Detectable Byzantine Agreement Secure Against Faulty Majorities

Matthias Fitzi, ETH Zürich
Daniel Gottesman, UC Berkeley
Martin Hirt, ETH Zürich
Thomas Holenstein, ETH Zürich
Adam Smith, MIT (currently at Microsoft Research)

Detectable Byzantine Agreement Secure Against Faulty Majorities

- "Broadcast" = Single-source Byzantine Agreement = Sender S wants to send value v to all other players
- In a synchronous network with no previous setup, broadcast requires $t < n/3$ [PSL,KY,DD]
- Detectable Broadcast [FGM]: Allow abort.

Either all honest players abort,
or broadcast is achieved

- This work: Randomized DB protocols for all $t < n$

Outline

- What is Detectable Broadcast?
- Protocols for an arbitrary number of cheaters
- Extensions
- Conclusions, Open questions

Outline

- What is Detectable Broadcast?
  - Review: Standard Synchronous Model
  - Detectable Broadcast
  - Our Results
- Protocols for all $t < n$
- Extensions
- Conclusions, Open questions

Review: Standard Synchronous Model

- Synchronous network of n players (= randomized TM's)
- Pairwise authenticated, unblockable channels
  - Know identity of all other players
- Common start time

Will be removed

- Adversary corrupts up to t players
  - Malicious coordination of corrupted players
  - Choice of corruptions is adaptive (= on the fly)
  - Messages may be rushed (= cheaters get to see honest players' round i messages before sending their own)

Review: Computational Power

Results for two models

- 'Computational' security
  - Adversary runs in polynomial time
  - Assume secure cryptographic primitives (e.g. signatures)
- 'Unconditional' security
  - Adversary has unbounded computational power
  - Assume secure channels between honest players

Theorem ([CFGN'96], "non-committing encryption"):

Unconditional security $\Rightarrow$ Computational security

Broadcast (Single-source Byzantine Agreement)

- Designated sender S with input $v \in \{0,1\}^m$
- Each player P_i outputs $v^{(i)} \in \{0,1\}^m$
- Consistency

  P_i , P_j honest $\Rightarrow$ $v^{(i)} = v^{(j)} = v'$
- Validity

  S honest $\Rightarrow$ $v' = v$

Detectable Broadcast

- Designated sender S with input $v \in \{0,1\}^m$
- Each player P_i outputs $v^{(i)} \in \{0,1\}^m \cup \{\perp\}$
- Consistency

  P_i, P_j honest $\Rightarrow v^{(i)} = v^{(j)} = v'$
- Validity

  S honest $\Rightarrow v' \in \{v, \perp\}$
- Completeness

  All players honest $\Rightarrow v' = v$

Either broadcast is achieved or all players abort

Detectable Broadcast

- Motivation:
  - Reduce setup assumptions to a minimum
- Applications: settings in which
  - In case of faults there is recourse to some other system
  - Adversary already has power to disrupt (secure function evaluation)
- Introduced by [Fitzi, Gisin, Maurer 2001] in context of quantum cryptography
- Stronger than "Weak Broadcast" [Lamport 83]

Previous Work on (Strong) Broadcast

With no previous setup (other than identities + start time)

- $t \geq n/3$ is impossible [LSP,PSL]
  (even randomized or computational) [KY,DD]

Other models (additional setup)

- Signature PKI (pre-distributed verification keys)
  $\Rightarrow$ Computational security for any $t < n$ [DS83]
- Preprocessing phase with broadcast [PW92]
  $\Rightarrow$ Unconditional security for any $t < n$

Previous Work on Detectable Broadcast

- [Lamport ‘83]

  Impossible for deterministic protocols when $t \geq n/3$

- Mistaken Folklore

  Impossible even for randomized or computational protocols

- [FGMR ‘02]

  Randomized protocol for $t < n/2$

  (unconditionally secure, very complex)

This Work

- Simple transformation

Broadcast protocol which requires previous setup

$\Downarrow$

no previous setup, but possibility of aborting

- Similar idea used in [Goldwasser-Lindell 2002] in context of multi-party computing

Contributions

- Protocols for Detectable Broadcast for any $t < n$
  - Computational security (signature schemes)

    $t + 3$ rounds, $O(\ n^3k\ )$ message bits per player
  - Unconditional security

    $t + 5$ rounds, $O(\ n^6(\log n + k)^3\ )$ message bits per player
- Extensions:
  - ‘Detectable’ Clock Synchronization
  - Secure Function Evaluation (Multi-party Computing)

Outline

- What is Detectable Broadcast?
- Protocols for all $t < n$
- Extensions
- Conclusions, Open questions

Outline

- What is Detectable Broadcast?
- Protocols for all $t < n$
  - General methodology
  - Illustration: Computationally secure protocol
  - Unconditionally secure protocol (in the paper)
- Extensions
- Conclusions, Open questions

Methodology

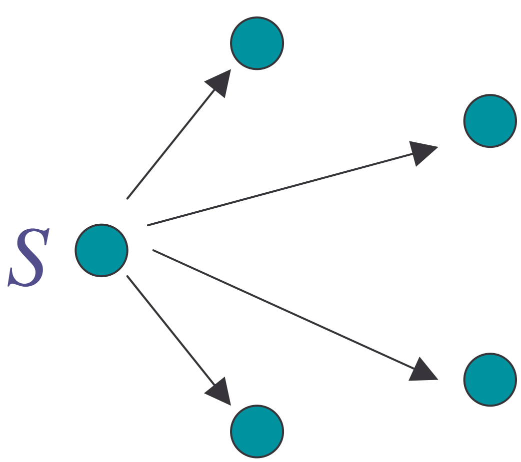
- Start from preprocessing-based protocol
  - Initial `Setup` phase assumes secure broadcast
  - Subsequent `Broadcast` phase uses only pair-wise channels
- Modify preprocessing to remove broadcast:
  - Replace with simple 'Send-Echo' protocol
- Use agreement phase to decide whether or not the preprocessing was successful

Basic Step: Send-Echo

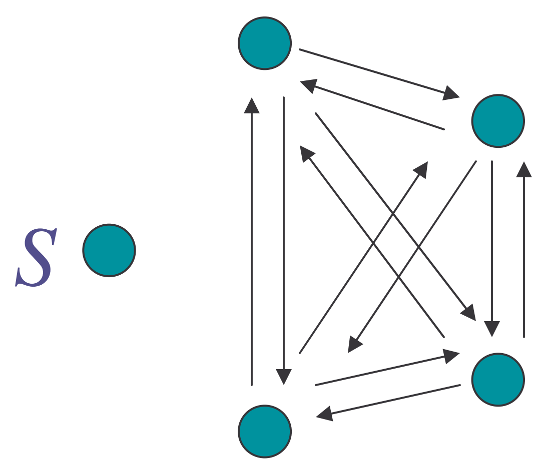
(1) Sender S sends value v to all other players

(2) Each player echoes received value $v^{(i)}$ to all other players

- Player P_i outputs
  - Value $v^{(i)}$ received from S
  - Bit $b^{(i)} =$ 1 if all echoed values agree; 0 otherwise



Round 1



Round 2

Basic Step: Send-Echo

(1) Sender S sends value v to all other players

(2) Each player echoes received value $v^{(i)}$ to all other players

- Player P_i outputs - Value $v^{(i)}$ received from S

  - Bit $b^{(i)}$ = 1 if all echoed values agree

    0 otherwise

- S honest

  $\Rightarrow$ All honest players output $v^{(i)} = v$

- If any honest player has $b^{(i)}$=1

  $\Rightarrow$ All honest players output same value v'

  i.e. Broadcast was achieved

Computationally Secure Protocol

Starting point: Dolev-Strong authenticated broadcast

1. Setup Phase (`DSSetup`)
   - P_i picks (VK_i,SK_i)
   - P_i broadcasts VK_i

2. Agreement Phase (`DSBroadcast`)
   - Achieves broadcast in $t+1$ rounds

For each $i = 1,2,\ldots,n$:

- Each P_i picks (VK_i, SK_i)

(1) Run n copies of Send-Echo (for each i: $S=P_i$ and $v = VK_i$)

- Set $b_i =$ 1 if all n Send-Echo protocols succeed
  0 otherwise

  $VK_j^{(i)}$ = Key received from P_j

(2) Run n copies of `DSBroadcast` (for each i: $S = P_i$, $v = b_i$)
using keys $VK_1^{(i)},\ldots,VK_n^{(i)}$

- Accept $VK_1^{(i)},\ldots,VK_n^{(i)}$ as valid if all received $b_j = 1$

(3) If valid, run `DSBroadcast` with real message and sender
Else abort

> If any honest player has $b_i = 1$
> $\Rightarrow$ all honest players have consistent keys
> $\Rightarrow$ `DSBroadcast` achieves broadcast

For each $i = 1,2,\ldots,n$:

- Each P_i picks (VK_i, SK_i)

(1) Run n copies of Send-Echo (for each i: $S=P_i$ and $v = VK_i$)

- Set $b_i = \begin{cases} 1 & \text{if all } n \text{ Send-Echo protocols succeed} \\ 0 & \text{otherwise} \end{cases}$

  $VK_j^{(i)}$ = Key received from P_j

(2) Run n copies of `DSBroadcast` (for each i: $S = P_i$, $v = b_i$) using keys $VK_1^{(i)},\ldots,VK_n^{(i)}$

- Accept $VK_1^{(i)},\ldots,VK_n^{(i)}$ as valid if all received $b_j = 1$

(3) If valid, run `DSBroadcast` with real message and sender

Else abort

If all honest players have $b_i = 0$

$\Rightarrow$ All honest players reject at end of phase (2)

Outline

- What is Detectable Broadcast?
- Protocols for all $t < n$
- Extensions
  - Desynchronized clocks
  - Secure Function Evaluation
    a.k.a. "Multi-party Computing" (in the paper)
- Conclusions, Open questions

Desynchronized clocks

- What if
  - Players don't start at the same time?
  - Some player wants to initiate a broadcast?

(Idea: minimize assumptions about system)

Classic Problem: Firing Squad

- Synchronous system but not all clocks start at same time
- Goal: Synchronize clocks
- Impossible for $t \geq n/3$ even with previous setup [CDDS'85]

What weaker tasks are achievable for all $t < n$?

Detectable Firing Squad

- Synchronous system but not all clocks start at same time
- Each P_i outputs $v^{(i)} \in \{\text{FIRE}, \text{FAIL}\}$ eventually
- Consistency

  P_i, P_j honest $\Rightarrow v^{(i)} = v^{(j)} = v'$

- Synchrony

  $v' = \text{FIRE} \Rightarrow$ all honest P_i terminate simultaneously

- Termination

  Any execution terminates within T steps

In case of FAIL, no synchronization

Theorem [this work]: Can tolerate any $t < n$ without previous setup

Outline

- What is Detectable Broadcast?
- Protocols for all $t < n$
- Extensions
- Conclusions, Open questions

Conclusions

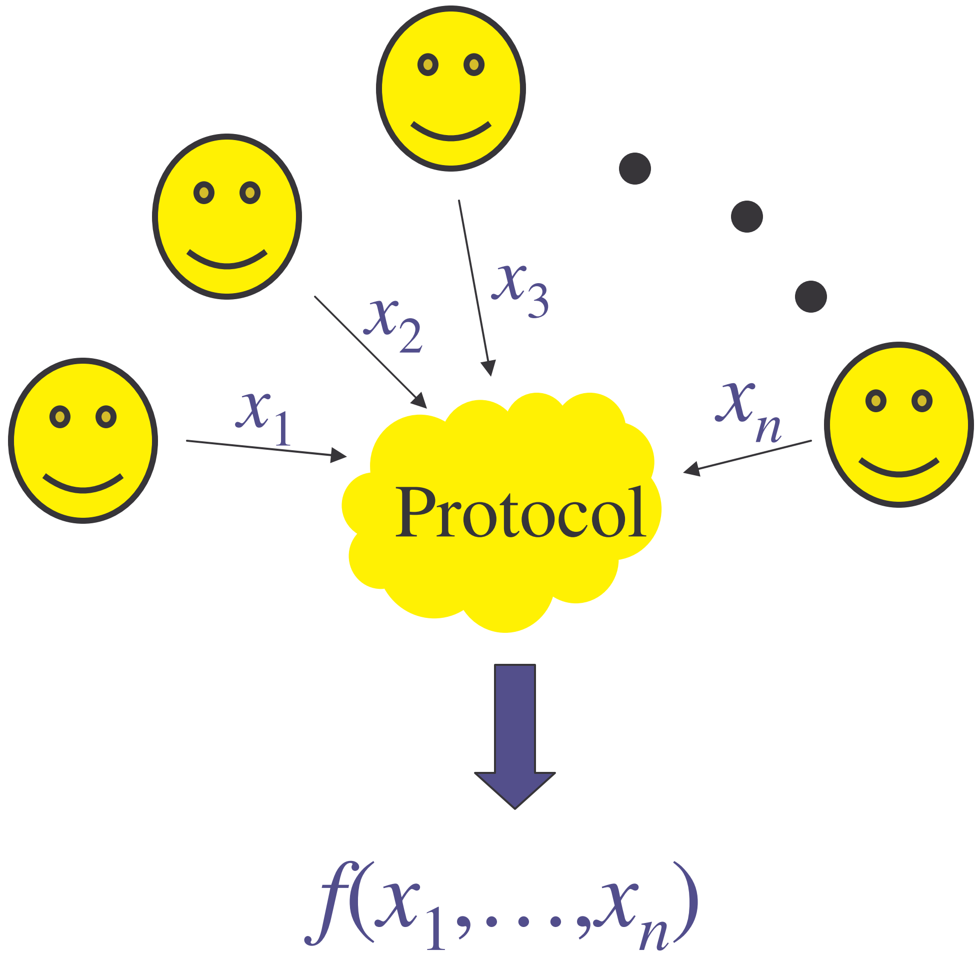
- Motivation:
  - Reduce assumptions to a minimum (identities)
- Quite a lot can be done, if you're willing to give up correction of faults in favor of detection by allowing cheaters to force abort
- Detectable broadcast protocols exist for $t < n$

Open Questions

- Reduce expected rounds below t (solved for $t < n/3$)
  - In Broadcast with Preprocessing?
  - In Detectable Broadcast?
- Asynchronous Settings
  - Does preprocessing allow any improvements?
    [partial answers known: Cachin et al.]
  - Could such protocols be modified to allow abort and remove preprocessing?

Secure Function Evaluation

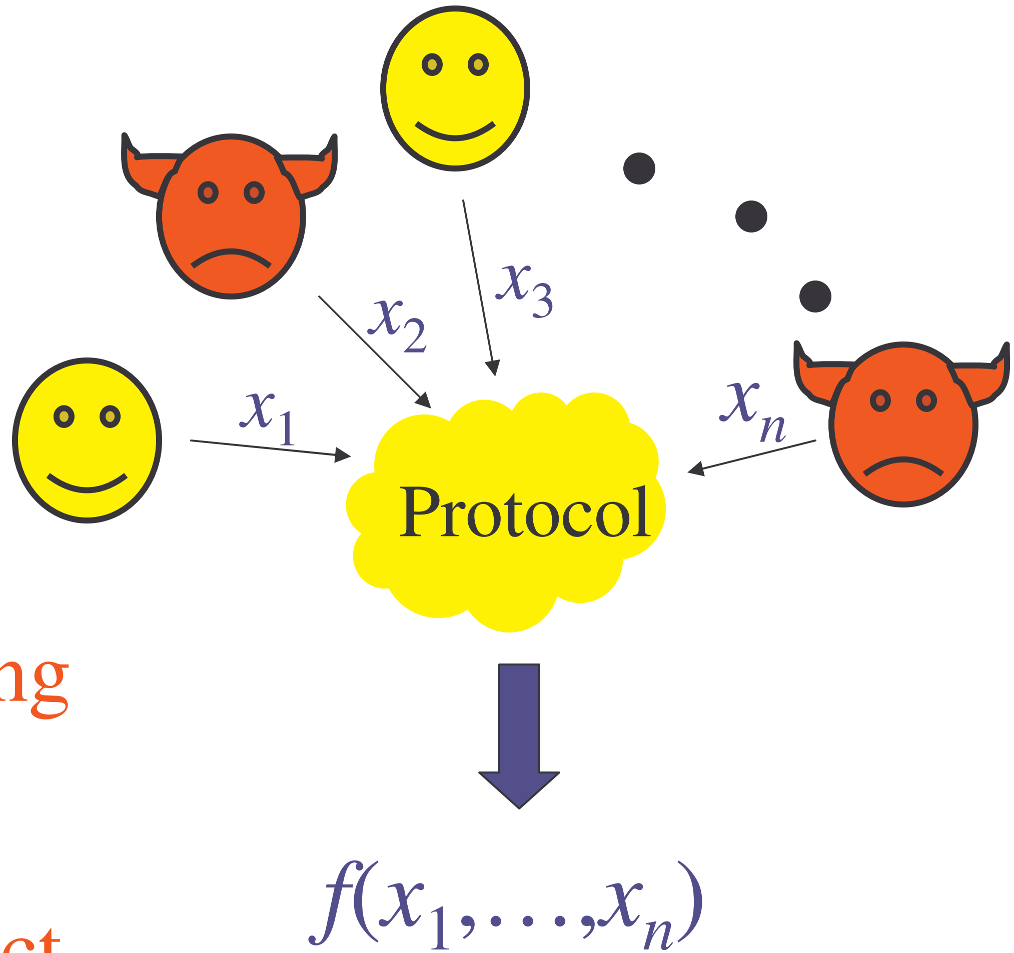
- Also called "Multi-Party Computing"
- Network of n players
- Each has input x_i
- Want to compute $f(x_1,\ldots,x_n)$ for some known function f
- *E.g.* electronic voting



Secure Function Evaluation

Even if t out of n

players try to cheat:



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

Secure Function Evaluation

Theorem [folklore]: With a faulty majority, adversary can always force SFE to abort (reduction to 2 player setting)

Theorem [BG,BMG,CR,FS,KO]: Using broadcast channel, there is a $O(\log n)$ round protocol for Secure Function Evaluation with abort tolerating any $t < n$.

Corollary [this work]: With pairwise authentic channels, we get $O(n \log n)$-round protocols for SFE with abort.

Broadcast with Non-Unison Start

- Designated sender S with input $v \in \{0,1\}^m$
- Each player P_i outputs $v^{(i)} \in \{0,1\}^m$ eventually
- Consistency

  P_i , P_j honest $\Rightarrow$ $v^{(i)} = v^{(j)} = v'$
- Validity

  S honest $\Rightarrow$ $v' = v$

Bound Δ on interval between outputs

Detectable Broadcast, Non-Unison Start

- Designated sender S with input $v \in \{0,1\}^m$
- Each player P_i outputs $v^{(i)} \in \{0,1\}^m \cup \{\perp\}$ eventually
- Consistency

  P_i, P_j honest $\Rightarrow v^{(i)} = v^{(j)} = v'$
- Validity

  S honest $\Rightarrow v' \in \{v, \perp\}$
- Completeness

  All players honest $\Rightarrow v' = v$

Bound Δ on interval between outputs

Results

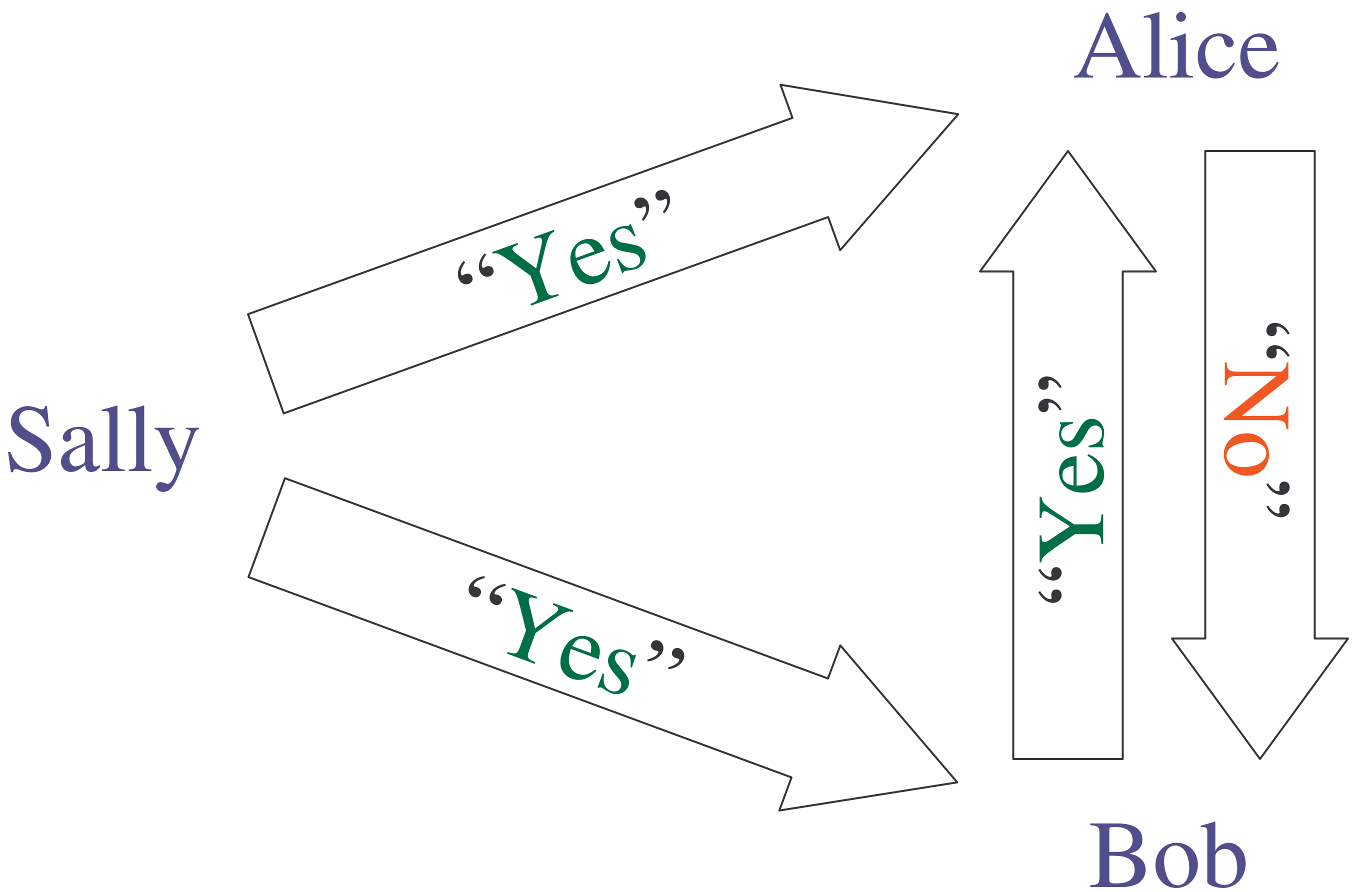
- Firing Squad
  - Possible if and only if $t < n/3$
- Broadcast with Non-Unison Start
  - No Preprocessing: Possible if and only if $t < n/3$
  - Pre-Agreed Signature Keys: Can tolerate any $t < n$
- Detectable Broadcast with Non-Unison Start
  - Possible for any $t < n$
  - Bonus: Protocol accepts $\Rightarrow$ outputs are synchronized

Detectable Byzantine Agreement Secure Against Faulty Majorities

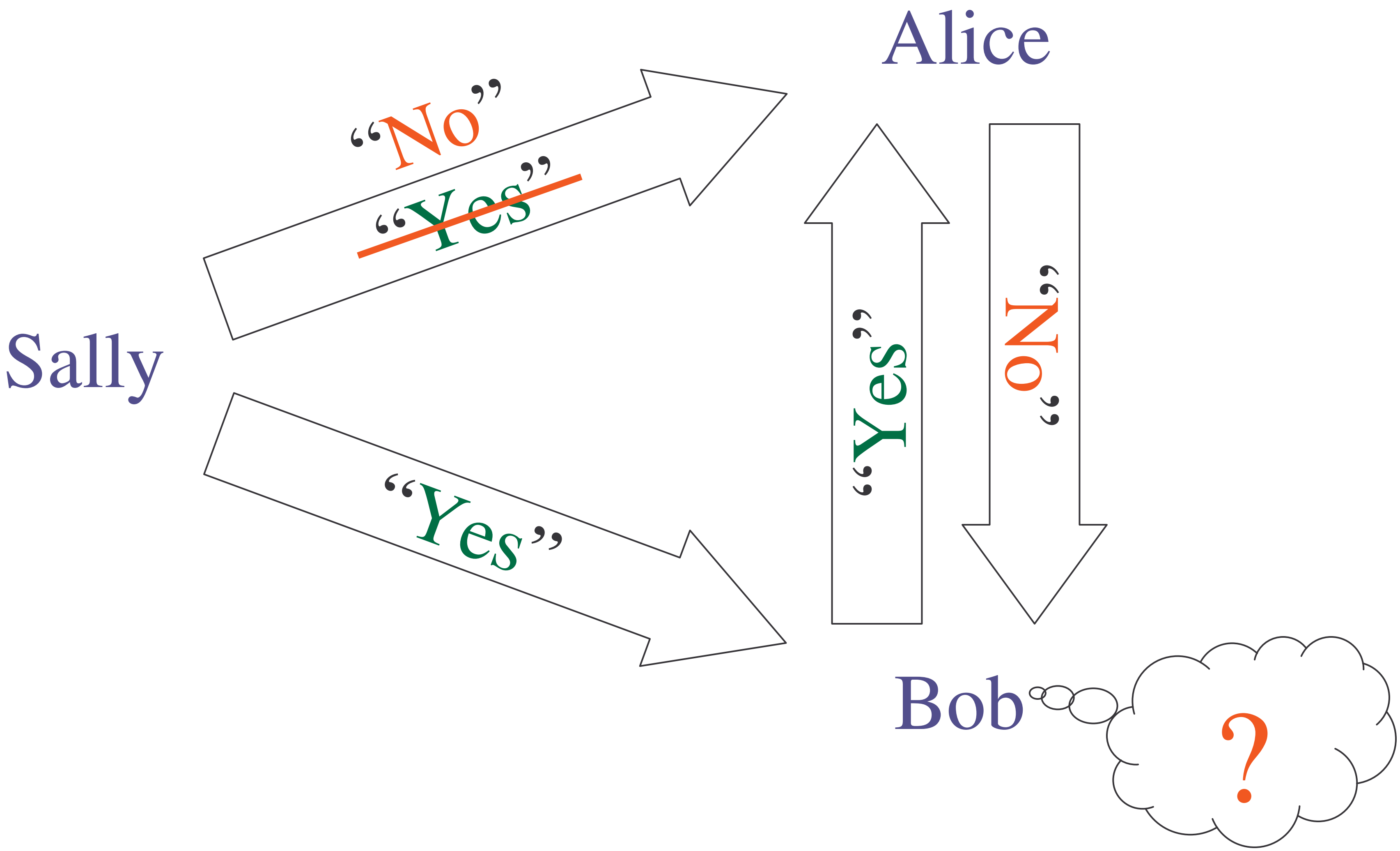
or

How (Well) You Can Agree When You've Never Agreed Before

Agreement is difficult....



Agreement is difficult....



Review: Broadcast (Byzantine Agreement)

- Designated sender S with input $v \in \{0,1\}^m$
- Each player P_i outputs $v^{(i)} \in \{0,1\}^m$
- Consistency

  P_i, P_j honest $\Rightarrow v^{(i)} = v^{(j)} = v'$
- Validity

  S honest $\Rightarrow v' = v$

Previous Work on (Strong) Broadcast

With no previous setup (other than identities + start time)

- $t < n/3$: Unconditionally secure protocols [LSP80]
- $t \geq n/3$: Impossible (even computational or randomized)

With preprocessing (broadcast available)

- Signature PKI (pre-distributed verification keys)
  $\Rightarrow$ Computational security for any $t < n$ [DS83]
- Preprocessing phase with broadcast
  $\Rightarrow$ Unconditional security for any $t < n$ [PW92]

Previous Work on Detectable Broadcast

- [Lamport '83]

  Impossible for deterministic protocols when $t \geq n/3$

- Mistaken Folklore

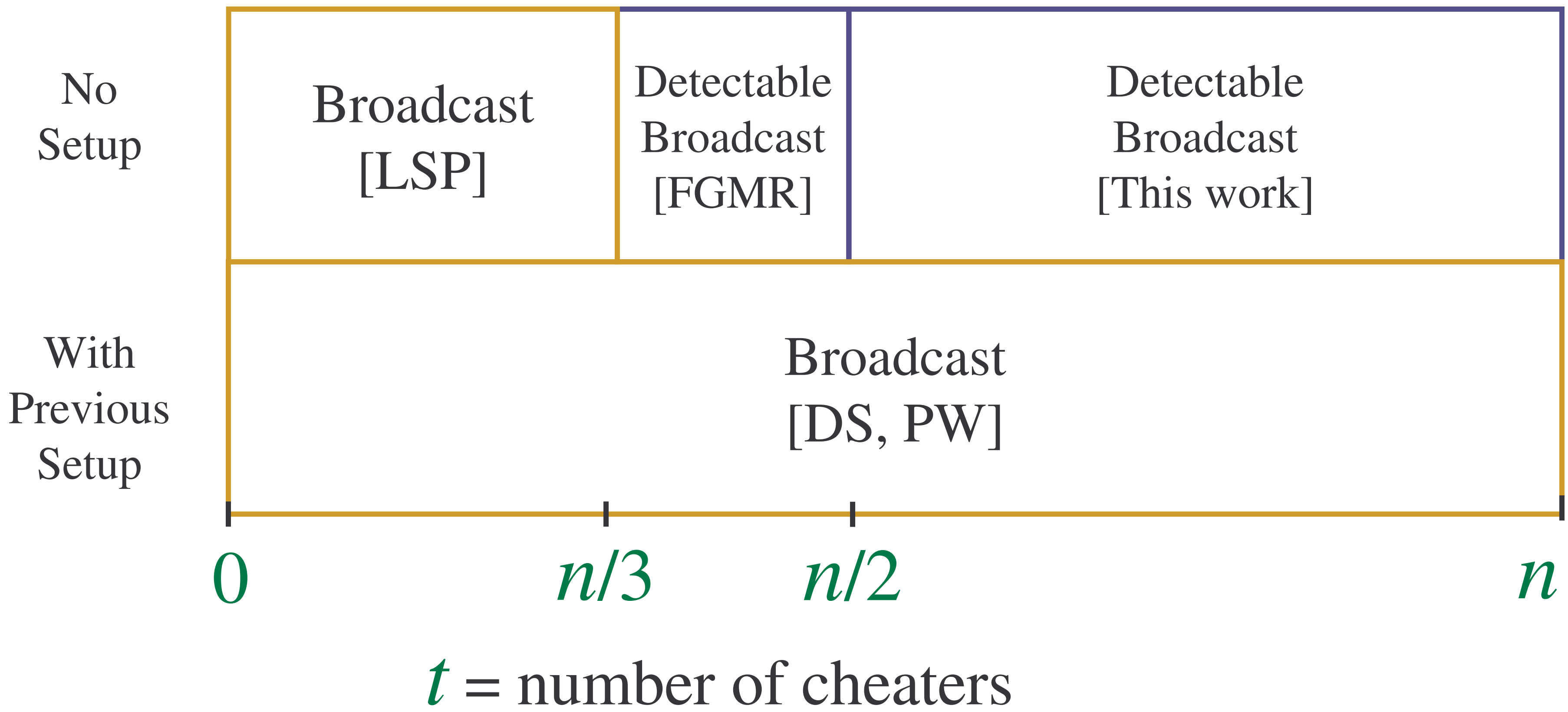
  Impossible even for randomized or computational protocols

- [FGMR '02]

  Randomized protocol for $t < n/2$

  (unconditionally secure, very complex)

Feasibility



Unconditionally Secure Protocol

- Start from Pfitzmann-Waidner protocol for broadcast from preprocessing
- Step 1: Modify Pfitzmann-Waidner for efficiency
- Step 2: Get rid of preprocessing as before

Starting point: Pfitzmann-Waidner

1. Setup Phase (`PWSetup`)
   - Broadcast available
   - $O(n^2)$ rounds
2. Agreement Phase (`PWBroadcast`)
   - Achieves broadcast in $t+1$ rounds
   - One setup allows $poly(n,k)$ broadcasts

Always succeeds (with high prob.)

Step 1: Modified PW protocol

1. Simplified Setup Phase (`SPWSetup`)
   - Broadcast available
   - ~~$O(n^2)$ rounds~~ 3 rounds

Either succeeds
or
all honest players
abort

2. Agreement Phase (`PWBroadcast`)
   - Achieves broadcast in $t+1$ rounds
     if setup phase succeeded
   - One setup allows *poly*(n,k) broadcasts

For each $i = 1,2,\ldots,n$:

- Run SPWSetup using Send-Echo instead of broadcasts
- Set $b_i =$ 1 if all Send-Echo protocols succeed
  0 otherwise
  $V^{(i)} =$ Values received during SPWSetup
- Run PWBroadcast with $S = P_i$, $v = b_i$ and parameters $V^{(i)}$
- Accept $V^{(i)}$ as valid if all received $b_j = 1$
- If valid, run PWBroadcast with real message and sender

  Else abort

Any honest player has $b_i = 1$
$\Rightarrow$ Setup completed successfully
$\Rightarrow$ PWBroadcast achieves broadcast