Distributed Computing Meets Game Theory

Robust Mechanisms for Secret Sharing and Implementation with Cheap Talk

Joe Halpern

Joint with: (a) Vanessa Teague – STOC ’04

(b) Ittai Abraham, Danny Dolev, Rica Gonen – PODC ’06

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either “good” or “bad”
- good agents follow the protocol
- bad agents do all they can to subvert it

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either “good” or “bad”
- good agents follow the protocol
- bad agents do all they can to subvert it

Game theory assumes

- all agents are rational
- they try to maximize their utility

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either “good” or “bad”
- good agents follow the protocol
- bad agents do all they can to subvert it

Game theory assumes

- all agents are rational
- they try to maximize their utility

Both views make sense in different contexts

- We want to combine them

Secret Sharing

Agent 0 has a secret s (an integer) that she wants to share among n other agents in such a way that any m of them can reconstruct it.

Shamir's protocol (1977):

- Agent 0 chooses a polynomial f of degree $m-1$ with $f(0)=s$.
- Agent 0 tells $f(i)$ to agent i, $i=1,\ldots,n$
  - $f(i)$ is agent i's share of the secret
- Any subset of m agents can pool their shares and reconstruct f, and hence $f(0)$.
- No subset of size $<m$ can figure out the secret.

Rational Secret Sharing

Why do we want protocols where m agents can reconstruct the secret?

- Implicit assumption: up to $n - m$ agents may be “bad”, and may not follow the protocol
  - The rest of the agents are “good”, and follow the protocol
- Bad agents can’t prevent good agents from recovering the secret.

The partition into “bad” and “good” is not always appropriate.

- A possibly better assumption:
  - agents aren’t bad or good, just rational.
  - they have preferences, and try to maximize their expected utility

Preferences

Assume each agent i's preferences are such that

- getting the secret is better than not getting it
- secondarily, the fewer of the other agents that get it, the better

These are not the only possible preferences, but they are reasonable.

Why Secret Sharing is Important

Secret sharing is an interesting problem in it's own right, but perhaps more importantly

- it is critical for *multiparty computation*
  - privately computing a function of inputs given to the agents
- it is in a sense a *complete* problem for implementing mediators
  - If we can implement the secret sharing mediator under certain conditions, then we can implement a mediator under the same conditions

An Impossibility Result

Claim: Rational agents won't broadcast their shares.

All results in the first half of the talk are joint with Vanessa Teague.

Proof: Consider agent i:

- If $m-1$ other agents send him their shares, he can compute the secret; otherwise he can't.
  - His action has no influence on this
- If exactly $m-1$ other agents send their shares, then sending his share enables other agents to compute the secret.

An Impossibility Result

Claim: Rational agents won't broadcast their shares.

All results in the first half of the talk are joint with Vanessa Teague.

Proof: Consider agent i:

- If $m-1$ other agents send him their shares, he can compute the secret; otherwise he can't.
  - His action has no influence on this
- If exactly $m-1$ other agents send their shares, then sending his share enables other agents to compute the secret.

Bottom line: sending a share is weakly dominated by not sending:

- it's never better and sometimes worse

Further Impossibility

Theorem 1: No protocol with bounded running time where some agents learn secret survives iterated deletion of weakly dominated strategies.

Proof: In the last round, not sending a message weakly dominates sending (just as in previous argument)

- Can delete strategies where message is sent in last round
- Once these are deleted, can delete strategies where messages sent in second-last round
- Continue deleting strategies by backward induction
  - This is iterated deletion
  - The actual argument is much more subtle.

A Possibility Result

Theorem 2: There is a randomized protocol for secret sharing with constant expected running time that

(a) is a *Nash equilibrium*

- No agent has any incentive to deviate, even if he knows everyone else's strategy

A Possibility Result

Theorem 2: There is a randomized protocol for secret sharing with constant expected running time that

(a) is a *Nash equilibrium*

- No agent has any incentive to deviate, even if he knows everyone else's strategy

(b) survives iterated deletion of weakly dominated strategies

Multiparty Computation

Now assume that each agent has a secret input.

- Goal: to compute some function of that input, without revealing any information other than the function's output.
  - Just as if a trusted mediator had computed the function

Example: secret input is salary, the function computes highest salary.

Multiparty Computation

Now assume that each agent has a secret input.

- Goal: to compute some function of that input, without revealing any information other than the function's output.
  - Just as if a trusted mediator had computed the function

Example: secret input is salary, the function computes highest salary.

There are protocols for multiparty computation, assuming that less than $1/2$ or $1/3$ (depending on underlying assumptions) of agents are bad

- Again, the good/bad dichotomy is often inappropriate

Results on Multiparty Computation

Theorem 3: No protocol for multiparty computation with bounded running time survives iterated deletion of weakly dominated strategies.

- Not surprising, in light of earlier result.
- But all the protocols for multiparty computation in the literature have a commonly known upper bound on the number of steps.

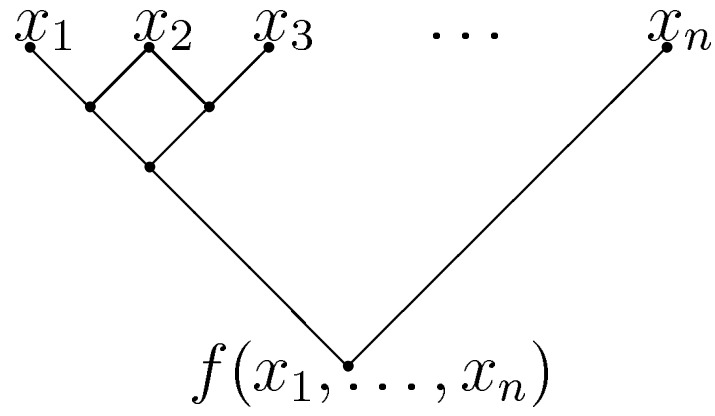
Theorem 4: Every function f that can be computed by rational agents with a trusted mediator can be computed without one.

- We use the protocol for secret sharing as a building block.
- (Shoham-Tennenholtz:) Parity can't be computed by rational agents with a trusted mediator

Idea of Protocol

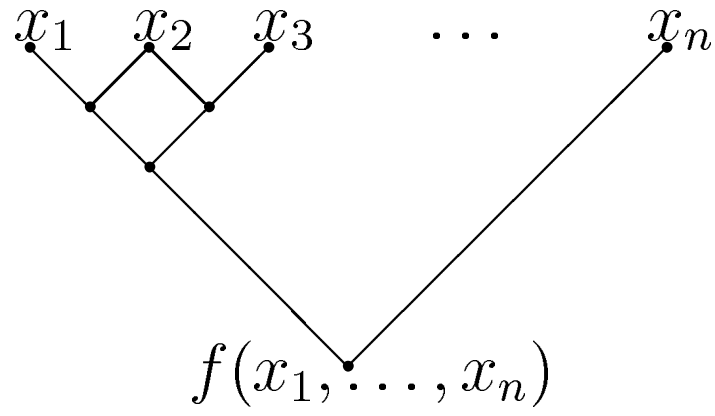
[Goldreich-Micali-Wigderson:]

Consider a circuit for computing function f:



Idea of Protocol

[Goldreich-Micali-Wigderson:] Consider a circuit for computing function f:



- Want each agent to have a share of value of each node in circuit
  - Each agent sends a share of its input to all other agents.
  - At the end, each agent has a share of $f(x_1, \ldots, x_n)$
- Agents use rational secret sharing to learn $f(x_1, \ldots, x_n)$
- If agent has incentive to lie about its input, then it could have lied the same way with the mediator

Dealing with Collusion

Remainder of talk covers joint work with Ittai Abraham, Danny Dolev, and Rica Gonen

Dealing with Collusion

Remainder of talk covers joint work with Ittai Abraham, Danny Dolev, and Rica Gonen

Nash equilibrium tolerates one defection.

- It's perfectly consistent with Nash equilibrium that two agents can do better by colluding
  - lots of examples in practice:
    - airline ticket pricing
    - unions
    - …

k-Resilience

A k-resilient equilibrium tolerates deviations by coalitions of size $\leq k$.

- if any coalition C with $|C| \leq k$ deviates from the equilibrium, then not everyone in C is better off.

Studying coalition resistance goes back to Aumann [1959]

- Other notions have also been considered
- Our results apply to them too

There is always a Nash equilibrium (= 1-resilient equilibrium)

- Some games do not even have 2-resilient equilibria
- We provide k-resilient protocols for secret sharing
- Then use protocol to implement *any* mediator using cheap talk.

Idea of Construction

- Start with a k-resilient protocol for rational secret sharing with a mediator
  - Assume for this talk that shares are signed (we don't need this)
  - This means that lies about initial shares are detected
- Then remove the mediator, using ideas from multiparty computation
  - The multiparty computation uses secret sharing as a subroutine, but we show there is no circularity
- Same ideas used to do k-resilient implementation of any mediator.

A Solution With a Mediator

Suppose f of degree $m-1$. Assume wolog $f(0)$ (the secret) is not 0.

- In stage 0, each agent i sends its secret share $f(i)$ to mediator
- The mediator computes f

A Solution With a Mediator

Suppose f of degree $m-1$. Assume wolog $f(0)$ (the secret) is not 0.

- In stage 0, each agent i sends its secret share $f(i)$ to mediator
- The mediator computes f
- In stage $r \geq 0$, if the game is not over, the mediator
  - chooses $c \in \{0,1\}$ ($\Pr(c=1)$ to be determined) and random polynomial g^r of degree $m-1$ such that $g^r(0)=0$
  - computes $h^r = cf + g^r$ and sends $h^r(i)$ to each agent i

A Solution With a Mediator

Suppose f of degree $m-1$. Assume wolog $f(0)$ (the secret) is not 0.

- In stage 0, each agent i sends its secret share $f(i)$ to mediator
- The mediator computes f
- In stage $r \geq 0$, if the game is not over, the mediator
  - chooses $c \in \{0,1\}$ ($\Pr(c=1)$ to be determined) and random polynomial g^r of degree $m-1$ such that $g^r(0)=0$
  - computes $h^r = cf + g^r$ and sends $h^r(i)$ to each agent i
- The players then share their shares and compute h^r.
  - if $h^r(0) \neq 0$, then $h^r(0) = f(0)$; the game is over
  - if $h^r(0) = 0$, the agents learn nothing; go to stage $r+1$.

Why This Works

- There is no advantage to lying about the initial value.
  - Shares are signed, liars are caught.

Why This Works

- There is no advantage to lying about the initial value.
  - Shares are signed, liars are caught.
- There is no advantage to not revealing your share. Two reasons:
  - If $c = 0$, then you do not learn the secret, but are caught
  - If m is small enough, it doesn't help anyway.

  Choice of $\Pr(c = 1)$ depends on relation of m to n.

Why This Works

- There is no advantage to lying about the initial value.
  - Shares are signed, liars are caught.
- There is no advantage to not revealing your share. Two reasons:
  - If $c = 0$, then you do not learn the secret, but are caught
  - If m is small enough, it doesn't help anyway.

  Choice of $\Pr(c = 1)$ depends on relation of m to n.
- There is no advantage to lying about your share
  - If $c = 0$, $h^r(0) \neq 0$, then other players (wrongly) stop playing
  - If m is small enough the secret can be reconstructed anyway
    - Reed-Solomon decoding

Removing the mediator

We replace the mediator by using multiparty computation

- Protocol uses m out of n secret sharing
- the choice of m depends on
  - the size k of coalitions we want to tolerate,
  - whether we have cryptography available
  - whether we know the exact utility
- Rational agents do not gain during circuit evaluation phase
- Agents reveal shares during secret-sharing phase because
  - may get caught, learn no information (depends on $\Pr(c = 1)$)
  - They gain no advantage in any case (if m is small)

Resilient Multiparty Computation

Theorem 5:

(a) If $3k < n$, there is a k-resilient multiparty computation protocol with expected running time 2.

Resilient Multiparty Computation

Theorem 5:

(a) If $3k < n$, there is a k-resilient multiparty computation protocol with expected running time 2.

(b) If $2k < n$ and utilities are known, there is a k-resilient multiparty computation protocol with expected running time 2.

Resilient Multiparty Computation

Theorem 5:

(a) If $3k < n$, there is a k-resilient multiparty computation protocol with expected running time 2.

(b) If $2k < n$ and utilities are known, there is a k-resilient multiparty computation protocol with expected running time 2.

(c) if $k < n$, utilities are known, and 1-way functions exist, there is a k-resilient multiparty computation protocol with constant expected running time (depends on utilities) and ϵ error probability.

- This result uses cryptography; ϵ is the probability that the cryptography is “broken”

Immunity

In large systems, there are certainly some agents who won't respond to incentives, perhaps because they

- have "unexpected" utilities
- are irrational
- have faulty computers

Immunity

In large systems, there are certainly some agents who won't respond to incentives, perhaps because they

- have "unexpected" utilities
- are irrational
- have faulty computers

A protocol is t-*immune* if the payoffs of "good" agents are not affected by the actions of up to t other agents.

- Somewhat like *Byzantine agreement* in distributed computing.
- Good agents reach agreement despite up to t faulty agents.

Immunity

In large systems, there are certainly some agents who won't respond to incentives, perhaps because they

- have "unexpected" utilities
- are irrational
- have faulty computers

A protocol is t-*immune* if the payoffs of "good" agents are not affected by the actions of up to t other agents.

- Somewhat like *Byzantine agreement* in distributed computing.
- Good agents reach agreement despite up to t faulty agents.

A (k, t)-*robust* protocol tolerates coalitions of size k and is t-immune.

Trusted Mediators

Trusted mediators make life easier:

- for multiparty computation
- for negotiation
- to achieve improved outcomes in many games

If we can get a good outcome with a mediator, can we get the same outcome without the mediator?

- This is the goal of the multiparty computation protocols

Implementing A Mediator

Consider the following mediator for secret sharing/multiparty computation: mediator in the case of secret sharing

- tell the mediator the truth; the mediator sends out the secret

Our earlier results give conditions under which we can implement this mediator in a robust and resilient way, using “cheap talk”.

Key observation: Secret sharing is essentially a complete problem—if we can do share secrets, we can implement a mediator!

- When we need to know utilities, need a “punishment strategy”
  - E.g., a punishment in secret sharing is to quit playing

Upper Bounds

Theorem 6: Suppose that σ is a (k, t)-robust protocol using a mediator. There is a (k, t)-robust implementation of σ using cheap talk

(a) If $3(k + t) < n$ even if exact utilities are not known; protocol is *bounded* and does not require punishment.

(b) If $2k + 3t < n$ and there is a punishment strategy; protocol is randomized, has finite *expected* running time

(c) If $2k + 2t < n$ and there is a broadcast channel, with an ϵ error

(d) If $k + t < n$, 1-way functions exist, and there is a punishment strategy, with an ϵ error

Key idea: reduce to secret sharing $+$ multiparty computation.

Matching Lower Bounds

Theorem 7:

(a) If $3(k+t) \geq n$, $\exists$ a (k,t)-robust strategy using a mediator that cannot be implemented without a mediator without knowing the utilities/without a punishment strategy/in bounded time.

(b) If $2k+3t \geq n$, $\exists$ a (k,t)-robust strategy with a mediator that cannot be simulated without a mediator, even if there is a punishment strategy and utilities are known.

(c) If $2k+2t \geq n \ldots$

(d) $k+t \geq n \ldots$

Some proofs exploit techniques used in lower bound proofs for Byzantine agreement.

Lower Bounds on Running Time

Theorem 7: If $2k + 2t \geq n$, then

(a) there is a game Γ with a (k, t)-robust strategy with a mediator that cannot be implemented by *any* deterministic cheap talk strategy.

(b) for all b, there is a game Γ_b with a (k, t)-robust strategy with a mediator that cannot be implemented using cheap talk with expected running time $\leq b$.

(c) there is a game Γ with a (k, t)-robust strategy with a mediator such that for all ϵ, there exists b_ϵ such that we cannot implement the mediator with ϵ error with a cheap-talk strategy that runs in $\leq b_\epsilon$ steps.

Related Work

Lots of related work on implementation in both CS and game theory:

- work of Forges + Barany [≈1990] gives Theorem 6(a) with $k = 1$
- work on secure multiparty [BGW88,CCD88] computation gives Theorem 6(a) for all (k, t)!
- Ben-Porath ('03): Theorem 6(b) with $k = 1$ (no crypto, known utilities, but does sequential equilibrium)
- Heller ('05): extends B-P to all k; proves matching lower bound
- Theorem 7(a) shows that B-P's strategy is incorrect (because bounded); Heller's has problems too
- Rabin/Ben-Or's work gives Theorem 6(c) for all (k, t)

More Related Work

- Urbano-Vila ('04) and Dodis-Halevi-Rabin ('00) get Theorem 6(d) if $k = 1, n = 2$
- Theorem 7(a) shows UV's strategy is incorrect
- Izmalkov, Micali, Lepinski; Lepinski, Micali, Shelat ('05) prove general implementation results assuming strong primitives (*envelopes* and *ballot-boxes*) that cannot be implemented over broadcast channels
- Lysanskaya-Triandopoulos: Theorem 6(c) for $k = 1$

Conclusions

- Issues of coalitions and fault-tolerance are critical in distributed computing, game theory, and cryptography.
- By combining ideas from all three areas we can gain new insights.
  - Better understanding of role of cryptography in games.
  - Better understanding of cheap talk
    - Should we assume that it is common knowledge when the cheap talk phase has ended?
  - Resource bounded equilibria
  - Synchrony vs. asynchrony
- We have not considered the asynchronous case here.
  - Somewhat surprisingly, similar results seem to hold