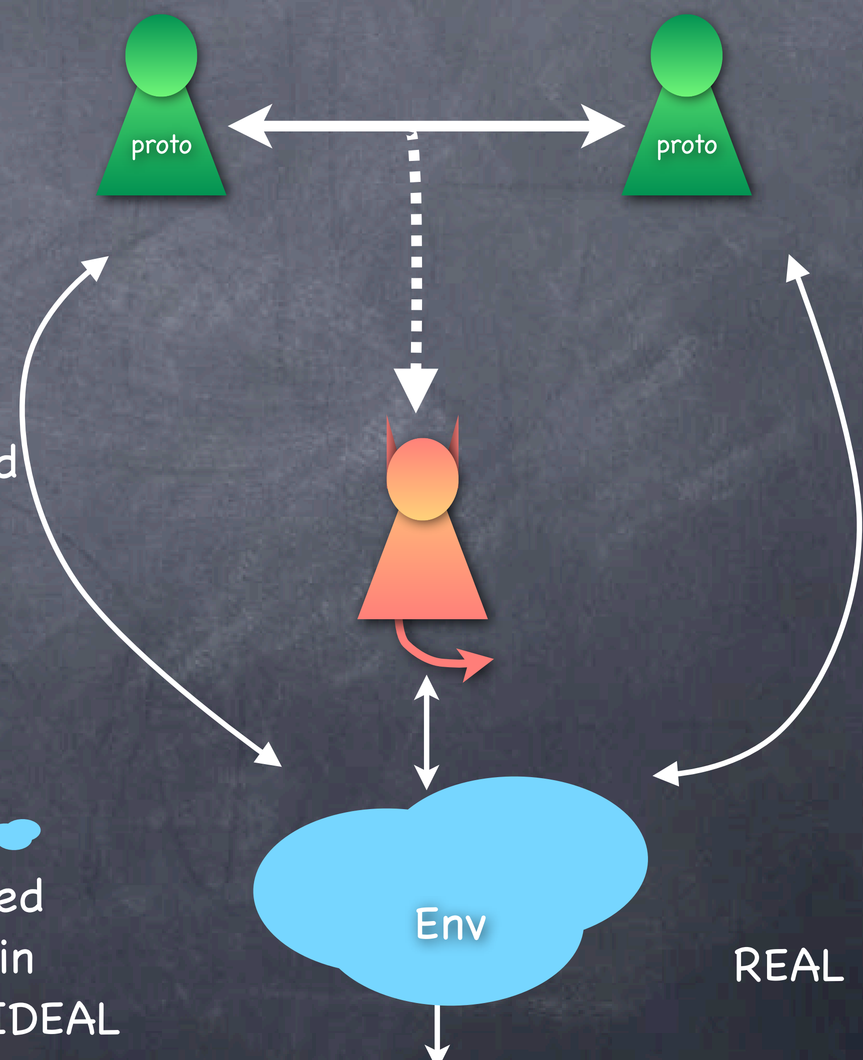
Secure (and correct) if:

∀

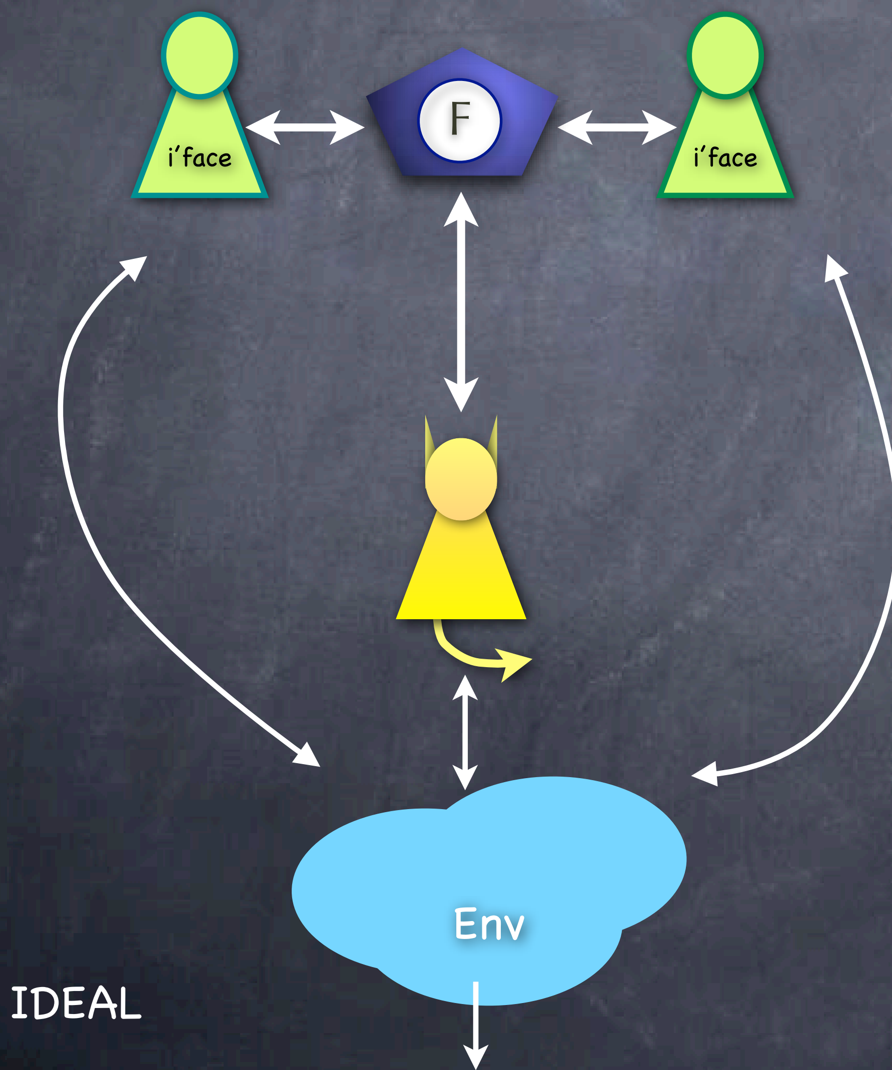
∃ s.t.

∀

output of is distributed identically in REAL and IDEAL



SIM-Secure MPC



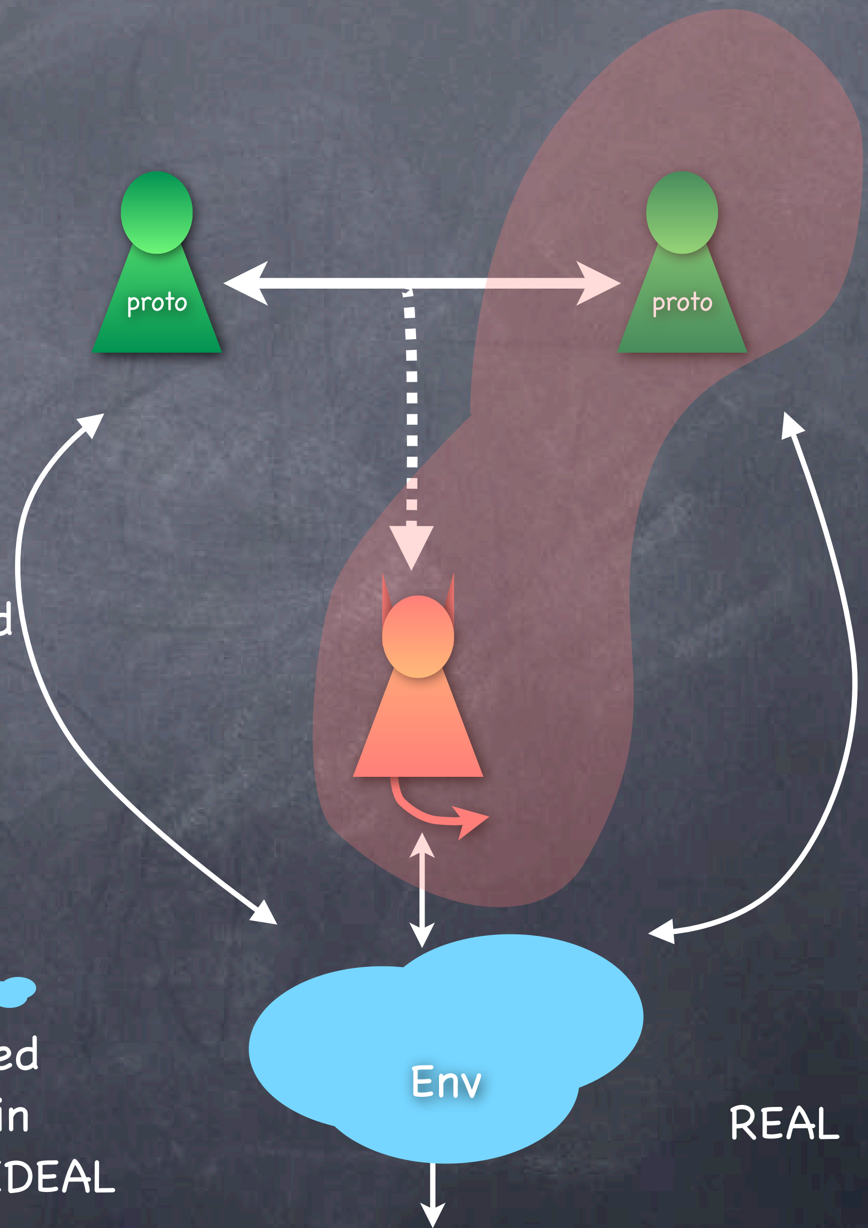
Secure (and correct) if:

∀

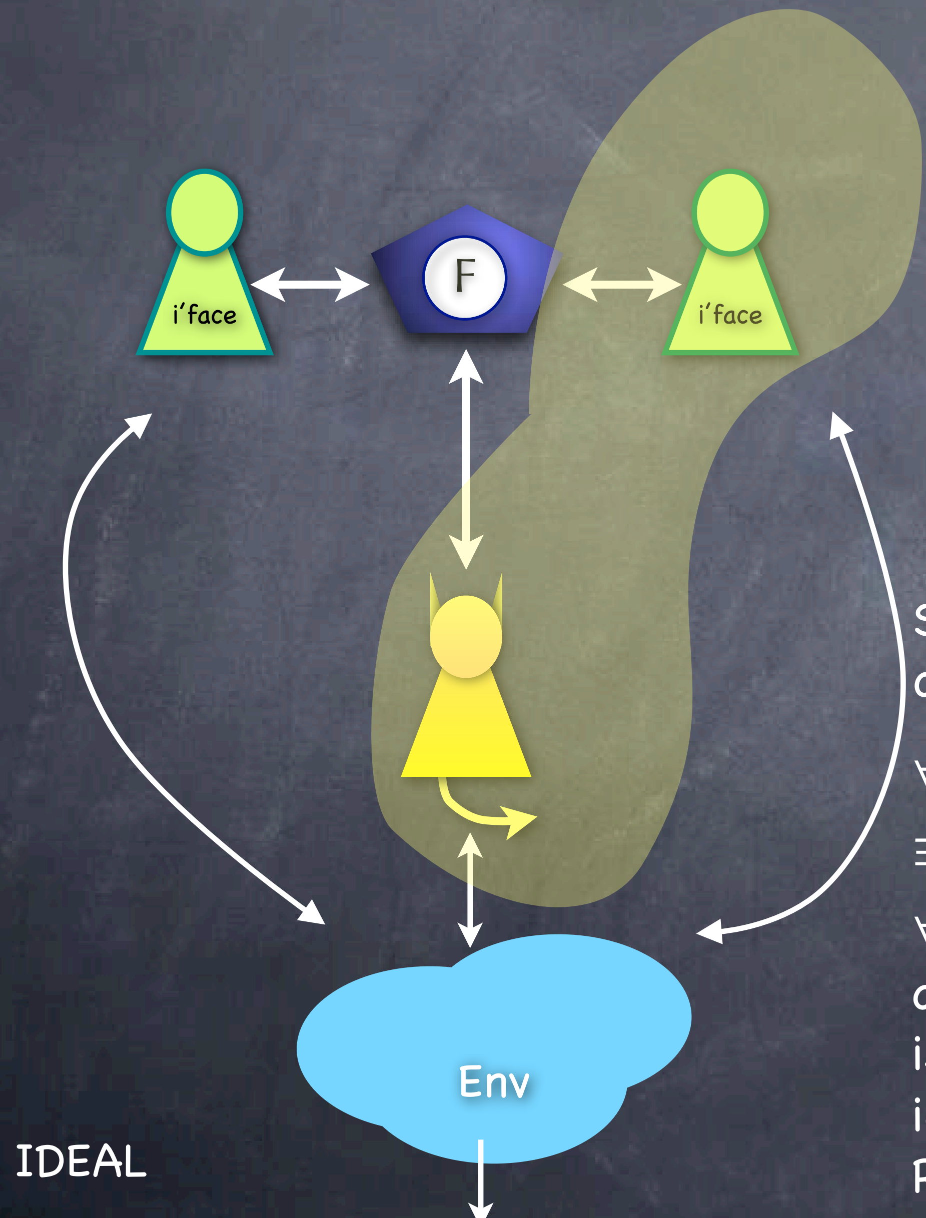
∃ s.t.

∀

output of is distributed identically in REAL and IDEAL



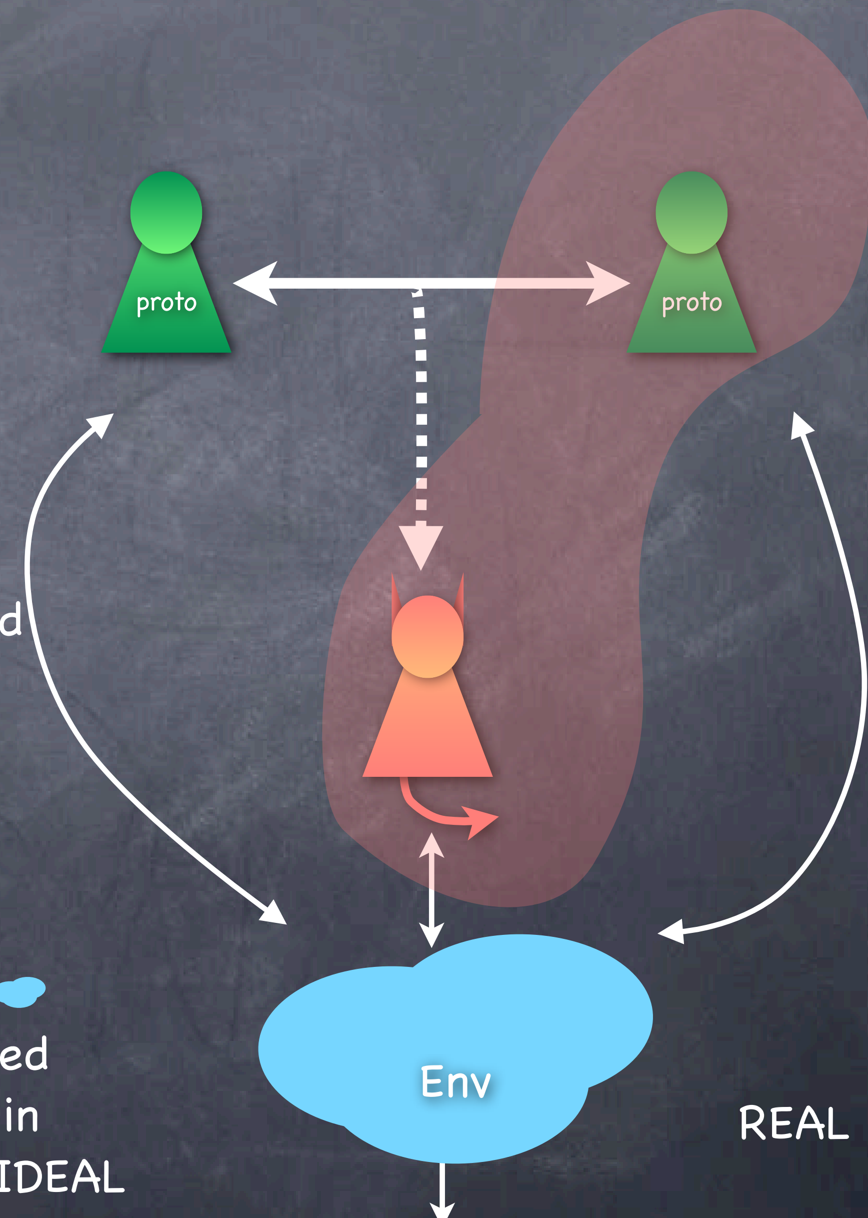
SIM-Secure MPC



Secure (and correct) if:

$\forall$ [adversary] $\exists$ [simulator] s.t. $\forall$ [environment]

output of [environment] is distributed identically in REAL and IDEAL



Trust Issues Considered

Trust Issues Considered

- Protocol may leak a party's secrets

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining
- Protocol may give adversary illegitimate influence on the outcome

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining
- Protocol may give adversary illegitimate influence on the outcome
  - Say in poker, if adversary can influence hands dealt

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining
- Protocol may give adversary illegitimate influence on the outcome
  - Say in poker, if adversary can influence hands dealt
- SIM security covers these concerns

Trust Issues Considered

- Protocol may leak a party's secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining
- Protocol may give adversary illegitimate influence on the outcome
  - Say in poker, if adversary can influence hands dealt
- SIM security covers these concerns
  - Because IDEAL trusted entity would allow neither

Adversary

Adversary

- REAL-adversary can corrupt any set of players

Adversary

- REAL-adversary can corrupt any set of players
  - In security requirement IDEAL-world adversary should corrupt the same set of players

Adversary

- REAL-adversary can corrupt any set of players
  - In security requirement IDEAL-world adversary should corrupt the same set of players
    - i.e., environment gets to know the set of corrupt players

Adversary

- REAL-adversary can corrupt any set of players
  - In security requirement IDEAL-world adversary should corrupt the same set of players
    - i.e., environment gets to know the set of corrupt players
- More sophisticated notion: adaptive adversary which corrupts players dynamically during/after the execution

Adversary

- REAL-adversary can corrupt any set of players
  - In security requirement IDEAL-world adversary should corrupt the same set of players
    - i.e., environment gets to know the set of corrupt players
- More sophisticated notion: adaptive adversary which corrupts players dynamically during/after the execution
  - We'll stick to static adversaries

Adversary

- REAL-adversary can corrupt any set of players
    - In security requirement IDEAL-world adversary should corrupt the same set of players
        - i.e., environment gets to know the set of corrupt players
- More sophisticated notion: adaptive adversary which corrupts players dynamically during/after the execution
    - We'll stick to static adversaries
- Passive vs. Active adversary: Passive adversary gets only read access to the internal state of the corrupted players. Active adversary overwrites their state and program.

Passive Adversary

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively
- Simplifies several cases

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively
- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively
- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]
- Oddly, sometimes security against a passive adversary is more demanding than against an active adversary

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively
- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]
- Oddly, sometimes security against a passive adversary is more demanding than against an active adversary
  - Active adversary: too pessimistic about what guarantee is available even in the IDEAL world

Passive Adversary

- Gets only read access to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively
- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]
- Oddly, sometimes security against a passive adversary is more demanding than against an active adversary
  - Active adversary: too pessimistic about what guarantee is available even in the IDEAL world
  - e.g. 2-party SFE for OR, with output going to only one party (trivial against active adversary; impossible without computational assumptions against passive adversary)

Example Functionalities

Example Functionalities

- Can consider “arbitrary” functionalities

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
  - Secure Function Evaluation

Example Functionalities

- Can consider “arbitrary” functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
  - Secure Function Evaluation
    - e.g. Oblivious Transfer (coming up)

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
  - Secure Function Evaluation
    - e.g. Oblivious Transfer (coming up)
    - Can be randomized: e.g. Coin-tossing

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
  - Secure Function Evaluation
    - e.g. Oblivious Transfer (coming up)
    - Can be randomized: e.g. Coin-tossing
  - "Reactive" functionalities (maintains state over multiple rounds)

Example Functionalities

- Can consider "arbitrary" functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
  - Secure Function Evaluation
    - e.g. Oblivious Transfer (coming up)
    - Can be randomized: e.g. Coin-tossing
  - "Reactive" functionalities (maintains state over multiple rounds)
    - e.g. Commitment (coming up)

Commitment

Commitment

- Commit now, reveal later

Commitment

- Commit now, reveal later
    - Intuitive properties: hiding and binding

Commitment

IDEAL World

- Commit now, reveal later
  - Intuitive properties: hiding and binding

Commitment

IDEAL World

- Commit now, reveal later
  - Intuitive properties: hiding and binding

Commitment

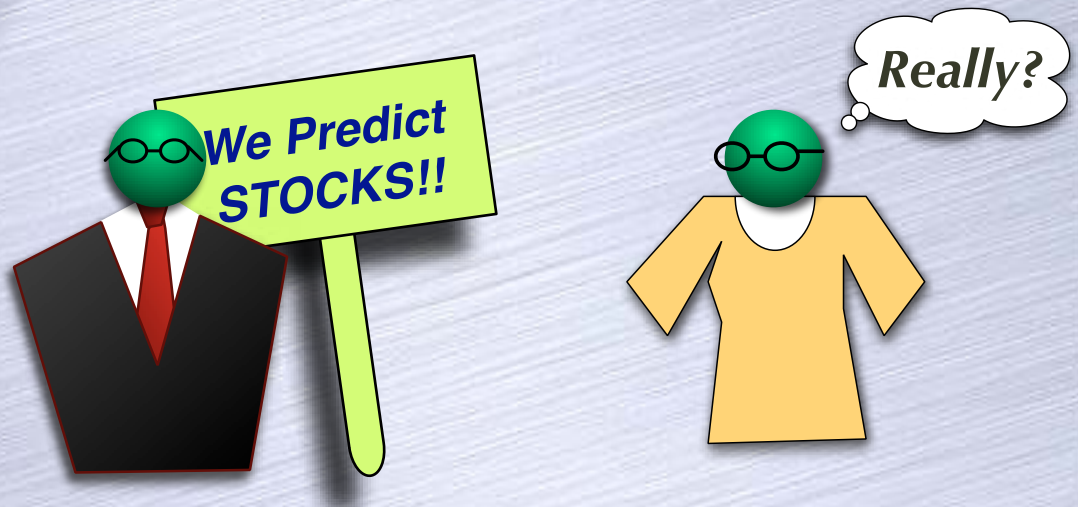
- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

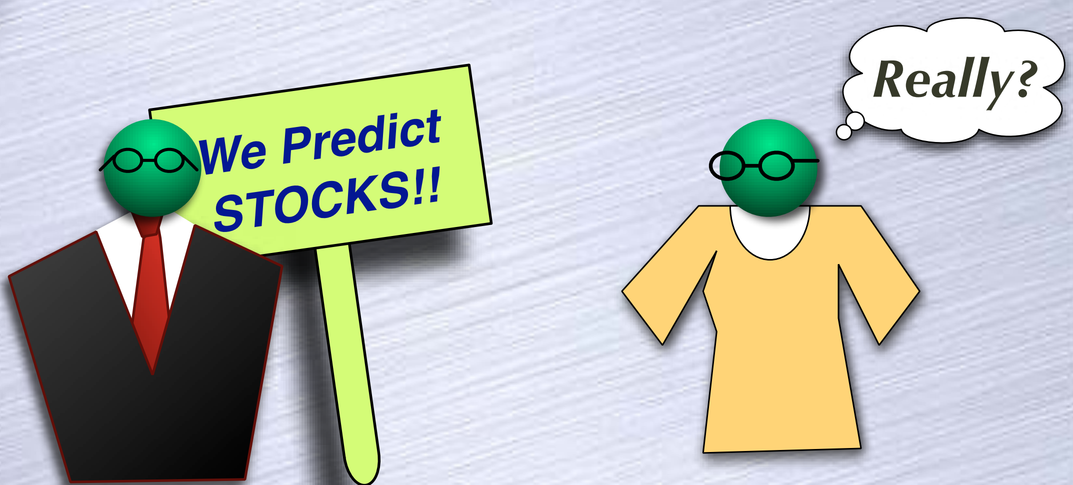


Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

30 Day Free Trial

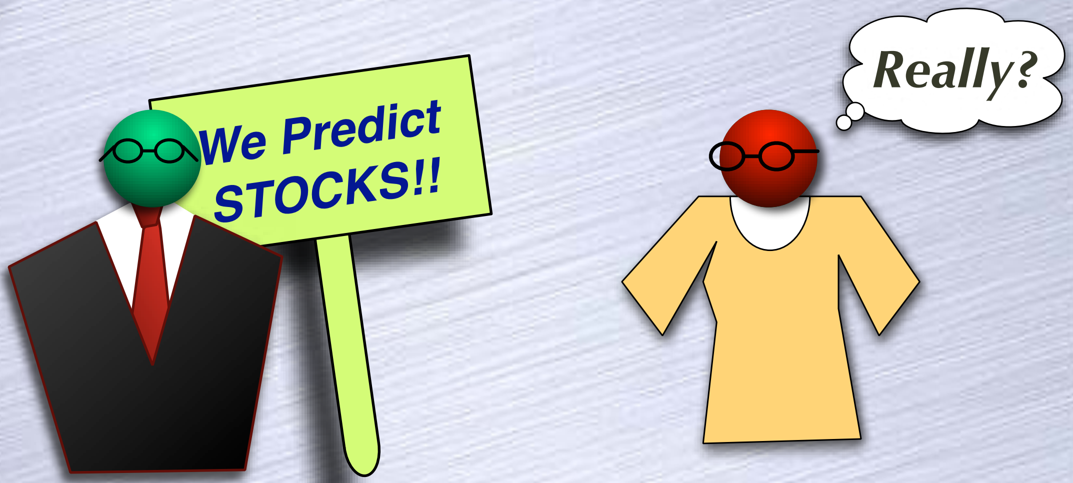


Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

30 Day Free Trial



Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

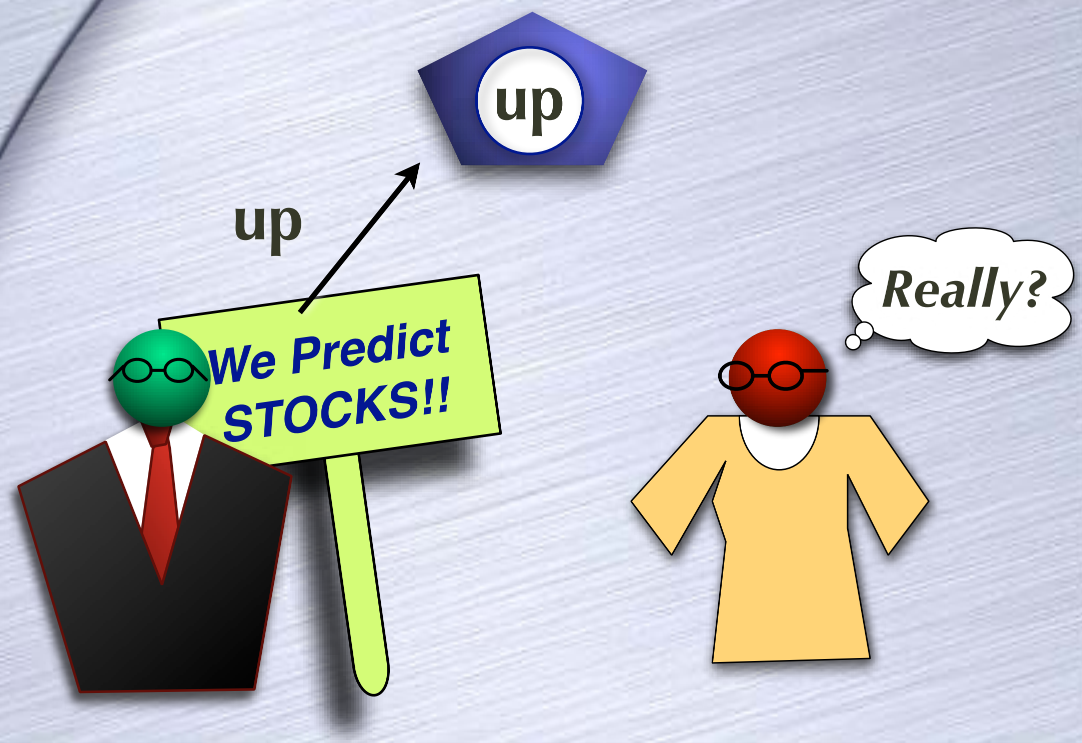
30 Day Free Trial

Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

IDEAL World

30 Day Free Trial

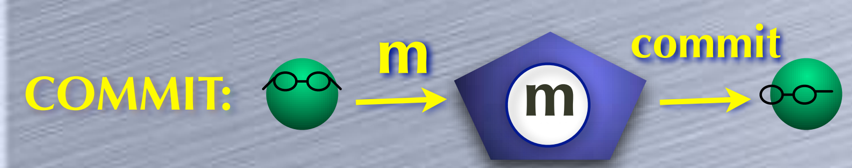


Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

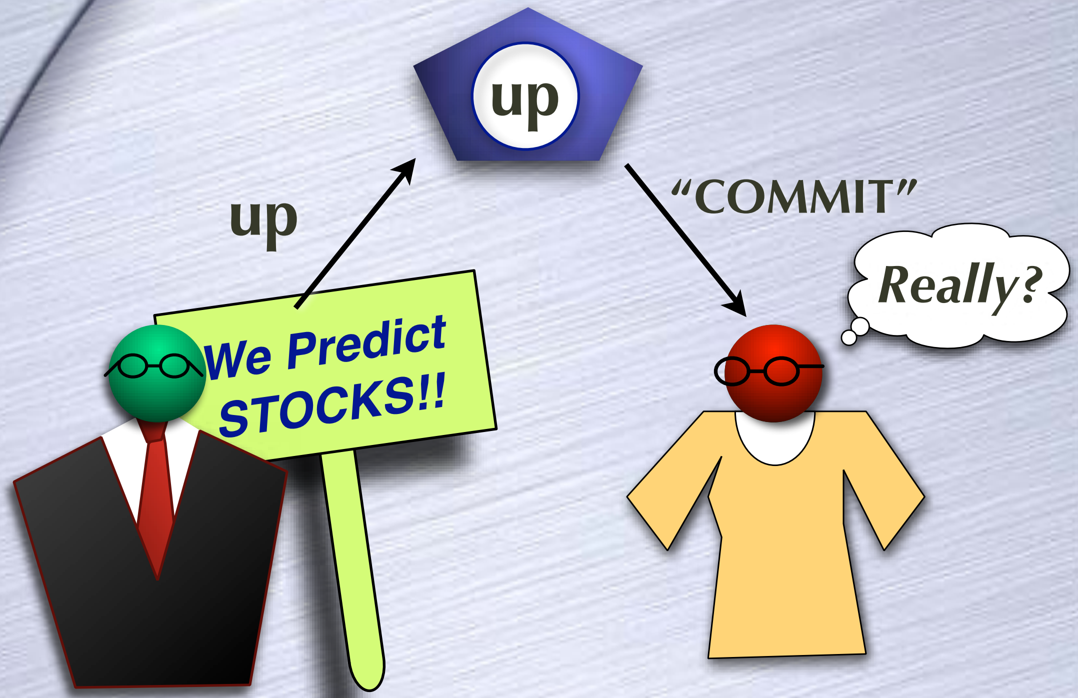
Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding



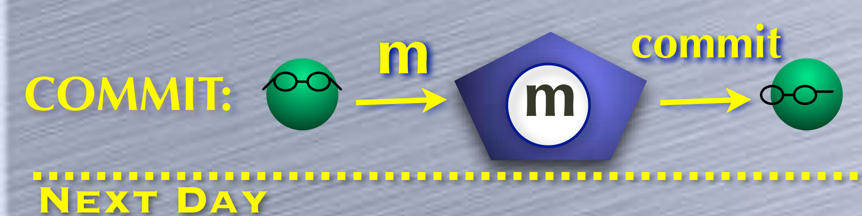
IDEAL World

30 Day Free Trial



Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding

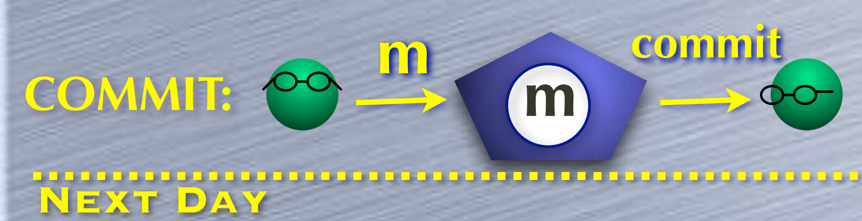


IDEAL World

30 Day Free Trial

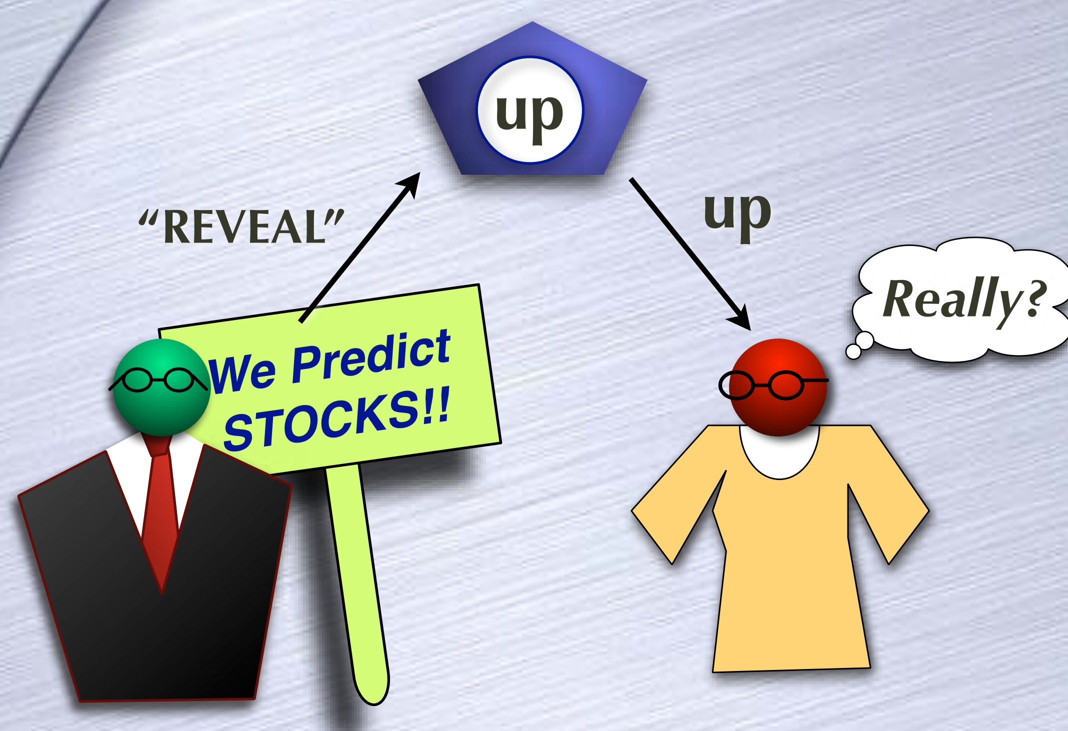
Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding



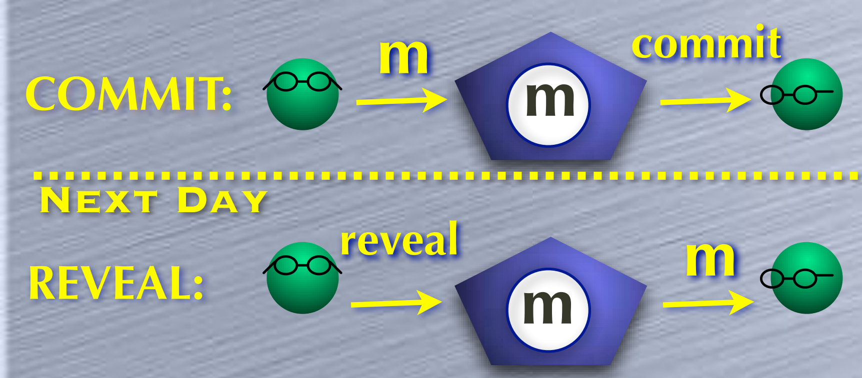
IDEAL World

30 Day Free Trial



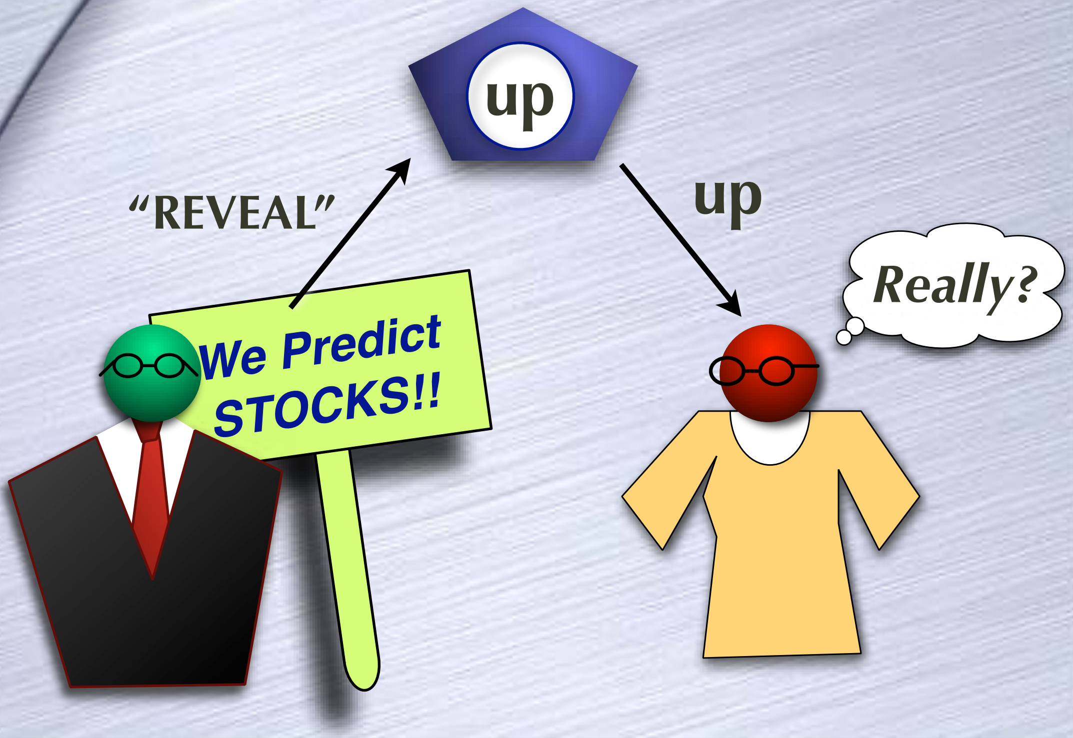
Commitment

- Commit now, reveal later
  - Intuitive properties: hiding and binding



IDEAL World

30 Day Free Trial



Oblivious Transfer

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

IDEAL World

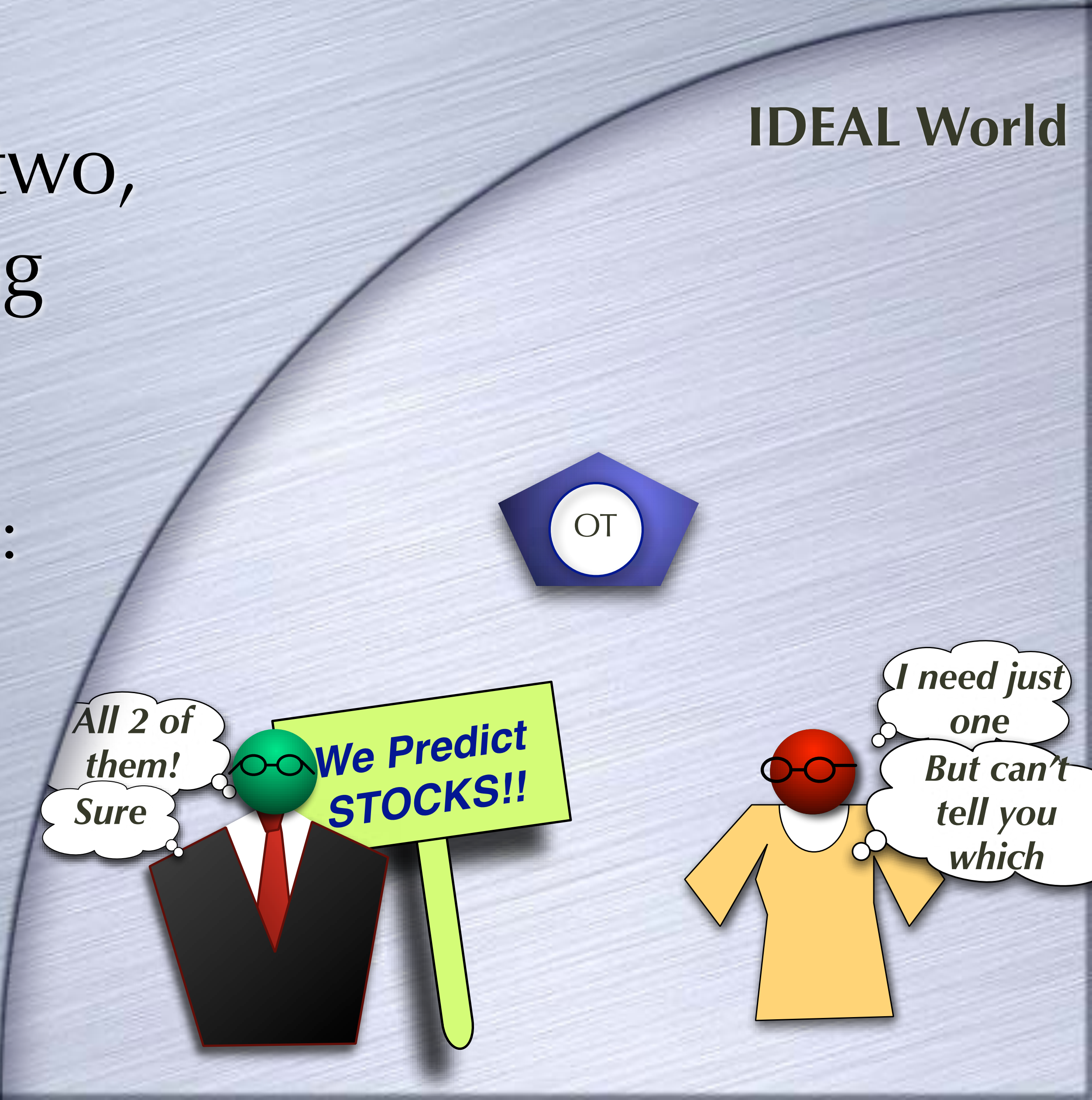
Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

IDEAL World

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

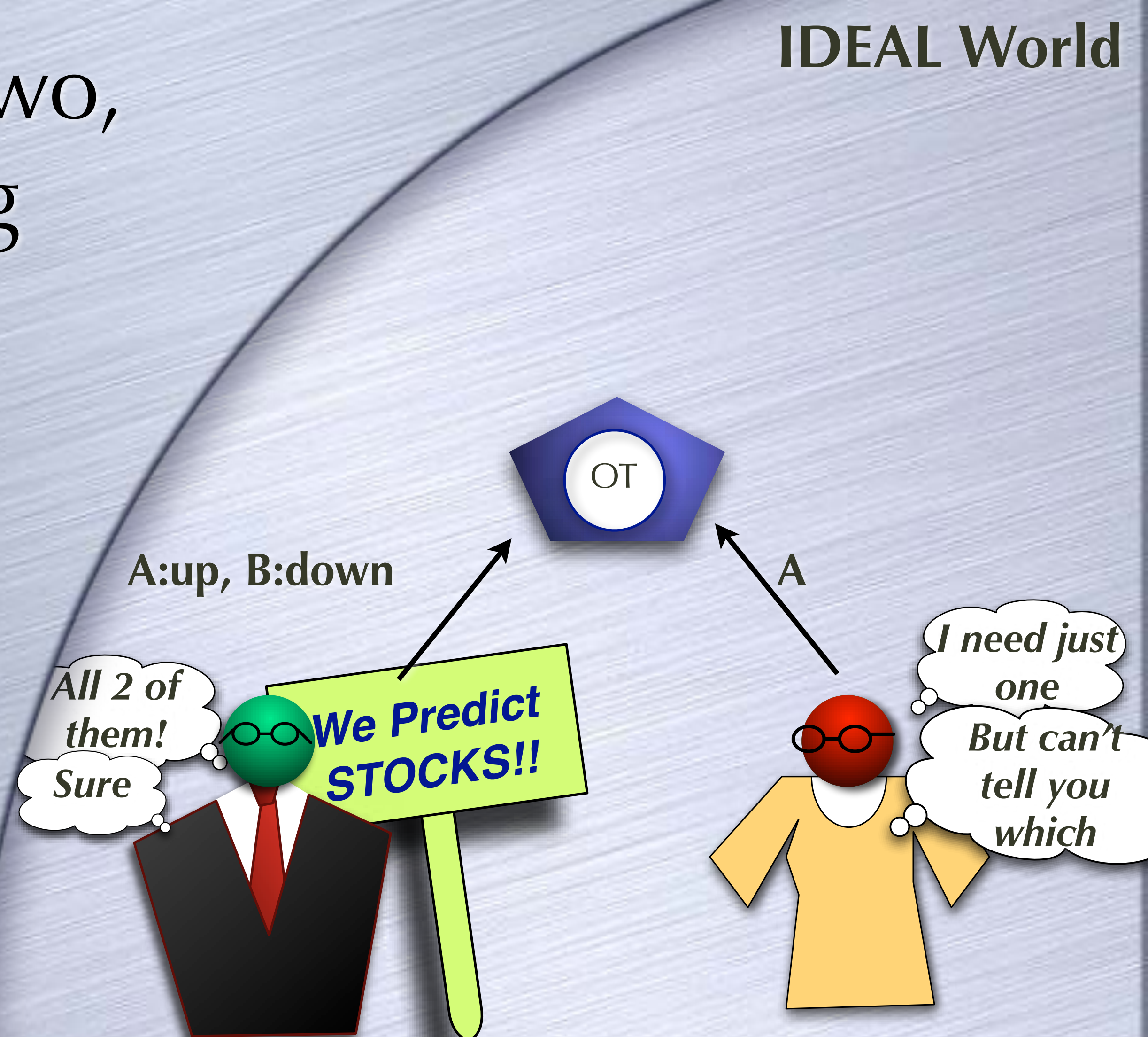


Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

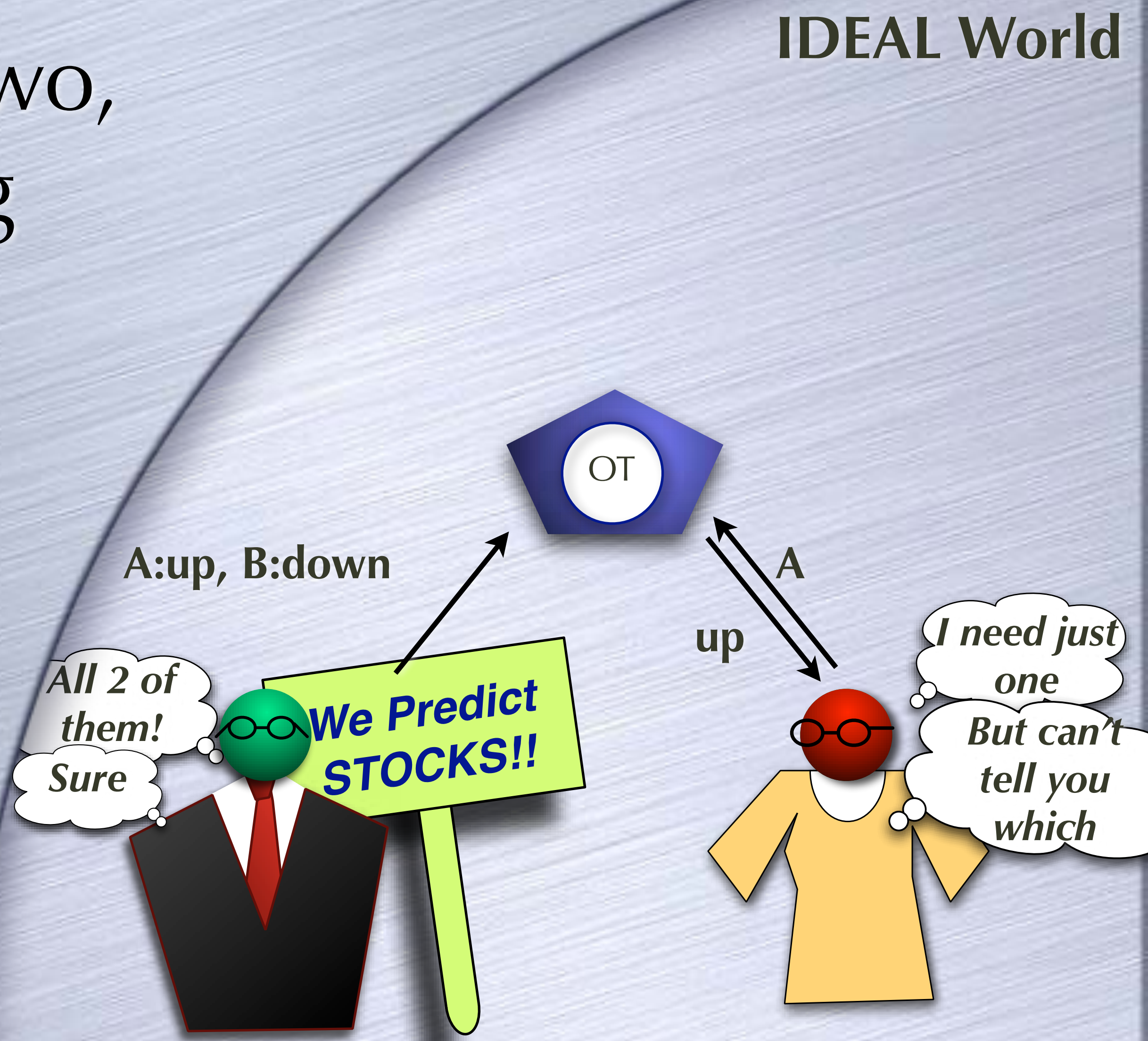
Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"



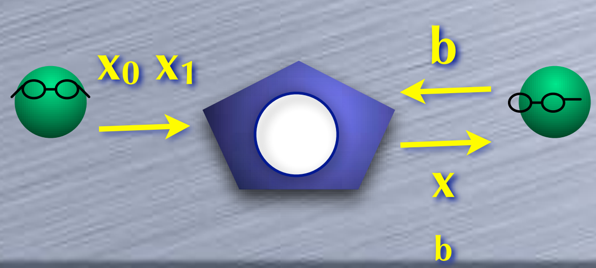
Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"

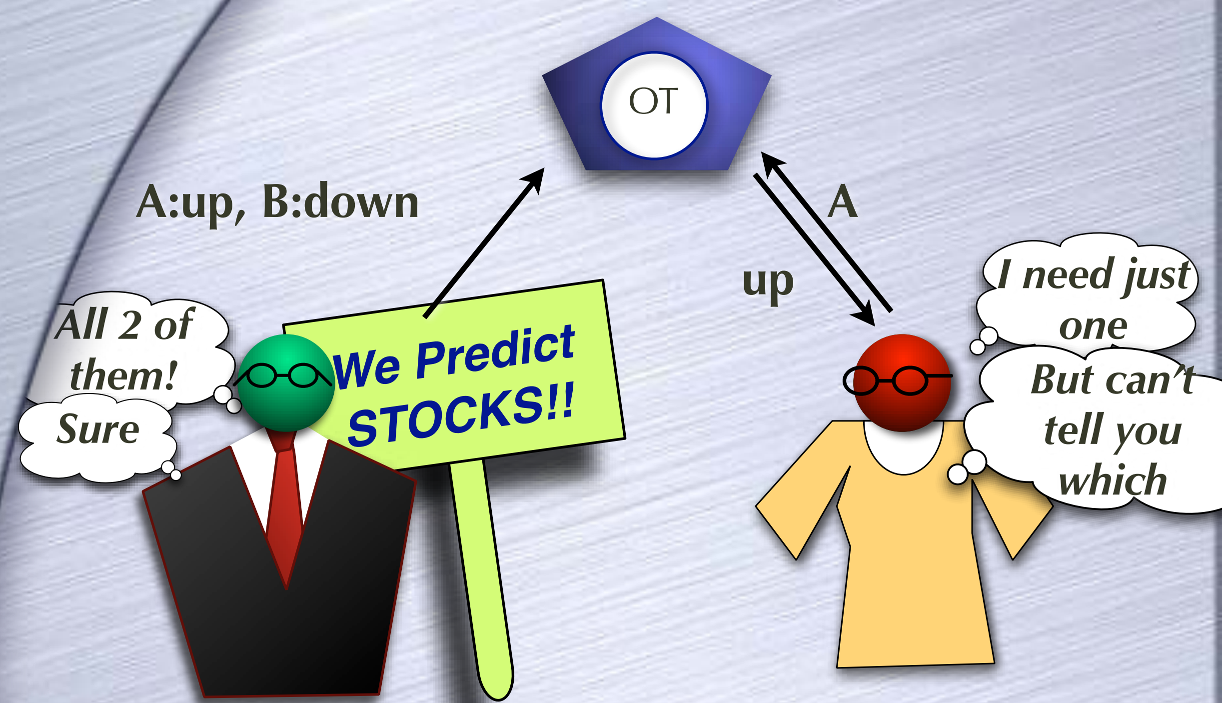


Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"



IDEAL World



Can we REAL-ize them?

Can we REAL-ize them?

- Are there protocols which securely realize these functionalities?

Can we REAL-ize them?

- Are there protocols which securely realize these functionalities?
  - Securely Realize: A protocol for the REAL world, so that SIM security definition satisfied

Can we REAL-ize them?

- Are there protocols which securely realize these functionalities?
  - Securely Realize: A protocol for the REAL world, so that SIM security definition satisfied
- Turns out SIM definition "too strong"

Can we REAL-ize them?

- Are there protocols which securely realize these functionalities?
    - Securely Realize: A protocol for the REAL world, so that SIM security definition satisfied
- Turns out SIM definition "too strong"
    - Unless modified carefully...

Alternate Security Definitions

Alternate Security Definitions

- Standalone security: environment is not "live": interacts with the adversary before and after (but not during) the protocol

Alternate Security Definitions

- Standalone security: environment is not "live": interacts with the adversary before and after (but not during) the protocol
- Honest-majority security: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)

Alternate Security Definitions

- Standalone security: environment is not "live": interacts with the adversary before and after (but not during) the protocol
- Honest-majority security: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)
- Passive (a.k.a honest-but-curious) adversary: where corrupt parties stick to the protocol (but we don't want to trust them with information)

Alternate Security Definitions

- Standalone security: environment is not “live”: interacts with the adversary before and after (but not during) the protocol
- Honest-majority security: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)
- Passive (a.k.a honest-but-curious) adversary: where corrupt parties stick to the protocol (but we don’t want to trust them with information)
- Functionality-specific IND definitions: usually leave out several attacks (e.g. malleability related attacks)

Alternate Security Definitions

- Standalone security: environment is not "live": interacts with the adversary before and after (but not during) the protocol
- Honest-majority security: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)
- Passive (a.k.a honest-but-curious) adversary: where corrupt parties stick to the protocol (but we don't want to trust them with information)
- Functionality-specific IND definitions: usually leave out several attacks (e.g. malleability related attacks)
- Protocols on top of a real trusted entity for a basic functionality

Alternate Security Definitions

- Standalone security: environment is not "live": interacts with the adversary before and after (but not during) the protocol
- Honest-majority security: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)
- Passive (a.k.a honest-but-curious) adversary: where corrupt parties stick to the protocol (but we don't want to trust them with information)
- Functionality-specific IND definitions: usually leave out several attacks (e.g. malleability related attacks)
- Protocols on top of a real trusted entity for a basic functionality
- Modified SIM definitions (super-PPT adversary for ideal world)

2-Party Secure Function Evaluation

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$
- Symmetric SFE: both parties get the same output

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$
- Symmetric SFE: both parties get the same output
  - e.g. $f(x_0,x_1;b,z) = g(x_0,x_1;b,z) = x_b \oplus z$ [OT from this! How?]

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$
- Symmetric SFE: both parties get the same output
  - e.g. $f(x_0,x_1;b,z) = g(x_0,x_1;b,z) = x_b \oplus z$ [OT from this! How?]
- More generally, any SFE from an appropriate symmetric SFE

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$
- Symmetric SFE: both parties get the same output
  - e.g. $f(x_0,x_1;b,z) = g(x_0,x_1;b,z) = x_b \oplus z$ [OT from this! How?]
- More generally, any SFE from an appropriate symmetric SFE
  - i.e., there is a protocol securely realizing SFE functionality G, which accesses a trusted party providing some symmetric SFE functionality F

2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - $f(x_0,x_1;b)$ = none; $g(x_0,x_1;b) = x_b$
- Symmetric SFE: both parties get the same output
  - e.g. $f(x_0,x_1;b,z) = g(x_0,x_1;b,z) = x_b \oplus z$ [OT from this! How?]
- More generally, any SFE from an appropriate symmetric SFE
  - i.e., there is a protocol securely realizing SFE functionality G, which accesses a trusted party providing some symmetric SFE functionality F
    - Exercise

2-Party Secure Function Evaluation

2-Party Secure Function Evaluation

- Randomized Functions: $f(X;Y;r)$

2-Party Secure Function Evaluation

- Randomized Functions: $f(X;Y;r)$
  - r is chosen randomly by the trusted party

2-Party Secure Function Evaluation

- Randomized Functions: f(X;Y;r)
  - r is chosen randomly by the trusted party
  - Neither party should know r (beyond what is revealed by output)

2-Party Secure Function Evaluation

- Randomized Functions: $f(X;Y;r)$
  - r is chosen randomly by the trusted party
  - Neither party should know r (beyond what is revealed by output)
  - Consider evaluating $f'(X,a;Y,b) := f(X;Y;a \oplus b)$

2-Party Secure Function Evaluation

- Randomized Functions: $f(X;Y;r)$
  - r is chosen randomly by the trusted party
  - Neither party should know r (beyond what is revealed by output)
  - Consider evaluating $f'(X,a;Y,b) := f(X;Y;a \oplus b)$
    - Note f' is deterministic

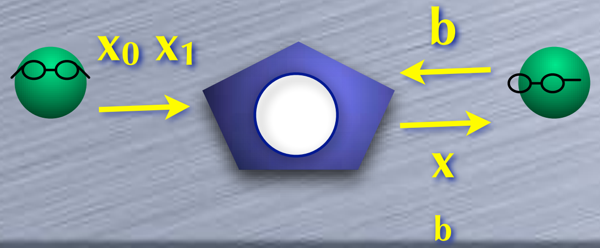
2-Party Secure Function Evaluation

- Randomized Functions: $f(X;Y;r)$
  - r is chosen randomly by the trusted party
  - Neither party should know r (beyond what is revealed by output)
  - Consider evaluating $f'(X,a;Y,b) := f(X;Y;a\oplus b)$
    - Note f' is deterministic
    - If either a or b is random $a\oplus b$ is random and hidden from each party

2-Party Secure Function Evaluation

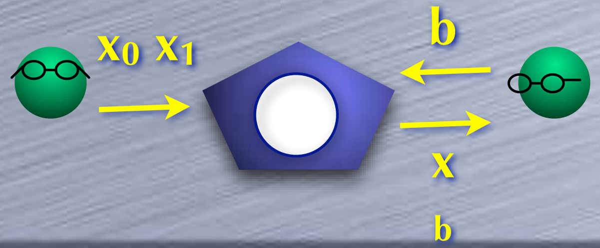
- Randomized Functions: $f(X;Y;r)$
  - r is chosen randomly by the trusted party
  - Neither party should know r (beyond what is revealed by output)
  - Consider evaluating $f'(X,a;Y,b) := f(X;Y;a\oplus b)$
    - Note f' is deterministic
    - If either a or b is random $a\oplus b$ is random and hidden from each party
    - Gives a protocol using access to f', to securely realize f Exercise

An OT Protocol
(passive receiver corruption)



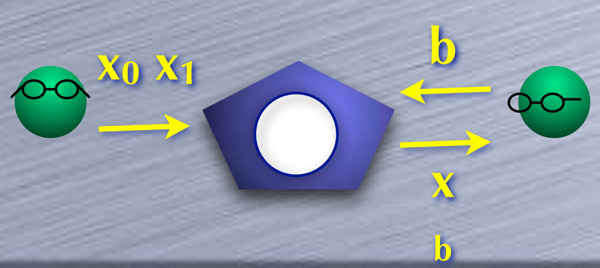
An OT Protocol
(passive receiver corruption)

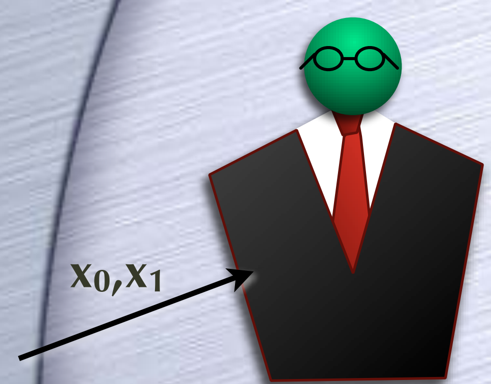
- Using a T-OWP

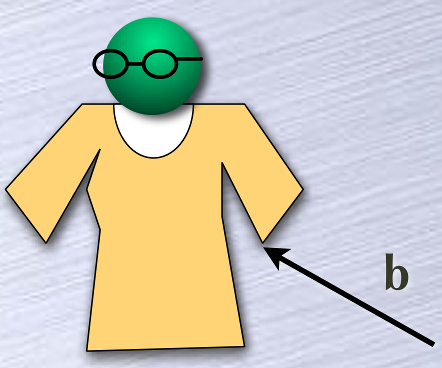


An OT Protocol
(passive receiver corruption)

- Using a T-OWP

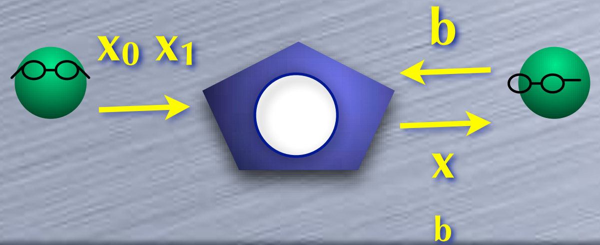


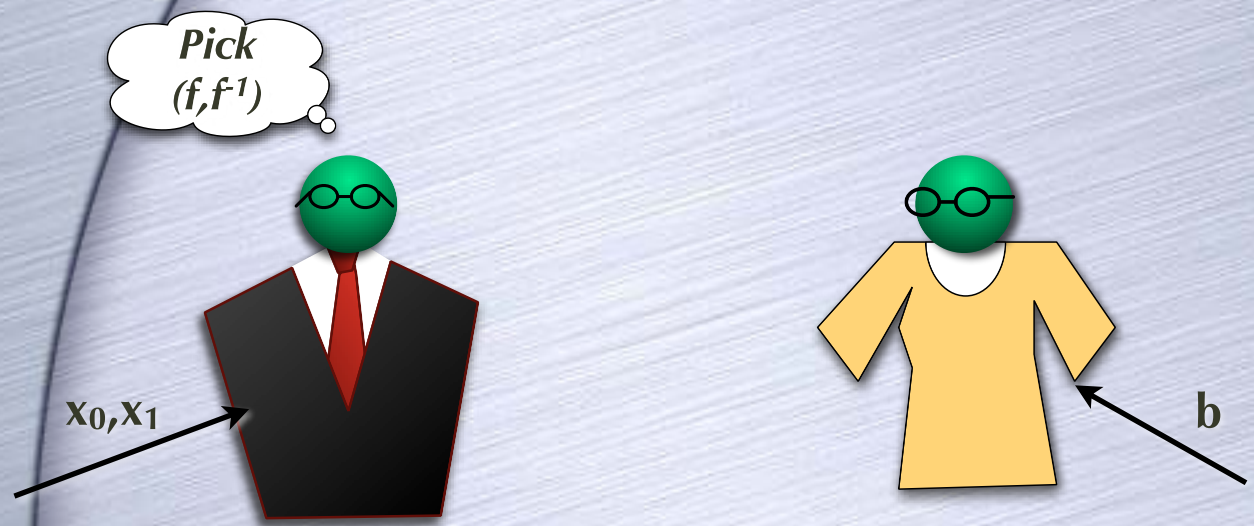




An OT Protocol
(passive receiver corruption)

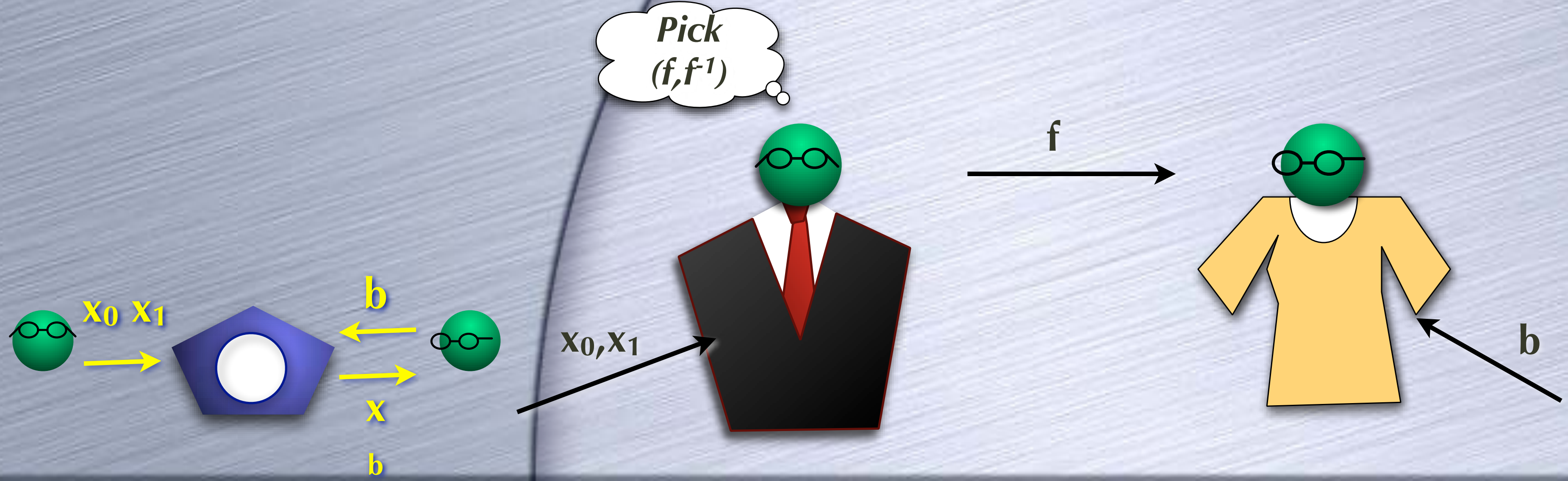
- Using a T-OWP





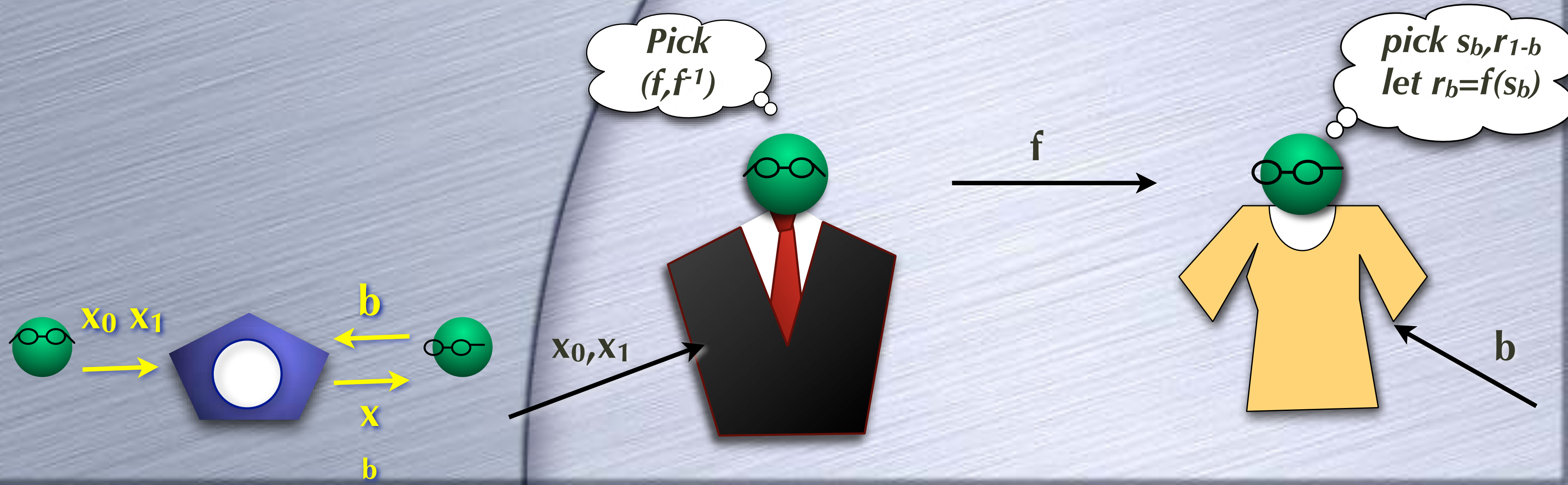
An OT Protocol
(passive receiver corruption)

- Using a T-OWP



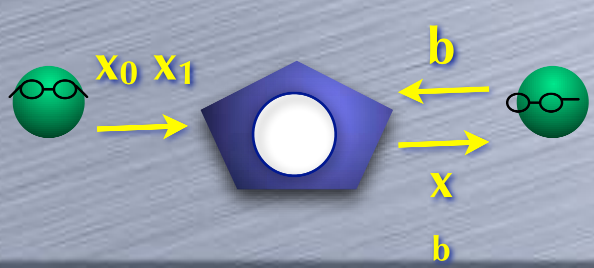
An OT Protocol
(passive receiver corruption)

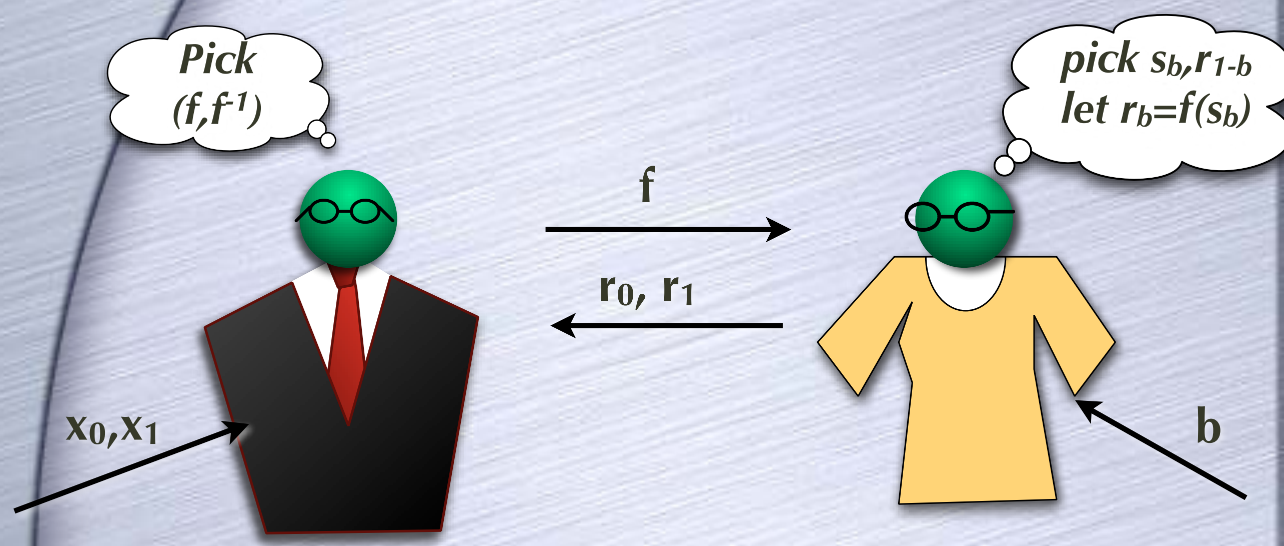
- Using a T-OWP



An OT Protocol
(passive receiver corruption)

- Using a T-OWP

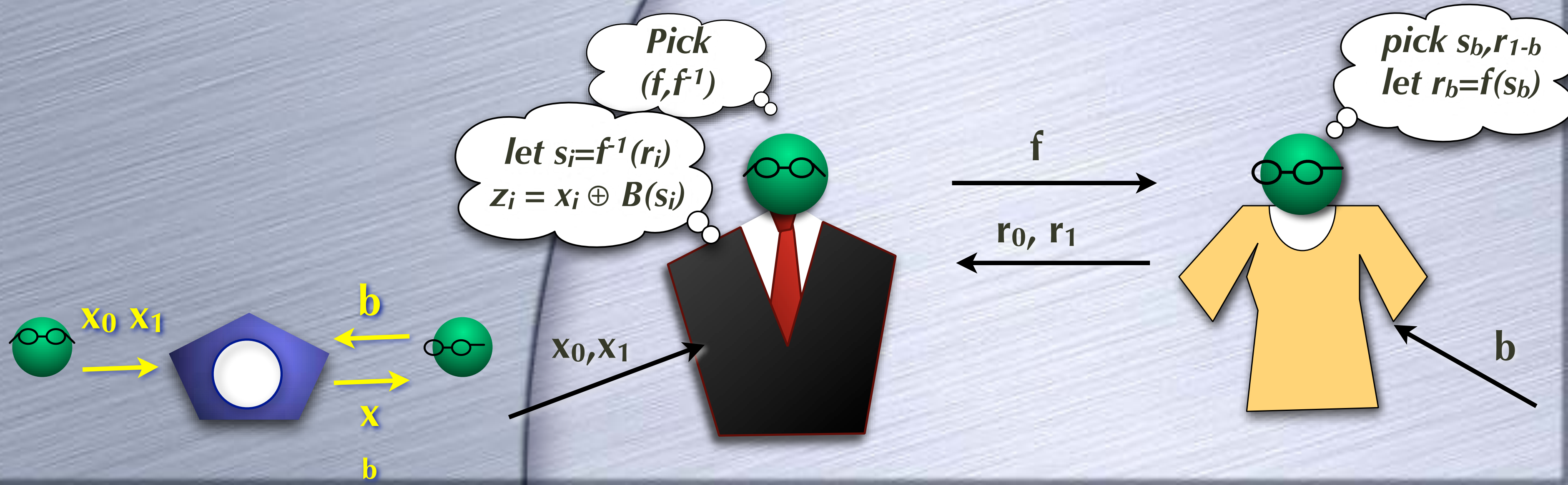




An OT Protocol

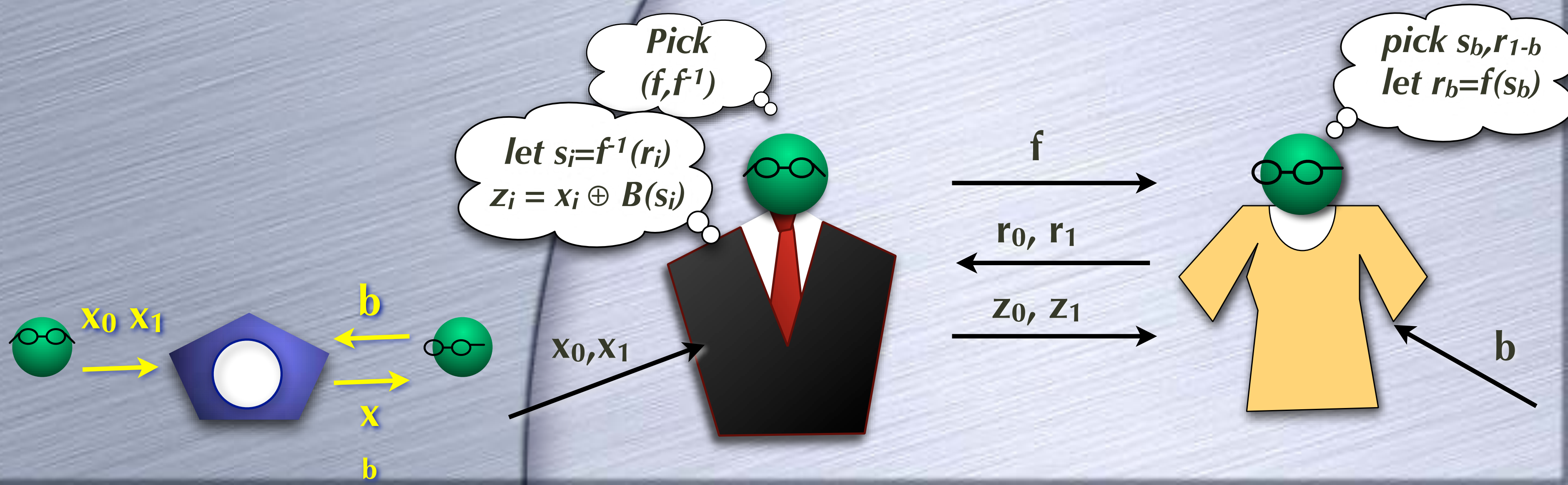
(passive receiver corruption)

- Using a T-OWP



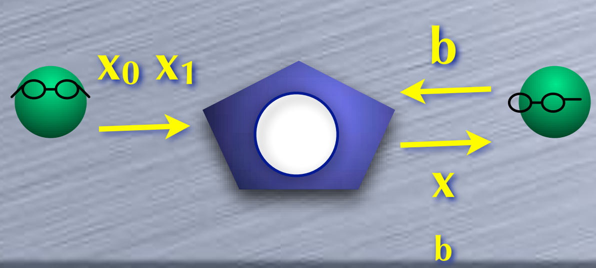
An OT Protocol
(passive receiver corruption)

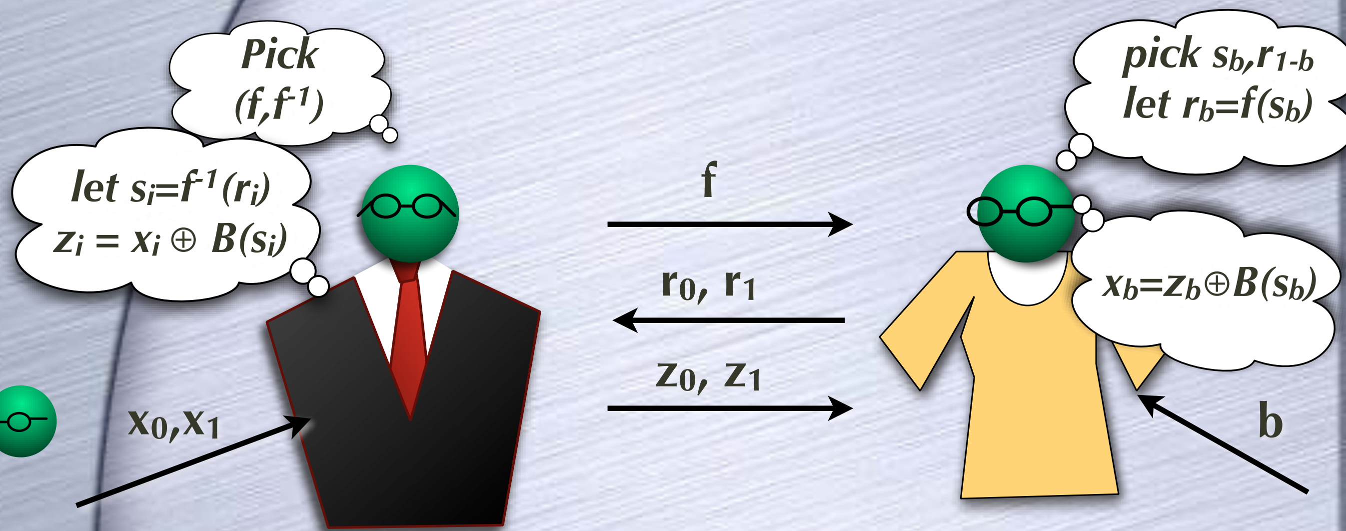
- Using a T-OWP



An OT Protocol
(passive receiver corruption)

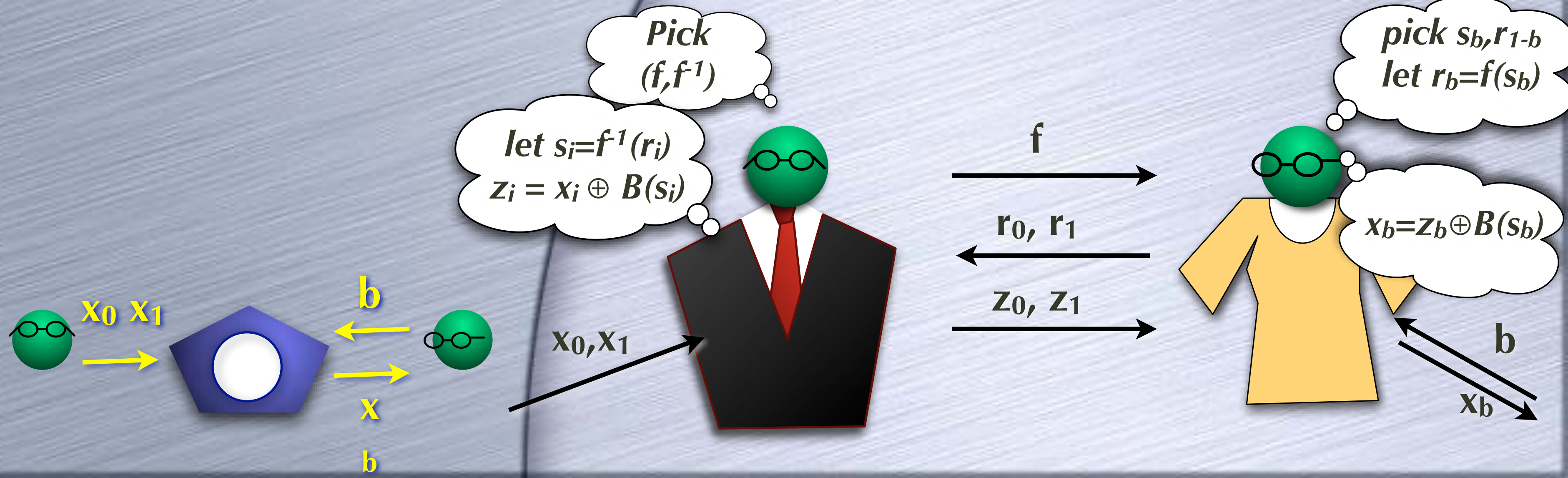
- Using a T-OWP





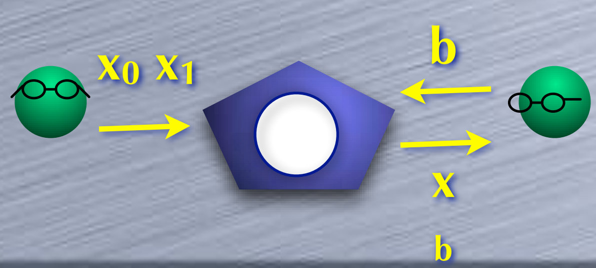
An OT Protocol
(passive receiver corruption)

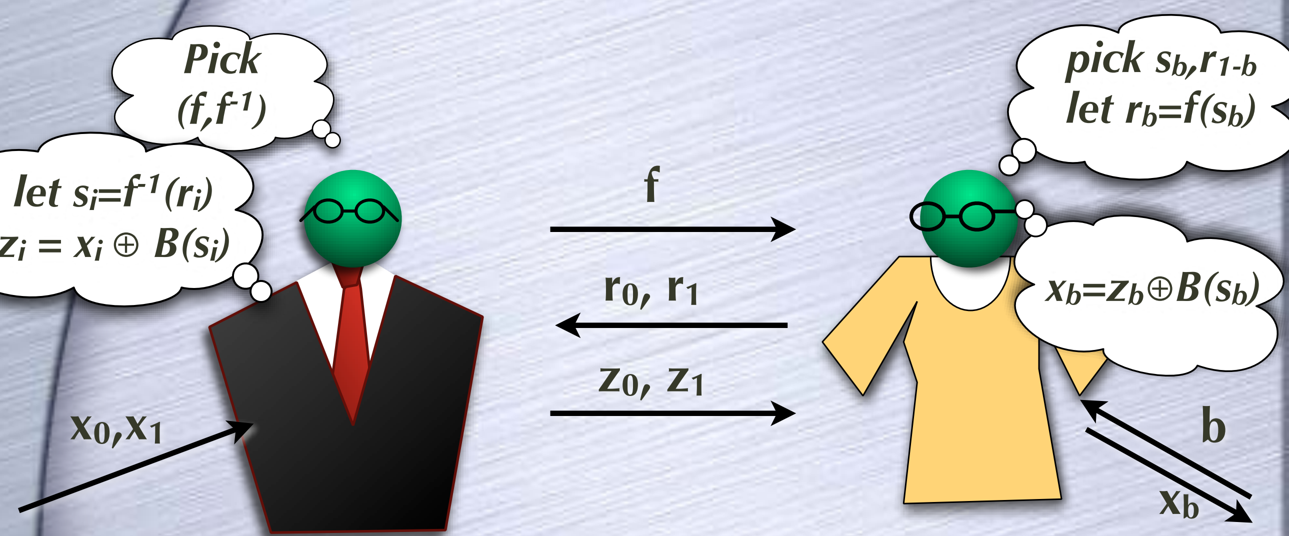
- Using a T-OWP



An OT Protocol
(passive receiver corruption)

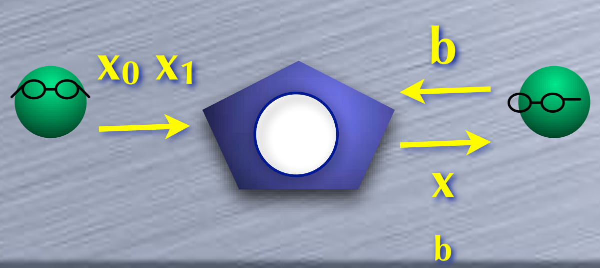
- Using a T-OWP
  - Depends on receiver to pick x_0, x_1 as prescribed

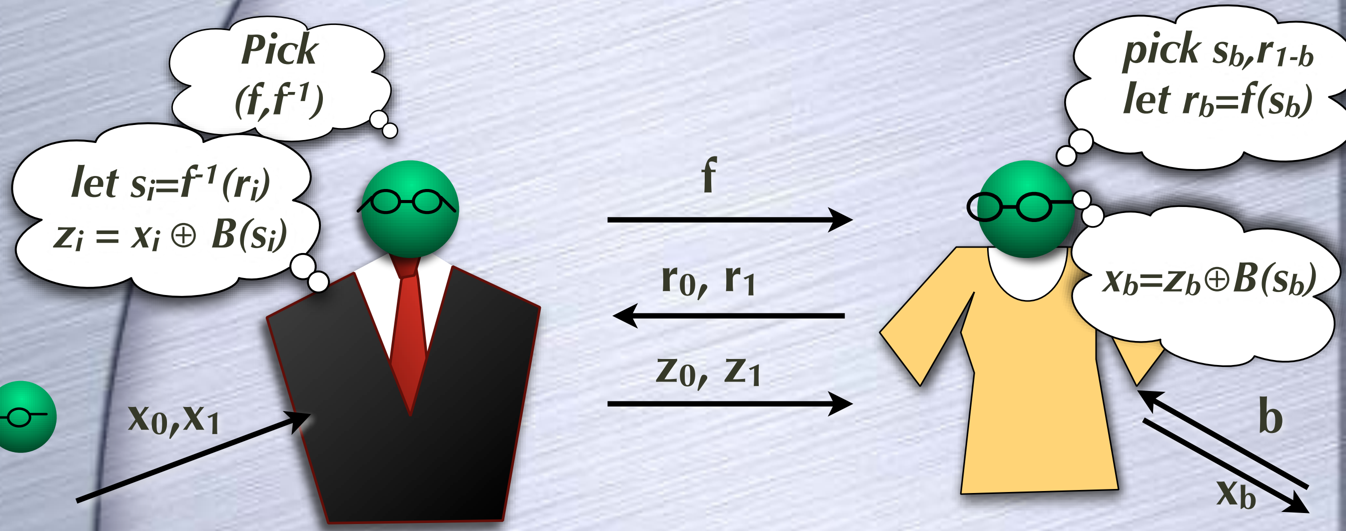




An OT Protocol
(passive receiver corruption)

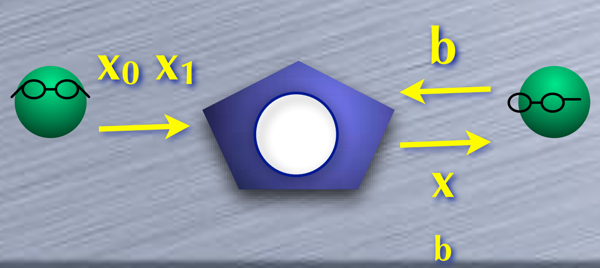
- Using a T-OWP
  - Depends on receiver to pick x_0, x_1 as prescribed
- Simulation for passive corrupt receiver: simulate z_0, z_1 knowing only x_b (use random z_{1-b})

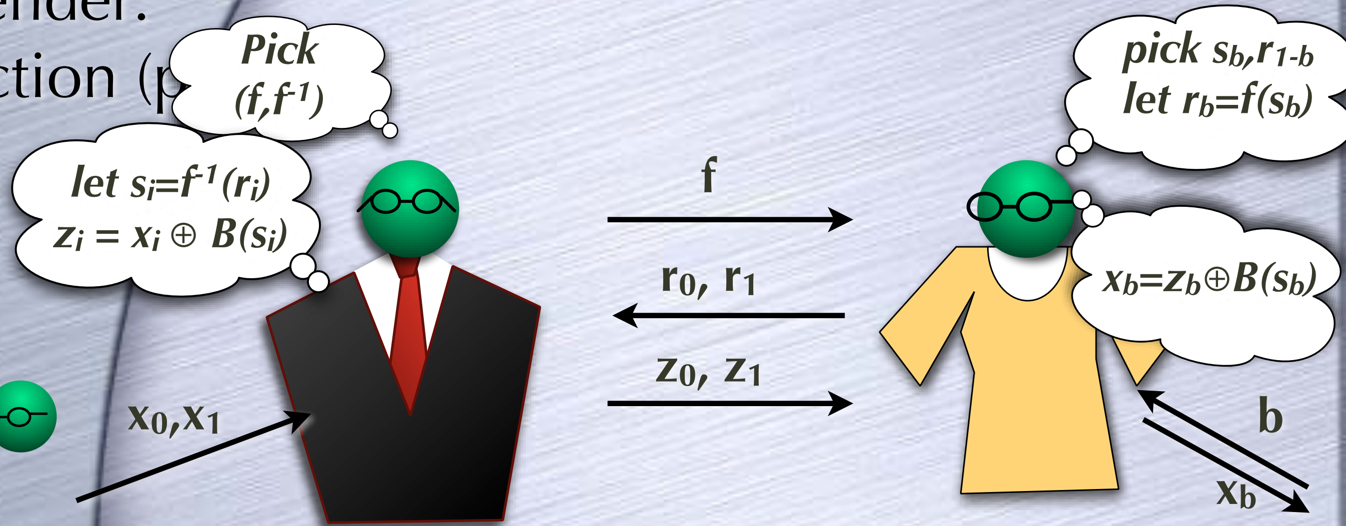




An OT Protocol (passive receiver corruption)

- Using a T-OWP
  - Depends on receiver to pick x_0, x_1 as prescribed
- Simulation for passive corrupt receiver: simulate z_0, z_1 knowing only x_b (use random z_{1-b})
- Simulation for corrupt sender: Extract x_0, x_1 from interaction (p
s_{1-b} also)





Today

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity
- Examples: poker, auction, privacy-preserving data-mining

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity
- Examples: poker, auction, privacy-preserving data-mining
- Basic Examples: SFE, Oblivious Transfer, Commitment

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity
- Examples: poker, auction, privacy-preserving data-mining
- Basic Examples: SFE, Oblivious Transfer, Commitment
- Weaker security requirements: security against passive (honest-but-curious) adversary, standalone security

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity
- Examples: poker, auction, privacy-preserving data-mining
- Basic Examples: SFE, Oblivious Transfer, Commitment
- Weaker security requirements: security against passive (honest-but-curious) adversary, standalone security
- Example of a protocol: OT secure against passive adversary

Today

- Secure MPC: formalized using IDEAL world with trusted computational entity
- Examples: poker, auction, privacy-preserving data-mining
- Basic Examples: SFE, Oblivious Transfer, Commitment
- Weaker security requirements: security against passive (honest-but-curious) adversary, standalone security
- Example of a protocol: OT secure against passive adversary
- Coming up: SFE protocols for passive security. Zero-Knowledge proofs. Issues of composition. Universal Composition.