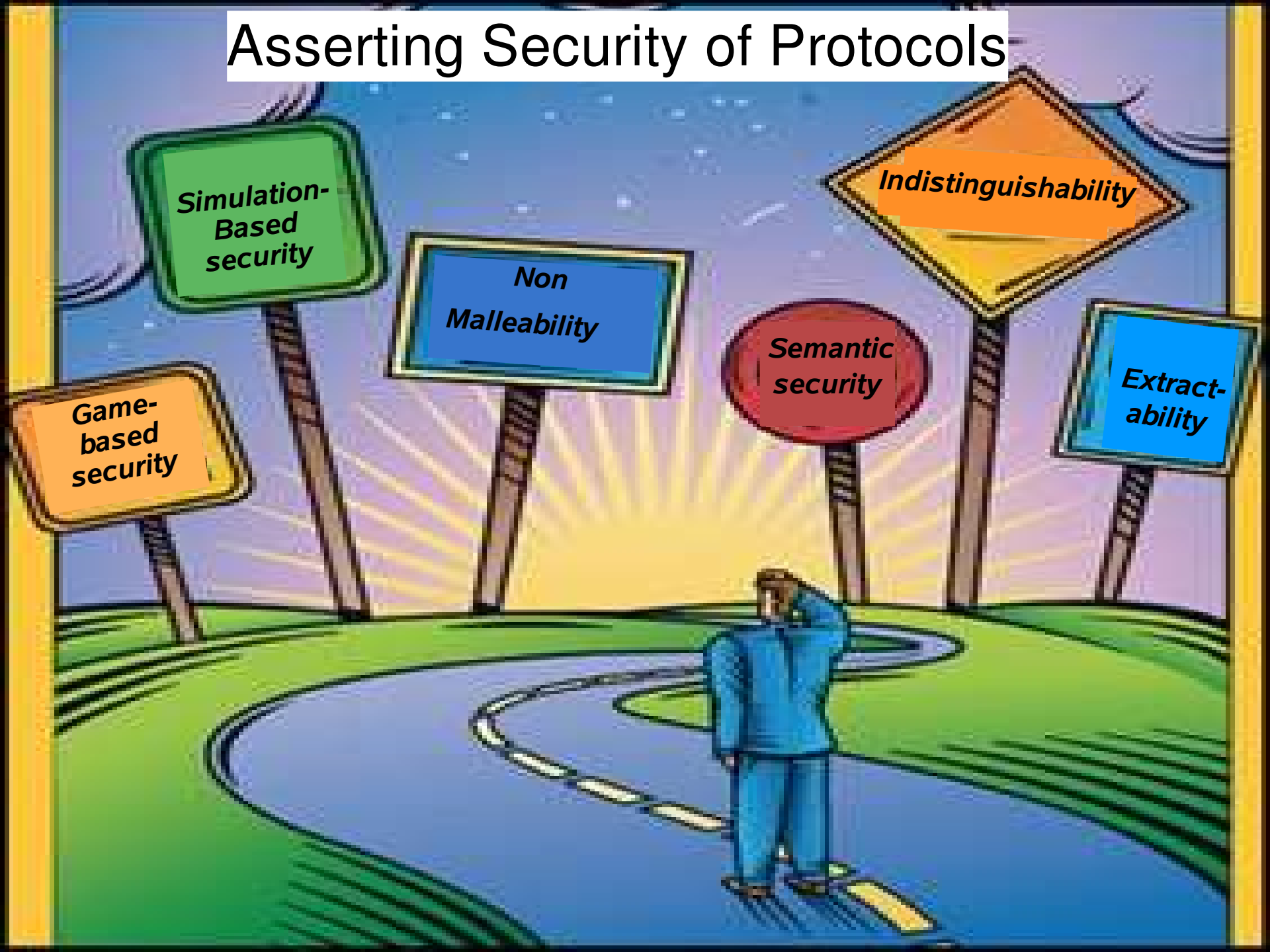
Obtaining Universally Composable Security: Towards the Bare Bones of Trust

Ran Canetti

IBM

Asserting Security of Protocols



A desired property: Composability

Meta-definition:

A notion of security is composable if the security guarantees it provides remain valid even when the analysed protocol is composed with (i.e., runs alongside) other protocols.

The advantages of composable security

- Simplifies the analysis:

  Can partition a complex system to small components, analyze each component separately as a “stand alone” protocol, and then deduce the security of the overall system.

The advantages of composable security

- Simplifies the analysis:

  Can partition a complex system to small components, analyze each component separately as a “stand alone” protocol, and then deduce the security of the overall system.

- Provides stronger security:

  Provides security even in unknown/dynamic systems where some of the components are unknown at the time of analysis.

How to assert composable security

Need:

- Composition operation(s) that capture common ways in which protocols are combined in actual systems.

- A way to argue that security properties of protocols are preserved under these composition operations.

The UC security framework

Provides:

- A general methodology for asserting security properties of protocols.

  A general composition operation - universal composition - that captures most standard composition methods.
  (E.g.: sequential, concurrent, subroutine calls, on same/different inputs, by same/different parties)

  A guarantee that security properties asserted within the framework are preserved even under universal composition with arbitrary other protocols.

The UC security framework

Provides:

- A general methodology for asserting security properties of protocols.
- A general composition operation - universal composition - that captures most standard composition methods.
  (E.g.: sequential, concurrent, subroutine calls, on same/different inputs, by same/different parties)
- A guarantee that security properties asserted within the framework are preserved even under universal composition with arbitrary other protocols.

The UC security framework

Provides:

- A general methodology for asserting security properties of protocols.
- A general composition operation - universal composition - that captures most standard composition methods.
  (E.g.: sequential, concurrent, subroutine calls, on same/different inputs, by same/different parties)
- A guarantee that security properties asserted within the framework are preserved even under universal composition with arbitrary other protocols.

The strength of UC security

Can:

- Analyze security of a protocol in a stand alone setting
- Be guaranteed that the security properties of the protocol remain intact in an arbitrary, multi-party, multi-protocol, multi-instance system.

The strength of UC security

Can:

- Analyze security of a protocol in a stand alone setting
- Be guaranteed that the security properties of the protocol remain intact in an arbitrary, multi-party, multi-protocol, multi-instance system.

However...

Impossibility of UC security

- Authenticated communication cannot be achieved in the bare model.

But, this is true even for “basic” (non-composable) security...

Some sort of initial authentication is necessary.

Impossibility of UC security

- Many two-party tasks (e.g., commitment, zero-knowledge, coin tossing, oblivious transfer) are impossible to realize even given ideal authentication.

  This is NOT true for "basic" security (e.g. [GMW87]).

- Impossibility generalizes to multi-party tasks with dishonest majority.

Relaxations of UC security

Some proposed relaxations:

- "Angel-Based" UC [Prabhakaran-Sahai04]
- UC with Quasi-polynomial simulation [Barak-Sahai05]
- Indistinguishability-based security [Micali-Pass-Rosen06]

These notions avoid the above impossibility.
But, neither notion provides all that we want...

Can UC security be relaxed, while maintaining both security and composability?

Can we have an alternative definition of security that:

- Guarantees basic (stand-alone) security
- Is preserved under universal composition
- Is more relaxed than UC-security?

No, if one takes “basic security” to mean simulation-based security, e.g. a la [C00]:

Thm [Lindell03,04,C06]: Assume protocol π realizes a task (ideal functionality) F with basic security, even when composed with arbitrary protocols. Then π UC-realizes F.

Conclusion

- UC-security is too strong a notion.
- Obtaining security that remains intact under general composition operations with arbitrary protocols is not possible. We should make do with weaker notions.

Thank you!

An alternative approach: Use trusted set-up

- Design and analyze protocols in an idealized model where some system components are trusted to behave in a certain way.
- Security will be guaranteed in actual systems where the idealizations are physically guaranteed.

An alternative approach: Use trusted set-up

- Design and analyze protocols in an idealized model where some system components are trusted to behave in a certain way.
- Security will be guaranteed in actual systems where the idealizations are physically guaranteed.

Several trusted set-up models for realizing UC security have been proposed.

Evaluating trusted set-up models

Main parameters:

- Ease of constructing and analyzing protocols
- Ease of physical realization, or "level of trust"

In the rest of this talk

- Review the UC framework
- Review impossibility of UC commitment w/o set-up
- Review possibility in some trusted set-up models:
  - Common reference string (CRS)
  - “Sunspot” model
  - Key registration model
  - Tamper-proof hardware
  - Global and augmented CRS
- Discuss set-up models for authentication

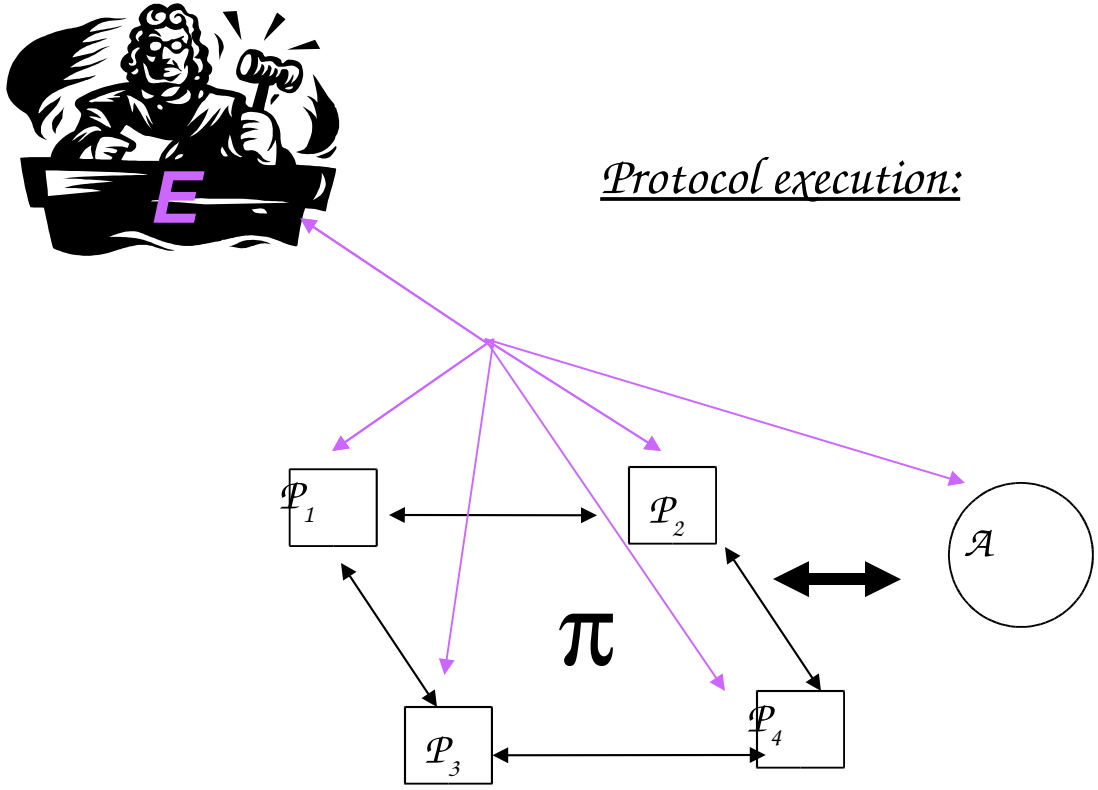
The trusted-party paradigm
[Goldreich-Micali-Wigderson87]

'

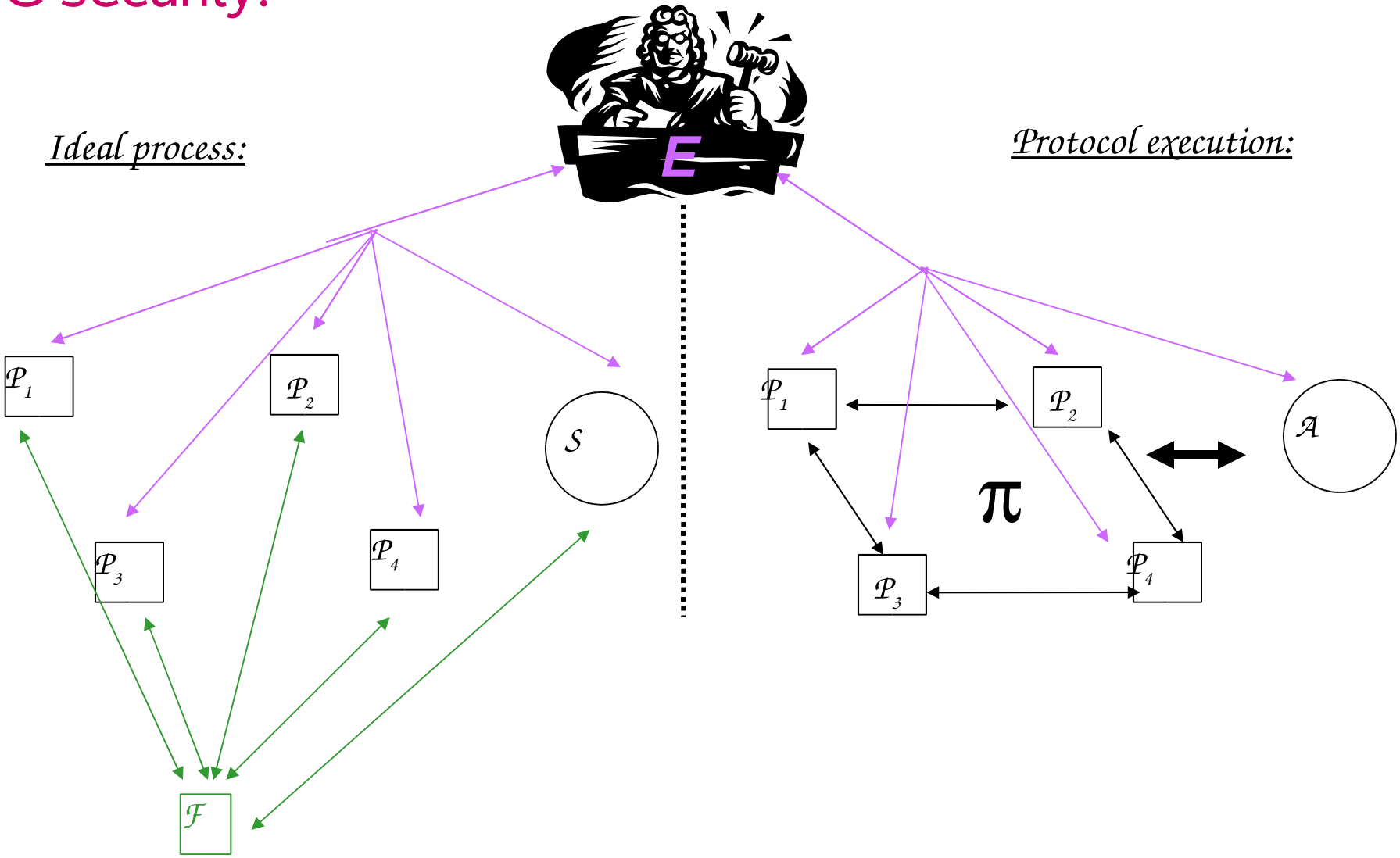
'A protocol is secure for some task if it "emulates" an "ideal process" where the parties hand their inputs to a "trusted party", who locally computes the desired outputs and hands them back to the parties.'

Many formalizations exist, with varying levels of composability guarantees, e.g. [Goldwasser-Levin 90, Micali-Rogaway 91, Beaver 91, Canetti 93,95,00,01, Pfitzmann-Waidner 94,00,01, Hirt-Maurer 00, Dodis-Micali 00]

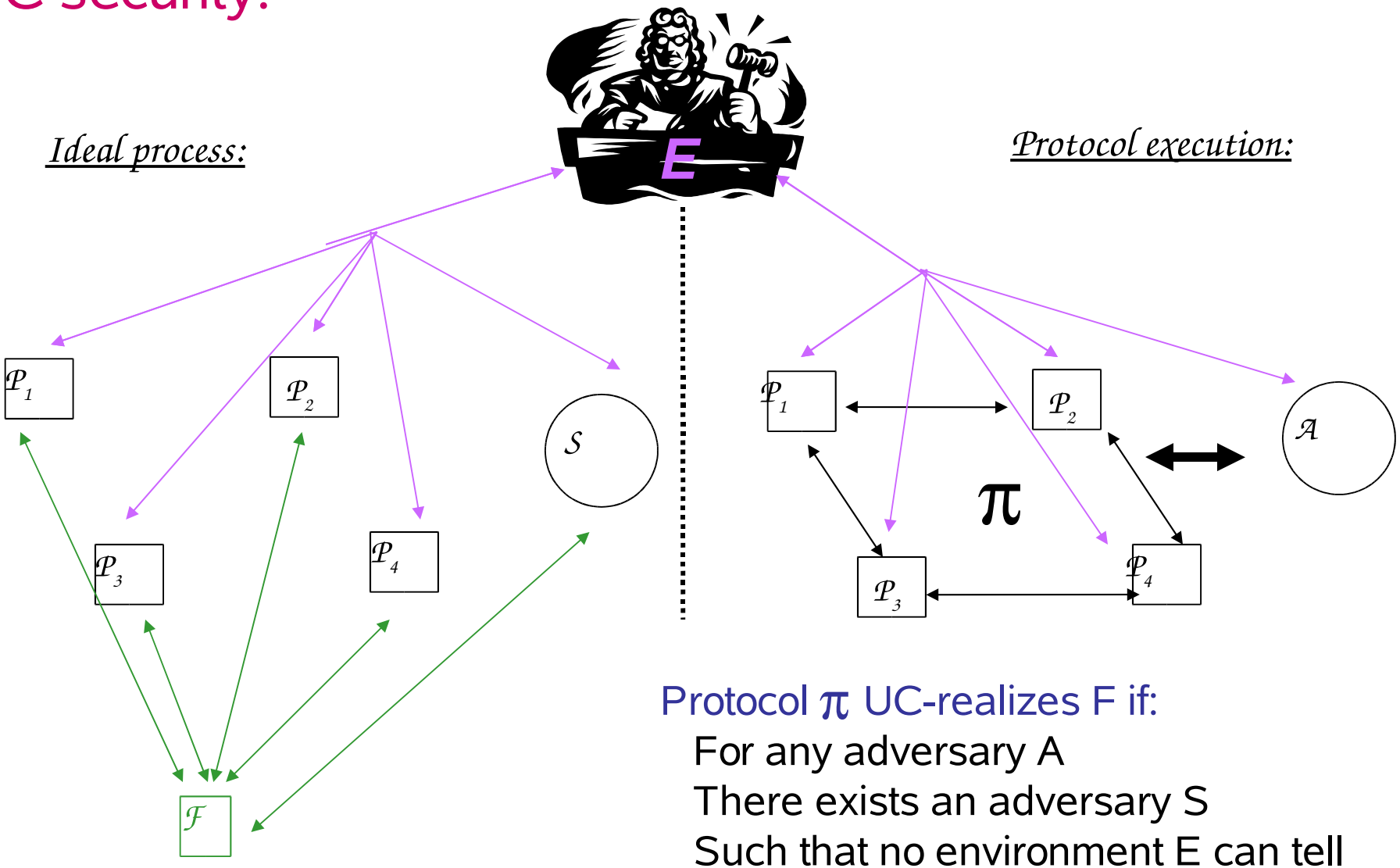
UC security:


E
Protocol execution:
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{A}$
π
$\mathcal{P}_3$
$\mathcal{P}_4$

UC security:


Ideal process:
E
Protocol execution:
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{S}$
$\mathcal{P}_3$
$\mathcal{P}_4$
$\mathcal{F}$
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{A}$
π
$\mathcal{P}_3$
$\mathcal{P}_4$

UC security:



Protocol π UC-realizes F if:

For any adversary A

There exists an adversary S

Such that no environment E can tell whether it interacts with:

- A run of π with A
- An ideal run with F and S

The commitment functionality, F_{com} (simplified)

1. Upon receiving ("commit",C,R,x) from party C, record x, and send (C, "receipt") to R.
2. Upon receiving ("open") from C, send (C,x) to R and halt.

Recall

- R is assured that the value it received in step 2 was fixed in step 1.
- C is assured that R learns nothing about x before it is opened.

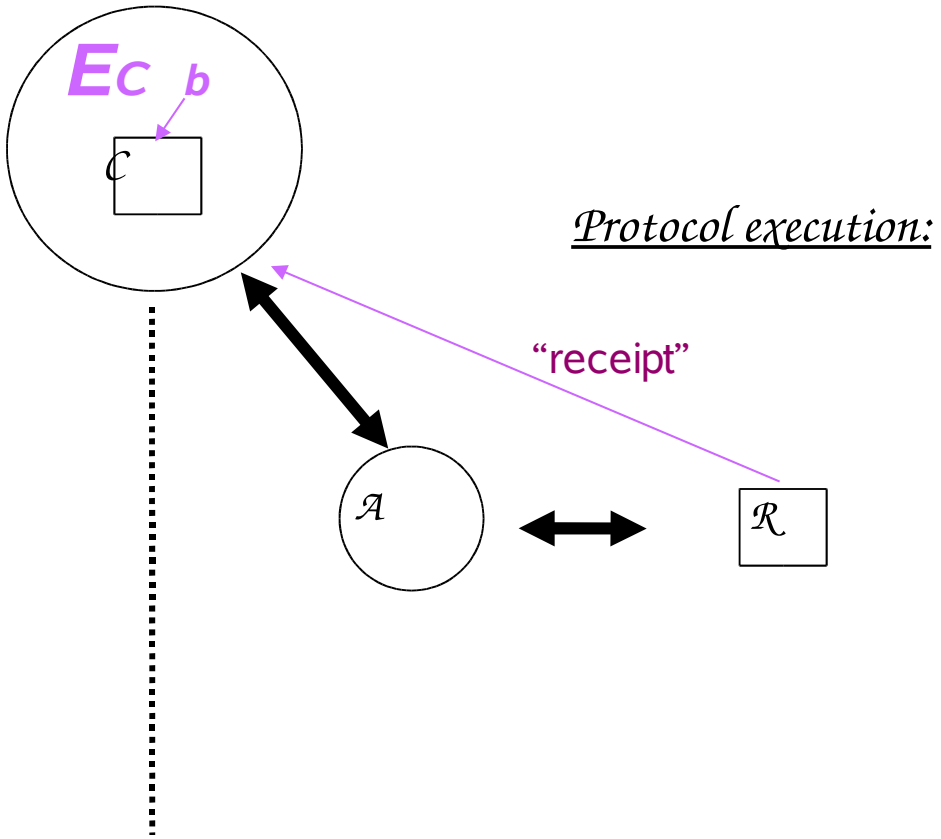
Impossibility of UC commitment

Theorem [C-Fischlin01]: There exist no two-party protocols that UC-realize F_{com}, even when given ideal authentication.

Proof idea:

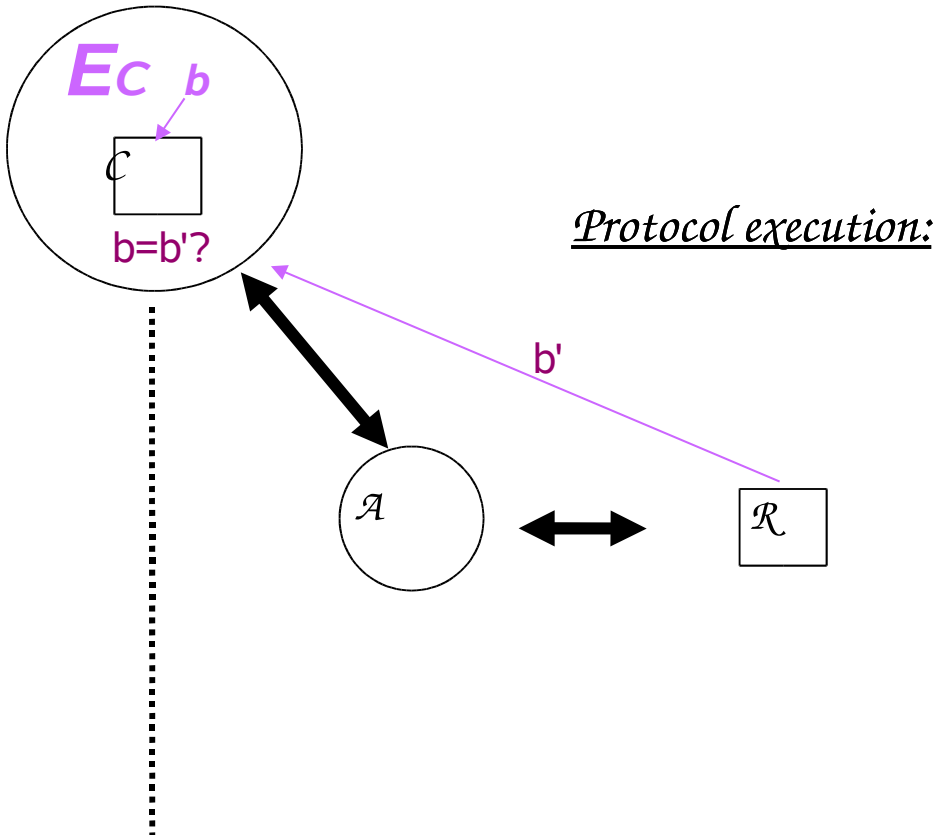
Let π be a two-party protocol that UC-realize F_{com}. Then...

Consider E_c and A_c :



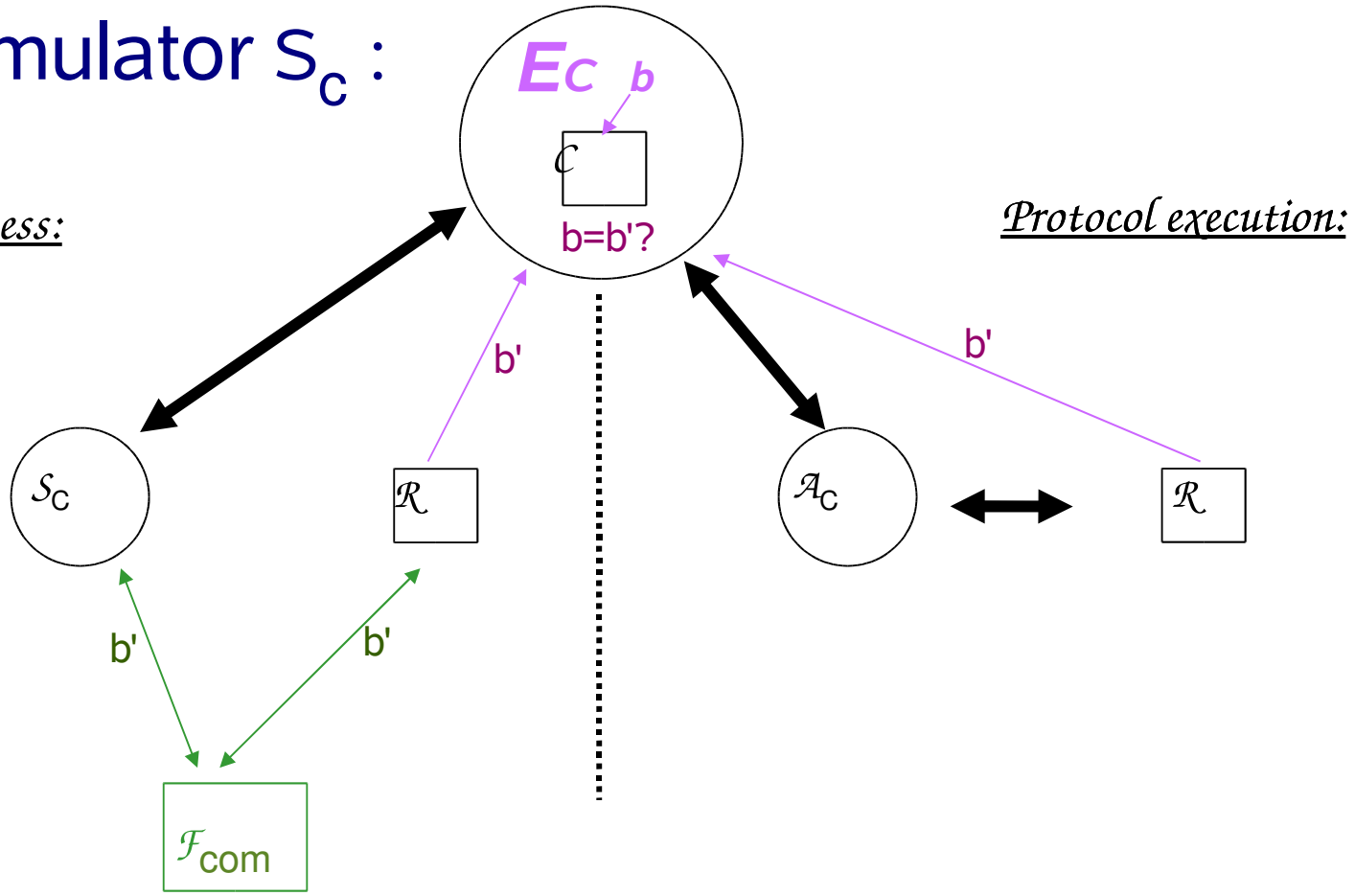
- The committer C is corrupted
- A_c delivers all messages between R and E_c.
- E_c behaves as an honest committer C on random input b.

Consider E_c and A_c :



- The committer C is corrupted
- A_c delivers all messages to/from E_c.
- E_c behaves as an honest committer C on random input b.
- After getting "receipt" from R, E_c honestly opens b and checks if b=b'.

The simulator S_c :

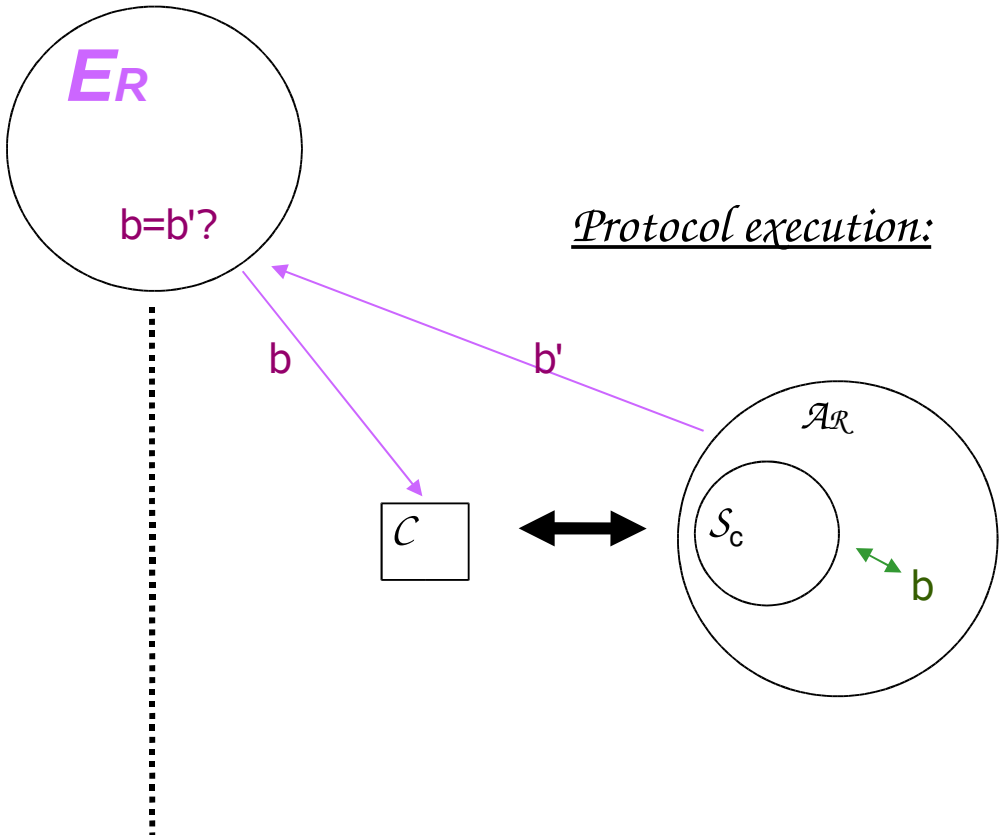


S_c needs to:

- Interact with E_c as an honest receiver.
- Provide the correct bit b to F_{com}

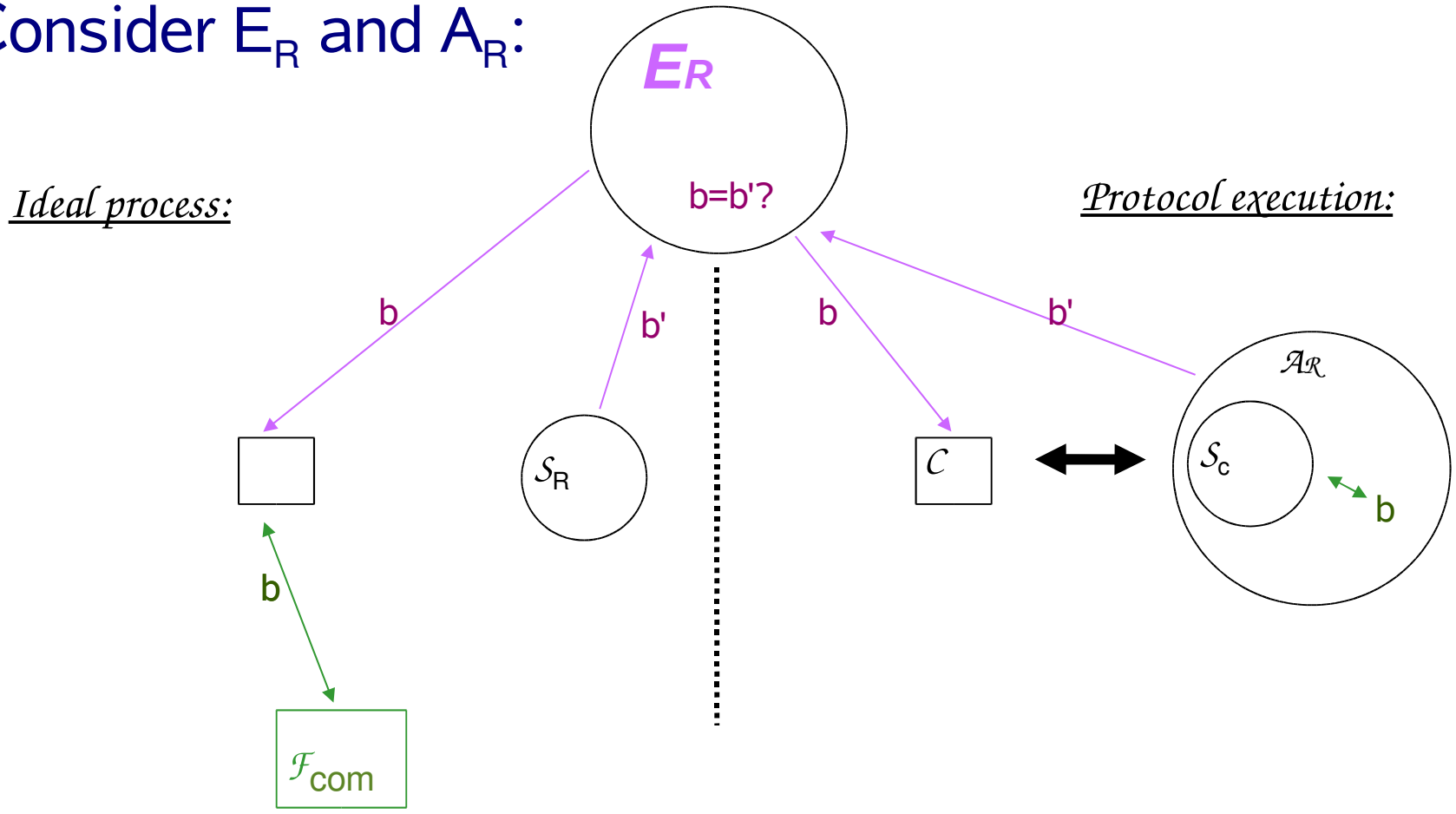
That is, S_c is forced to "extract" b when communicating with C.

Consider E_R and A_R:



- The receiver R is corrupted
- E_R gives a random b to the honest committer, and never opens
- A_R runs S and can thus guess b.

Consider E_R and A_R:



- The receiver R is corrupted
- E_R gives a random b to the honest committer, and never opens
- A_R runs S and can thus guess b.
- But S_R 's view is independent from b...

The crux of the proof:

The simulator must be able to extract the committed bit by simple interaction with the honest committer, just as an adversary would.

- *Any useful set-up model had better get around this limitation.*

The common reference string model

The idea:

- All protocol participants obtain a value r that is trusted to be taken from a known distribution D.
- No “side information” related to D is known.

Potential realizations:

- Measurements of the physical environment
- Trusted entities or devices

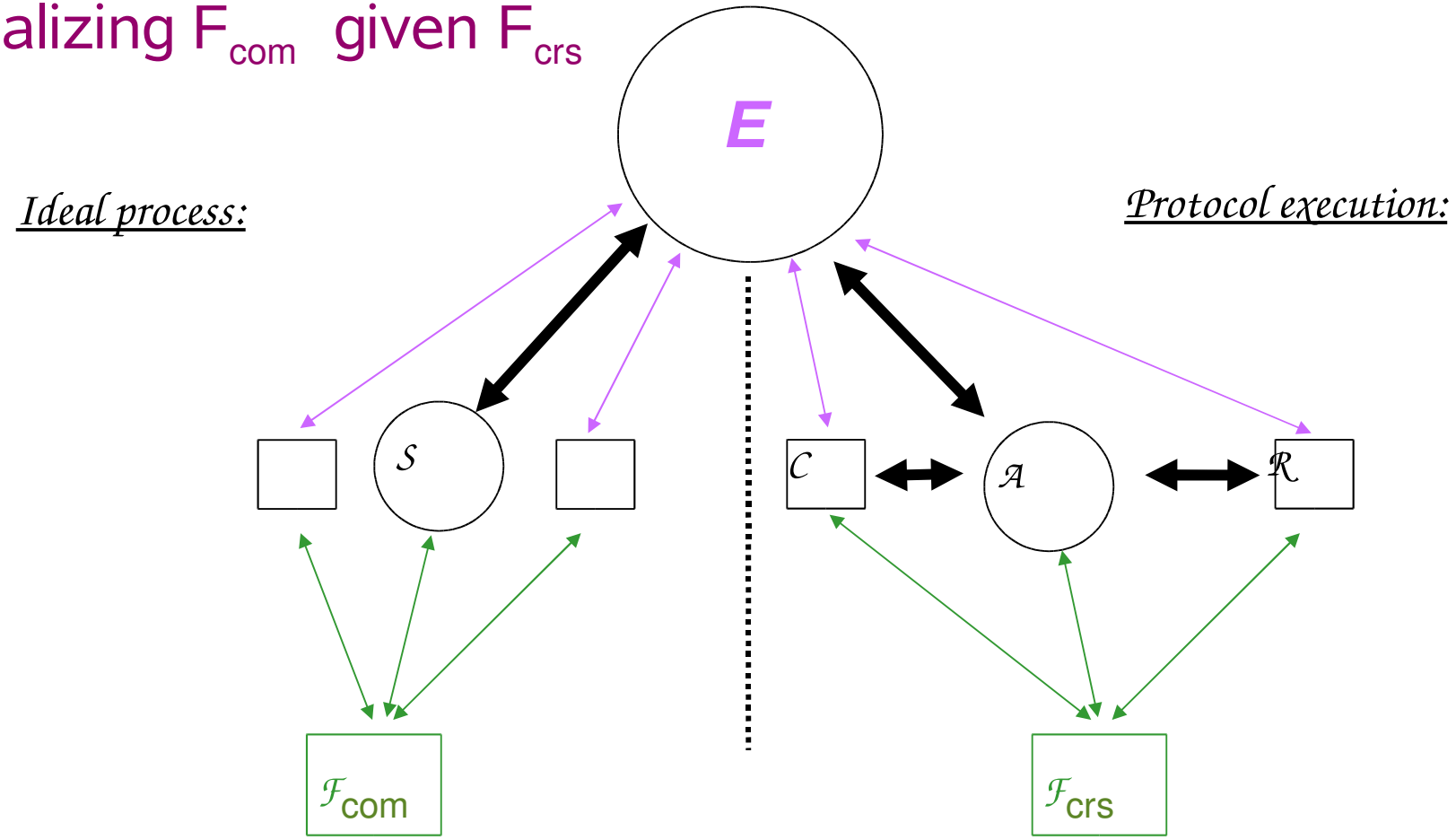
The formal modeling: The common reference string functionality

Functionality F_{crs}

(with prescribed distribution D and set R of recipients)

- Choose a value r from distribution D, and send r to the adversary.
- Upon receiving ("CRS") from party P in R, return r to P.

Realizing F_{com} given F_{crs}



- E expects to learn the CRS from A
- In the ideal process, S can make up its own CRS
- Very similar to the CRS model of [Blum-Feldman-Micali89]
- Impossibility no longer holds

Realizing F_{com} given F_{crs}

- Roughly speaking, we need to make sure that S can:
  - Extract the committed value from a corrupted committer.
  - Generate commitments that can be opened in multiple ways to a corrupted receiver.
  - Explain internal state of committer and verifier upon corruption (for adaptive security).

The [C-Fischlin01] protocol
(for static adversaries)

CRS: - a claw-free pair f_0, f_1, of traproor permutations
- an encryption key e of a CCA-secure scheme

To commit to b:

- choose random x, r_0, r_1
- send $f_b(x), E_e(r_b, x|sid), E_e(r_{1-b}, 0)$

To open: send b, x, r_b. (don't send r_{1-b})

Analysis idea:

- To extract, simulator decrypts both ciphertexts and finds x.
- To equivocate, simulator chooses x_0, x_1, r_0, r_1, such that $f_0(x_0)=f_1(x_1)=y$ and sends $y, E_e(r_0, x_0), E_e(r_1, x_1)$.

Other works in the CRS model

- Can realize ZK, any “well formed” functionality [C-Lindell-Ostrovsky-Sahai02,...]
- Many other protocols with a variety of nice properties

But have to put much trust in the CRS...

"Have no fear" said the cat.

" I will not let you fall"

Difficulties in implementing the CRS model in actual systems

- The reference string must come from a precisely specified distribution
- All parties must put trust in a single, external construct (entity)
- Need to have a different reference string for each protocol session

"Have no fear" said the cat.
" I will not let you fall"

We'll address these one by one...

How to deal with an "imperfect" CRS?

In the CRS model the reference string is guaranteed to be taken from a known distribution.

This is hard to implement in reality:

- Does not allow for "noisy" physical measurements where the actual distribution is not known.
- Does not allow for "semi-trusted" processes that may leak partial trapdoor information or allow partial adversarial influence.

Initial thoughts

- A partial solution: Allow only very simple distributions (e.g. the uniform distribution)

But, even the uniform distribution is hard to guarantee precisely...

Can we have protocols that work even if the distribution of the CRS is unknown?

Modeling a CRS with unknown distribution: The "sunspot" functionality (simplified)

Functionality F_{sun}

(with a set R of recipients)

- At activation, receive a sampling algorithm D from the adversary, set s=D(r) for random r, and return s to the adversary.
- Upon receiving ("CRS") from party P in R, return s to P.

Realizing F_{com} given F_{sun}

Theorem [C-Pass-shelat07]:

Assume CRHF, dense cryptosystems, OWFs with sub-exponential hardness. Then there exists a two-party protocol that UC-realizes F_{com} given F_{sun} if and only if it is guaranteed that:

- D's runtime is polynomial in n=|s|
- Minentropy(D) > Description(D) + polylog n

Proof idea of positive result

(A ZK protocol, for non-adaptive corruptions)

The idea (following [Feige-Shamir89,Barak01]):

The prover will prove "either the theorem is true or the CRS has a short description". (here "short" = polylog n)

Soundness:

The CRS has min-entropy > polylog n, so can cheat only with negligible probability

Zero-Knowledge:

The simulator will give E a CRS s=D(PRG(r')), where |r'|=polylog n. Now:

-E cannot tell the simulated s from a real one (PRG)

-S knows a short description for s

How to minimize trust in a single entity?

In the CRS model, all parties must put full trust a single external entity(mechanism)...

"Have no fear" said the cat.
" I will not let you fall"

The Key Registration model[Barak-C-Nielsen-Pass04]

The Idea:

- Parties obtain public keys from “registration authorities”.
- There can be multiple authorities, each party registers with one.
- A party needs to trust:
  - Its own authority to keep its secret key secret
  - Other authorities to have seen the secret keys of its registrants

-> *No single entity needs to be fully trusted by all...*

The Key Registration functionality, F_{kr}

(with function f and set R of participants)

Upon receiving ("Register") from a party P in R do:

- If P is uncorrupted, then record (P,f(r)) for random r.
- If P is corrupted, and provides value s, record (P,f(s)).

Upon receiving ("retrieve", P), from a party in R:

If there is a recorded entry (P,s), then return s.

Only "corrupted parties" see their private keys. This means that:

- Protocols in this model cannot use the secret keys.
- Parties *can* know (and choose) their keys, by deviating from the protocol.

Realizing F_{com} given F_{kr}

An idea:

Use the [CF01] protocol, but replace the CRS with PKs.

- The verifier PK is a claw-free pair f_0, f_1, of traproor permutations
- The committer PK is an encryption key e of a CCA-secure scheme.

Analysis goes through, for non-adaptive corruptions.

Other works in the KR model

- Can construct UC NIZK (for non-adaptive corruptions) [BCPR04]
- Can construct UC commitments against adaptive corruptions, at the price of more interaction [C-Dodis-Pass-Walfish07]

How to eliminate trust in external entities?

In all models so far, the parties had to trust an external entity or mechanism.

Can we eliminate that?

"Have no fear" said the cat.
" I will not let you fall"

The “hardware token” model [Katz07]

The idea:

Parties can encapsulate a program in a tamper-proof hardware device (“token”), and hand the device to other parties.

Need to trust that:

- Parties cannot see or modify the internal state of a given device.
- Parties can prevent a given device from communicating directly with other parties.

Parties need not trust the code run by a given device!

The hardware token functionality, F_{wrap}

(simplified)

Upon receiving ("wrap",M,B) from party A do:

- Internally invoke an instance of program M
- Output ("initialized",A,B) to party B and the adversary.

Upon receiving ("input", x), from B:

- Hand x to M, and forward to B any output of M.

Note:

- M can maintain state and have its own randomness
- M can communicate only with B

Realizing F_{com} given F_{wrap}

Theorem [Katz07]:

Under the DDH assumption, there exists a two-party protocol that UC-realizes F_{com} given F_{wrap}.

How to use a “globally known” trusted mechanism?

In all models so far, the parties had to use a different instance of the trusted mechanism per protocol instance.

Can we use a single global trusted mechanism, while maintaining composability?

“Have no fear” said the cat.

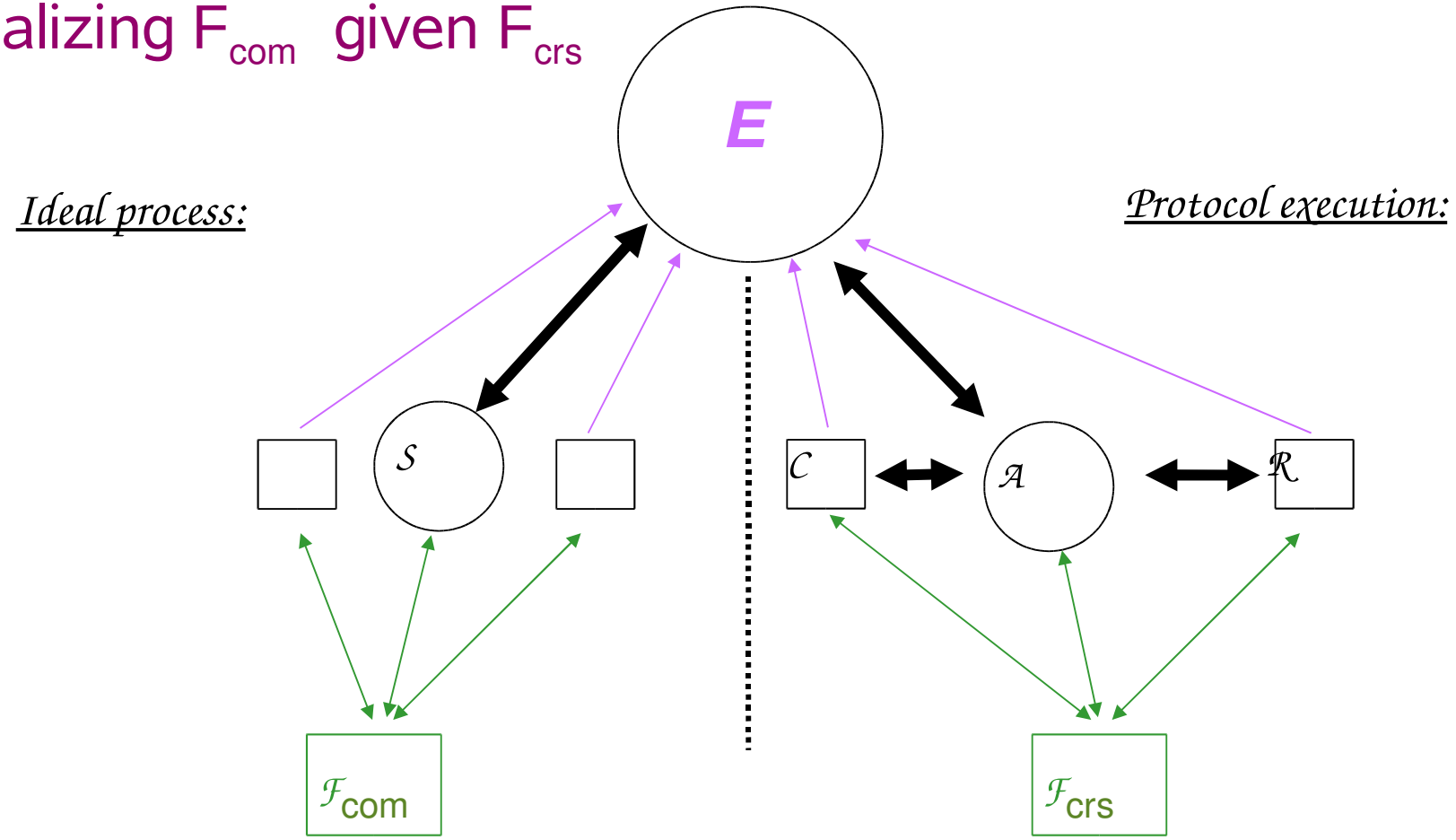
“ I will not let you fall”

Towards a globally known trusted set-up

Questions:

- How to model the situation where the same set-up information is known to and trusted by all --- including other parties that run arbitrary protocols?
- How to construct protocols that use such set-up?

Realizing F_{com} given F_{crs}



Recall:

F_{crs} gives the reference string only to the participants of this instance.

→E expects to learn the CRS *only* from A.

→In the ideal process, S can make up its own CRS.

How to extend the CRS model?

Option 1: Model all protocol instances that use the same CRS as a single instance of a larger protocol (used in, e.g. [CLOS02]).

Drawbacks:

- Modularity of analysis is lost. (Can be mostly regained using UC with joint state [C-Rabin03].)
- Composability with other protocols that use the same CRS is lost (e.g. “deniability”).

How to extend the CRS model?

Option 1: Model all protocol instances that use the same CRS as a single instance of a larger protocol (used in, e.g. [CLOS02]).

Drawbacks:

- Modularity of analysis is lost. (Can be mostly regained using UC with joint state [C-Rabin03].
- Composability with other protocols that use the same CRS is lost (e.g. “deniability”).

Option 2: Directly model the fact that the CRS is available globally to all parties.

The ***global*** common reference string functionality

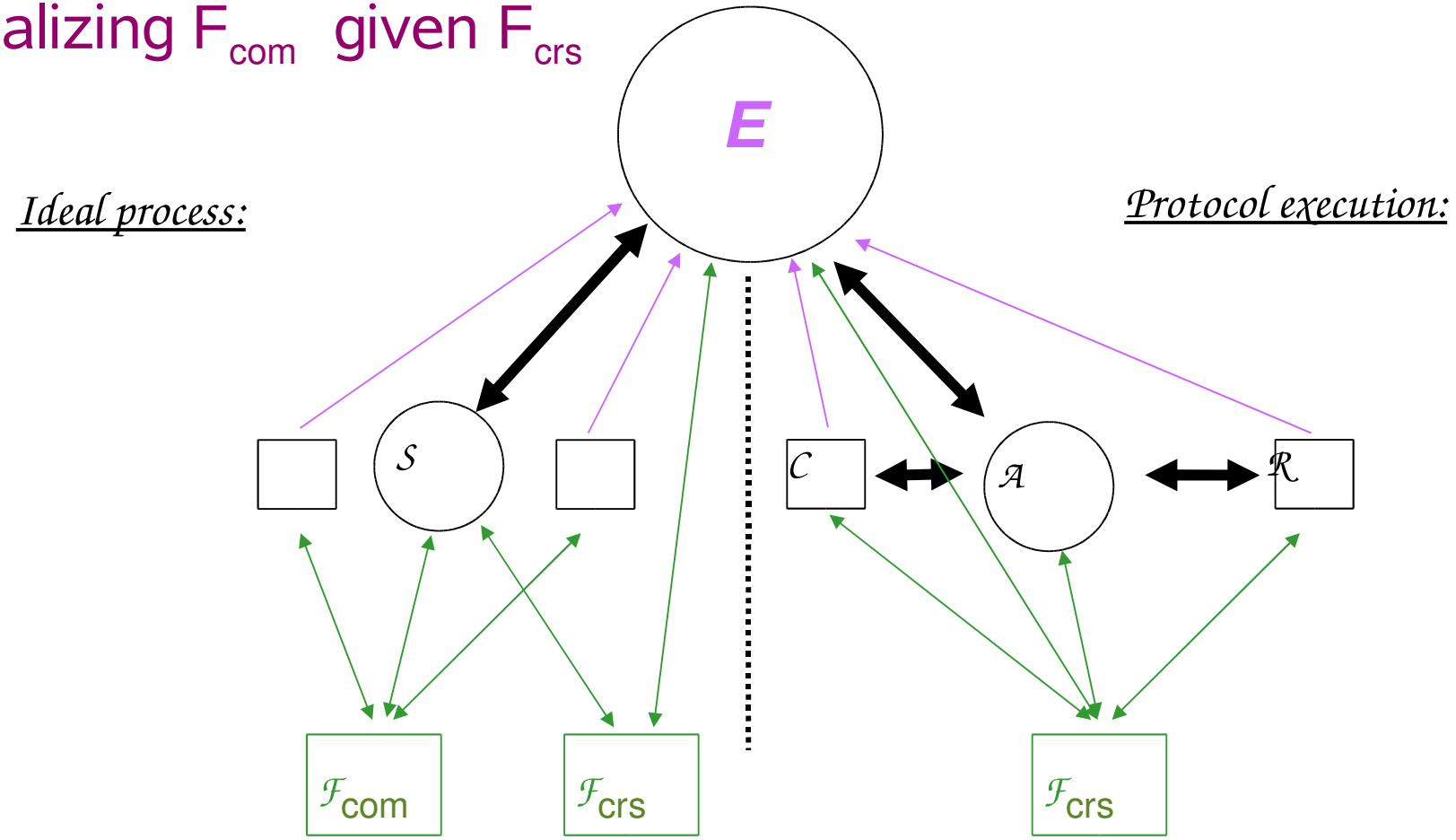
Functionality F_{crs}

(with prescribed distribution D and set R of recipients)

- Choose a value r from distribution D, and send r to the adversary.
- Upon receiving ("CRS") from party P in R, return r to P.

Note: This modeling requires an extension to the original UC formalism (to allow a subroutine to exchange I/O with external entities).
This is the Generalized UC framework of [CDPW07].

Realizing F_{com} given F_{crs}



E has access to the CRS independently of the protocol, and even In the ideal process.

- Consequently, S has to work with an externally determined CRS.
- The impossibility result of [CF01] again holds...
- Can be extended to any “public-only set-up”

Realizing F_{com} given a "global F_{kr}"

F_{gkr} (for "global F_{kr}"):

Modify F_{kr} so that any party can register and retrieve keys.

Then, modify the CF/BCNP protocol so that each party registers a key once for the lifetime of the system.

Analysis still works, assuming "party-wise, non-adaptive corruptions".

Extensions [CDPW07]:

- Can have a short reference string with "ID-based secret keys" for corrupted parties (Augmented CRS model)
- Can withstand adaptive corruptions (by adding interaction)

Set-up models I didn't talk about

- Timing assumption [Kalai,Lindell,Prabhakaran 05]
- Many CRSs, some of which are corrupted [Groth,Ostrovsky 07]
- The “Signature card” setup [Hofheinz,Quade-Muller,Unruh 06]
- Set-up models for obtaining authentication

Modeling authentication with global registration

- All results until now assume ideal authentication (captured as an ideal functionality F_{auth}).
- Obtaining authentication via a protocol requires some trusted set-up.
- Can realize F_{auth} using a simple registration set-up, F_{reg}.
  - But, F_{reg} is not modeled as a global service (it hands the registered values only to the protocol participants [C04].)

Questions:

- How to model and obtain authentication with globally available registration (a "public bulletin board")?
  - Deniable authentication?
  - Standard (non-deniable) authentication?

Conclusion

- Obtaining composable security requires *some* form of trusted set-up.
- There are many different trusted set-up models, each requiring trusting different components and in different ways.
- Our understanding of the nature of security and composability keeps evolving.
- This is a fun and fascinating research area!