Cryptographic Complexity of Multi-party Computation Problems: Classifications and Separations

Manoj Prabhakaran & Mike Rosulek

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

CRYPTO 2008 – August 19, 2008

Multi-Party Computation

Secure Multi-party Computation (MPC)

Parties engage in a protocol to securely accomplish some task, in the presence of adversaries.

Example tasks:

- Communication
- Function evaluation
- Zero-knowledge proof

Multi-Party Computation

Secure Multi-party Computation (MPC)

Parties engage in a protocol to securely accomplish some task, in the presence of adversaries.

Example tasks:

- Communication
- Function evaluation
- Zero-knowledge proof

Fundamental Question

Which MPC tasks have secure protocols? (answer depends on MPC model)

Universal Composition framework [Canetti 2000]

UC framework cast of characters (all PPT machines):

Universal Composition framework [Canetti 2000]

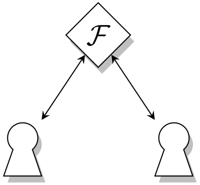
UC framework cast of characters (all PPT machines):

- Parties: components for doing a task

Universal Composition framework [Canetti 2000]

UC framework cast of characters (all PPT machines):

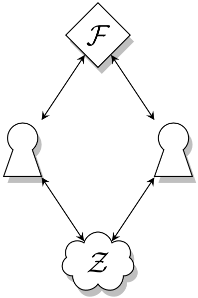
- Parties: components for doing a task
- Functionality: trusted party that performs some task



Universal Composition framework [Canetti 2000]

UC framework cast of characters (all PPT machines):

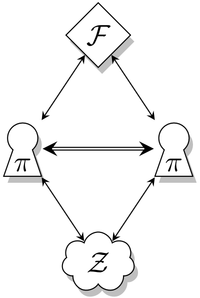
- Parties: components for doing a task
- Functionality: trusted party that performs some task
- Environment: supplies inputs, receives outputs from parties; outputs a bit



Universal Composition framework [Canetti 2000]

UC framework cast of characters (all PPT machines):

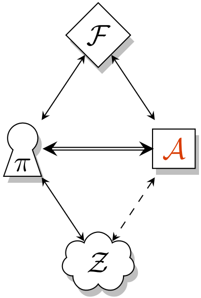
- Parties: components for doing a task
- Functionality: trusted party that performs some task
- Environment: supplies inputs, receives outputs from parties; outputs a bit
- Protocol: prescribes parties' interaction on channel



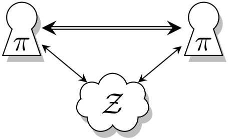
Universal Composition framework [Canetti 2000]

UC framework cast of characters (all PPT machines):

- Parties: components for doing a task
- Functionality: trusted party that performs some task
- Environment: supplies inputs, receives outputs from parties; outputs a bit
- Protocol: prescribes parties' interaction on channel
- Adversary: influences environment, corrupts parties

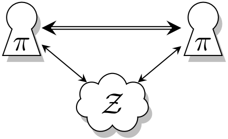


UC Definition of Security

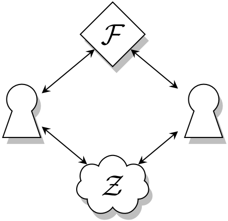


Real world interaction

UC Definition of Security

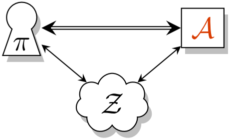


Real world interaction

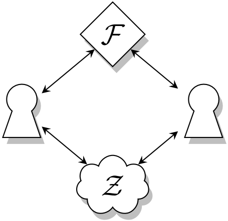


Ideal world interaction

UC Definition of Security



Real world interaction



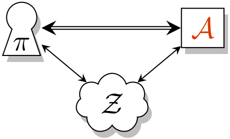
Ideal world interaction

Definition

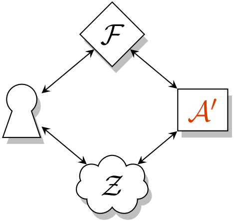
π is a secure realization of $\mathcal{F}$ if:

- For every *real-world* adversary $\mathcal{A}$

UC Definition of Security



Real world interaction



Ideal world interaction

Definition

π is a secure realization of $\mathcal{F}$ if:

- For every *real-world* adversary $\mathcal{A}$
- There exists an *ideal-world* adversary $\mathcal{A}'$

UC Definition of Security

Real world interaction

Ideal world interaction

Definition

π is a secure realization of $\mathcal{F}$ if:

- For every *real-world* adversary $\mathcal{A}$
- There exists an *ideal-world* adversary $\mathcal{A}'$
- Two worlds indistinguishable to *all* environments $\mathcal{Z}$

(Un)Feasibility Results in UC Framework

Main Question

Which functionalities are realizable in UC framework?

Ad hoc techniques (e.g., [C00,CF01]):

- Positive results (protocol constructions): e.g., can securely realize private channels using public channels
- Negative results (separations, impossibilities): bit commitment, ZK proofs, oblivious transfer, coin-tossing, etc...

(Un)Feasibility Results in UC Framework

Main Question

Which functionalities are realizable in UC framework?

Ad hoc techniques (e.g., [C00,CF01]):

- Positive results (protocol constructions): e.g., can securely realize private channels using public channels
- Negative results (separations, impossibilities): bit commitment, ZK proofs, oblivious transfer, coin-tossing, etc...

Towards more general-purpose techniques:

- Broad impossibility results for 2-party secure function evaluation [CKL03]
- Argument goes through even with certain "trusted set-up" functionalities [KL07]

Our Results

General-purpose tools to classify functionalities.

Our Results

General-purpose tools to classify functionalities.

Splittability:

- Complete characterization of realizability for *arbitrary* functionalities
- Can *easily* re-derive all previous UC impossibility results

Our Results

General-purpose tools to classify functionalities.

Splittability:

- Complete characterization of realizability for *arbitrary* functionalities
- Can *easily* re-derive all previous UC impossibility results

Deviation-revealing property:

- Relate separations from passive corruption model
- Can make distinctions among higher-complexity functionalities

Outline

Motivation: Man-in-the-Middle Attacks

Implicit in all previous impossibility results in UC framework:

- Undetectable man-in-the-middle attack on protocol

Motivation: Man-in-the-Middle Attacks

Implicit in all previous impossibility results in UC framework:

- Undetectable man-in-the-middle attack on protocol

Key Idea

Make man-in-the-middle attack explicit in the *ideal* world:

- Against 2 instances of the functionality, not protocol

2-Party Splittability

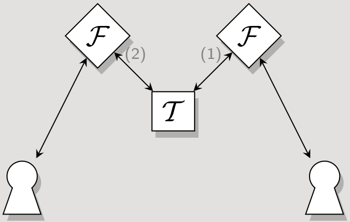
Definition

A functionality $\mathcal{F}$ is splittable if there exists $\mathcal{T}$ such that:

2-Party Splittability

Definition

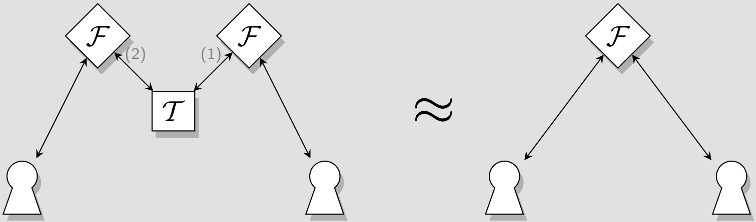
A functionality $\mathcal{F}$ is splittable if there exists $\mathcal{T}$ such that:



2-Party Splittability

Definition

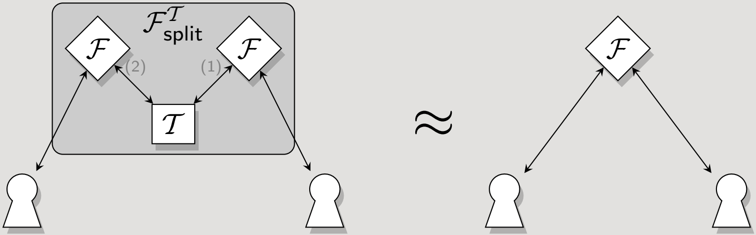
A functionality $\mathcal{F}$ is splittable if there exists $\mathcal{T}$ such that:



2-Party Splittability

Definition

A functionality $\mathcal{F}$ is splittable if there exists $\mathcal{T}$ such that:



Formally, if $\mathcal{F}$ and $\mathcal{F}^{\mathcal{T}}_{\text{split}}$ indistinguishable for all environments.

Example: Unsplittable Functionalities

Can be very easy to show unsplittability.

Example: Coin-tossing functionality

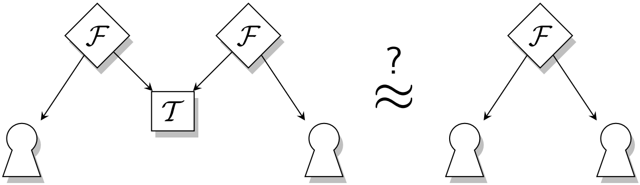
Choose random $b \leftarrow \{0,1\}$, send b to both parties.

Example: Unsplittable Functionalities

Can be very easy to show unsplittability.

Example: Coin-tossing functionality

Choose random $b \leftarrow \{0,1\}$, send b to both parties.

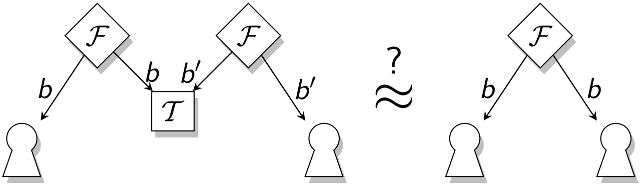


Example: Unsplittable Functionalities

Can be very easy to show unsplittability.

Example: Coin-tossing functionality

Choose random $b \leftarrow \{0,1\}$, send b to both parties.



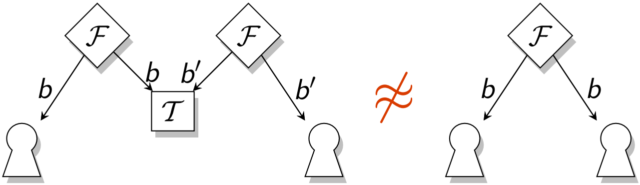
- Independent instances of $\mathcal{F}$ generate independent bits

Example: Unsplittable Functionalities

Can be very easy to show unsplittability.

Example: Coin-tossing functionality

Choose random $b \leftarrow \{0,1\}$, send b to both parties.



- Independent instances of $\mathcal{F}$ generate independent bits
- Environment can easily distinguish with probability $1/2$.

$\Longrightarrow$ Coin-tossing is not splittible.

Main Characterization

Also very easy to show unsplittable:

- Oblivious transfer
- Commitment
- ZK for NP $\setminus$ BPP

Main Characterization

Also very easy to show unsplittable:

- Oblivious transfer
- Commitment
- ZK for NP $\setminus$ BPP

Main Theorem

$\mathcal{F}$ splittable $\iff$ $\mathcal{F}$ securely realizable.

Also very easy to show unsplittable:

- Oblivious transfer
- Commitment
- ZK for NP $\setminus$ BPP

Main Theorem

$\mathcal{F}$ splittable $\iff$ $\mathcal{F}$ securely realizable.

Significance:

- First alternate characterization of realizability in UC model:
- Splittability defined in terms of black-box interactions with $\mathcal{F}$.
- $\mathcal{F}$ may be arbitrary (randomized, interactive, etc.)
- Can very easily re-derive essentially all UC impossibility results

Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

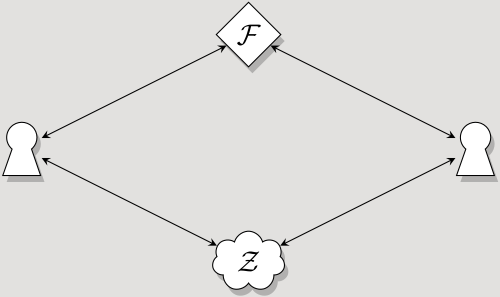
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



$\square$

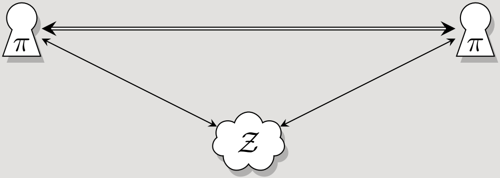
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



□

Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:

□

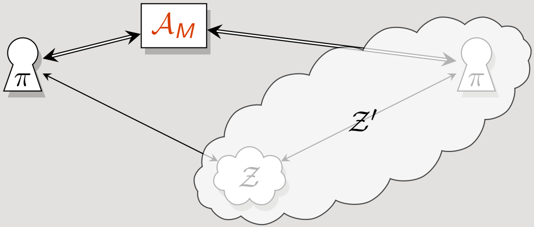
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



$\square$

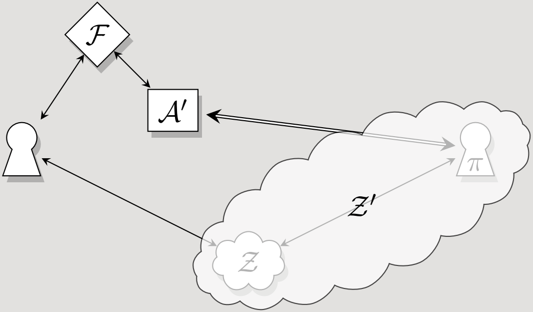
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



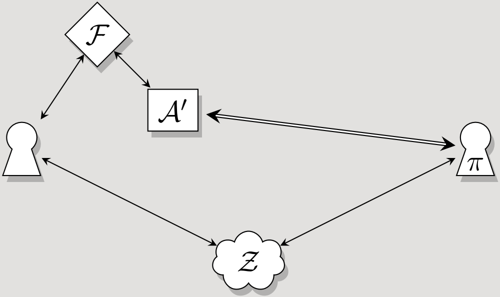
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



$\square$

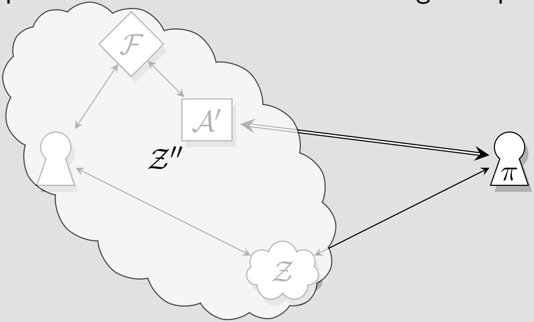
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



□

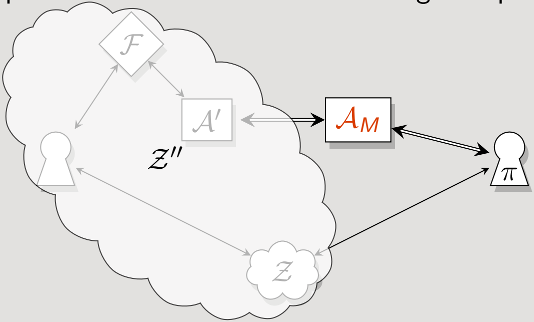
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



$\square$

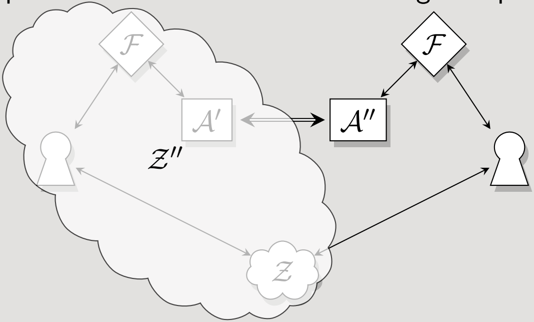
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\implies$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



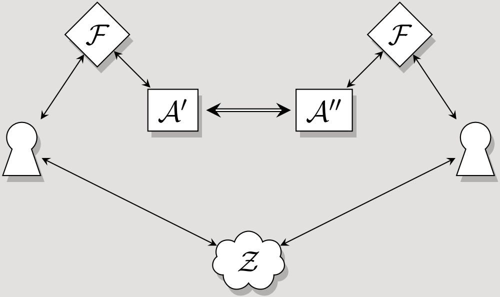
Proof Overview

Main Theorem (one direction)

$\mathcal{F}$ realizable $\Longrightarrow$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



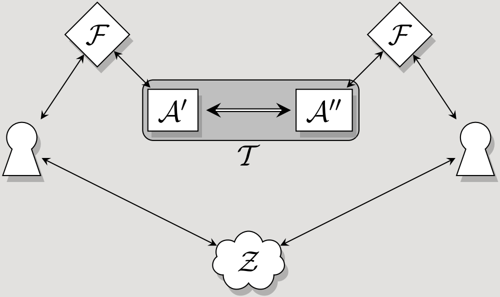
□

Main Theorem (one direction)

$\mathcal{F}$ realizable $\Longrightarrow$ $\mathcal{F}$ splittable

Proof Sketch (generalizes [CKL03] technique)

Let π be a protocol for $\mathcal{F}$. Want to derive a good splitting of $\mathcal{F}$:



□

Additional Results

Give *purely combinatorial* characterization for 2-party SFE.

- Completes and subsumes [CKL03]

Additional Results

Give *purely combinatorial* characterization for 2-party SFE.

- Completes and subsumes [CKL03]

Easy impossibility results also hold w.r.t. “set-up” functionalities.

- We characterize such set-up functionalities
- Completes and subsumes [KL07]

Additional Results

Give *purely combinatorial* characterization for 2-party SFE.

- Completes and subsumes [CKL03]

Easy impossibility results also hold w.r.t. "set-up" functionalities.

- We characterize such set-up functionalities
- Completes and subsumes [KL07]

Summary:

- Simple, unified paradigm for showing UC impossibility

Outline

Motivation: "Complexity Theory" for MPC

Have a characterization of realizability in UC framework, but..

- Can't make distinctions among "higher complexity" functionalities. (e.g., ZK, OT, commitment)

Motivation: “Complexity Theory” for MPC

Have a characterization of realizability in UC framework, but..

- Can’t make distinctions among “higher complexity” functionalities. (e.g., ZK, OT, commitment)

Idea: Use Reductions

Say $\mathcal{F}$ reduces to $\mathcal{G}$ if there is a secure protocol for $\mathcal{F}$ that uses $\mathcal{G}$ as a black box.

- Easy to model in UC framework (hybrid world)
- Transitive relation in UC framework

Motivation: "Complexity Theory" for MPC

Have a characterization of realizability in UC framework, but..

- Can't make distinctions among "higher complexity" functionalities. (e.g., ZK, OT, commitment)

Idea: Use Reductions

Say $\mathcal{F}$ reduces to $\mathcal{G}$ if there is a secure protocol for $\mathcal{F}$ that uses $\mathcal{G}$ as a black box.

- Easy to model in UC framework (hybrid world)
- Transitive relation in UC framework

Goal:

- Build "complexity theory" for MPC tasks
- Understand structure of high-complexity tasks

Motivation: Leveraging Other MPC Models

Can we leverage results from other MPC models?

- e.g., passive corruptions

Motivation: Leveraging Other MPC Models

Can we leverage results from other MPC models?

- e.g., passive corruptions

Quiz

If there is a UC protocol for $\mathcal{F}$, is there necessarily a protocol for $\mathcal{F}$ secure against passive corruptions?

Motivation: Leveraging Other MPC Models

Can we leverage results from other MPC models?

- e.g., passive corruptions

Quiz

If there is a UC protocol for $\mathcal{F}$, is there necessarily a protocol for $\mathcal{F}$ secure against passive corruptions? No!

Counterexample $\mathcal{F}$: Receive bits x, y from Alice, Bob. Give $x \vee y$ to Bob.

- No protocol in passive model (unbounded parties)
- Secure UC protocol: Alice sends x to Bob
  - Bob could learn x in ideal world by sending $y = 0$.

Deviation-Revealing

Boolean-OR functionality allowed Bob to “cheat” in ideal world!

- Not all functionalities are so weird.

Deviation-Revealing

Boolean-OR functionality allowed Bob to "cheat" in ideal world!

- Not all functionalities are so weird.

Definition

$\mathcal{F}$ is deviation-revealing if an environment can tell whether an adversary deviates from honest ideal-world behavior

Deviation-Revealing

Boolean-OR functionality allowed Bob to "cheat" in ideal world!

- Not all functionalities are so weird.

Definition

$\mathcal{F}$ is deviation-revealing if an environment can tell whether an adversary deviates from honest ideal-world behavior

Notes:

- Property of $\mathcal{F}$, not the protocol!
- Definition applies to arbitrary $\mathcal{F}$

Main Theorem

Theorem

When $\mathcal{F}$ is deviation-revealing, then separations in passive model imply separations in UC model.

Many separations already known for passive corruptions, with unbounded parties.

- Can now be translated to separations in UC model.

Main Theorem

Theorem

When $\mathcal{F}$ is deviation-revealing, then separations in passive model imply separations in UC model.

Many separations already known for passive corruptions, with unbounded parties.

- Can now be translated to separations in UC model.

Applying this new technique in unbounded UC model:

- Can identify several intermediate levels of complexity
- Neither realizable nor complete

Outline

Summary

New tools for analyzing complexity of UC functionalities:

- Apply to completely arbitrary functionalities

Apply new tools to obtain:

- Complete characterization of UC realizability
- Very easy paradigm for showing UC impossibility
- Combinatorial characterization for SFE
- Way to relate passive & active corruption settings

Future Work

Extend combinatorial characterizations to interactive functionalities.

Future Work

Extend combinatorial characterizations to interactive functionalities.

Classify all intermediate complexity levels (unbounded model)

- Infinite strict hierarchy? Infinitely many incomparable functionalities?

Future Work

Extend combinatorial characterizations to interactive functionalities.

Classify all intermediate complexity levels (unbounded model)

- Infinite strict hierarchy? Infinitely many incomparable functionalities?

Characterize *completeness* of arbitrary functionalities:

- 0/1 Conjecture: Every functionality is either splittable or complete in PPT model.

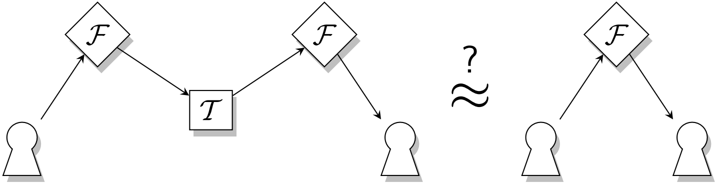
fin.

Thanks for your attention.

Special thanks to Qualcomm, PGP, and Marconi Society

Outline

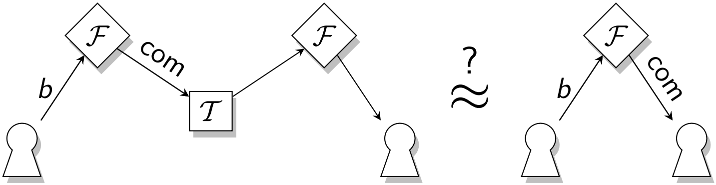
Bit commitment:



Environment that can distinguish two worlds:

Another Example

Bit commitment:

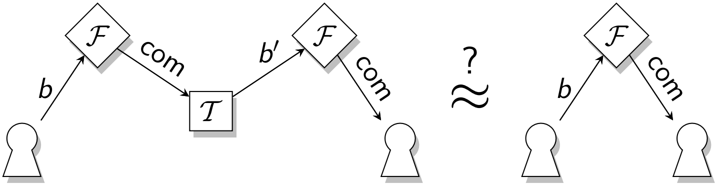


Environment that can distinguish two worlds:

1. Sender commits to random b

Another Example

Bit commitment:

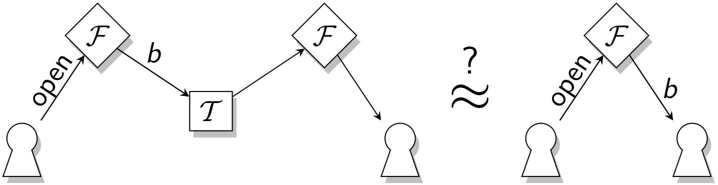


Environment that can distinguish two worlds:

1. Sender commits to random b
2. $\mathcal{T}$ must commit to a bit b' (its view independent of b)

Another Example

Bit commitment:

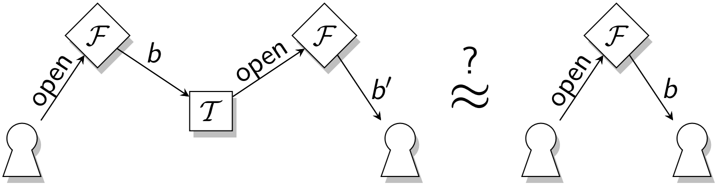


Environment that can distinguish two worlds:

1. Sender commits to random b
2. $\mathcal{T}$ must commit to a bit b' (its view independent of b)
3. Sender opens;

Another Example

Bit commitment:

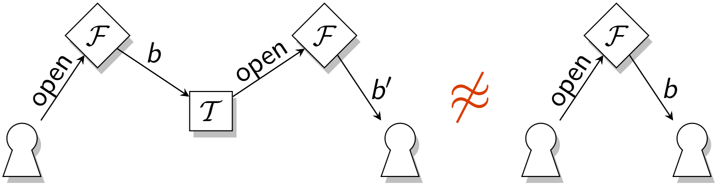


Environment that can distinguish two worlds:

1. Sender commits to random b
2. $\mathcal{T}$ must commit to a bit b' (its view independent of b)
3. Sender opens; $\mathcal{T}$ can only open to b'

Another Example

Bit commitment:



Environment that can distinguish two worlds:

1. Sender commits to random b
2. $\mathcal{T}$ must commit to a bit b' (its view independent of b)
3. Sender opens; $\mathcal{T}$ can only open to b'

$b \neq b'$ with probability $1/2$.