Collusion-Preserving Computation

Joël Alwen (ETH Zürich)
Jonathan Katz (U. Maryland)
Ueli Maurer (ETH Zürich)
Vassilis Zikas (U. Maryland)

Overview

- Motivation & Goals
- Definition
- Fall-back Security
- Synchronization Pollution
- Implications for Game Theory
- Future Directions

Goals (1)

Goals (1)

- Primary Goal: A realization notion bounding the capabilities of deviating coalitions even in the presence of arbitrary composition.

Goals (1)

- Primary Goal: A realization notion bounding the capabilities of deviating coalitions even in the presence of arbitrary composition.
  - “R realizes F” = R can be used in place of F

Goals (1)

- Primary Goal: A realization notion bounding the capabilities of deviating coalitions even in the presence of arbitrary composition.
    - “R realizes F” = R can be used in place of F
    - “capabilities of deviating coalitions” = such that even collaborating “dishonest” players can do no more with R then they could with F

Goals (1)

- Primary Goal: A realization notion bounding the capabilities of deviating coalitions even in the presence of arbitrary composition.
  - "R realizes F" = R can be used in place of F
  - "capabilities of deviating coalitions" = such that even collaborating "dishonest" players can do no more with R then they could with F
  - "arbitrary composition" = regardless of any concurrent activities in which they may be involved.

Example Use Cases

Example Use Cases

- Composable Game Theory.
    - Extreme case of deviating coalitions.

Example Use Cases

- Composable Game Theory.
  - Extreme case of deviating coalitions.
- Collusion-Free (CF) MPC robust in the presence of side-channels.
  - CF (provably) not concurrently composable

Example Use Cases

- Composable Game Theory.
    - Extreme case of deviating coalitions.
- Collusion-Free (CF) MPC robust in the presence of side-channels.
    - CF (provably) not concurrently composable
- Other (intuitive) examples requiring bounds on collaborating dishonest players.
    - Incoercability: Coercer/Informant & Coercee.
    - Auctions: Bid fixing by corrupt bidders.
    - Bounded Isolation: Useful for say, poker or bridge

Goals (2)

Goals (2)

- Generic definition independent of communication resource R.
  - Better for comparing different constructions.
  - Allows investigating minimal properties for resource R used to realize a given F.

Goals (2)

- Generic definition independent of communication resource R.
  - Better for comparing different constructions.
  - Allows investigating minimal properties for resource R used to realize a given F.
- Non-triviality: strong fall-back security even if R “miss-behaves”.

Goals (2)

- Generic definition independent of communication resource R.
  - Better for comparing different constructions.
  - Allows investigating minimal properties for resource R used to realize a given F.
- Non-triviality: strong fall-back security even if R “miss-behaves”.
- Concrete communication resource R & construction for many F.

Goals (2)

- Generic definition independent of communication resource R.
  - Better for comparing different constructions.
  - Allows investigating minimal properties for resource R used to realize a given F.
- Non-triviality: strong fall-back security even if R “miss-behaves”.
- Concrete communication resource R & construction for many F.
- Explore implications for composable Game

Related Work

Related Work

- SFE/MPC [GMW, BGW,...]
  - First generic realization notions.
    - Not generally composable
    - Gives deviating coalitions arbitrary (internal) capabilities (monolithic adversary)

Related Work

- SFE/MPC [GMW, BGW,...]
  - First generic realization notions.
    - Not generally composable
    - Gives deviating coalitions arbitrary (internal) capabilities (monolithic adversary)
- Arbitrary composition [Can, PW, CLOS, CDPW,...]
  - Exa: UC, GUC, JUC, etc.
    - But monolithic adversary

Related Work

- SFE/MPC [GMW, BGW,...]
  - First generic realization notions.
    - Not generally composable
    - Gives deviating coalitions arbitrary (internal) capabilities (monolithic adversary)
- Arbitrary composition [Can, PW, CLOS, CDPW,...]
  - Exa: UC, GUC, JUC, etc.
    - But monolithic adversary
- Collusion-Free (CF) computation [LMPS, ILM, ASV, AKLPSV]
  - Bounds deviating coalitions (via split adversaries)

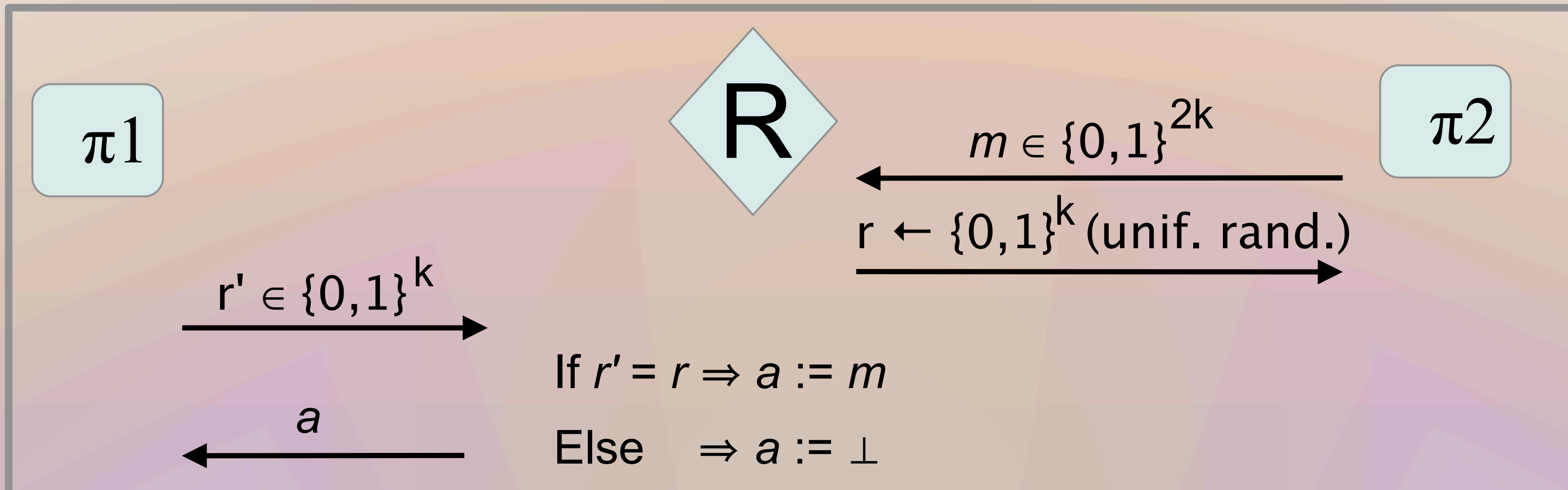
CF is not Composable

CF is not Composable

- F = 2-party null functionality (does nothing)
- Define R and protocol π = ($\pi 1$, $\pi 2$)

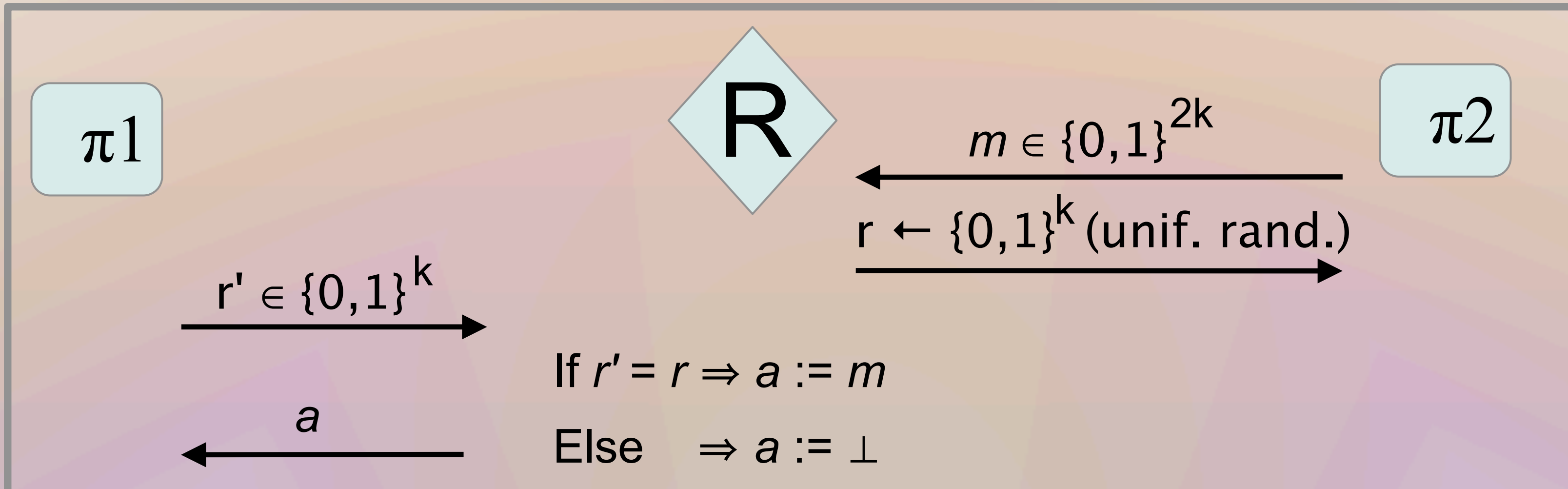
CF is not Composable

- F = 2-party null functionality (does nothing)
- Define R and protocol π = ($\pi 1$, $\pi 2$)



CF is not Composable

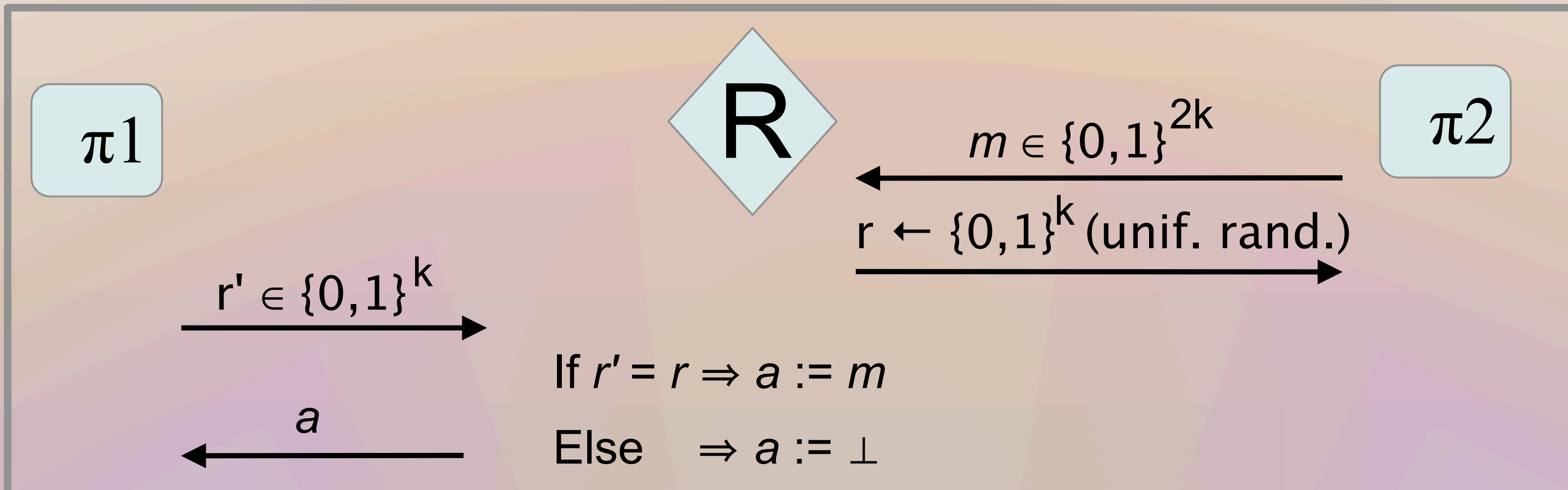
- F = 2-party null functionality (does nothing)
- Define R and protocol π = ($\pi 1$, $\pi 2$)



- r is uniform random and F allows no communication between simulators. $\Rightarrow$ Can always simulate for $\pi 1$ with $a = \perp$.
  $\Rightarrow$ R CF-realizes F via π.

CF is not Composable

- F = 2-party null functionality (does nothing)
- Define R and protocol π = ($\pi 1$, $\pi 2$)



- r is uniform random and F allows no communication between simulators. $\Rightarrow$ Can always simulate for $\pi 1$ with $a = \perp$.
  $\Rightarrow$ R CF-realizes F via π.
- Now compose with C ; a k-bit channel from $P2 \rightarrow P1$. Use it transmit r. So P2 can learn m from R . But using F & C the simulators can communicate at most k. I.e. π is no longer simulatable!

Composable CF → Collusion-Preservation

Composable CF → Collusion-Preservation

- Goal: Add composability to CF.

Composable CF → Collusion-Preservation

- Goal: Add composability to CF.
- Idea: Add an environment (as in UC-style realization notions) to CF → CP.

Composable CF → Collusion-Preservation

- Goal: Add composability to CF.
- Idea: Add an environment (as in UC-style realization notions) to CF → CP.
- Immediate results:
    - Dummy (adversary) lemma and (G)UC composition theorems hold essentially unchanged.

Composable CF → Collusion-Preservation

- Goal: Add composability to CF.
- Idea: Add an environment (as in UC-style realization notions) to CF → CP.
- Immediate results:
  - Dummy (adversary) lemma and (G)UC composition theorems hold essentially unchanged.
  - CP strictly generalizes (G)UC realization notions.

Construction (1)

Construction (1)

- CP Construction for F using resource R:
  - Trivial Idea: Resource R = Functionality F.

Construction (1)

- CP Construction for F using resource R:
    - Trivial Idea: Resource R = Functionality F.
- Issues:
    - R depends on F
        - We show that to some extent such a dependency is unavoidable.
        - However at least R must only be “programmable” but not fully “non-uniform”.

Construction (1)

- CP Construction for F using resource R:
  - Trivial Idea: Resource R = Functionality F.
- Issues:
  - R depends on F
    - We show that to some extent such a dependency is unavoidable.
    - However at least R must only be “programmable” but not fully “non-uniform”.
  - If R mis-behaves all bets are off.
    - Usually we don't care about this case. But trust is a rare commodity.

Fallback Security

Fallback Security

- Def. “Fallback Security” = Security attained when protocol is run using an arbitrary communication resource.

Fallback Security

- Def. "Fallback Security" = Security attained when protocol is run using an arbitrary communication resource.
- Example: Protocol π CP-realizes R from F with GUC-Fallback Security.
    - If π is run with R then F is CP-realized.
    - If π is run with any R* then F is GUC-realized.

Fallback Security

- Def. “Fallback Security” = Security attained when protocol is run using an arbitrary communication resource.
- Example: Protocol π CP-realizes R from F with GUC-Fallback Security.
    - If π is run with R then F is CP-realized.
    - If π is run with any R* then F is GUC-realized.
- Now trivial construction no longer works because it achieves no fallback security.

Construction (2)

Construction (2)

- Recall CF construction of Mediated Model of [ASV, AKLPSV]. Idea: “assisted SFE in the mediator's head”
    - For functionality F, let protocol π = GMW(F).
    - “Mediator” resource M runs π on behalf of players “in her head”.
    - Player Pi's internal state in π shared between Pi and M.
    - Next protocol msg generated and Pi's state updated via 2-party SFE between Pi and M.

Construction (2)

- Recall CF construction of Mediated Model of [ASV, AKLPSV]. Idea: “assisted SFE in the mediator's head”
    - For functionality F, let protocol π = GMW(F).
    - “Mediator” resource M runs π on behalf of players “in her head”.
    - Player Pi's internal state in π shared between Pi and M.
    - Next protocol msg generated and Pi's state updated via 2-party SFE between Pi and M.
- CP Construction Idea:
    - Use π = GUC(F) with setup S.
        - GUC allows us to reuse S across protocols.

Synchronization Pollution (1)

Synchronization Pollution (1)

- Did we get CP with GUC fall-back?
    - No! “Synchronization Pollution”

Synchronization Pollution (1)

- Did we get CP with GUC fall-back?
    - No! “Synchronization Pollution”
- Recall Intuitive Goal: Ensure corrupt colluding parties get no more from R then from F.
    - Technically: Can simulate with split simulators

Synchronization Pollution (1)

- Did we get CP with GUC fall-back?
  - No! “Synchronization Pollution”
- Recall Intuitive Goal: Ensure corrupt colluding parties get no more from R then from F.
  - Technically: Can simulate with split simulators
- Solutions:
  1. Remove subliminal communication channels (“steganography freeness”) [Sim84]
  2. Remove “randomness pollution” for CF [LMS05, ILM05,...]

Synchronization Pollution (2)

Synchronization Pollution (2)

- This work: Identify and mitigate new security concern.

Synchronization Pollution (2)

- This work: Identify and mitigate new security concern.
- Def. “Synchronization Pollution” = Adversaries obtain more synchronization of events using R then using F.

Synchronization Pollution (2)

- This work: Identify and mitigate new security concern.
- Def. "Synchronization Pollution" = Adversaries obtain more synchronization of events using R then using F.
    - Intuitive problem: more observable events from R than from F ⇒ Adversaries more coordinated.

Synchronization Pollution (2)

- This work: Identify and mitigate new security concern.
- Def. “Synchronization Pollution” = Adversaries obtain more synchronization of events using R then using F.
    - Intuitive problem: more observable events from R than from F ⇒ Adversaries more coordinated.
    - Technical Problem: F doesn't provide simulators enough synchronization for them to coordinate the events in their on-line simulations.
        - Not an issue for CF because distinguisher (unlike environment) is off-line.

Mitigating Synchronization Pollution

Mitigating Synchronization Pollution

- Idea: Resource R runs ρ = GUC(F) “in the head”. Minimize number of observable events generated by assisting R.

Mitigating Synchronization Pollution

- Idea: Resource R runs ρ = GUC(F) “in the head”. Minimize number of observable events generated by assisting R.
- Problem: ρ is multi-round $\Rightarrow$ Has many public ordered events.
- Q: What is minimal synchronization obtained from 2-party SFEs used to “assist” R in running ρ?

Mitigating Synchronization Pollution

- Idea: Resource R runs ρ = GUC(F) “in the head”. Minimize number of observable events generated by assisting R.
- Problem: ρ is multi-round $\Rightarrow$ Has many public ordered events.
- Q: What is minimal synchronization obtained from 2-party SFEs used to “assist” R in running ρ?
- A: Surprisingly, only output-delivery synchronization.
    - Ideal World: F delivers output only after players activated enough to “fuel” an execution of ρ.
    - Technically: 2-party SFEs now hide all events in ρ.
    - e.g. Round number? Message received? From who? Message sent? To who? State changed? (!!!)
    - [AKLPSV]: hides only internal state of Pj and message contents for ρ.

Result

Result

- Show the necessity of several properties of our real-world resource R.
    - Probabilistic, Isolating, Programmable.

Result

- Show the necessity of several properties of our real-world resource R.
    - Probabilistic, Isolating, Programmable.

⇒ Rule out using most standard resources for realizing practically any interesting F.

- Broadcast channel, insecure/secure/perfect channels.

Result

- Show the necessity of several properties of our real-world resource R.
  - Probabilistic, Isolating, Programmable.

⇒ Rule out using most standard resources for realizing practically any interesting F.

  - Broadcast channel, insecure/secure/perfect channels.

⇒ Minimality of Mediator resource.

Result

- Show the necessity of several properties of our real-world resource R.
  - Probabilistic, Isolating, Programmable.

⇒ Rule out using most standard resources for realizing practically any interesting F.

  - Broadcast channel, insecure/secure/perfect channels.

⇒ Minimality of Mediator resource.

- **Theorem**: For a large class of F we give a resource and protocol that CP-realize F with GUC-fallback.

Applications to GT

Applications to GT

1) Define a model of rational, computational and concurrent mediated game play

    Goal: Bring GT models closer to reality (Crypto :)).

    Principle: Local actions. Global intentions and consequences

Applications to GT

1) Define a model of rational, computational and concurrent mediated game play

   Goal: Bring GT models closer to reality (Crypto :)).

   Principle: Local actions. Global intentions and consequences

2) Show how to replace ideal mechanism with cryptographic protocol games on a network s.t.

   - Game theorists can design and analyze ideal and fully trusted mechanisms
   - but games can be played by computers over (special) networks s.t.
   - less trust placed in network than mechanism achieving essentially the same game.

Future Directions

Future Directions

- Further constructions.
  - Weaker fallback → realize more funcs. more efficiently.
  - When can R be stateless?
  - Can output synchronization be removed from F?
  - Efficient constructions for auctions?

Future Directions

- Further constructions.
    - Weaker fallback → realize more funcs. more efficiently.
    - When can R be stateless?
    - Can output synchronization be removed from F?
    - Efficient constructions for auctions?
- New security notions leveraging split-simulators.
    - Example: Capturing enforced properties like incoercability.
    - Currently: if a single process (ITI) on a machine is corrupted entire party is considered corrupt. Can we do better? What do we get from Sandboxes, VMs, chroot jails, restricted UIDs? E.g LUC [CV12]

Future Directions

- Further constructions.
  - Weaker fallback → realize more funcs. more efficiently.
  - When can R be stateless?
  - Can output synchronization be removed from F?
  - Efficient constructions for auctions?
- New security notions leveraging split-simulators.
  - Example: Capturing enforced properties like incoercability.
  - Currently: if a single process (ITI) on a machine is corrupted entire party is considered corrupt. Can we do better? What do we get from Sandboxes, VMs, chroot jails, restricted UIDs? E.g LUC [CV12]
- Wanted: Local stability notion for Concurrent GT.

Future Directions

- Further constructions.
  - Weaker fallback → realize more funcs. more efficiently.
  - When can R be stateless?
  - Can output synchronization be removed from F?
  - Efficient constructions for auctions?
- New security notions leveraging split-simulators.
  - Example: Capturing enforced properties like incoercability.
  - Currently: if a single process (ITI) on a machine is corrupted entire party is considered corrupt. Can we do better? What do we get from Sandboxes, VMs, chroot jails, restricted UIDs? E.g LUC [CV12]
- Wanted: Local stability notion for Concurrent GT.
- Relations to Abstract Cryptography framework [MR11].

Thank You!