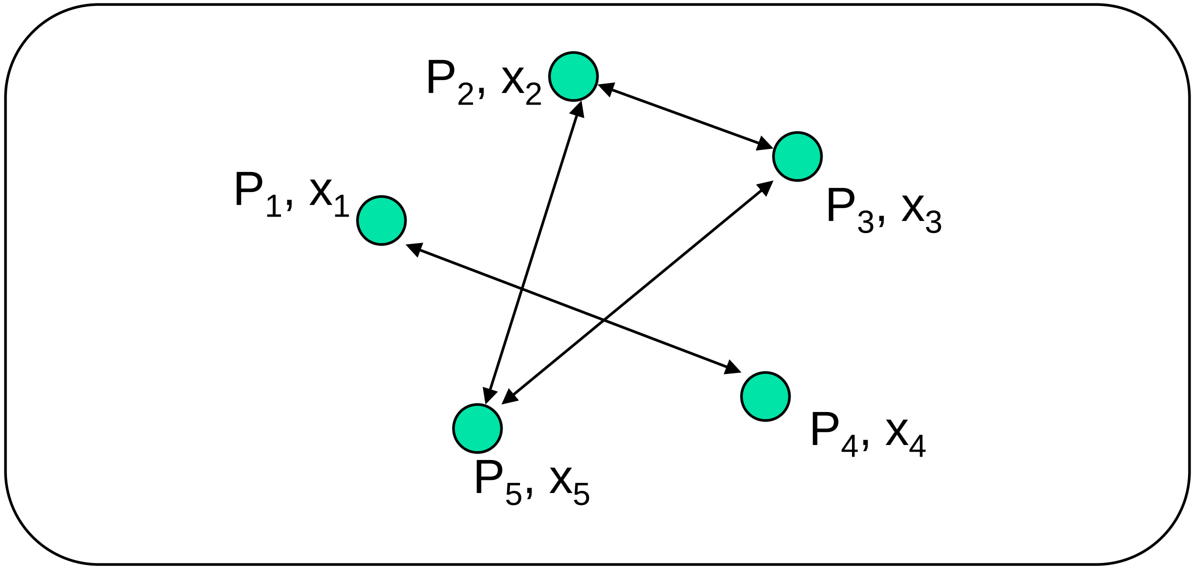
Efficient Two-party and Multiparty Computation against Covert Adversaries

Vipul Goyal
UCLA

Payman Mohassel
UC Davis

Adam Smith
Penn Sate

Secure Multiparty Computation



- Parties learn $f(x_1,\ldots,x_n)$
- But no other information

Adversary Models

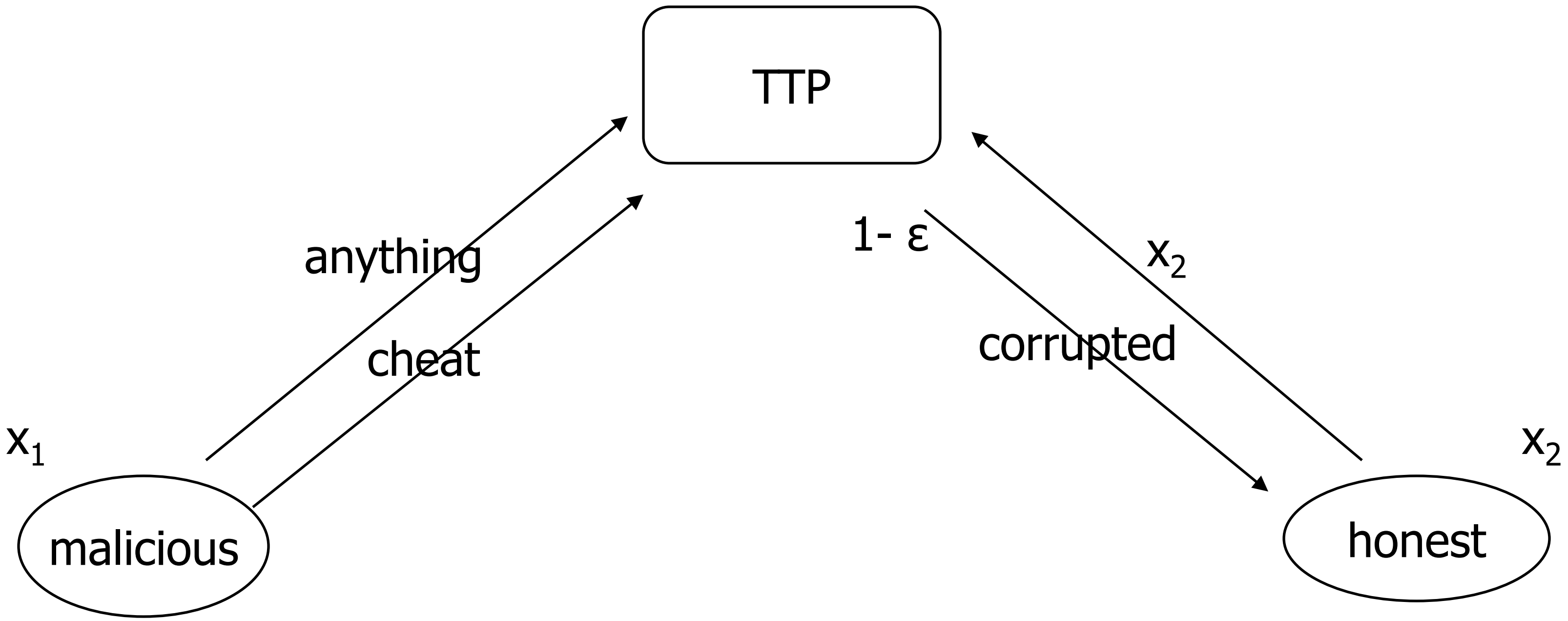
- Number of corrupted parties
  - Honest majority
  - General adversary structures
  - Dishonest majority
    - No fairness or output delivery guarantee
- Malicious vs. Semi-honest
- Static vs. Adaptive

Covert Adversaries

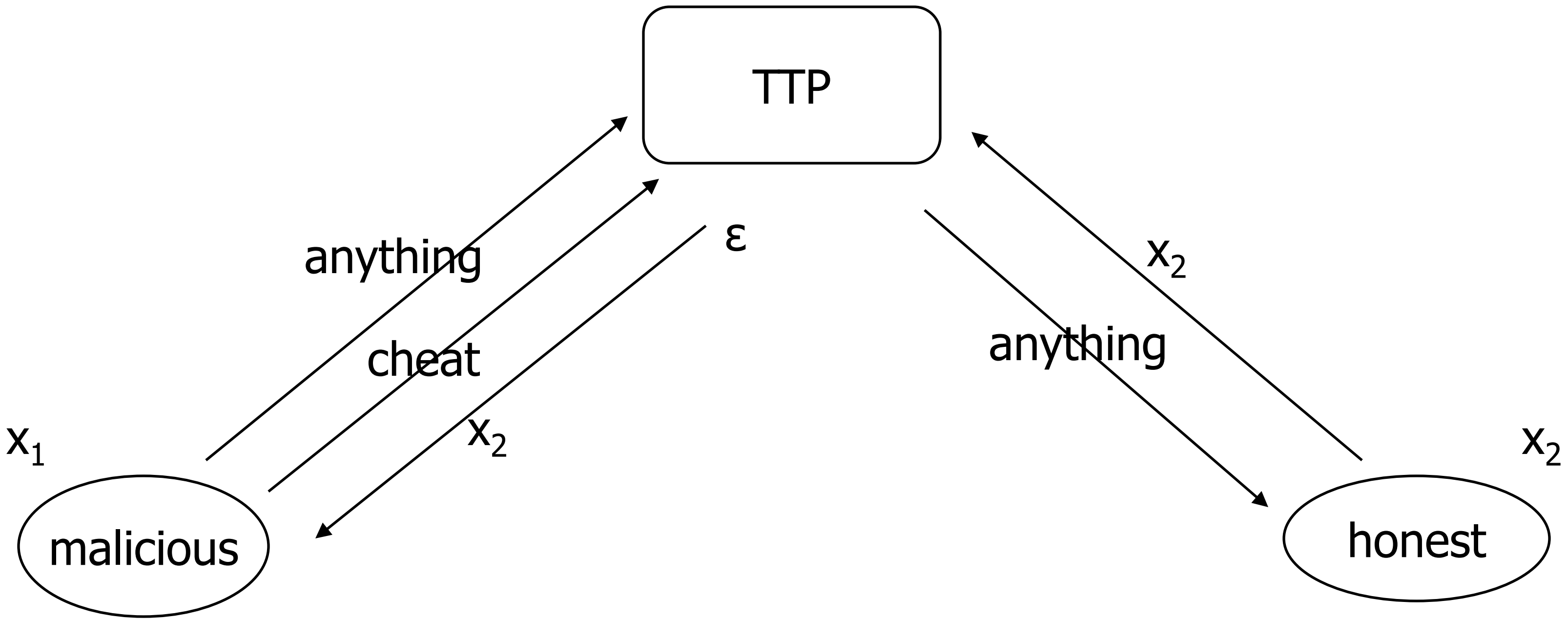
- Somewhere between malicious and semi-honest
- Adversary can cheat but,
  - Caught with reasonable probability
  - Detected cheaters are punished!
- Studied in several previous works
  - [FY92], [CO99], [AL07], etc.

Covert Adversaries

- Simulation-based definition [AL07]



Covert Adversaries



Current Situation

- Honest Majority
  - [DI05]
    - Constant Round
    - Blackbox reduction to PRG
- Dishonest Majority
  - [IKLP06]
    - Blackbox
    - Polynomial number of rounds
  - [KOS03]
    - generic ZK
    - O(log(n)) rounds
  - [MF06,Woo07,LP07,JS07]
    - Constant round
    - No generic ZK
    - Only two-party case

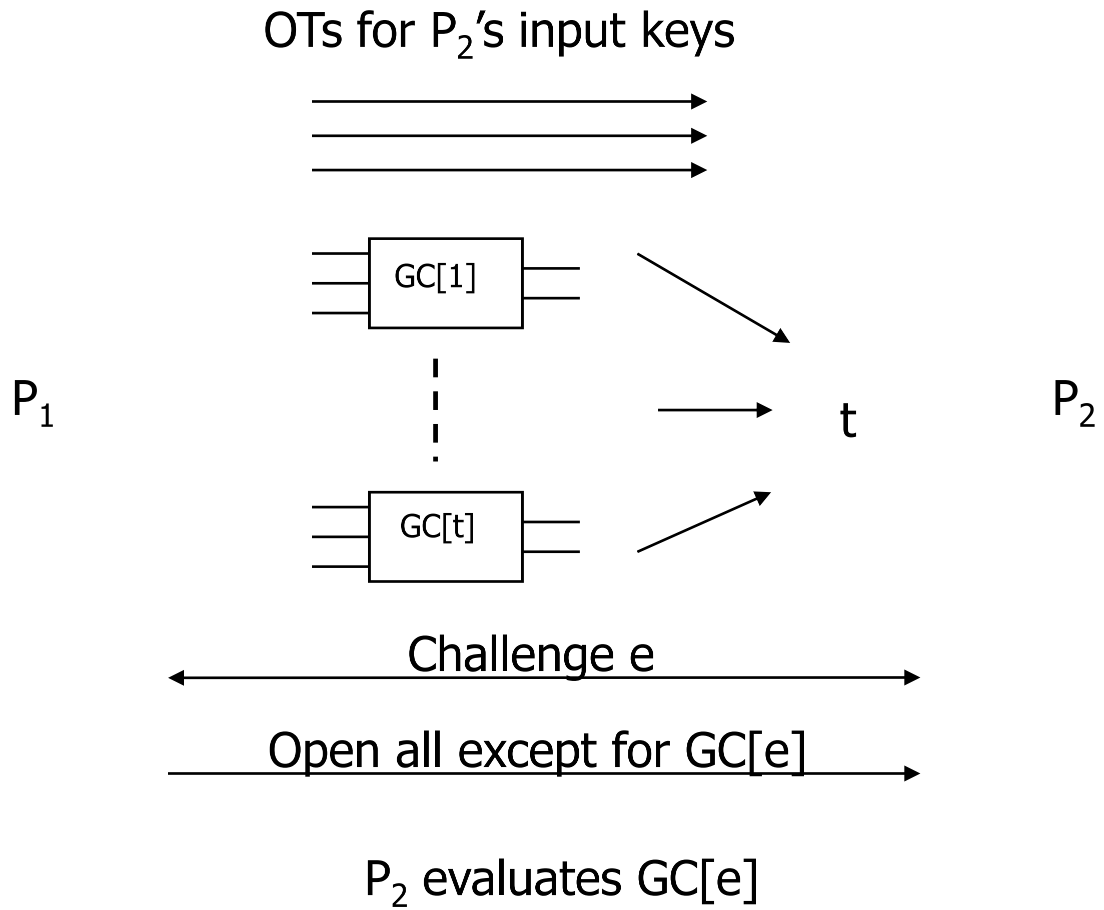
Goal

- Combine all the good properties
  - Round and communication efficiency
  - Avoiding generic ZK
  - Handle dishonest majority
- Settle for Covert Adversaries

Contributions

- Two-party Case
  - Improve communication
  - Malicious and covert adversaries
- Multiparty Case
  - Avoids generic ZK
  - O(log(n)) rounds
  - Covert Adversaries

Two-party Overview



TWO-Party Improvements

- Circuits generated pseudo randomly
- Only hashes of circuits sent over
- Seeds are revealed for opened circuits
- Reduced OT communication
  - Only first few steps of OTs are executed initially
  - Receiver committed to his inputs
  - Sufficient for simulation to go through

Two-party Improvements

h: hash function; G: PRG

P_1

$s_1 \leftarrow 1^k$, $G(s_1)$,
$GC[1] \leftarrow Garble(G(s_1))$

com(s_1), com(h(GC[1]))

$s_t \leftarrow 1^k$, $G(s_t)$,
$GC[t] \leftarrow Garble(G(s_t))$

com(s_t), com(h(GC[t]))

P_2

Challenge e

Reveal all seeds except s_e

Send GC[e], P_2 evaluates GC[e]

Two-party Improvements

- Communication
  - Undetected cheating prob. 1/t
  - O(|C| + t) instead of O(t|C|)
  - Can handle larger t
  - More incentive not to cheat
- Malicious adversaries
  - Similar techniques work
  - Have not analyzed asymptotically

Multiparty Case

- Modify [BMR90] garbled circuit construction
- Run the protocol in t session
- Each session performed using semihonest SFE
- Perform cut-and-choose

Modified BMR

- A mask bit λ^w for every wire w
  - P_i holds λ_i^w
  - $\lambda^w = \lambda_1^w \oplus \lambda_2^w \oplus \dots \oplus \lambda_n^w$
  - for P_i's input bit x^w let
    - $x^w \oplus \lambda_i^w$
- Two random keys $k^{w,0}$, $k^{w,1}$ for wire w
  - P_i holds $k_i^{w,0}$, $k_i^{w,1}$
  - $k^{w,j} = k_1^{w,j} \,||\, k_2^{w,j} \,||\, \dots \,||\, k_n^{w,j}$

Modified BMR

- P_i expands his keys to one-time pads
  - $p_i^{w,0}, q_i^{w,0} \leftarrow G(k_i^{w,0})$
  - $p_i^{w,1}, q_i^{w,1} \leftarrow G(k_i^{w,1})$
- Garbled NAND gate g:
  - input wires a,b
  - output wire c

Modified BMR

- $g(0,0) = p_1^{a,0} \oplus \cdots \oplus p_n^{a,0} \oplus p_1^{b,0} \oplus \cdots \oplus p_n^{b,0}$

$$\oplus \begin{cases} k_1^{c,0} \| \ldots \| k_n^{c,0} & \text{if } \lambda^a \text{ NAND } \lambda^b = \lambda^c \\ k_1^{c,1} \| \ldots \| k_n^{c,1} & \text{otherwise} \end{cases}$$

  - $x^a \oplus \lambda^a = 0; x^b \oplus \lambda^b = 0$
  - $(x^a \text{ NAND } x^b) \oplus \lambda^c = (\lambda^a \text{ NAND } \lambda^b) \oplus \lambda^c$
- Similarly for g(0,1), g(1,0) and g(1,1)

g(0,0)
g(0,1)
g(1,0)
g(1,1)

Main Modifications

- Inputs not embedded in garbled circuit
  - Opening a circuit does not reveal inputs
- Garbling done using a semi-honest SFE
  - Parties commit to their random coins
  - Run multiple semi-honest sessions
  - Cheating is detected through cut-and-choose

Sub-Protocols

- **PublicCoinFlip**
  - $(1^k, \ldots, 1^k) \rightarrow (\sigma, \ldots, \sigma)$
  - [CR87, KOS03] O(logn) rounds
- **Simulatable Commitments**
  - Commit: $(\sigma; x_1, \ldots, x_n) \rightarrow (\{com(x_i)\}, \ldots, \{com(x_i)\})$
  - Open: P_i opens $com(x_i)$
- **CommittedCoinFlipToAll**
  - $(\sigma; 1^k, \ldots, 1^k) \rightarrow (com(e), \ldots, com(e))$
- **CommittedCoinFlipToP$_i$**
  - $(\sigma; 1^k, \ldots, 1^k) \rightarrow (com(e), \ldots, e, \ldots, com(e))$

Main Protocol

CRS generation

$\sigma \leftarrow$ PublicCoinFlip

Challenge generation

Com(e) $\leftarrow$ CommittedCoinFlipToAll(σ)

Committing to randomness

For each player i, for each session S in [1..t]

- $r_i[S] \leftarrow$ CommittedCoinFlipTo$P_i(\sigma)$
- Expanded using pseudorandom generator
- used to generate mask bits, wire keys, semi-honest SFE randomness

Committing to Masked Inputs

P_i commits to $x^w \oplus \lambda_i^w[S]$ for his input wires w

Generating Garbled Circuits

Parties run t parallel sessions to generate garbled circuits GC[1], … , GC[t]

Verification Phase

Parties open the committed challenge e

For each session S ≠ e, parties open all commitments (except for masked inputs)

Evaluation Phase

For GC[e], parties open masked inputs and broadcast

Each party evaluates the garbled circuit on their own

Summary

- Multiparty
  - Covert Adversaries
  - Avoid generic ZK
  - Round efficient
- Two-party
  - Improved efficiency
  - Covert and malicious adversaries

Thank you!

Efficiency Measures

- Communication
  - Number of bits exchanged
- Rounds
  - Number of rounds of interaction
- Computation
  - Local work by each party
- Practical measures
  - Black-box use of underlying primitives
  - Avoiding generic ZK proofs
  - Efficiently implementable primitives