Universally Composable Security With Global Setup

Ran Canetti
IBM

Yevgeniy Dodis
NYU

Rafael Pass
Cornell

Shabsi Walfish
NYU

The Trusted Party paradigm [GMW87]

How to decide whether a protocol "securely realizes" a given task?

- Devise an "ideal protocol" for the task:
  - All parties hand their inputs to a "Trusted Party" T
  - T locally computes outputs and hands to parties.
- A protocol "securely realizes" the task if running the protocol "looks like" running the ideal protocol.

But, How to formalize?

The simulation approach

[GMR85,B91,C95,...]

What does

"running protocol P *looks like* running the ideal protocol"

mean?

- For any adversary A that interacts with P
- There exists an adversary S that interacts with T
- Such that no "external distinguisher" can tell an interaction with (P,A) from an interaction with (T,S).

The appeal

A priori, the approach looks great:

- General
- Intuitive
- “Inherently composable”

Do existing formulations deliver?

...It seems so:

- Many formulations [GL90,MR91,B91,DM00,PW00,C95,00,01...]
- Can capture practically any task
- Composability:
  - Non-concurrent [MR91,C00]
  - Concurrent [DM00,PW00,C01]

Trusted Set-up

Trusted set-up is common in cryptography:

- PKI for authentication and encryption
- CRS for protocols

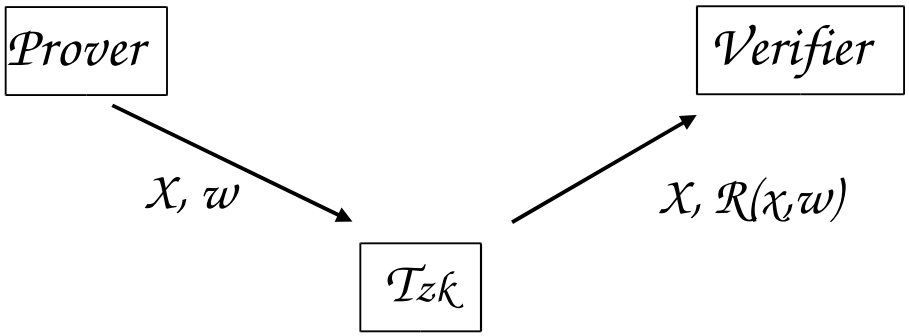
In fact, *some* type of set-up is often essential:

- Authentication
- NIZK
- UC commitment, ZK, general computation...

How do existing definitions fare with trusted set-up?

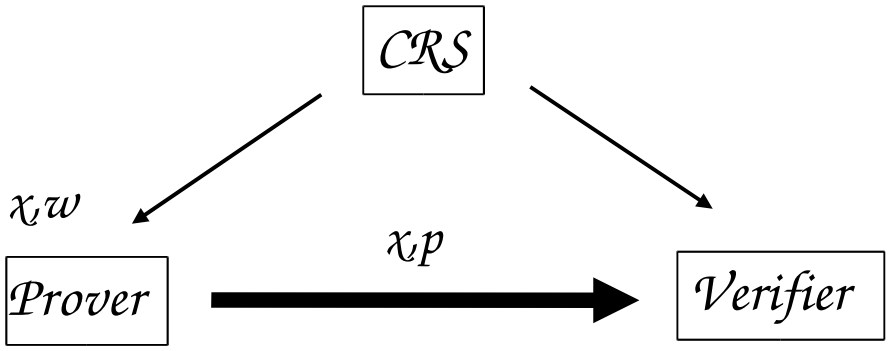
A “warning sign”: Transferability of NIZK in the CRS model

- Reminder: “Ideal Zero-Knowledge” (for relation R(x,w))



A “warning sign”: Transferability of NIZK in the CRS model

- Reminder: Non-Interactive Zero-Knowledge



A “warning sign”: Transferability of NIZK in the CRS model

- “Ideal ZK” is not “transferable”:

  The verifier cannot later “convince” a third party in the verity of the statement.

- In the standard model, ZK protocols have the same property, as expected.
- But, NIZK protocols in the CRS model are easily transferable, assuming the CRS is public...

A “warning sign”: Transferability of NIZK in the CRS model

- “Ideal ZK” is not “transferable”:

  The verifier cannot later “convince” a third party in the verity of the statement.

- In the standard model, ZK protocols have the same property (as expected).
- But, NIZK protocols in the CRS model are easily transferable, assuming the CRS is public...

So, what's going on?

An approach for a solution

[Jakobson-Sako-Impagliazzo96]

- Modify the se-tup: The verifier will have a public key PKv, and a corresponding secret key SKv.
- Modify the protocol. The prover will prove: “Either the statement is correct or I know the secret key that corresponds to Pkv”.

Now, the verifier can no longer use the proof to convince third parties in the verity of the statement.

...or so it seems....

Lingering doubts

- Is the proposed solution good enough?
- How to capture this property?
- Are there other such “loopholes” in the definitional approach, in the presence of set-up?
- Where did the modeling fail?

What about UC security?

Perhaps if we use stronger formulations of the TP paradigm we'll be ok? How about UC security?

- Nope... the same attack works even for UC-NIZK.
- There are also other examples:

  The single-value UC-commitment protocol in [C-Fischlin01] becomes completely malleable when two instances use the same CRS.

Some work-arounds and non-solutions

- Move to a relaxed notion that does not require set-up
  [Prabhakaran-Sahai04, Barak-Sahai05,....]

  But the security guarantees are inherently weaker...

Some work-arounds and non-solutions

- Move to a relaxed notion that does not require set-up
  [Prabhakaran-Sahai04, Barak-Sahai05]

  But the security guarantees are inherently weaker...
- Use “Universal Composition with Joint State (JUC)”
  [C-Rabin03]. But JUC only guarantees security wrt specially designed protocols... in particular the same attack works.

Some work-arounds and non-solutions

- Move to a relaxed notion that does not require set-up
  [Prabhakaran-Sahai04, Barak-Sahai05,....]

  But the security guarantees are inherently weaker...
- Use "Universal Composition with Joint State (JUC)"
  [C-Rabin03]. But JUC only guarantees security wrt specially designed protocols... in particular the same attack works.
- Move to a different set-up: "PKI with knowledge"
  [Barak-C-Neilsen-Pass04]. The protocols are UC and seem "non-transferrable" (akin to [JSI96]).

  But, again, how to capture???

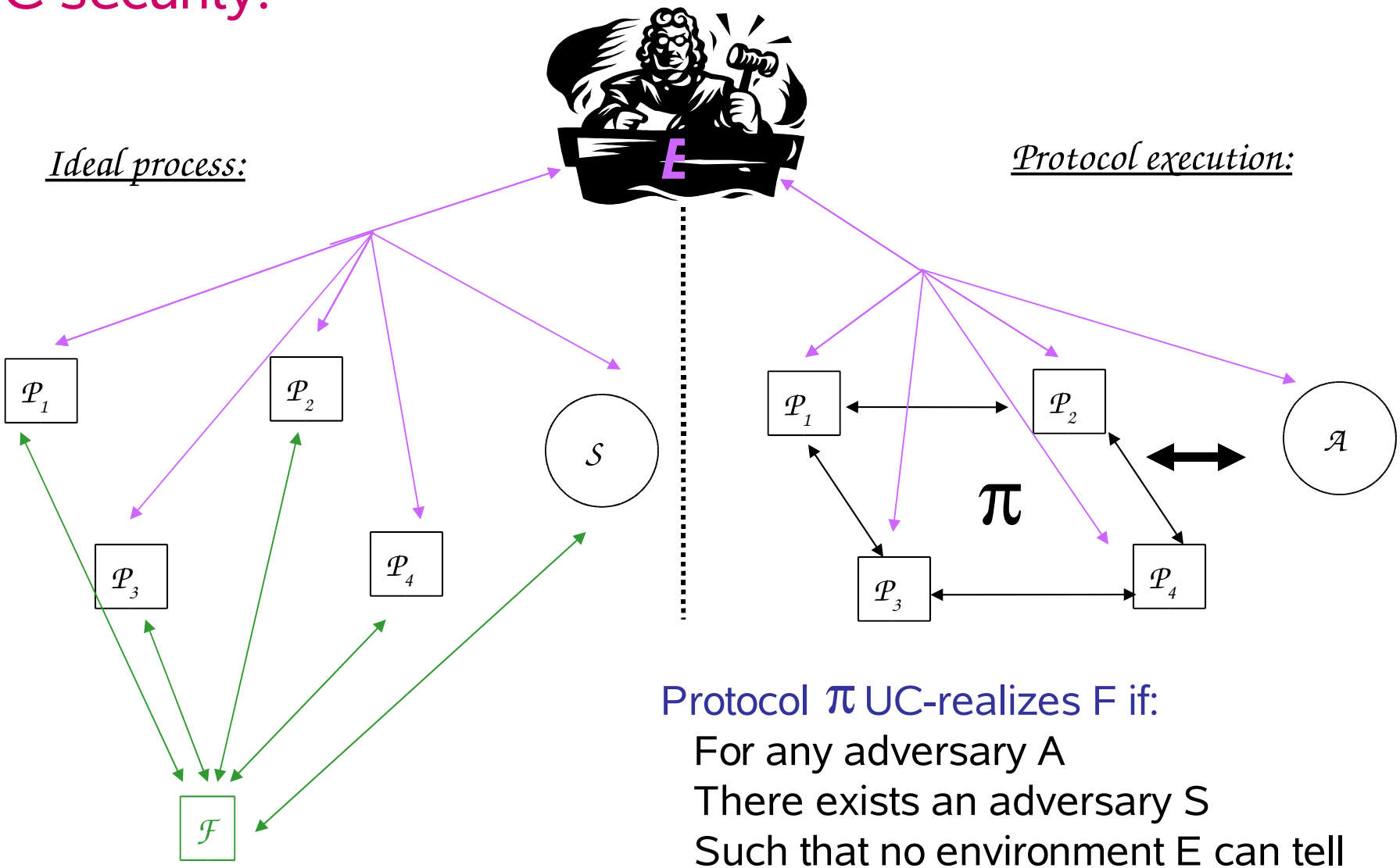
Our Contribution

- A new notion, Generalized UC (GUC) security, that:
  - Is preserved under universal composition
  - Guarantees security even with global setup
  - In particular, guarantees non-transferability

A construction for realizing any functionality in a GUC-secure way, given reasonable ("minimal") setup.

Results with a similar flavor in [Hofeinz,Muller-Quade,Unruh06]

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 UC model postulates a restricted environment:

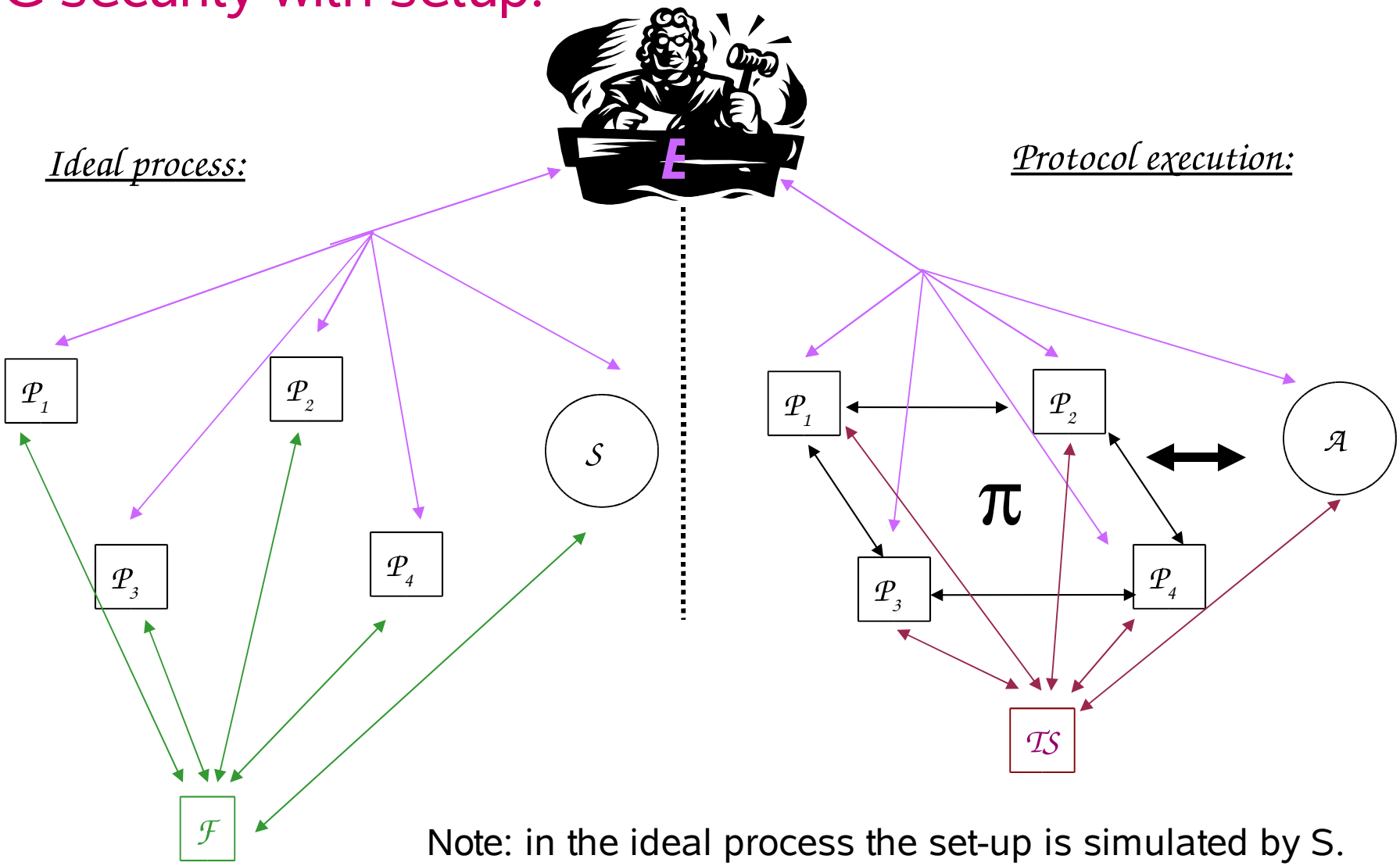
- The environment can interact with a single protocol instance only.
- No subroutines of the parties running the protocol can be directly accessed.

Pro: Greatly simplifies the model and analysis.

Con: The UC theorem holds only for "subroutine respecting" protocols.

In particular, cannot protocols that "share subroutines" with other protocols.

UC security with setup:



Note: in the ideal process the set-up is simulated by S.

This models a set-up that is available only to the protocol execution, and is not available to other protocols.

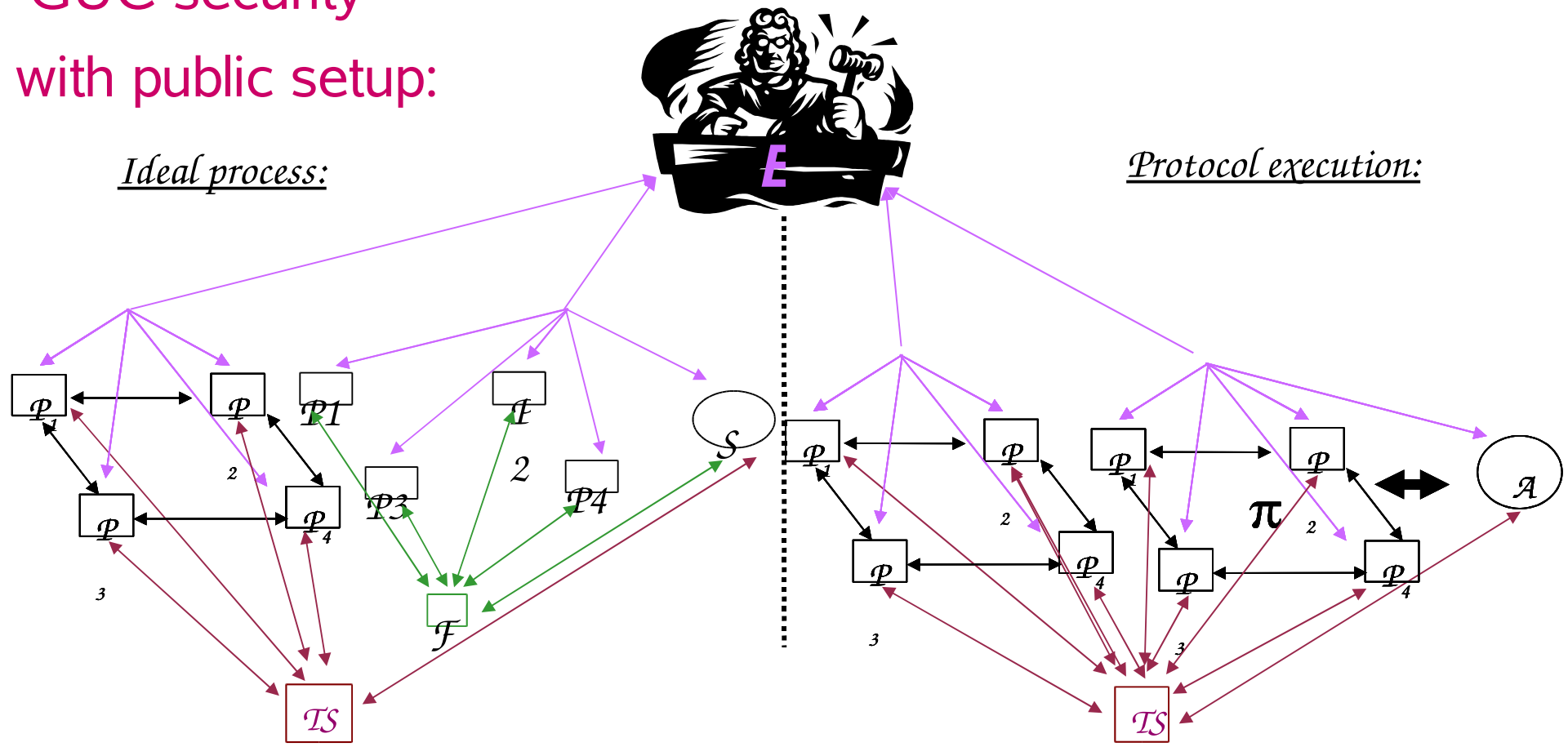
How to extend the model to capture shared subroutines?

Natural extension:

- Allow the environment to interact with any number and type of protocols, even when testing security of a single protocol.
- Allow protocols and subroutines to interact with each other arbitrarily.
- Otherwise, definition remains the same.

We call the resulting notion Generalized UC security.

GUC security with public setup:



Note: The set-up exists even in the ideal model, and the environment has direct access to it.

In particular, the simulator has to deal with an existing set-up.

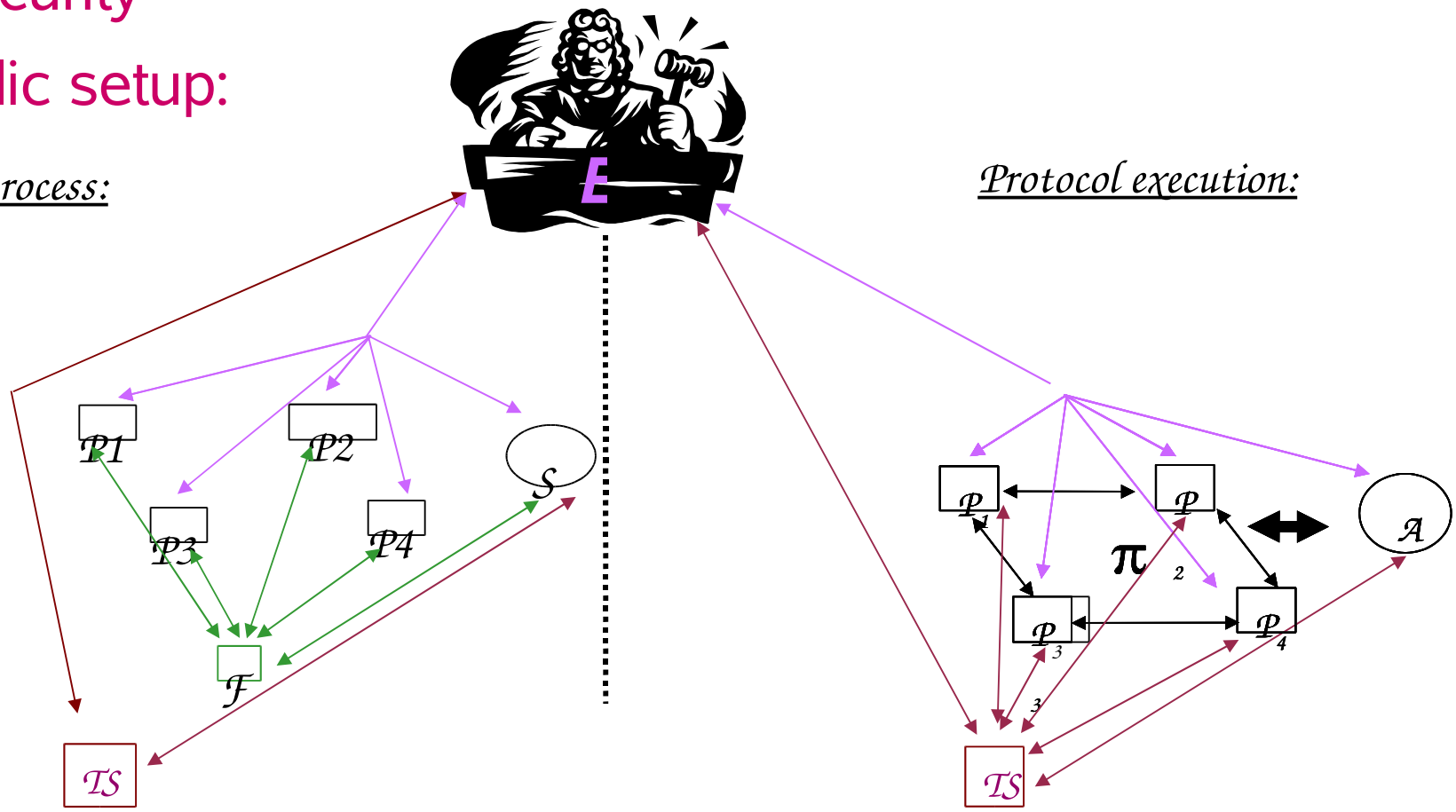
A simplified variant

- Allow the environment to interact with only a *single* other entity other than the protocol instance.
- Allow that entity to interact with the protocol.

We call the resulting notion

Externalized UC (EUC) security.

EUC security with public setup:



Also here, the set-up exists even in the ideal model, and the environment has direct access to it.

Also here, the simulator has to deal with an existing set-up.

Relating the notions:

Def: A protocol P is [F-]subroutine respecting if no subroutine of a party in P takes I/O with any non-subroutine of P [other than a single instance of F].

Thms:

- For F-subroutine respecting protocols, F-EUC-security is equivalent to GUC-security.
- For subroutine-respecting protocols, GUC security is equivalent to UC security.

Composability:

If π GUC-emulates φ then protocol ρ^{π} GUC-emulates protocol ρ^{φ}, for any ρ.

The CRS model

Recall the CRS functionality (for distribution D):

- Choose a string S from D, send S to the adversary.
- Whenever a party asks, reply with S.

UC interpretation: Only parties in the invoking protocol instance can get S.

GUC interpretation: *Any* party can get S.

EUC interpretation: Even the environment can get S.

An impossibility result

Thm: There do not exist two-party protocols that GUC-realize F_{com} in the GUC-CRS model.

Can be extended:

- Other primitives (ZK, OT, [CKL03] functions,...)
- Any set-up that “only gives public information”

Reflections

- Good news: The notion does seem to be capturing something good, since we know that protocols in the CRS model are “inherently bad”...
- Bad news: How can we hope to realize, say, F_{com}, with *any* reasonable set-up?

  (Existing proofs seem to hinge on the simulator's ability to “rig” the set-up...)

We show:

Can regain general realizability with the following set-ups:

Key registration:

- When a party registers, (PK,SK) are chosen.
- Anyone can obtain the PK of any registered party.
- Corrupted parties can get their SKs.

(As in [BCNP04], but here it's the the same global set-up for all protocols)

We show:

Can regain general realizability with the following set-ups:

Key registration:

- When a party registers, (PK,SK) are chosen.
- Anyone can obtain the PK of any registered party.
- Corrupted parties can get their SKs.

(As in [BCNP04], but here it's the the same global set-up for all protocols)

Augmented CRS (ACRS):

- A random string S is chosen and made available to all
- Corrupted parties can obtain a “personalized secret key” that's derived from S and their identities.

More reflections

- How to interpret the provision that only corrupted parties can obtain secret keys?
  - Protocol instructions cannot use the secret keys
  - Parties cannot trust other parties to *not* have their secret key.
- Minimality of ACRS:
  - The interface for honest parties is the same as plain CRS.
  - The impossibility is circumvented by giving secret information only to corrupted parties.

Yet more reflections

- GUC security with these set-ups leaves very little leeway to the simulator:
  - Cannot affect the set-up information of uncorrupted parties
  - Only learns the secrets of corrupted parties when the environment learns them
  - Cannot rewind the environment
  - Can only “fake” the coin-tosses of the uncorrupted parties.
- Seems “almost” as good as simulation in the plain model...

Main theorem

Assuming dense cryptosystems, there exists an ACRS-hybrid protocol for GUC-realizing any well-formed functionality, even for *adaptive corruptions with no erasures*, as long as the corruptions are PID-wise.

Def: Corruptions are PID-wise if all parties in the system, in all protocol instances, that have the same "party identity (PID)" must be corrupted together.

The construction

- Enough to show how to GUC-realize F_{com}.

  Given F_{com}, can use the [C-Lindell-Ostrovsky-Sahai] construction (with the [Pass] transformation to 1:M ZK).
- For non-adaptive corruptions and key registration set-up, the [BCNP] construction works.
- To deal with adaptive corruptions we use a special coin-tossing protocol that allows choosing an appropriate "CRS" per commitment.
- To make do with ACRS, we make use an "identity-based trapdoor commitment scheme".

Further research

- How to model standard “global PKI” for authentication?
  - How to capture “deniable authentication”?
  - How to capture standard PKI-based protocols, assuming global PK? (Authentication, Key Exchange, etc...)
- Can we find other set-up assumptions that would be be more readily implemented in reality and still allow for general UC feasibility? (e.g., the [HQU] set-up...)