Round-Efficient Multi-party Computation with a Dishonest Majority

Jonathan Katz, U. Maryland

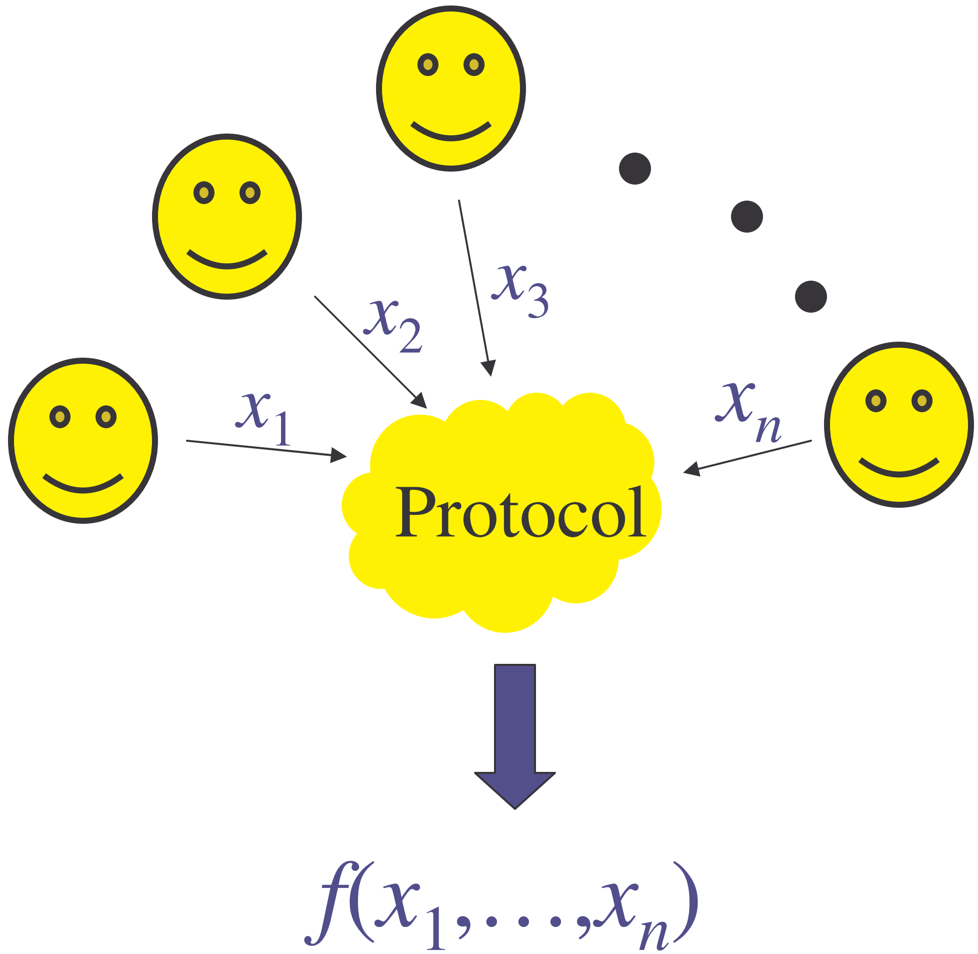
Rafail Ostrovsky, Telcordia

Adam Smith, MIT

Longer version on **http://theory.lcs.mit.edu/~asmith**

Multi-party Computation [GMW87]

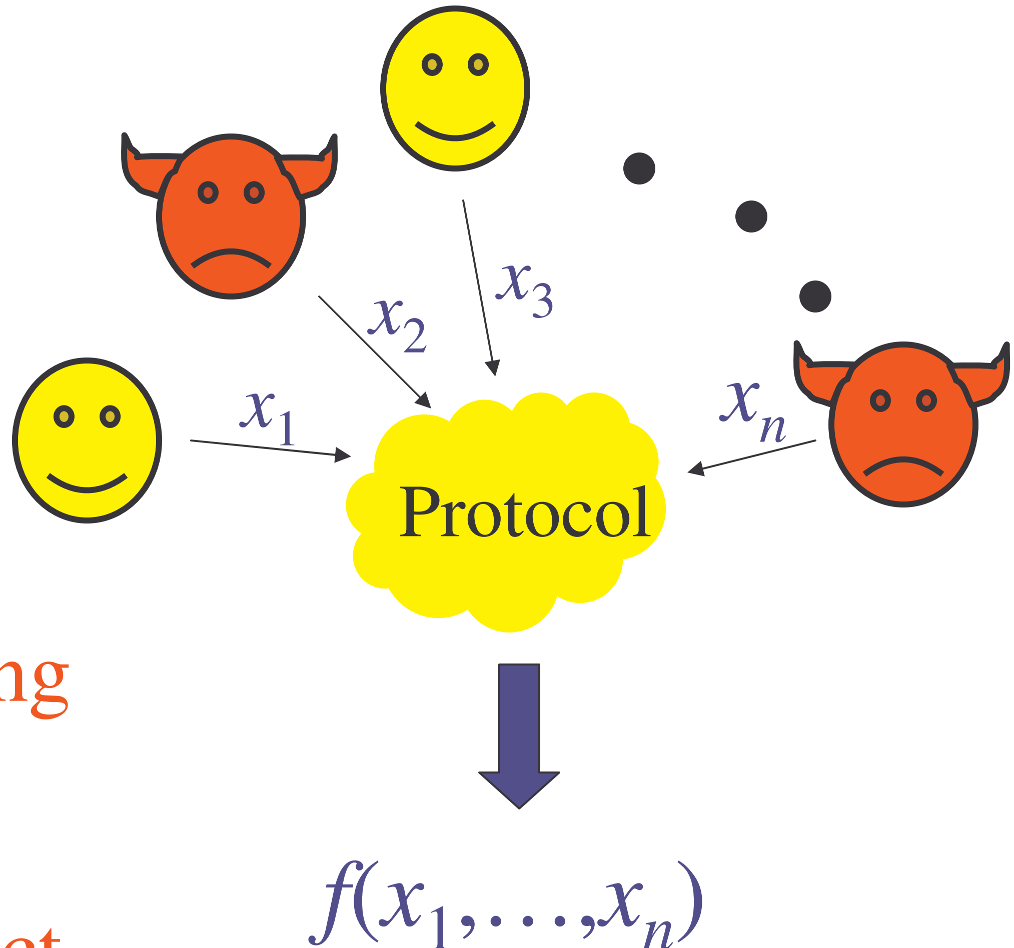
- Also called "Secure Function Evaluation"
- Network of n players
- Each has input x_i
- Want to compute $f(x_1,\ldots,x_n)$ for some known function f
- *E.g.* electronic voting



Multi-party Computation [GMW87]

Even if t out of n

players try to cheat:

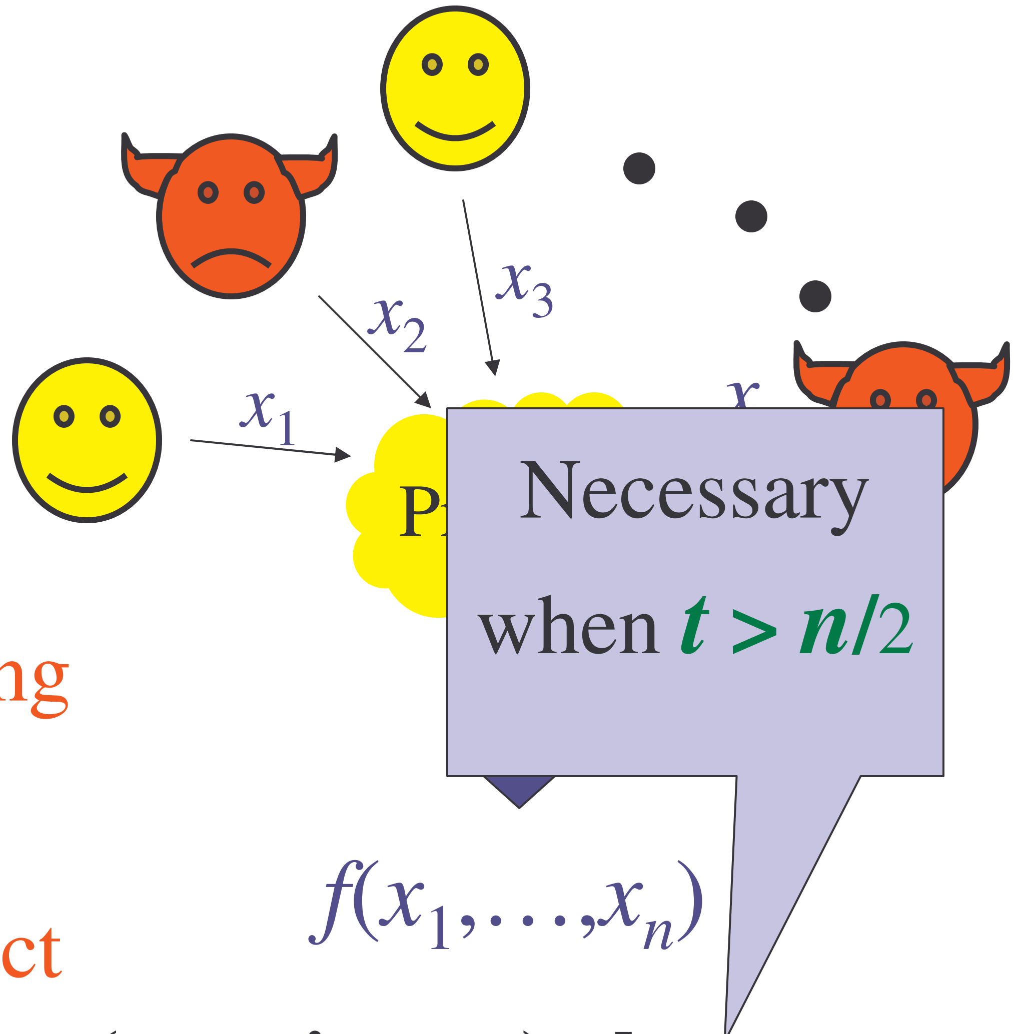


1. Cheaters learn nothing (except output)
2. Cheaters cannot affect output

Multi-party Computation [GMW87]

Even if t out of n

players try to cheat:



1. Cheaters learn nothing (except output)
2. Cheaters cannot affect output **except to force (unanimous) abort**

Round-efficient MPC tolerating any $t < n$

For any PPT $f(\ )$, we get (abortable, unfair) MPC:

- In $O(\log n)$ rounds… with black-box simulation
- In $O(1)$ rounds… with non-black-box simulation

- No assumption of Common Random String, but:
  - Given CRS, MPC takes $O(1)$ rounds [BMR, CLOS]
  - This talk: how to generate a CRS from scratch fast?

Review: Standard Synchronous Model

- Synchronous network of n players (= randomized TM's)
- Authenticated, unblockable Broadcast Channel
- Adversary corrupts $t < n$ players
  - Malicious coordination of corrupted players
  - Choice of corruptions is static (= before start of protocol)
  - Messages may be rushed
- Computationally bounded adversary

No initial common random string

Big Picture: Active Adversary

$t < n/2$

- O(depth) rds, unconditional security, adaptive [GMW87, CDDHR99]
- $O(1)$ rounds, static [GMW87, BMR90]

$t \geq n/2$

(Abortable)

- Robustness and fairness impossible [Cleve,GMW]
- $O(n+k)$ rounds static (?) […,BG,GL]
- $O(\log n)$ static with black box simulation
- $O(1)$ static with non-black-box simulation

Rest of talk

- Reduction of MPC to "simulatable coin-flipping"

Two protocols

1. $O(\log n)$ round protocol (black box)

   based on Chor-Rabin proof scheduling

2. $O(1)$ round protocol (non-black-box)

   based on Barak's non-malleable coin-flipping

Simulatable Coin-Flipping is Enough

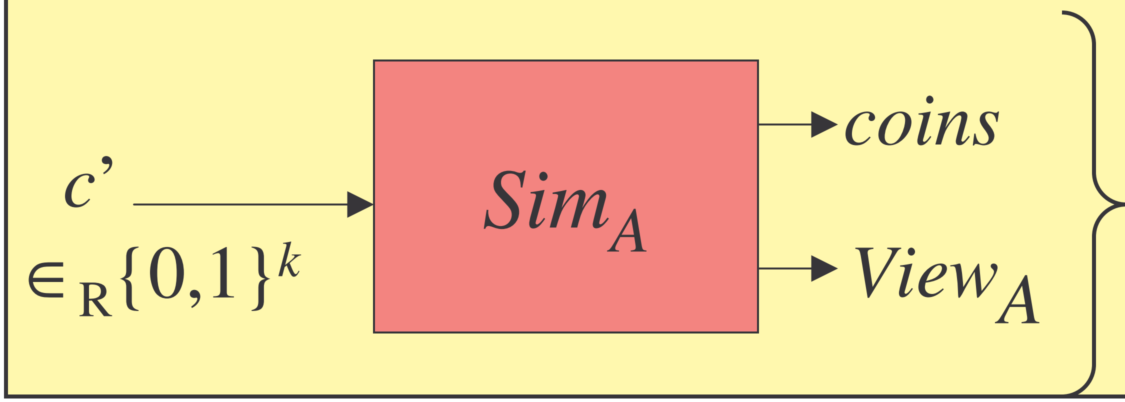
- Honest-but-Curious adversary:

  [BMR90] $O(1)$ rounds for any $t < n$

- Intuition: to go from Honest-But-Curious to Active, we want independence of zero-knowledge proofs [GMW]
- Possible in $\Omega(n)$ rounds (sequential proofs)
- Possible in $O(1)$ rounds [CLOS90]
  - Need a common random string
- To get CRS from scratch: simulatable coin-flipping

Simulatable Coin-Flipping I

$\forall$ PPT adversaries A, $\exists$ PPT Sim_A :



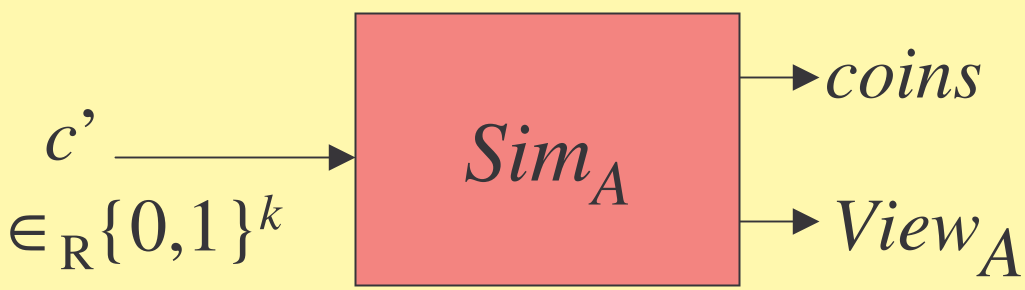
- Indistinguishable from real execution
- $coins \in \{c', \perp\}$

Output k coin flips (or abort) so that:

1) Adversary can bias outcome only by sometimes aborting
2) Simulator can set outcome to any desired string (needed for composition theorems)

Simulatable Coin-Flipping II

$\forall$ PPT adversaries A, $\exists$ PPT Sim_A :



- Indistinguishable from real execution
- $coins \in \{c', \perp\}$

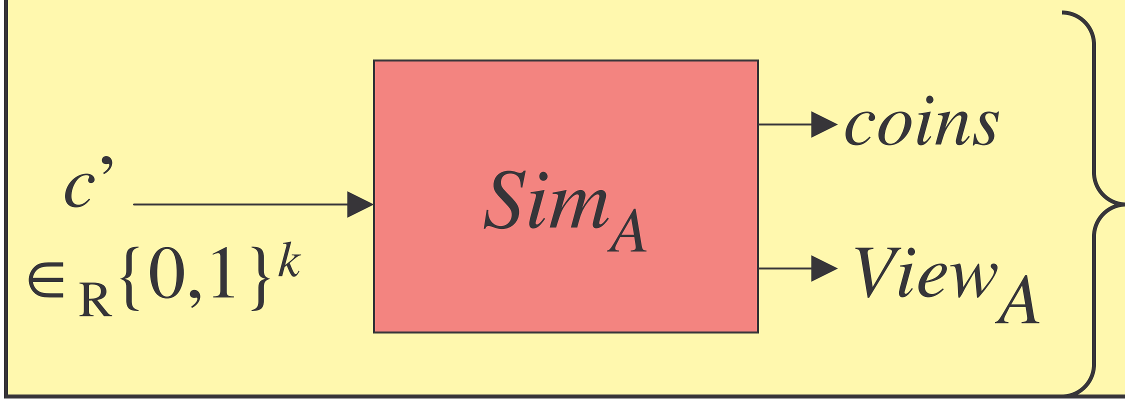
Composition Lemma:

Simulatable coin-flipping
+
MPC protocol based on CRS

= Secure MPC protocol (from scratch)

Simulatable Coin-Flipping III

$\forall$ PPT adversaries A, $\exists$ PPT Sim_A :



- Indistinguishable from real execution
- $coins \in \{c', \perp\}$

Two protocols:

- Proof scheduling of Chor-Rabin: $O(\log n)$ rounds
- Non-malleability technique of Barak: $O(1)$ rounds

Simulatable CF: Protocol Outline [Lindell02]

I) For all i:
1. P_i m_i = Commit(r_i)
2. P_i proves knowledge of r_i

II) For all i:
3. P_i r_i (no decommitment)
4. P_i proves consistency with m_i

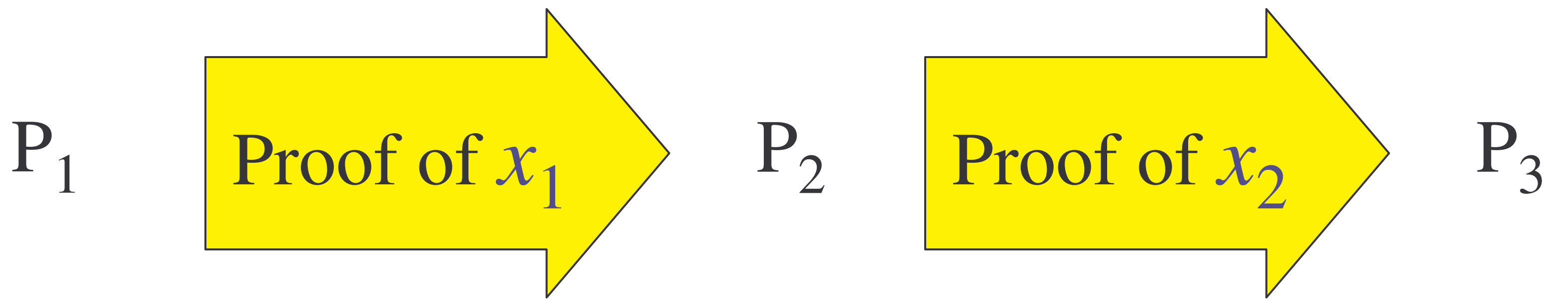
III) Output coins = r_1 XOR r_2 XOR … XOR r_n

Proofs must overlap to get $o(n)$ rounds

Simulator must
- **Extract** from cheaters
- Lie about x_i (i.e. **falsify** proofs)

Problem: Malleability of Proofs

- When proofs overlap, bad things can happen:

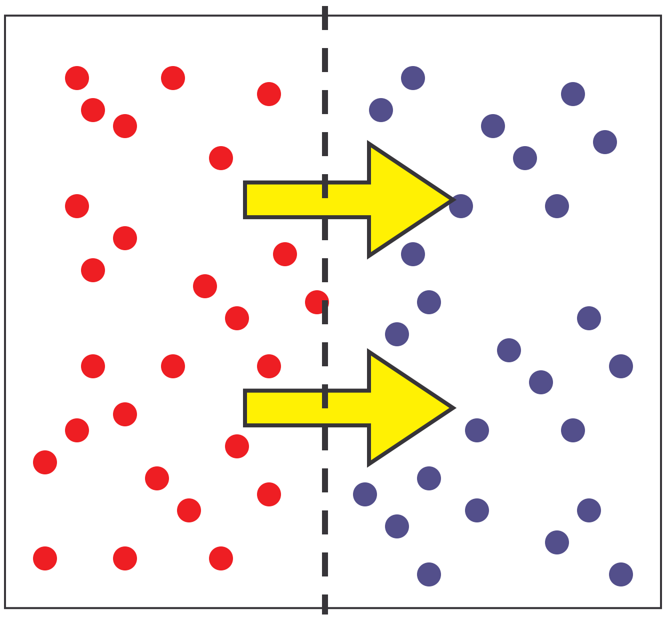


- P_2 can choose x_2 to depend on x_1
- Protocols often provably broken
- Non-malleable Zero-Knowledge [DDN]:
  - Resists this attack
  - Huge round complexity*

* = more later in talk

Chor-Rabin Proof Scheduling

- For all i: P_i must prove some statement x_i in ZK
- $\log n$ phases, each with 2 blocks

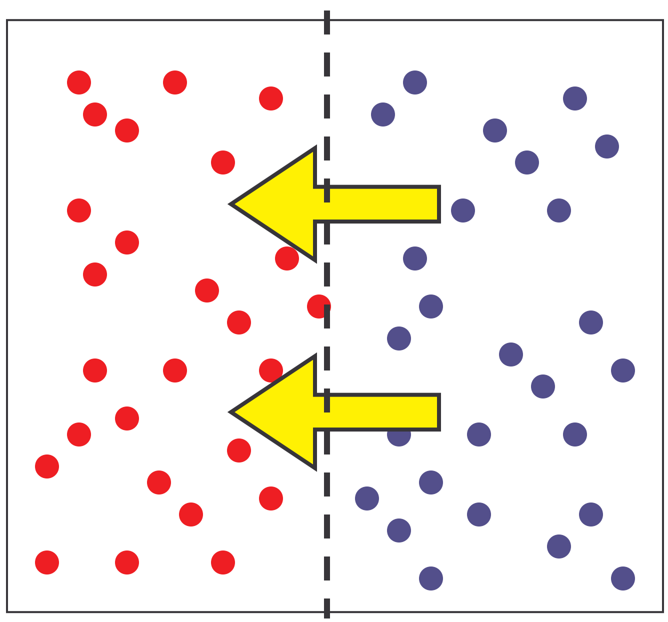


- Each phase:
  Players either **blue** or **red**
- At phase t:
  Blue = $\{P_i \mid t$-th bit of i is **0**$\}$
  Red = $\{P_i \mid t$-th bit of i is **1**$\}$
- 1st block: **Red** prove to **Blue**
  2nd block: **Blue** prove to **Red**

At every point, each player is **either** prover **or** verifier

Chor-Rabin Proof Scheduling

- For all i: P_i must prove some statement x_i in ZK
- $\log n$ phases, each with 2 blocks

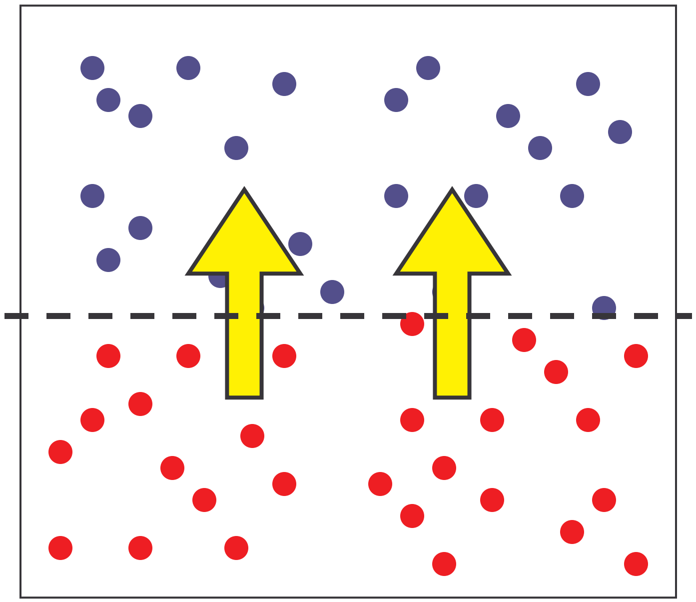


- Each phase:
  Players either **blue** or **red**
- At phase t:
  Blue = $\{P_i \mid t$-th bit of i is $\mathbf{0}\}$
  Red = $\{P_i \mid t$-th bit of i is $\mathbf{1}\}$
- 1st block: **Red** prove to **Blue**
  2nd block: **Blue** prove to **Red**

At every point, each player is **either** prover **or** verifier

Chor-Rabin Proof Scheduling

- For all i: P_i must prove some statement x_i in ZK
- $\log n$ phases, each with 2 blocks

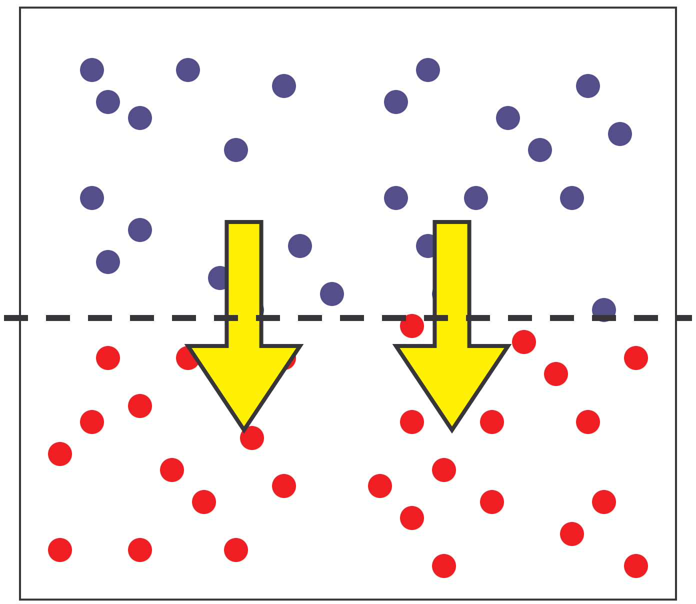


- Each phase:
  Players either **blue** or **red**
- At phase t:
  Blue = $\{P_i \mid t\text{-th bit of } i \text{ is } \mathbf{0}\}$
  Red = $\{P_i \mid t\text{-th bit of } i \text{ is } \mathbf{1}\}$
- 1st block: **Red** prove to **Blue**
  2nd block: **Blue** prove to **Red**

At every point, each player is **either** prover **or** verifier

Chor-Rabin Proof Scheduling

- For all i: P_i must prove some statement x_i in ZK
- $\log n$ phases, each with 2 blocks



- Each phase:
  Players either **blue** or **red**
- At phase t:
  Blue = $\{P_i \mid t$-th bit of i is $\mathbf{0}\}$
  Red = $\{P_i \mid t$-th bit of i is $\mathbf{1}\}$
- 1st block: **Red** prove to **Blue**
  2nd block: **Blue** prove to **Red**

At every point, each player is **either** prover **or** verifier

Chor-Rabin Scheduling: Analysis

- At every point, each player is **either** prover **or** verifier **but never both**
- For every pair i,j:

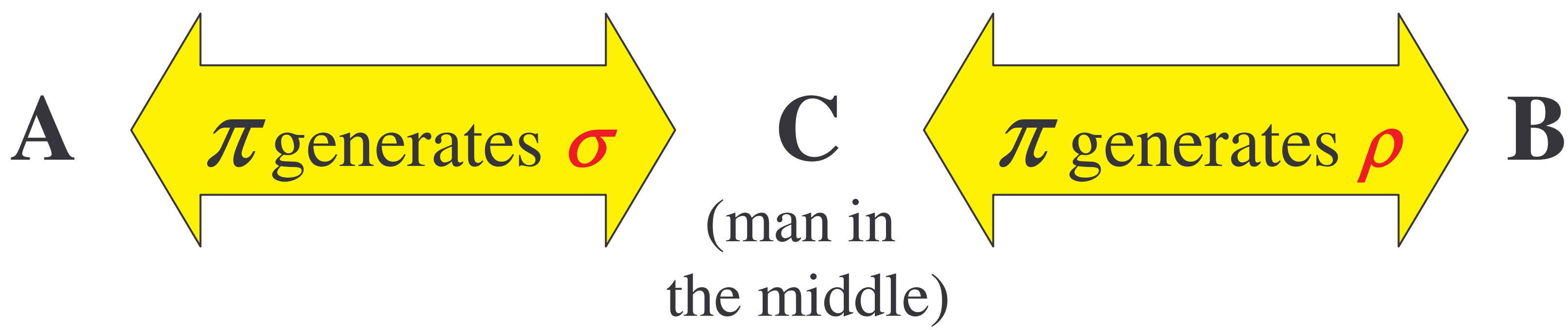
  Eventually P_i proves to P_j and P_j proves to P_i
- Simulator who controls a **single honest player** can
  - **Falsify** all proofs
  - **Extract** witnesses from all other players
- Sufficient for simulatable coin flipping (and MPC)
- (Not known if Chor-Rabin works directly in MPC)

Getting to Constant Rounds

- All pairs i,j of players run some pairwise coin flipping protocol π simultaneously
- Get $n(n\text{-}1)$ strings σ_{ij}
- Give proofs with respect to σ_{ij} in the global coin flip
- Need some kind of non-malleable coin flipping protocol

Non-Malleable Coin Flipping [Barak02]

- Two executions run concurrrently
- Resists man-in-the-middle attack

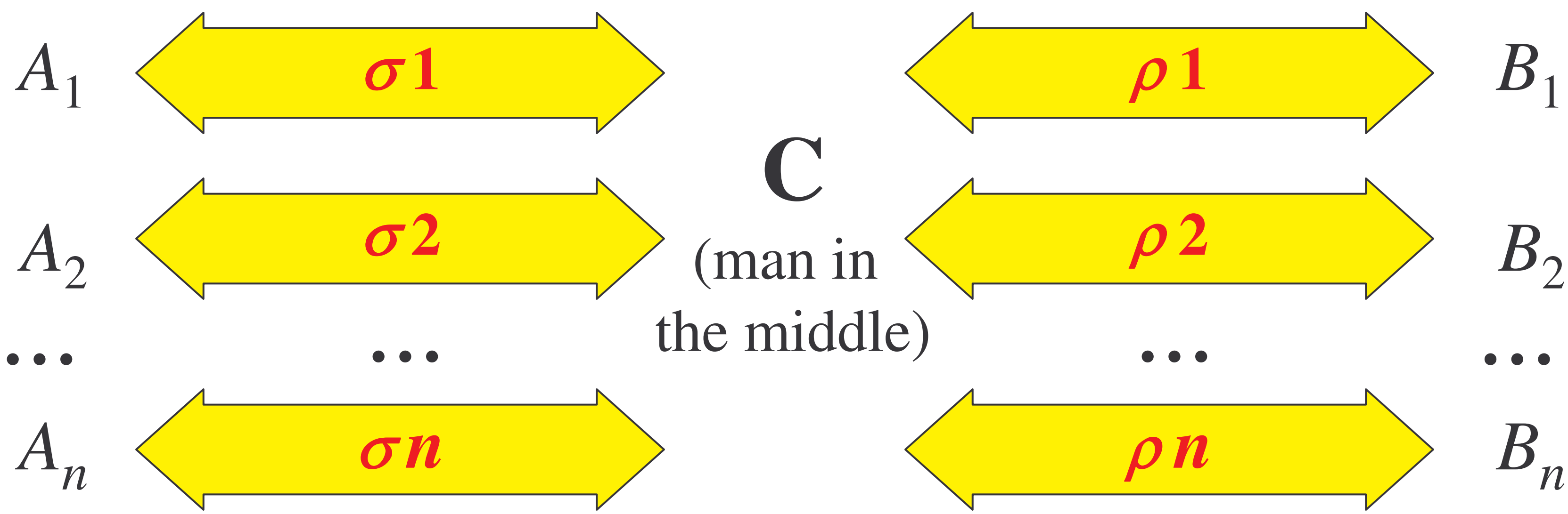


Either $\rho = \sigma$ **or** ρ, σ independent

- Constant rounds

Parallel Non-Malleable Coin Flipping

- Two sets of *n* **parallel** protocols



- All σ_i independent, random
- For each i: **either** $\rho_i \in \{\sigma_1, \ldots, \sigma_n\}$ **or** ρ_i independent

The end

- Improved round complexity for dishonest majority
- Protocols still far from practical… how well can we do?
- Adaptive adversaries?
- log(n)-round on black-box round complexity?
- What about composability?
  - Composability results useful even for “stand-alone” model and essential for practice
  - Concurrent composability: impossible [Lindell03]
  - Limited non-malleability?

Old slides graveyard

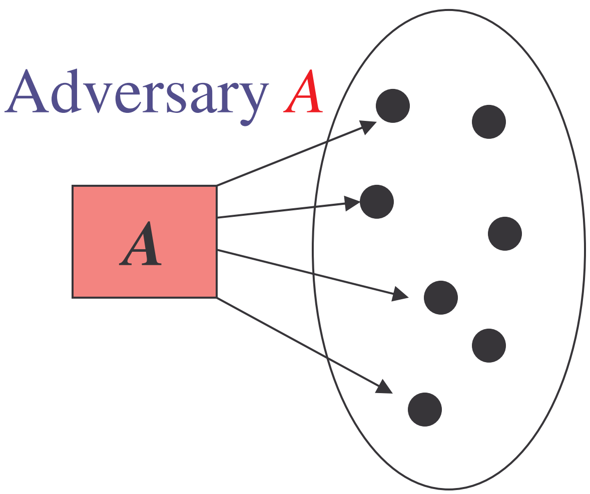
Review: Computational Power

Two main models:

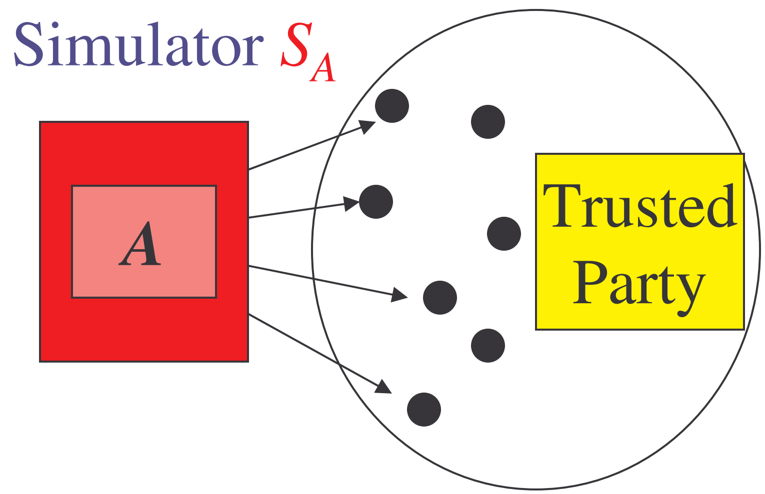
- ‘Computational’ security
  - Adversary runs in polynomial time
  - Assume secure cryptographic primitives (e.g. signatures)
- ‘Statistical’ security
  - Adversary has unbounded computational power
  - Assume secure channels between honest player

Definition of Security [...,Canetti99]

Real Protocol π



Ideal Protocol π'



Security: real protocol equivalent to ideal protocol with TP

$\forall$PPT A, $\exists$ PPT S_A : $\pi[A](1^k) \approx \pi'[S_A]\,(1^k)$

Ideal Protocol for function $f\,()$

1. $\forall i$: P_i sends x_i to TP
2. TP computes $y = f(x_1, \ldots, x_n)$
3. TP broadcasts y
4. Honest players output y

Abortable Ideal Protocol for $f()$

1. $\forall i$: P_i sends x_i to *TP*
2. *TP* computes $y = f(x_1,\ldots,x_n)$
3. *TP* sends y to *A*
4. *A* replies accept/reject
5. *TP* sends $y' = y$ (if accept) or $y' = \perp$ (if reject)
6. Honest players output y'

Protocol neither robust nor fair

Outline

- Passive adversaries: $O(1)$ rounds for any $t < n$
- Intuition: to go from passive to active, we want independence of zero-knowledge proofs
- Independence easy with Common Random String (NIZK)
- To generate a CRS: simulatable coin-flipping
  - Proof scheduling of Chor-Rabin: $O(\log n)$ rounds
  - Non-malleability technique of Barak: $O(1)$ rounds
- Open questions

Passive (honest-but-curious) adversaries

- All players follow protocol faithfully
- *A* tries to learn by looking at internal state of *t* parties (e.g. honest verifier ZK)
- [BMR90]: $O(1)$ rounds for any $t < n$ (static)

  All communication over broadcast channel

From passive to active adversaries [GMW]

General schema: real players P_i emulate passive players P_i'

1. $\forall i$: P_i commits to initial state of P_i' : input x_i, coins r_i
2. P_i proves knowledge of (x_i, r_i)
3. Repeat:
   - P_i commits to new state of P_i'
   - P_i broadcasts messages sent by P_i' at this round.
   - P_i proves consistency of new state and messages with previous round.

From passive to active adversaries [GMW]

Main challenge: independence in this emulation

- Committed input values should be independent
- Proofs should be independent. We want that
  - Simulator can prove false statements
  - Simultaneously extract witnesses from cheaters.

Rest of talk: how to guarantee independence

Why Coin Flipping is Enough

$\sigma = \boxed{\sigma_1} \boxed{\sigma_2} \boxed{\sigma_3} \boxed{\ldots} \boxed{\sigma_n}$

- Suppose all players see a common random string σ
- Divide σ into n pieces
- Player i gives commitments and proofs with respect to string σ_i
- Players' proofs are mutually independent
- Simulator can prove false statements and simultaneously extract from malicious players.