SCALABLE MPC WITH STATIC ADVERSARY

Mahnush Movahedi, Jared Saia, Valerie King, Varsha Dani
University of New Mexico
University of Victoria

November 2013
Multiparty Computation (MPC)

- $n$ players participate in an auction
- They want not to reveal their inputs to each other or any external party

Q: How can they determine the highest bid without revealing any information about the other bids?
MPC: Formal Definition

- **Given:**
  - $n$ players
  - Each player $i$ has a private input
  - Function $f$ over $n$ inputs, known to all players

- **Goals:**
  - All players learn the value $f(x_1, x_2, \ldots, x_n)$
  - The inputs remain as private as possible

No more information revealed than what can be determined by the output
Applications

A group can sign/read a document collectively, but not individually

- Auctions
- Threshold cryptography
- Anonymous message transmission
- Information aggregation

Messages can be broadcast to a network but originator remains anonymous
Applications as Functions

- **Auctions**
  \[ f = \max(x_1, x_2, \ldots, x_n) \]

- **Threshold cryptography**
  1) \( M, p, q \) are parameters of the function;
  2) \( s \) is the \( y \) intercept of a degree \((d-1)\) function
  with points given by the \( x_i \) values.
  \[ f = M^s \mod pq \]

- **Information aggregation**
  \[ f = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \left( \frac{\sum_{i=1}^{n} x_i}{n} \right)^2} \]
MPC in different Models

- **Adversary type:**
  - Static Adversary
    - adversary can take over players at starting of the algorithm
  - Adaptive Adversary
    - adversary can take over players any point
  - Rational players
    - Players are good but curious

- **Adversary Computational Power**
  - Bounded or Unbounded

- **Communication model:**
  - Have broadcasting or Peer-to-Peer
  - Synchronous or Asynchronous
Previous Work

- Ben-Or et al. in [BGW88], Cramer et al. in [CDD99], Hirt et al. in [HM01], Beerliova et al. in [BH06], Applebaum in [AIK10]
- Create a circuit based on function $f$
- Assume the circuit has $m$ gates
  - Each player sends $O(mn)$ messages
  - Each player performs $O(mn)$ computation.
Problem

Current algorithms to solve SMPC are not resource-efficient.
First Set of Assumptions

- Computationally-unbounded static adversary
  - Not a cryptographic algorithm
  - 1/3 of players are bad
- No memory leakage
- Only Pairwise secure channels
  - No broadcast channel
  - Simulating broadcast channel using byzantine agreement.
- Synchronous Communication model
Our Contribution

- Much improved computation & message cost
- Assume the circuit has m gates
  - Each player sends $\tilde{O}(\frac{m+n}{n} + \sqrt{n})$ messages
  - Each player performs $\tilde{O}(\frac{m+n}{n} + \sqrt{n})$ computation.
- We solve MPC w.h.p. meaning

$$1 - O(1/n^k)$$ for any fixed $k$
Algorithm Overview

- Make critical use of a quorum:
  - Has $\theta(\log n)$ players
  - Less than $1/3$ are bad
  - Each gate is computed by a quorum

- Preserve privacy
  - Masking inputs & gate outputs with random numbers
  - Random number are known collectively via verifiable secret sharing
Tools

- Make critical use of a quorum
  - Can get all processors to agree on $n$ quorums \( w.h.p. \) [KS 11]

- Preserve privacy
  - Masked with Random number that are known collectively
  - Verifiable secret sharing algorithm [BGW 88]
  - HEAVY-WEIGHT-MPC algorithm [BGW 88]
Algorithm Overview

- Translate function $f$ to circuit $C$
- Build network $G$ based on $C$
  - Gates $\rightarrow$ Internal nodes
  - Inputs $\rightarrow$ Input nodes
  - Wire $\rightarrow$ Edges
- Build quorums
- Assign each quorum to a node
Circuit and Network
The Algorithm

- Input commitment using VSS
- Random number generation

Generate $R_G$ jointly
Computation of a Gate

1) $G$ Shares of $R_G$

$X + R_x$
Shares of $R_x$

$Y + R_y$
Shares of $R_y$

2) $G(X, Y) + R_G$

$G(X, Y) + R_G$
Propagating Output

- Output reconstruction
- Output propagation
Second Set of Assumptions

- Computationally unbounded \textit{static} adversary
  - Not a cryptographic algorithm
  - $1/4$ of players are bad
- No memory leakage
- Only Pairwise secure channels
  - No broadcast channel
  - Simulating broadcast channel using byzantine agreement.
- Asynchronous Communication model
Asynchronous Communication Model

- The adversary has control over the latency of the channel.
  - Can arbitrarily delay messages.
  - Can not lose them outright.

- MPC simulates a trusted third party
  - Waits for \( n-t \) messages.
  - Compute \( f \).
  - Send back the output and the set of indices from which input was received.
What is the difference?

- We have to count the correct inputs
  - We develop count-tree to deal with that.
- Players can not wait for all the messages
  - Decrease number of bad players
  - Wait until receiving sufficient number of same messages.
Count-Tree

- Count the ready inputs.
  - A tree with input quorums as its leaves

- Naïve approach
  - Sends message every time it gets a new input.
  - Problem: Load balancing.
Count-Tree

- Halving trick
  - Sends message only when 
  # new inputs > 1/2 # inputs the 
  parent is waiting for.
We are Working on

- Make quorum building algorithm assuming asynchronous communication.

- More specific:
  - Building an scalable “leader election” or ”byzantine agreement” algorithm assuming asynchronous communication.
Third Set of Assumptions

- Rational Players
- Only Pairwise secure channels
  - No broadcast channel
  - Simulating broadcast channel using byzantine agreement.
- Synchronous Communication model
Rational Players

- Players prefer
  - Personally learn the output of the function
  - Prevent others from learning
- $n$ rational players with any utility functions based on above preferences
- We developed an algorithm for rational players
Our Results

- There exists a protocol such that
  - It is a Nash equilibrium
  - Players can solve SMPC w.h.p.

- Each player sends $\tilde{O}\left(\frac{m+n}{n}\right)$ messages
- Each player performs $\tilde{O}\left(\frac{m+n}{n}\right)$ computation.
Conclusion

- We have created a scalable algorithms to perform MPC
  - needs less resources.
  - Synchronous case: tolerates less than $n/3$ bad players.
  - Asynchronous case: tolerates less than $n/4$ bad players.
  - Rational case: tolerates rational players.
Open Problems

- What is the probability of my death because of the Immune System Failure?

- It is my long term hobby question.
Open Problems

- Adaptive adversary
  - The adversary can take over players any point during the algorithm
  - Is it possible to propose an scalable algorithm?
    - We know it is not possible for all functions
  - Is there a lower bound?

- What is the lower bound for number of messages in different MPC models?
Open Problems

- Large constants and logarithmic coefficient
  - We use our algorithm to implement an Anonymous message transmission protocol
  - It still sends a lot of messages.
  - Can we improve it?

- Can Cryptography help?
  - Using cryptographic algorithms can decrease the number of messages.
Thanks!