universal composability from essentially any trusted setup

Mike Rosulek | The University of Montana | CRYPTO 2012

secure computation. . .

Several parties wish to carry out an agreed-upon computation.

- Parties have individual inputs / output
- Security guarantees:
  - Privacy (learn no more than your prescribed output)
  - Input independence
  - Output consistency, etc..
- Parties are mutually distrusting, some possibly malicious

secure computation. . .

Several parties wish to carry out an agreed-upon computation.

- Parties have individual inputs / output
- Security guarantees:
  - Privacy (learn no more than your prescribed output)
  - Input independence
  - Output consistency, etc..
- Parties are mutually distrusting, some possibly malicious

Example:

- Set intersection $A \cap B$ (*function evaluation*)
- Generate a fair coin toss (*randomized*)
- Online poker without a dealer (*reactive*)

good news, bad news. . .

Good news [Canetti01]

Universal Composition (UC) framework = realistic security model for Internet protocols.

good news, bad news...

Good news [Canetti01]

Universal Composition (UC) framework = realistic security model for Internet protocols.

Bad news [CanettiFischlin01,CanettiKushilevitzLindell06]

UC security is impossible for almost all tasks that we care about ☹

the next best thing. . .

Slightly relax UC framework:

- Assume bounded network latency [KalaiLindellPrabhakaran05]
- Uniform adversaries, non-uniform simulators [LinPassVenkitasubramaniam09]
- Superpolynomial-time simulators [Pass03, PrabhakaranSahai04, BarakSahai05, MalkinMoriartyYakovenko06, CanettiLinPass10, ...]

the next best thing. . .

Slightly relax UC framework:

- Assume bounded network latency [KalaiLindellPrabhakaran05]
- Uniform adversaries, non-uniform simulators [LinPassVenkitasubramaniam09]
- Superpolynomial-time simulators [Pass03, PrabhakaranSahai04, BarakSahai05, MalkinMoriartyYakovenko06, CanettiLinPass10, ...]
- Trusted setup: Protocols can use ideal functionality
  - Bit-commitment [CanettiLindellOstrovskySahai02]
  - Common random string [CanettiLindellOstrovskySahai02,...]
  - Oblivious transfer [IshaiPrabhakaranSahai08]
  - Trusted hardware device [Katz07]

the next best thing...

Slightly relax UC framework:

- Assume bounded network latency [KalaiLindellPrabhakaran05]
- Uniform adversaries, non-uniform simulators [LinPassVenkitasubramaniam09]
- Superpolynomial-time simulators [Pass03, PrabhakaranSahai04, BarakSahai05, MalkinMoriartyYakovenko06, CanettiLinPass10, ...]
- Trusted setup: Protocols can use ideal functionality
  - Bit-commitment [CanettiLindellOstrovskySahai02]
  - Common random string [CanettiLindellOstrovskySahai02,...]
  - Oblivious transfer [IshaiPrabhakaranSahai08]
  - Trusted hardware device [Katz07]

fundamental question...

How useful is $\mathcal{F}$ as a trusted setup?

- What tasks have UC-secure protocols in the presence of $\mathcal{F}$?

fundamental question. . .

How useful is $\mathcal{F}$ as a trusted setup?

- What tasks have UC-secure protocols in the presence of $\mathcal{F}$?

Possible “levels of power” for $\mathcal{F}$

- **Useless**: access to $\mathcal{F}$ is equivalent to *no* trusted setup.
  - $\Leftrightarrow$ $\mathcal{F}$ already has a UC-secure protocol without setups

fundamental question...

How useful is $\mathcal{F}$ as a trusted setup?

- What tasks have UC-secure protocols in the presence of $\mathcal{F}$?

Possible "levels of power" for $\mathcal{F}$

- **Useless**: access to $\mathcal{F}$ is equivalent to *no* trusted setup.

  $\Leftrightarrow$ $\mathcal{F}$ already has a UC-secure protocol without setups

- **Complete**: *all* tasks have UC-secure protocols in presence of $\mathcal{F}$

fundamental question...

How useful is $\mathcal{F}$ as a trusted setup?

- What tasks have UC-secure protocols in the presence of $\mathcal{F}$?

Possible "levels of power" for $\mathcal{F}$

- **Useless**: access to $\mathcal{F}$ is equivalent to *no* trusted setup.
  - $\Leftrightarrow$ $\mathcal{F}$ already has a UC-secure protocol without setups
- **Intermediate:** something between these two extremes
- **Complete**: *all* tasks have UC-secure protocols in presence of $\mathcal{F}$

take-home message. . .

1. Which 2-party setups are **useless**?

2. Which 2-party setups are **complete**?

take-home message. . .

1. Which 2-party setups are **useless**?
   - Complete characterization [PrabhakaranRosulek08]
2. Which 2-party setups are **complete**?

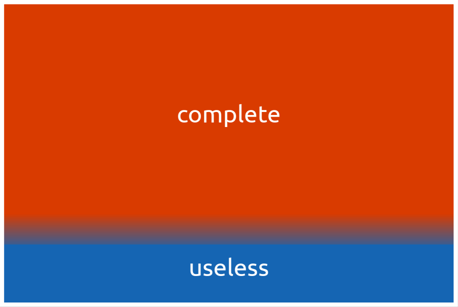
take-home message. . .

1. Which 2-party setups are **useless**?
   - Complete characterization [PrabhakaranRosulek08]
2. Which 2-party setups are **complete**?
   - Almost-complete characterization [This talk]

take-home message. . .

1. Which 2-party setups are **useless**?
   - Complete characterization [PrabhakaranRosulek08]
2. Which 2-party setups are **complete**?
   - Almost-complete characterization [This talk]

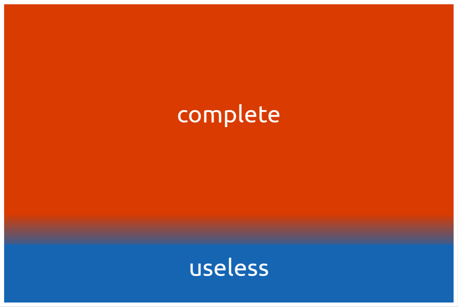
⇒ Nearly every setup is either useless or complete.



take-home message...

1. Which 2-party setups are **useless**?
   - Complete characterization [PrabhakaranRosulek08]
2. Which 2-party setups are **complete**?
   - Almost-complete characterization [This talk]

$\Rightarrow$ Nearly every setup is either useless or complete.

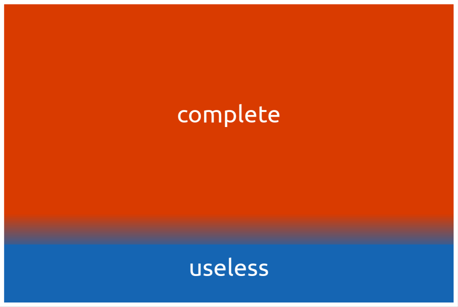


Characterize *reactive, randomized* functionalities, w/ behavior depending on security parameter!

take-home message. . .

1. Which 2-party setups are **useless**?
   - Complete characterization [PrabhakaranRosulek08]
2. Which 2-party setups are **complete**?
   - Almost-complete characterization [This talk]

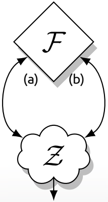
⇒ Nearly every setup is either useless or complete.

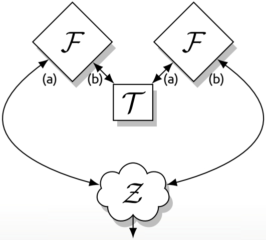


Characterize *reactive, randomized* functionalities, w/ behavior depending on security parameter!

[MajiPrabhakaranRosulek10] restricted to deterministic & constant-sized.

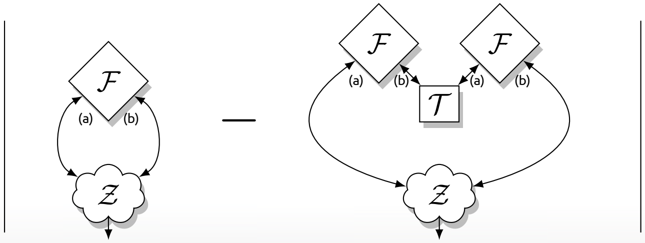
“splitting game” for $\mathcal{F}$…



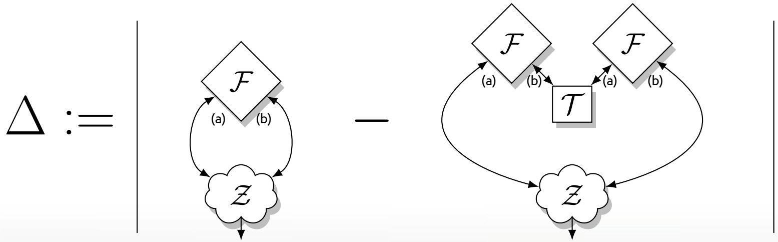


“splitting game” for $\mathcal{F}$…

$\Delta :=$



“splitting game” for $\mathcal{F}$...



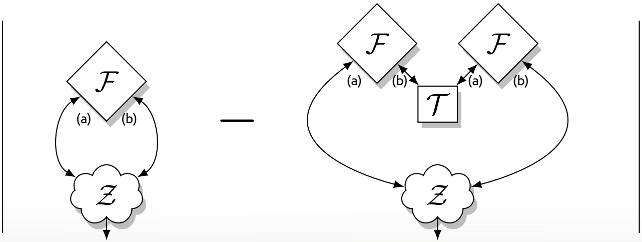
Definitions

$\mathcal{F}$ is **splittable** if $\mathcal{T}$ has a winning strategy. [PrabhakaranRosulek08]

$\Leftrightarrow$ $\exists \mathcal{T} : \forall \mathcal{Z} : \Delta$ negligible. *(“$\mathcal{T}$ fools all environments”)*

“splitting game” for $\mathcal{F}$...

$\Delta :=$



Definitions

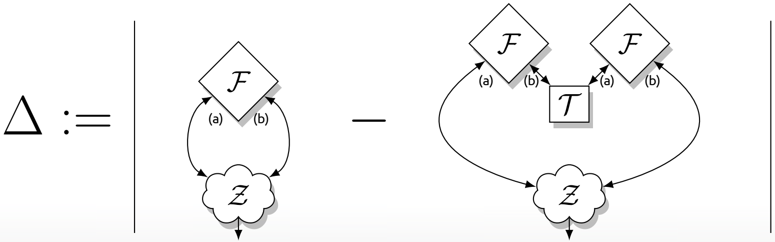
$\mathcal{F}$ is **splittable** if $\mathcal{T}$ has a winning strategy. [PrabhakaranRosulek08]

$\Leftrightarrow$ $\exists \mathcal{T} : \forall \mathcal{Z} : \Delta$ negligible. *(“$\mathcal{T}$ fools all environments”)*

$\mathcal{F}$ is **strongly unsplittable** if $\mathcal{Z}$ has a winning strategy.

$\Leftrightarrow$ $\exists \mathcal{Z} : \forall \mathcal{T} : \Delta$ 1/poly. *(“$\mathcal{Z}$ detects all splitting strategies”)*

“splitting game” for $\mathcal{F}$...



Definitions

$\mathcal{F}$ is **splittable** if $\mathcal{T}$ has a winning strategy. [PrabhakaranRosulek08]

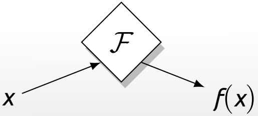
$\Leftrightarrow$ $\exists \mathcal{T} : \forall \mathcal{Z} : \Delta$ negligible. *(“$\mathcal{T}$ fools all environments”)*

$\mathcal{F}$ is **strongly unsplittable** if $\mathcal{Z}$ has a winning strategy.

$\Leftrightarrow$ $\exists \mathcal{Z} : \forall \mathcal{T} : \Delta$ 1/poly. *(“$\mathcal{Z}$ detects all splitting strategies”)*

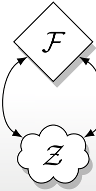
- Some (arguably unnatural) $\mathcal{F}$ admit no winning strategy for $\mathcal{Z}$ or $\mathcal{T}$!
- Applies to arbitrary (reactive, randomized, etc) functionalities.

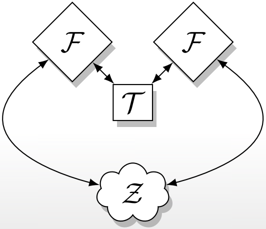
quiz: splittable or not?...



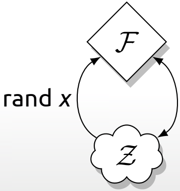
... where f is a OWF

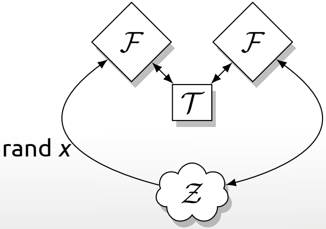
quiz: splittable or not?…



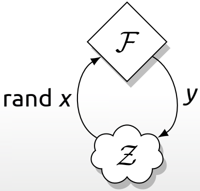


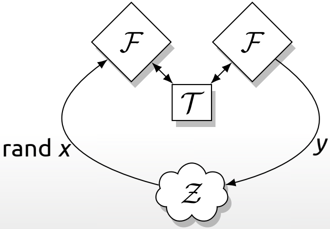
quiz: splittable or not?…



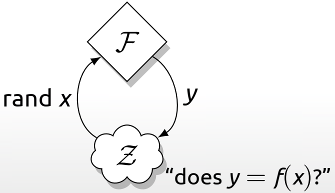


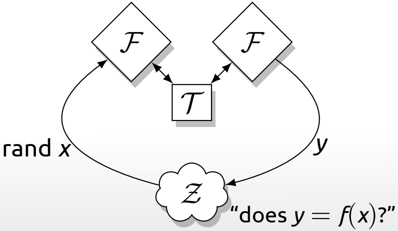
quiz: splittable or not?…



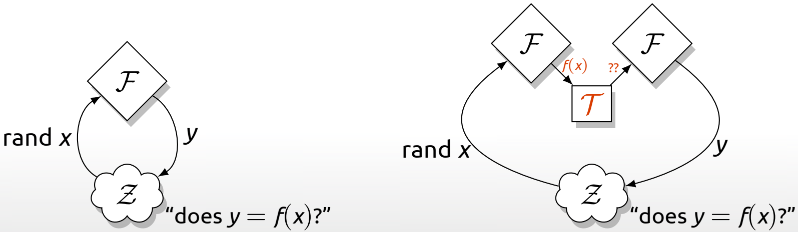


quiz: splittable or not?…



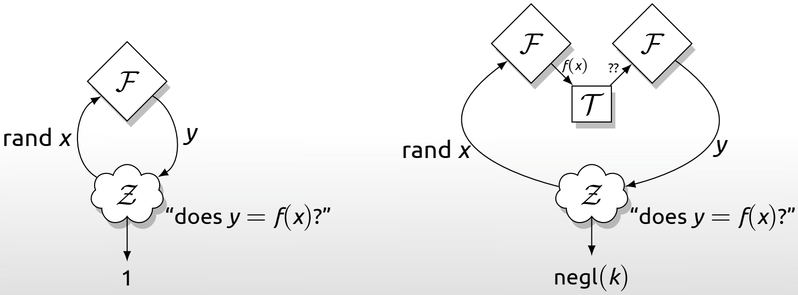


quiz: splittable or not?...



- To make interactions similar, $\mathcal{T}$ must be able to invert f

quiz: splittable or not?. . .

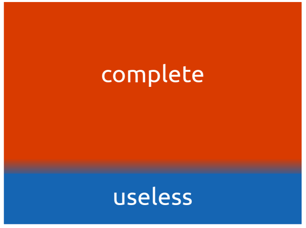


- To make interactions similar, $\mathcal{T}$ must be able to invert f

$\Rightarrow$ This $\mathcal{Z}$ detects every $\mathcal{T}$

$\Rightarrow$ $\mathcal{F}$ is strongly unsplittable

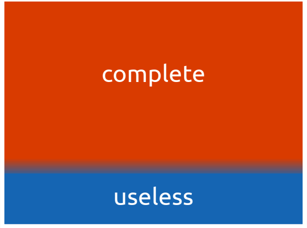
the characterization. . .



$\mathcal{F}$ useless $\Leftrightarrow$ $\mathcal{F}$ splittable

[PrabhakaranRosulek08]

the characterization. . .



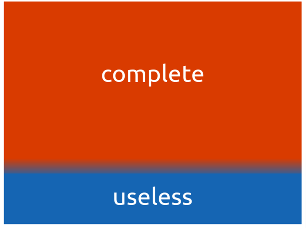
$\mathcal{F}$ complete $\stackrel{*}{\Leftarrow}$ $\mathcal{F}$ strongly unsplittable

[This talk]

$\mathcal{F}$ useless $\Leftrightarrow$ $\mathcal{F}$ splittable

[PrabhakaranRosulek08]

the characterization. . .



$\mathcal{F}$ complete $\stackrel{*}{\Leftarrow}$ $\mathcal{F}$ strongly unsplittable

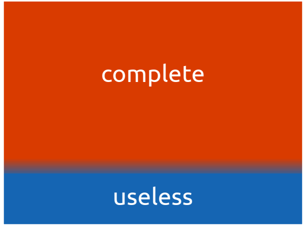
[This talk]

*: slightly more involved statement for *reactive* $\mathcal{F}$

$\mathcal{F}$ useless $\Leftrightarrow$ $\mathcal{F}$ splittable

[PrabhakaranRosulek08]

the characterization. . .



$\mathcal{F}$ complete $\overset{*}{\Leftarrow}$ $\mathcal{F}$ strongly unsplittable

[This talk]

*: slightly more involved statement for *reactive* $\mathcal{F}$

$\mathcal{F}$ useless $\Leftrightarrow$ $\mathcal{F}$ splittable

[PrabhakaranRosulek08]

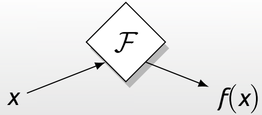
Outline: Strong Unsplittability $\Rightarrow$ Complete

Suffices to construct UC-secure **commitment protocol**

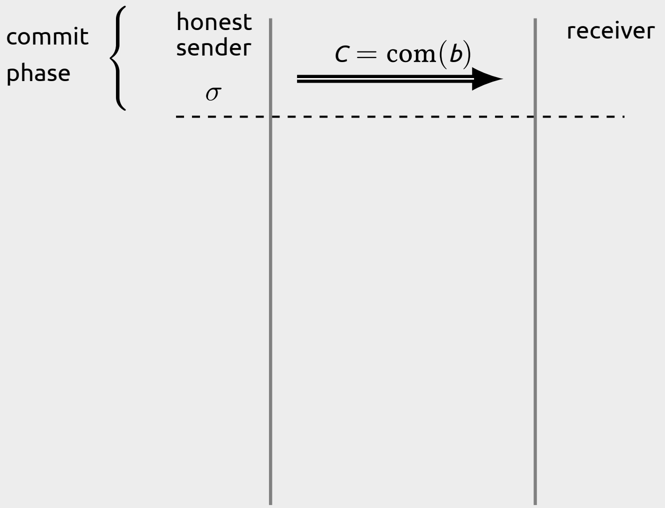
1. UC-commitment is complete [CanettiLindellOstrovskySahai02]

commitment protocol...

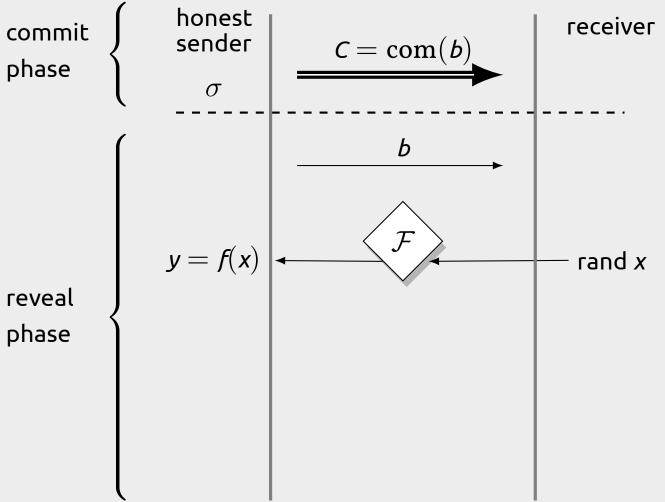
How to do it (using our example)...



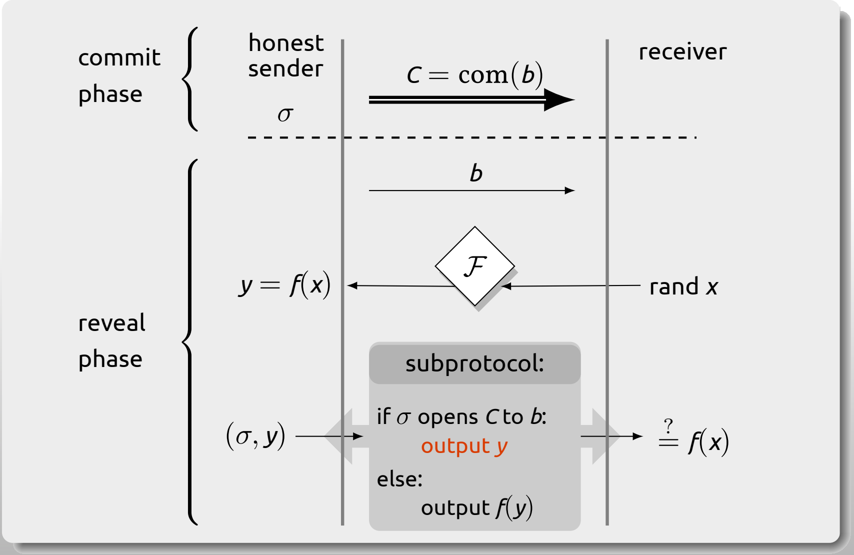
commitment protocol. . .



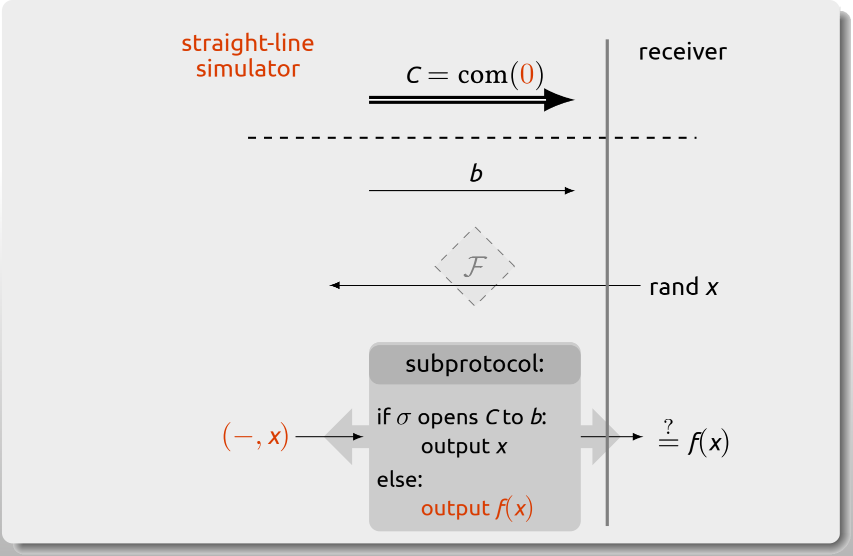
commitment protocol. . .



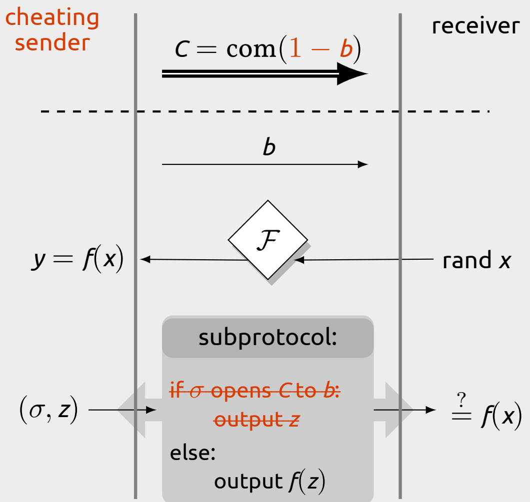
commitment protocol...



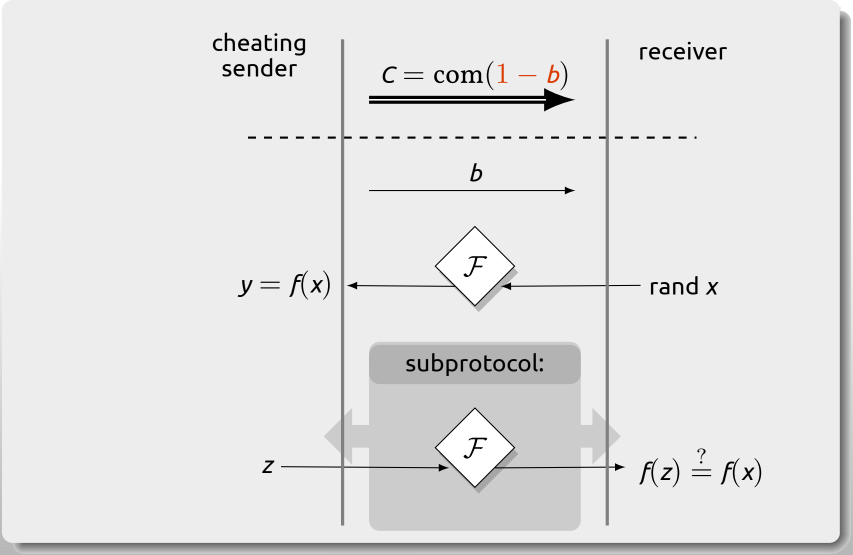
commitment protocol...



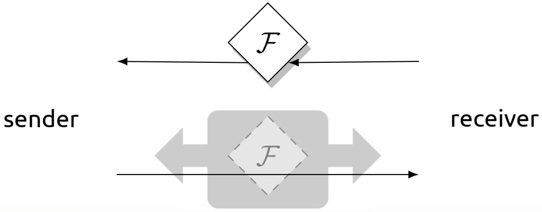
commitment protocol. . .



commitment protocol. . .

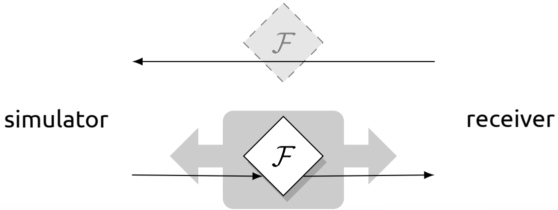


protocol: key idea. . .



Honest sender: Bypass “instance of $\mathcal{F}$” within subprotocol

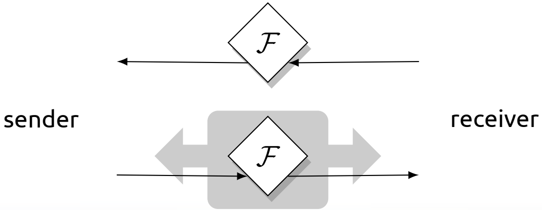
protocol: key idea. .



Honest sender: Bypass “instance of $\mathcal{F}$” within subprotocol

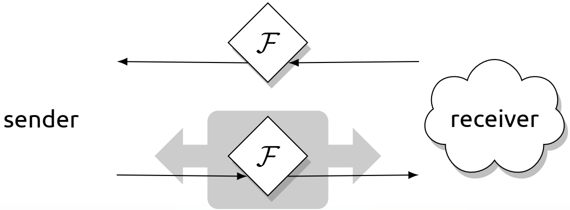
Simulator: Bypass ideal instance of $\mathcal{F}$

protocol: key idea. .



Honest sender:	Bypass "instance of $\mathcal{F}$" within subprotocol
Simulator:	Bypass ideal instance of $\mathcal{F}$
Cheating sender:	"Stuck between" two instances of $\mathcal{F}$

protocol: key idea. .



Honest sender: Bypass "instance of $\mathcal{F}$" within subprotocol

Simulator: Bypass ideal instance of $\mathcal{F}$

Cheating sender: "Stuck between" two instances of $\mathcal{F}$

Strong Un-Splittability

There is a way for receiver to behave which can distinguish:

- Interacting with a single instance of $\mathcal{F}$ (#1, #2)
- Interacting with *any* "split" $\mathcal{F}$ (#3)

wrap-up. . .

Other things in the paper (full version @ `eprint/2011/240`):

- Get from “one-sided” to full-fledged UC commitment
- Subtleties, caveats for *reactive* $\mathcal{F}$
- Complete $\Rightarrow$ strongly unsplittable? (almost!)

wrap-up. . .

Other things in the paper (full version @ `eprint/2011/240`):

- Get from "one-sided" to full-fledged UC commitment
- Subtleties, caveats for *reactive* $\mathcal{F}$
- Complete $\Rightarrow$ strongly unsplittable? (almost!)

Summary:

> *Every "natural" functionality (reactive, randomized, etc.) is either* **useless** *or* **complete** *as a UC setup.*

The End