Problem addressed

- To compute a multiparty computation \( f(x_1, \ldots, x_n) \)
  - Each \( n \) holds its input \( x_i \)
  - Reveals no other information (individual \( x_i \) not released)
- Want to determine threshold of corrupted nodes
  - Related work (Ben-Or, Goldwasser, & many others) can prove \( t < n/3 \) (related to Shamir’s secret key sharing)
  - This paper is able to prove a threshold of \( t < n/2 \) can be tolerated and still provide secure computations.
Introductory Definitions

- Common definition of security in multiparty protocols
  - Secrecy of a trusted third party (TTP) (i.e. IDEAL) is the standard/base to measure a given protocol against
  - i.e. the distributed calculation $f(x_1, ..., x_n)$ should reveal no more information than a TTP
- Resilience: security & reliability
  - Allows comparison to the ideal TTP
- Relative Resilience
  - Allows comparison between two protocols (non-ideal)
  - Supports modularity

Paper Solution

- ABC problem using Verifiable Secret Sharing (VSS)
  - Problem: with three secrets $a$, $b$, & $c$, satisfy $c = ab$ w/out revealing values
  - Results/consequences
    - Shows multiparty computations resilient to $t < n/2$
    - Allows a single member to prove in zero-knowledge a secret $y = p(x_1, ..., x_n)$ w/out revealing the secrets
Protocol Execution

- Protocol can be modeled as a set of \( n \) Turing machines
  - Input tape \( I = n#m#k#x#a \) and random tape \( R \)
  - \( n \) = # of machines in network
  - \( m \) = # bits of input
  - \( k \) = security parameter
  - \( a \) = auxiliary input (former run)
  - \( x \) = input

Protocol Execution

- The complete Turing model is for correctness, but can be confusing
- The general approach is to model separately
  - Player
  - Channel
  - Adversary
Protocol Execution-Player

- Player: tuple \((Q, q(0), \delta, Y)\)
  - \(Q\) = set of states (possibly infinite)
  - \(q(0)\) = start state such that \(q(0) \in Q\)
  - \(Y\) = output function \(Y: Q \rightarrow \Sigma^*\)
    - Maps superstate (tape contents + control) to a string onto the output tape
  - \(\delta\) = transition function \(\delta: Q \times 2^\Sigma^* \rightarrow \text{dist}(Q \times 2^\Sigma^*)\)
    - Transition determined by a probabilistic distribution
    - Idea similar to class notes on P TM. (recall each choice had an associated probability)
- Input/output is the critical behavior

Protocol Execution-Channel

- Channel defined by probabilistic function \(C: \Sigma^* \rightarrow \text{dist}(2^N \times N \times \Sigma^*)\)
  - Is the probability that a given input string will produce an output
    \((i,j_1,u_1), \ldots (i,j_l,u_l)\)
  - \(i\) = sender (local node)
  - \(j_l\) = target node
  - \(u_l\) = message
- Types:
  - Perfect
    - Private: \(Pr_{\text{CIW}}[\{(i,j,u)\}] = 1\)
    - Broadcast: \(Pr_{\text{CIW}}[\{(i,[n],u)\}] = 1\)
  - Noisy
    - \(Pr_{\text{CIW}}[\{(i,j,(1,u))\}] = \frac{1}{2}\) (success)
    - \(Pr_{\text{CIW}}[\{(i,j,(0,0))\}] = \frac{1}{2}\) (failure)
Protocol Execution-the protocol

- Protocol $\Pi$ can now be given as a set of players $P$ and a labeled set of channels
  $\Pi = \{(P_1, \ldots, P_n, C_n) \mid n \in \mathbb{N}\}$

- Protocol execution is better described as an example that maps $<x, \bar{s}>$ to final state of $<\tilde{q}(R), \tilde{v}(R)>$
Protocol Execution—Adding an Adversary

- **Adversary**
  - Can examine a subset of $\mu^{\text{del}}$ & substitute its own
  - **Types**
    - Passive: makes no changes
    - Byzantine: can make changes
    - Rushing: can change messages prior to delivery
  - Defined: a $t$-adversary is a pair $(\mathcal{A}, T)$
    - $\mathcal{A}$ is a player with:
      - $Q_{\mathcal{A}}, \delta_{\mathcal{A}}, Y_{\mathcal{A}}$
    - $T$ is a coalition function describing set of players corrupted by $\mathcal{A}$
      - $T: Q_{\mathcal{A}} \rightarrow [n]$

- **Execution with an adversary**
  - All non-faulty players perform local computation, generate messages, and pass to communication channel as normal
  - Adversary requests for a particular player $i \in [n]$:
    - Incoming from previous and current round
    - Outgoing messages
    - View
    - State
  - Adversary then corrupt/replace messages
Real & Ideal Protocols

- **Real protocol**
  - Associated with a t-adversary class with no player above corruption

- **Ideal protocol**
  - Contains one or more non-corruptible (trusted) hosts labeled as n+1, n+2, ...

---

Ideal Protocol Execution

1. \((1 \leq i \leq n)\) \(i \rightarrow n + 1: x_i\)
2. \((1 \leq i \leq n)\) \(n + 1: x_i' \leftarrow \begin{cases} x_i & \text{if } x_i \in \text{dom}(F) \\ 0 & \text{otherwise (default value)} \end{cases}\)
2.  \((1 \leq i \leq n)\) \(n + 1: \langle y_1, \ldots, y_n \rangle \rightarrow F(y_1, \ldots, y_n)\)
2.  \((1 \leq i \leq n)\) \(n + 1: y_i\)
2.  \((1 \leq i \leq n)\) \(n + 1: \text{output } y_i\)

**Fig. 2.** Ideal protocol for trusted host \((n + 1)\) to compute probabilistic multivalued function \(F\), which without loss of generality is extended to handle default values. When default values are used, the trusted host returns an \(n\)-bit vector with a 1 in the \(i\)th position if player \(i\) supplied a value outside the domain (not shown).
Focusing on Security

- Multiparty protocol security history
  - Privacy
  - Correctness (the right answer)
  - Independence of inputs
- This may create dependencies & make formal definitions difficult. . . must return to idea of achieving same results as if you were running on the ideal TTP (simulation)

Solution Attempts

- Fault-oracle approach
  - Does not allow modular proofs
    - Can prove two protocols secure against two respective oracles, but cannot prove the concatenation of the two
- The answer (in this paper)
  - Relative Resilience (comparing two protocols against each other)
    - Absolute resilience: comparison against a standard (ideal protocol).
    - A protocol is resilient iff it is as resilient as ideal case
Formally Defining Relative Resilience

- Relative Resilience
  \[ \alpha \geq (A_\alpha, A_\beta) \beta \]

  If there exists an interface from \( \alpha \) to \( \beta \) such that for all adversaries \( \mathcal{A} \in A_\alpha \),
  \[ [\alpha, \mathcal{A}] \approx [\beta, \varphi(\mathcal{A})] \]

  Interface \( \varphi \) has two lines
  - environment simulation line & adversarial line
  - and for every \( \mathcal{A} \in A_\alpha \), \( \varphi(\mathcal{A}) \in A_\beta \)

Resilience Comparisons

- 1st case: Ideal protocol
  - Trusted host can ensure adversary gains no advantage

- 2nd case: Protocol w/ interface
  - Using the interface to model real world adversary on ideal protocol

- 3rd case: Real protocol
  - No player is trusted but protocol should return same results as case 1
t-Resilient (resilience)

- t-resilient protocol satisfies
  \[ \Pi \geq (A', A_{ideal}) ID(F) \]
- Types of resilience:
  - Exponential, computational, or statistical indistinguishability

Cheating & Recovery

- Adversarial cheating is defined as producing messages that are not a possible output
  - An adversary cheats on message from a faulty player if a non-faulty player would have never sent that message
- Recovery
  - If cheating detected, the secrets of the cheating players are reconstructed and distributed
Questions

BACKUP SLIDES