

Fault-tolerant Distributed Consensus

(Slides courtesy : former student B. Laasya)

Ashish Choudhury

International Institute of Information Technology (IIIT) Bangalore

Roadmap

□ **Part I** : Byzantine agreement

- ❖ Problem definition and practical applications
- ❖ Known results

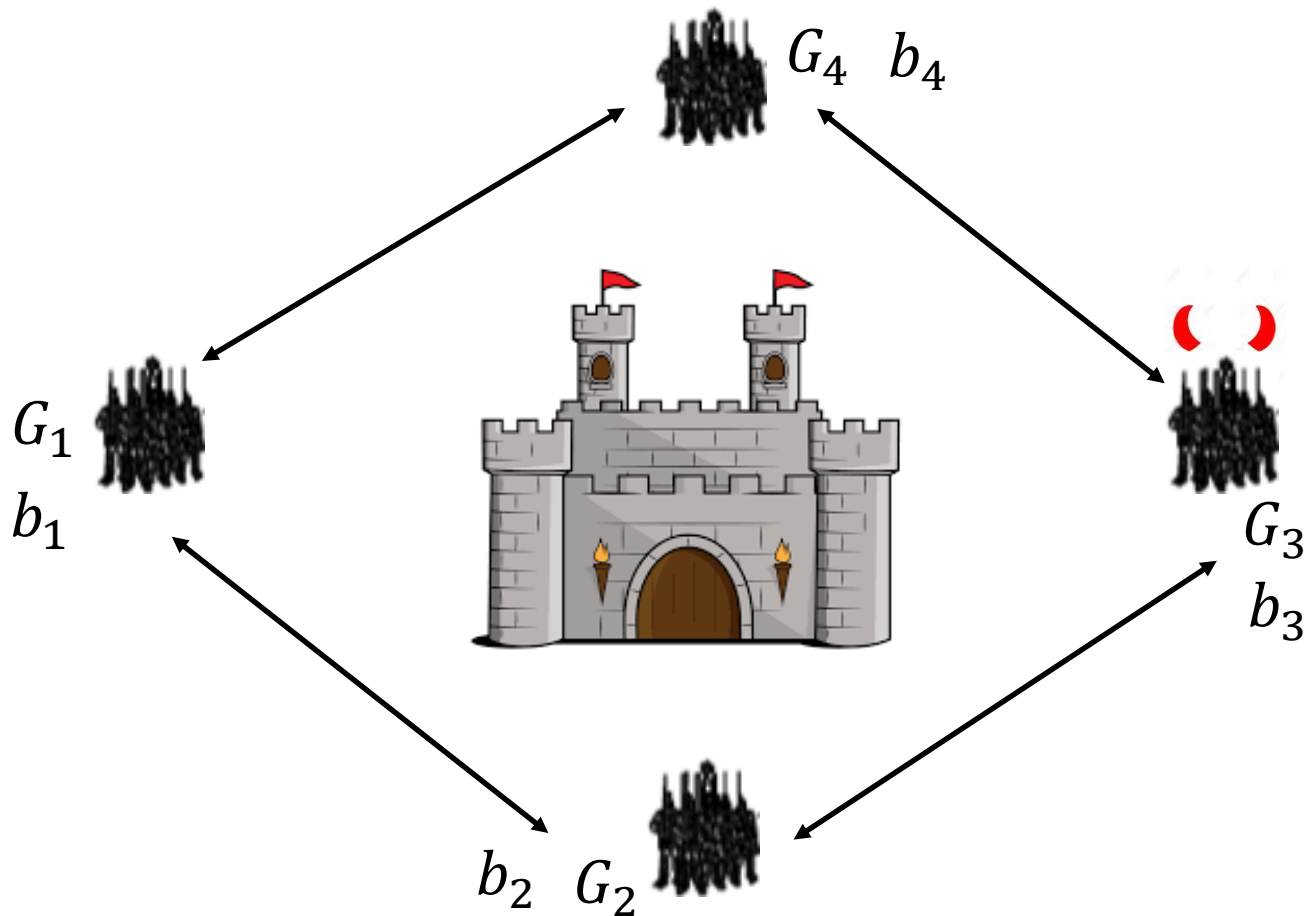
□ **Part II** : Randomized Byzantine agreement

- ❖ Framework of Rabin and BenOr
- ❖ Instantiating the framework using verifiable secret-sharing (VSS)

PART I

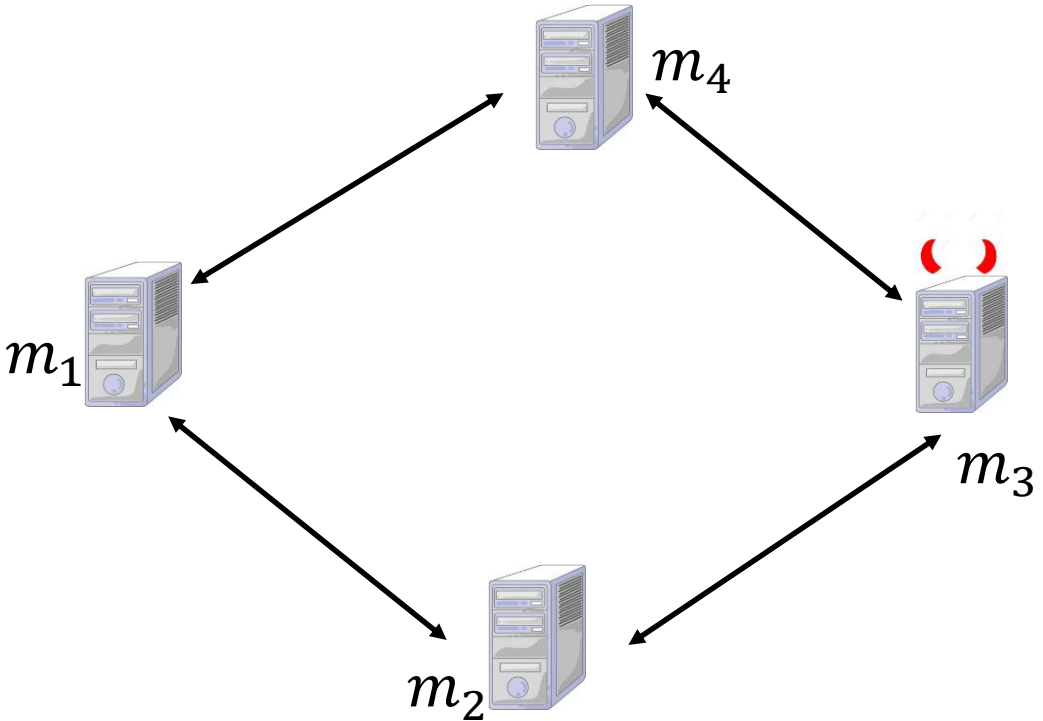
Byzantine Generals Problem

Leslie Lamport, Robert E. Shostak, Marshall C. Pease: ^[SEP]The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst. 4(3): 382-401 (1982)



- n generals, connected by **pair-wise private, authentic channel**
- Each general has a **secret plan (bit)**: retreat (0) or attack (1)
 - ❖ Up to t generals may be **traitor**
- Goal: to come up with a **common action plan for the honest generals**
 - ❖ Should be equal to the action plan of the **honest generals** if they all had the **same individual action plan**

Byzantine Generals Problem : CS Abstraction



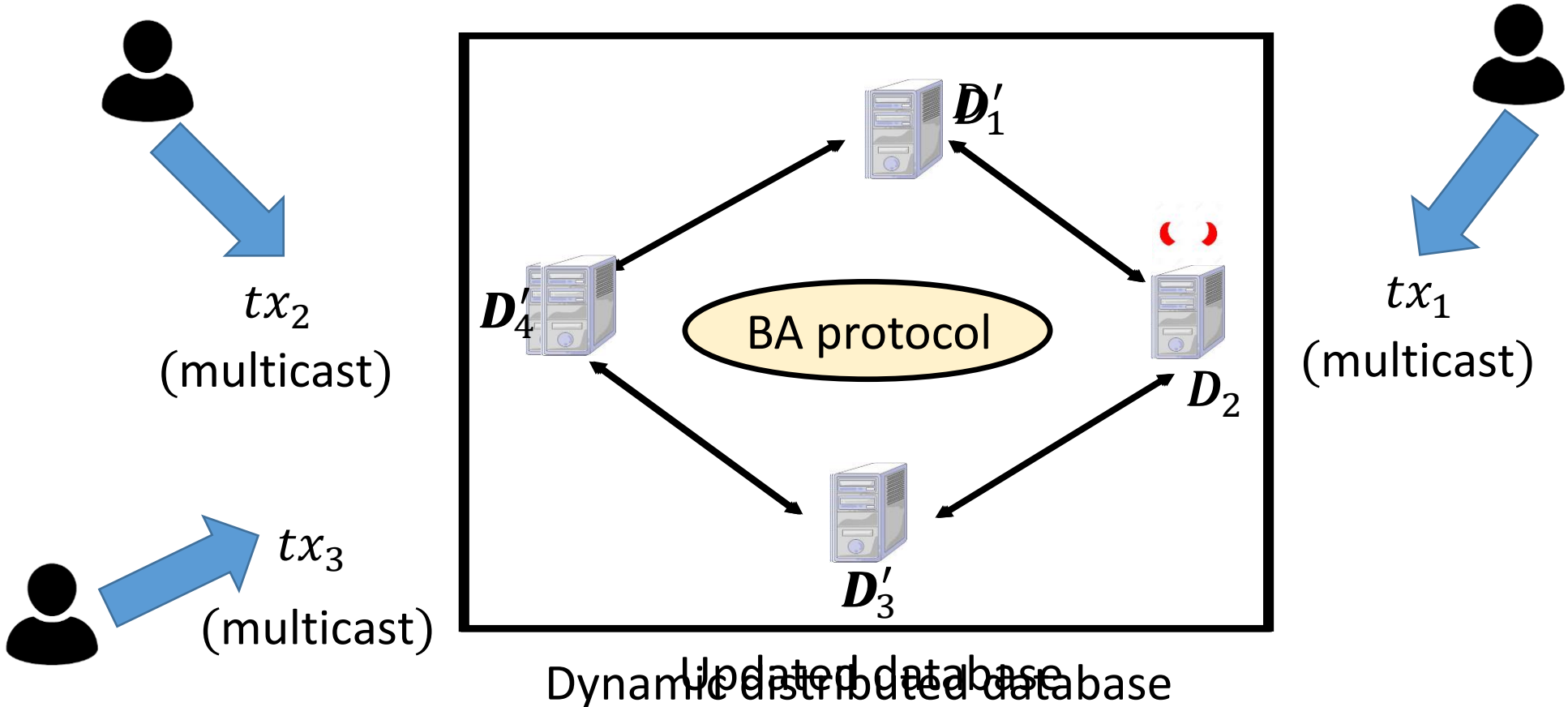
- n mutually-distrusting parties
- Up to t corruptions
- Goal: to design a distributed protocol, allowing the honest parties to agree on a common output

$m_1 \ m_2 \ m_3 \ m_4 \ \dots \ m_{n-1} \ m_n$ ----- m Agreement

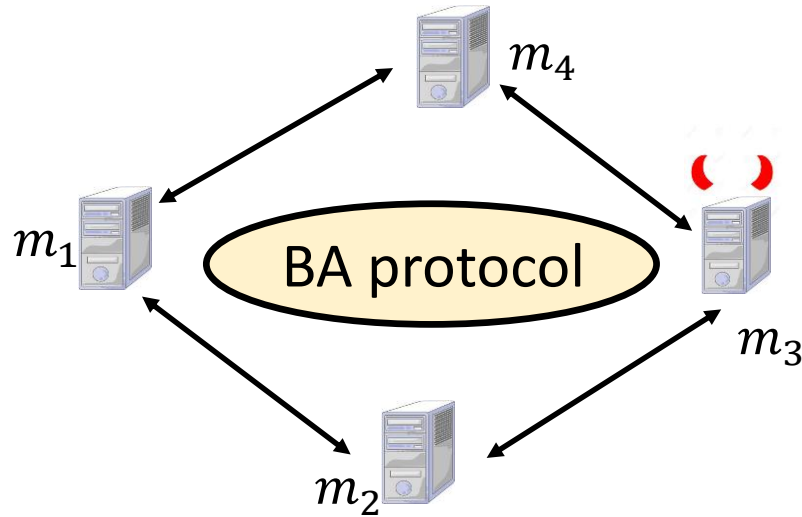
$m \ m \ m_3 \ m \ \dots \ m_{n-1} \ m$ ----- m Validity

Byzantine Agreement Problem : Applications

- ❑ Secure multi-party computation (MPC) protocols
- ❑ State-machine replication / distributed databases



The Landscape (Vishwaroopam) of BA Problem



- ❑ **Widely-studied** by both the distributed-computing, as well as CRYPTO community
 - ❖ JACM
 - ❖ PODC
 - ❖ STOC
 - ❖ FOCS
 - ❖ DISTRIBUTED COMPUTING
 - ❖ DISC

Some of the widely-studied settings

Level of synchronization

- ❖ Completely synchronous
- ❖ Partially synchronous
- ❖ Completely asynchronous

Type of faults

- ❖ Crash
- ❖ Malicious (Byzantine)

Setup available

- ❖ PKI setup
- ❖ No setup

Type of channels

- ❖ Private-and-authentic
- ❖ Authentic

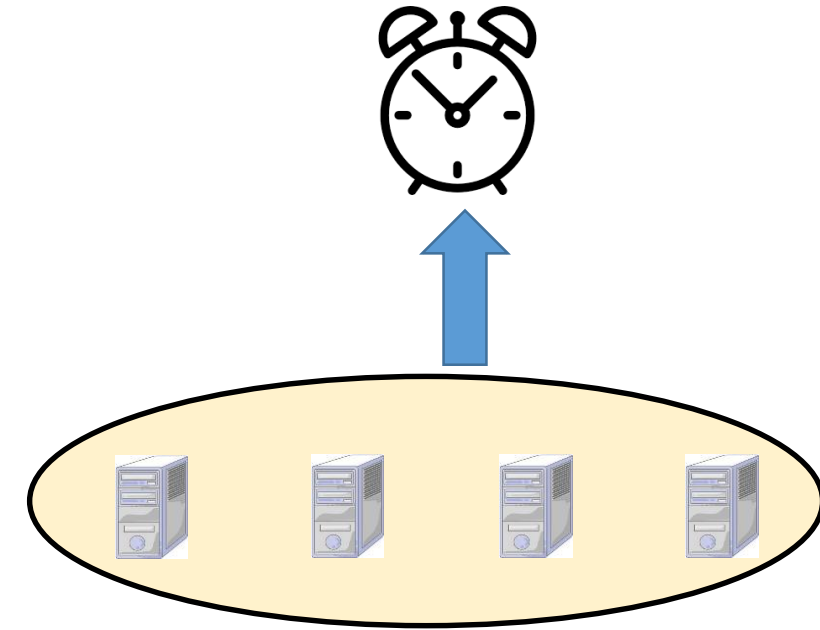
Difficulty level



Known Results in the Synchronous Setting

□ **Synchronous communication model**: parties synchronized by a **global clock**

- ❖ Protocol operates as a sequence of communication **rounds** : sequence of **compute-send-receive**
- ❖ Channels have a **fixed known delay**, say Δ
- ❖ An **expected message** not arriving within time $\Delta \Rightarrow$ **corrupt sender**



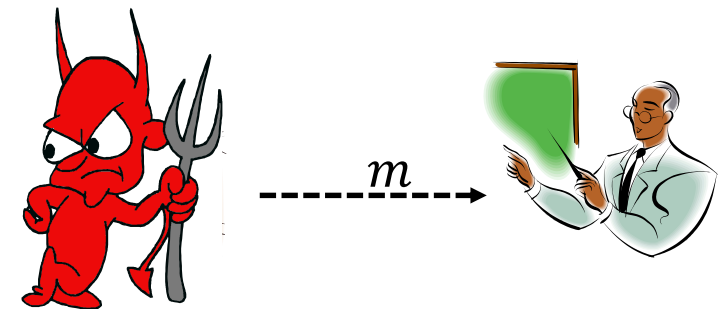
□ BA (**with or without any setup**) tolerating t **Byzantine faults** possible only if $t < n/2$

□ If a PKI setup is available, then BA possible with $t < n/2$
[D. Dolev and H. Strong, *Siam Journal of Computing* 1983]

□ With **no setup**, BA possible if and only if $t < n/3$ [L. Lamport, R. E. Shostak and M. C. Pease, *ACM Trans. Program. Lang. Syst.* 1982]

❖ Presented **inefficient protocols**

❖ **Efficient protocols**: [P. Berman, J. A. Garay and K. J. Perry, *FOCS* 1989], [J. A. Garay and Y. Moses, *STOC* 1993]

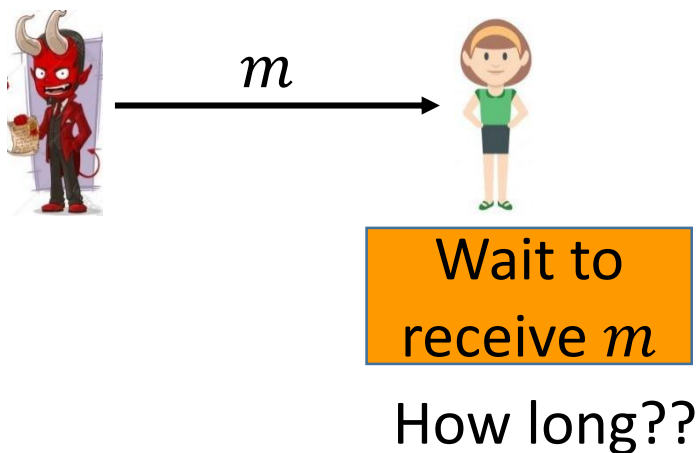


More Practical Setting: Asynchronous Model

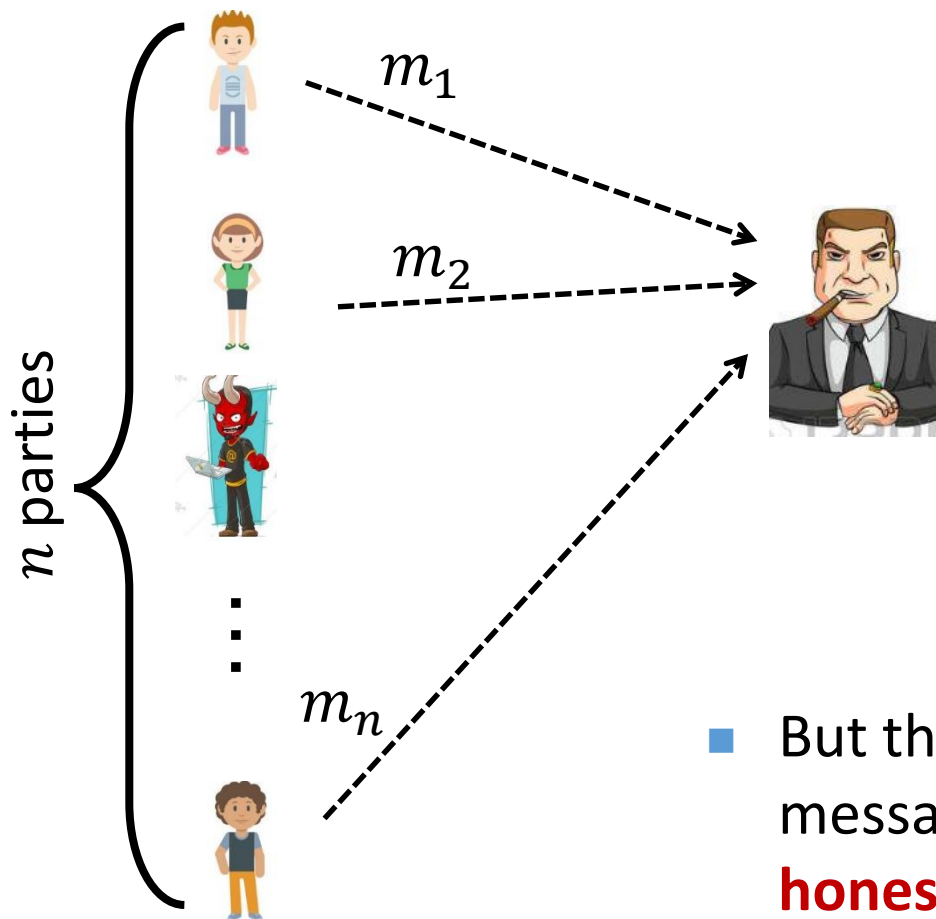
Asynchronous Network



- No Global Clock
- Channels unbounded delay
- Waiting time is not known



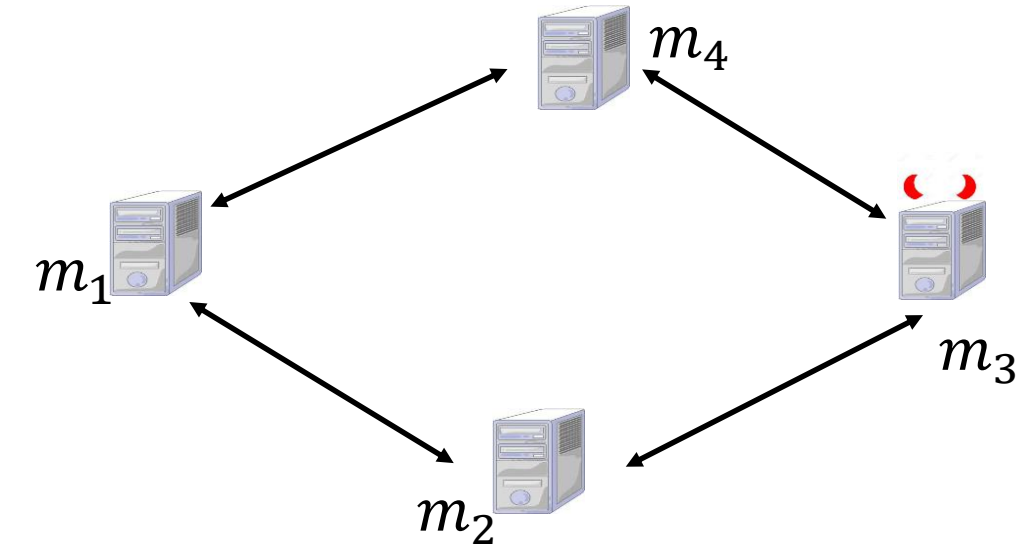
No distinction between a **slow** (but honest) **sender** and a **corrupt sender**



- Waiting for all results in **endless waiting!**
- Can afford to wait for $(n - t)$ parties
- But this can lead to **ignoring** messages of t **potentially honest parties**

In the asynchronous setting, the network itself is the adversary

BA Problem in the Asynchronous Setting : ABA



- n **mutually-distrusting parties**, up to t **corruptions**
- Completely **asynchronous network**
- **Goal**: to design a **distributed protocol**, allowing the **honest parties** to agree on a **common output**

$m_1 \ m_2 \ m_3 \ m_4 \ \dots \ m_{n-1} \ m_n \ \dashrightarrow \ m$

Agreement

$m \ m \ m_3 \ m \ \dots \ m_{n-1} \ m \ \dashrightarrow \ m$

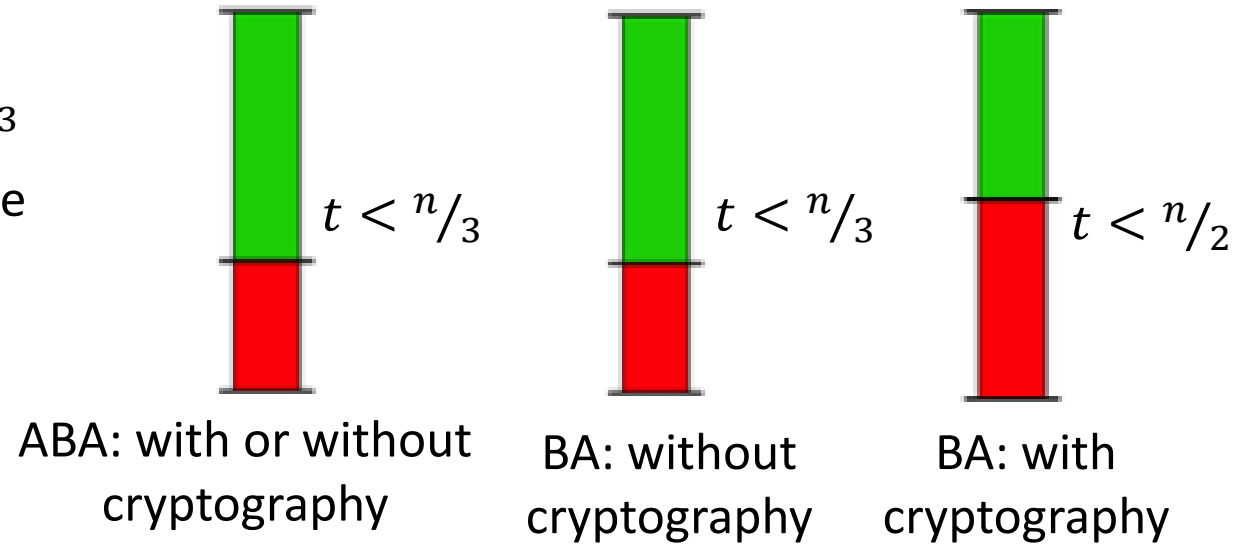
Validity

Termination

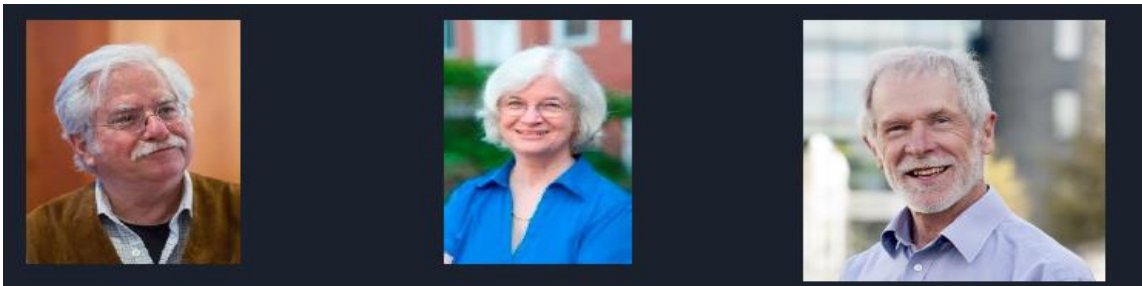
If all honest parties participate in the protocol, then all honest parties eventually terminate the protocol with an output

ABA Problem : Known Results

- ABA tolerating t **Byzantine faults** possible only if $t < n/3$
 - ❖ Holds, **even if a PKI setup is available** and parties are allowed to use cryptography
 - ❖ In the **synchronous setting**, using **cryptography increases the resilience** from $t < n/3$ to $t < n/2$



- **FLP Impossibility results for ABA:** Don't even dare to design a **deterministic** ABA protocol



[M. J. Fischer, N. A. Lynch and M. S. Paterson, JACM 1985]

Any **deterministic ABA** protocol will have **non-terminating runs**, even if one party crashes

How to Circumvent FLP Impossibility Result ?

[M. J. Fischer, N. A. Lynch and M. S. Paterson, JACM 1985]: any **deterministic ABA protocol** will have **non-terminating runs**, even if one party crashes



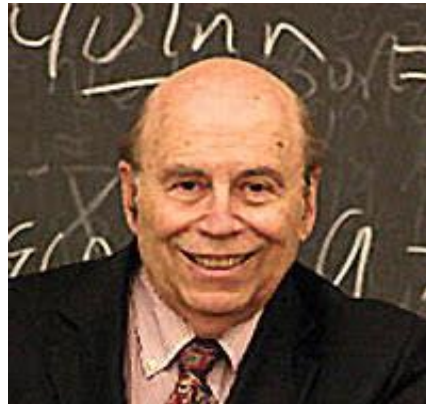
❑ Does FLP impossibility result mean the end of ABA ?

❖ No

❑ A common approach to circumvent FLP impossibility result --- “embrace” **randomness**



[M. Ben-Or, PODC 1983]

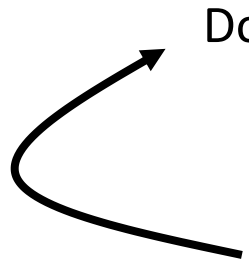


[M. Rabin, FOCS 1983]

❖ **$(1 - \lambda)$ -terminating ABA**: honest parties terminate, with probability $1 - \lambda$

❖ **Almost-surely terminating ABA**: honest parties terminate, with probability 1

Almost-surely vs $(1 - \lambda)$ -terminating




```
Do
{
  b ← Coin-Toss();
} while (b = 0)
```

- ❖ Probability that **loop does not terminate** after k iterations : $(\frac{1}{2})^k$
- ❖ Probability that **loop terminates** after k iterations : $1 - (\frac{1}{2})^k$
- ❖ Probability that loop **eventually terminates**:
$$\lim_{k \rightarrow \infty} 1 - (\frac{1}{2})^k = 1$$
- ❖ **Expected** number of iterations:

$$\sum_{k=1}^{\infty} k \cdot \left(\frac{1}{2}\right)^k = 2$$

```
Do
{
  b ← Coin-Toss();
  Wait for event E to occur;
} while (b = 0)
```



- ❖ **Conditioned** on the event that **event E occurs in each iteration**, the loop eventually terminates, with probability 1
- ❖ Conditioned on the event that event E occurs in each iteration, the expected number of iterations is 2
- ❖ If **event E does not occur in some iteration**, then the **loop never terminates**, even if $b = 1$

Necessity of $t < n/3$ for Asynchronous BA

□ Goals of an ABA protocol (apart from termination):

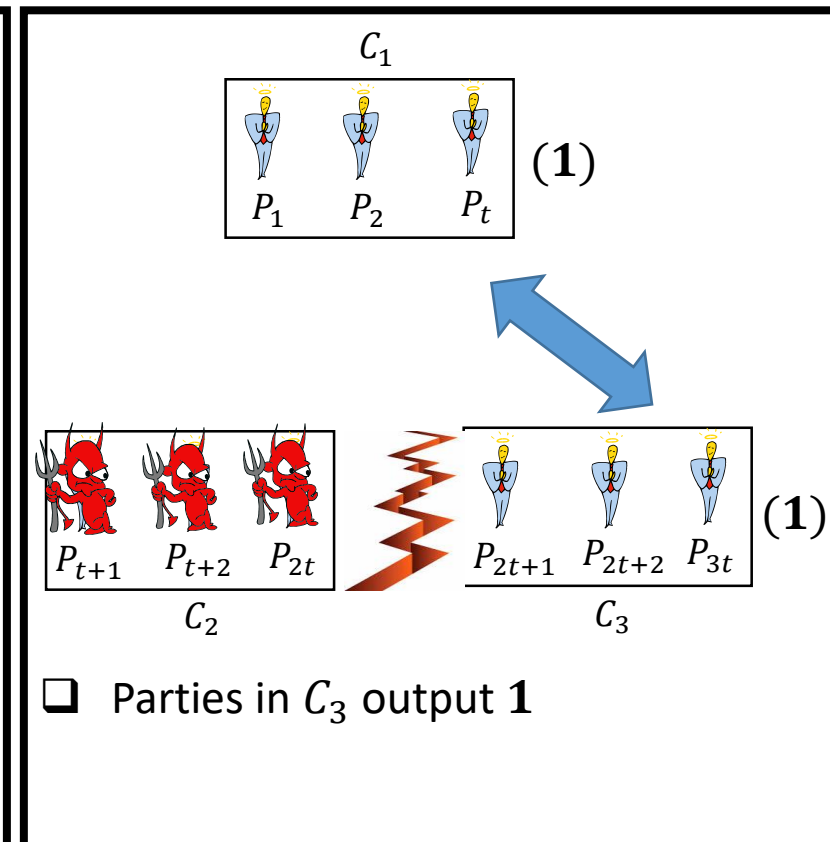
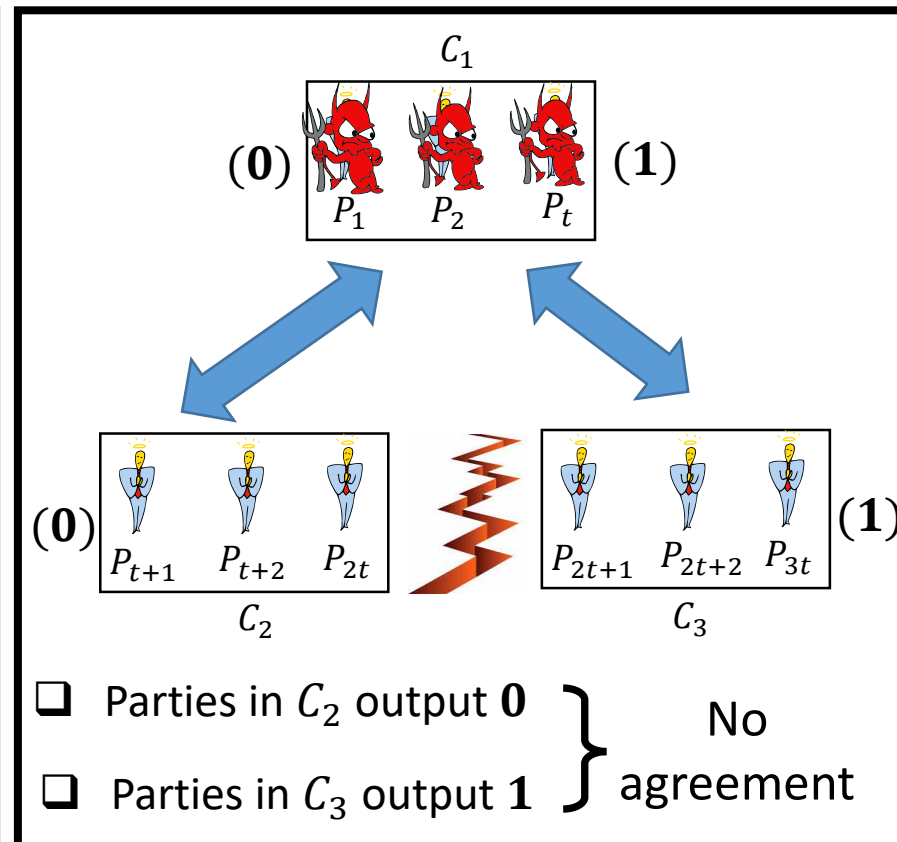
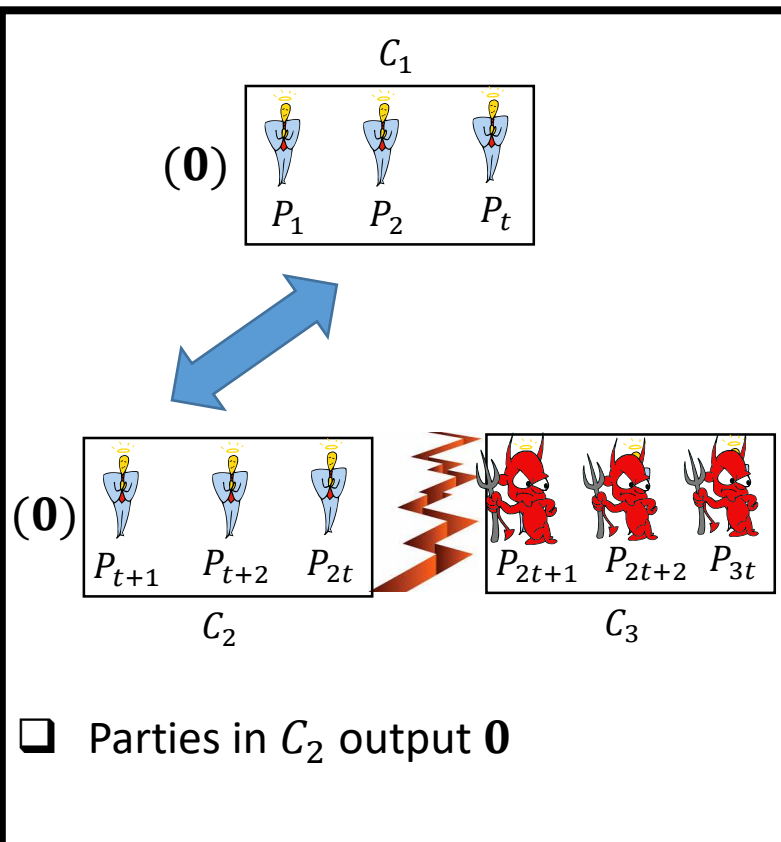
- ❖ Validity
- ❖ Agreement

□ Theorem: ABA possible only if $t < \frac{n}{3}$

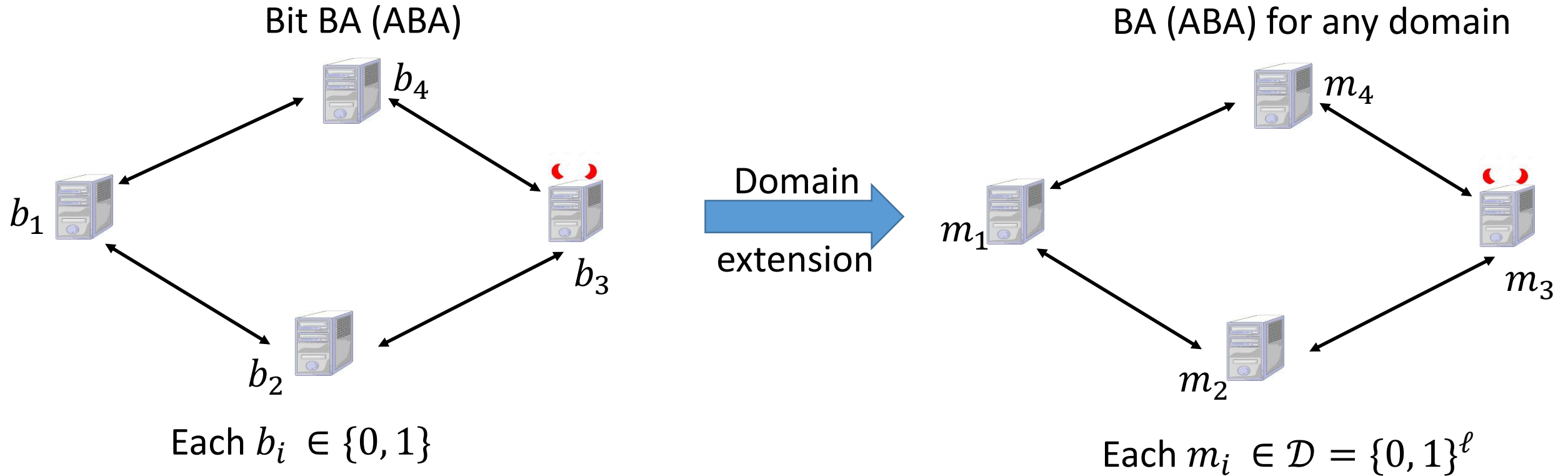
❖ Proof by **contradiction**

❖ Let Π be an ABA protocol with $n = 3t$

□ Consider the following executions of Π



From Bit BA (ABA) to BA (ABA) for Any Domain



[R. Turpin and B. A. Coan, IPL 1984] [M. Fitzi and M. Hirt, PODC 2006] [A. Patra, OPODIS 2011]

[M. Hirt and P. Raykov, ASIACRYPT 2014] [C. Ganesh and A. Patra, PODC 2016]

[[A. Choudhury](#), ICDCN 2017]

Roadmap

□ **Part I** : Byzantine agreement

- ❖ Problem definition and practical applications
- ❖ Known results

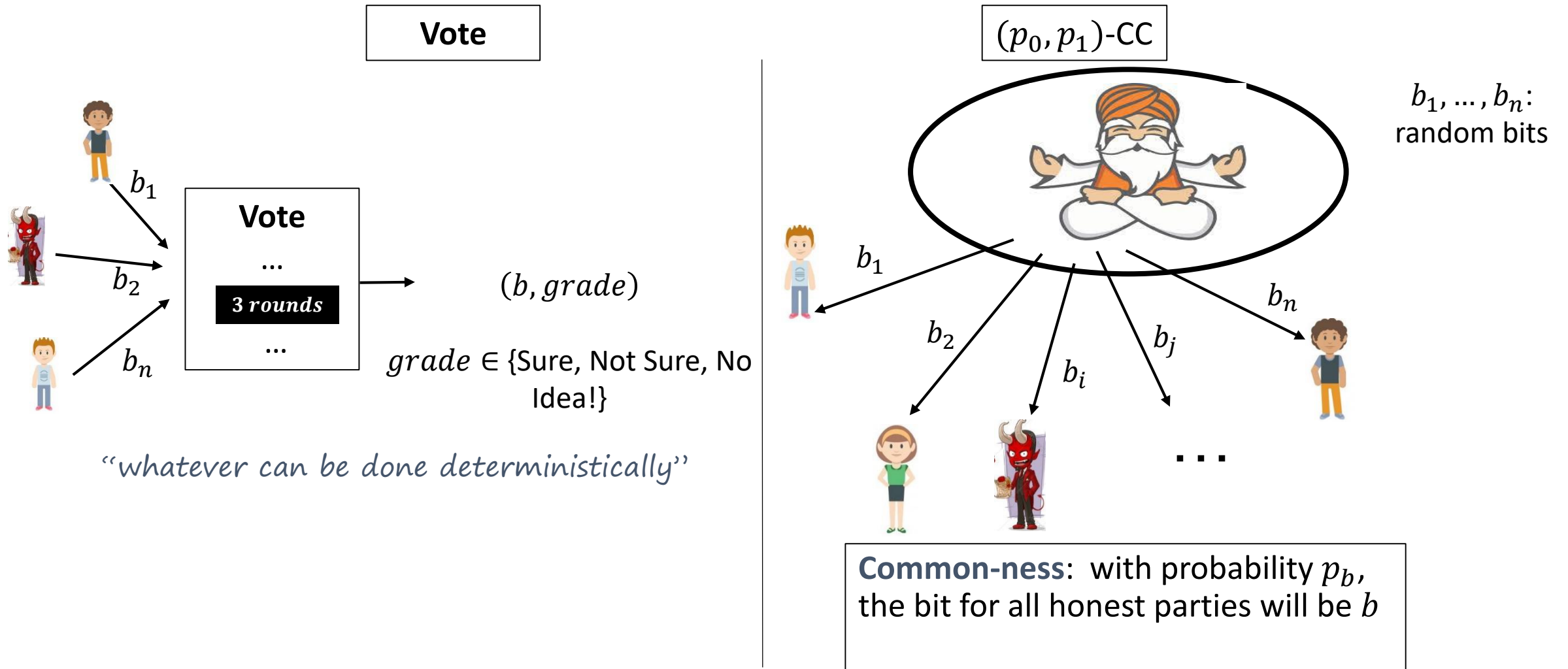


□ **Part II** : Randomized Byzantine agreement

- ❖ Framework of Rabin and BenOr
- ❖ Instantiating common-coin using verifiable secret-sharing (VSS)

PART II

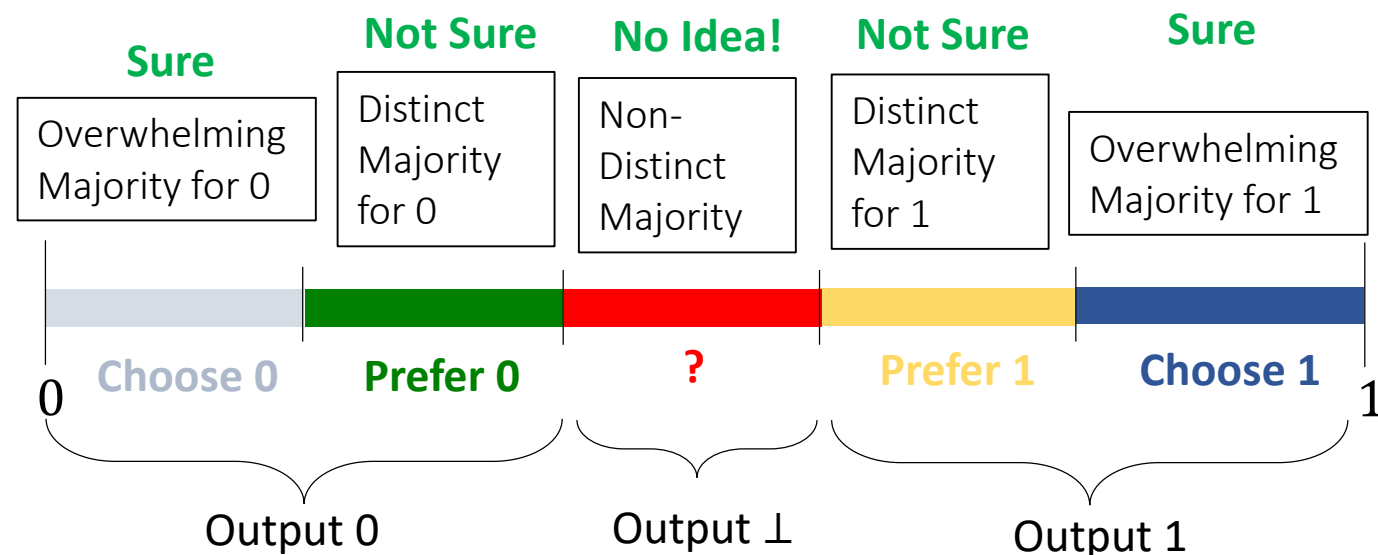
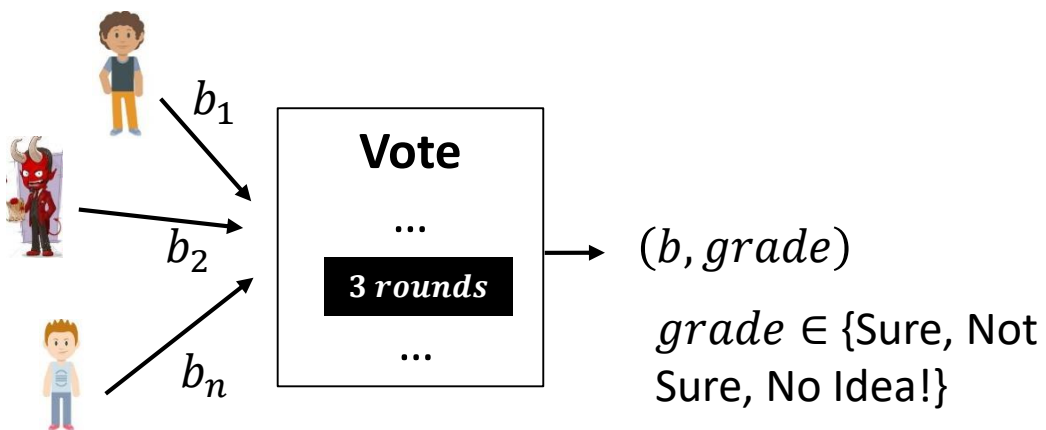
Common Framework for Randomized BA (Rabin, Ben-Or)



If p_0 and p_1 are constant, then expected constant number of iterations of **Vote + CC** \rightarrow ABA

(Asynchronous) Vote Protocol

- A **deterministic protocol**, with 3 rounds of **asynchronous communication**



- Properties

- ❖ **All honest parties same input bit b** \Rightarrow all honest parties output (b, Sure) } **Overwhelming majority** for bit b
- ❖ If **some honest party outputs (b, Sure)** \Rightarrow every other honest party outputs either (b, Sure) or $(b, \text{Not Sure})$ } **Distinct majority** for bit b
- ❖ If some honest party outputs $(b, \text{Not Sure})$ AND **no honest party has output (b, Sure)** \Rightarrow every other honest party outputs either $(b, \text{Not Sure})$ or $(\perp, \text{No Idea!})$ } **Non-distinct majority** for bit b

Vote + CC \Rightarrow Randomized BA

How to emulate trusted guruji ?

(p_0, p_1) -CC



b_1 b_2 b_3 b_4 ... b_{n-1} b_n

Vote

Let output be b for some parties and \perp for others.

b b \perp b ... \perp b

Call CC-TTP

Let output be r .

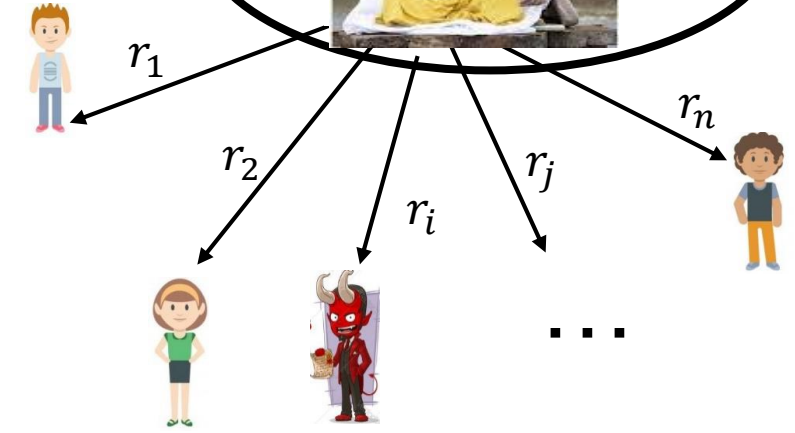
b b r b ... r b

If $r = b$?

Special case. ABA will terminate.

If $r \neq b$?

Similar to the current iteration



Iteration

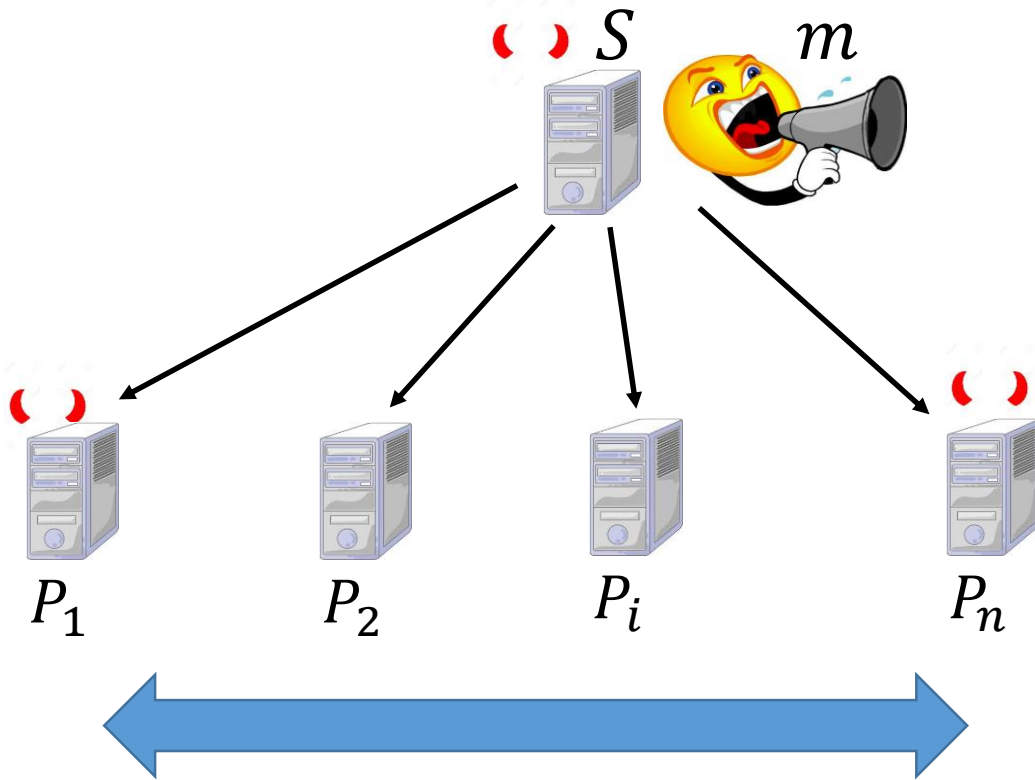
□ CC emulated by running a protocol

❖ $(1 - \lambda)$ -terminating CC \Rightarrow $(1 - \lambda)$ -terminating ABA

❖ always-terminating CC \Rightarrow almost-surely terminating ABA

Tools Deployed for Instantiating CC

□ Asynchronous reliable broadcast:



□ n mutually-distrusting parties

□ Up to t corruptions (potentially including S)

□ Termination:

❖ **Honest** $S \Rightarrow$ all honest parties eventually terminate

❖ Some honest P_i terminates \Rightarrow all honest parties eventually terminate

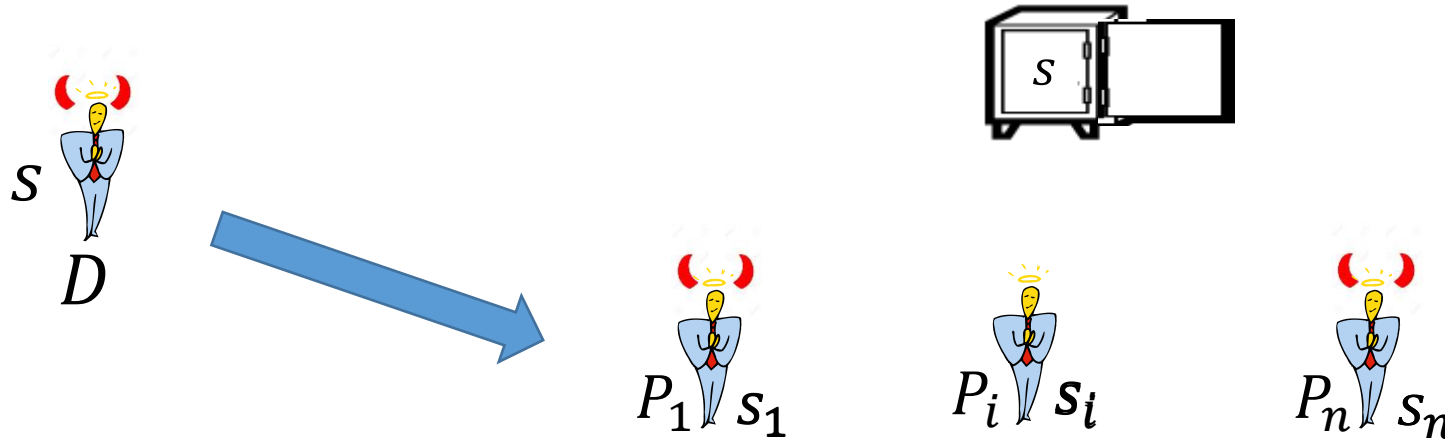
□ Agreement:

❖ All honest parties upon terminating output m^* , where $m^* = m$, if S is honest

□ [Bracha, PODC 1984]: asynchronous broadcast protocol with $n = 3t + 1$

Tools Deployed for Instantiating CC

- Asynchronous Verifiable Secret Sharing (**AVSS**): a pair of protocols (Sh, Rec)
- n **mutually-distrusting parties** □ Up to t corruptions (**potentially including D**)
- During Sh, an **unknown secret** s is shared □ During Rec, shared secret **publicly reconstructed**

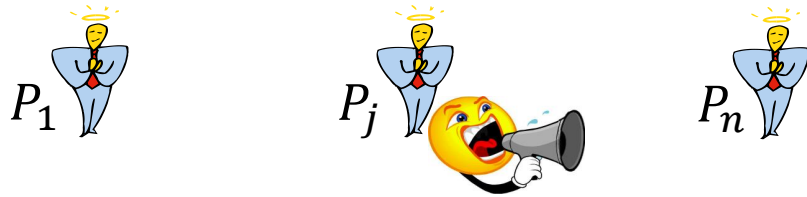


□ Termination:

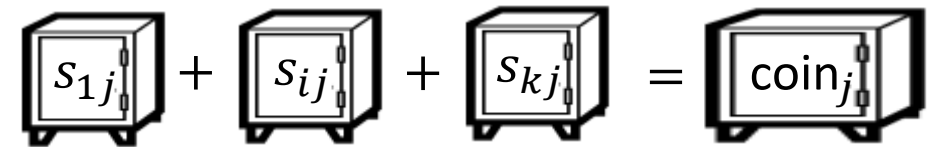
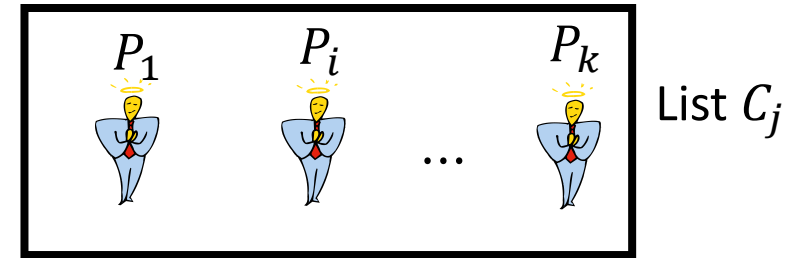
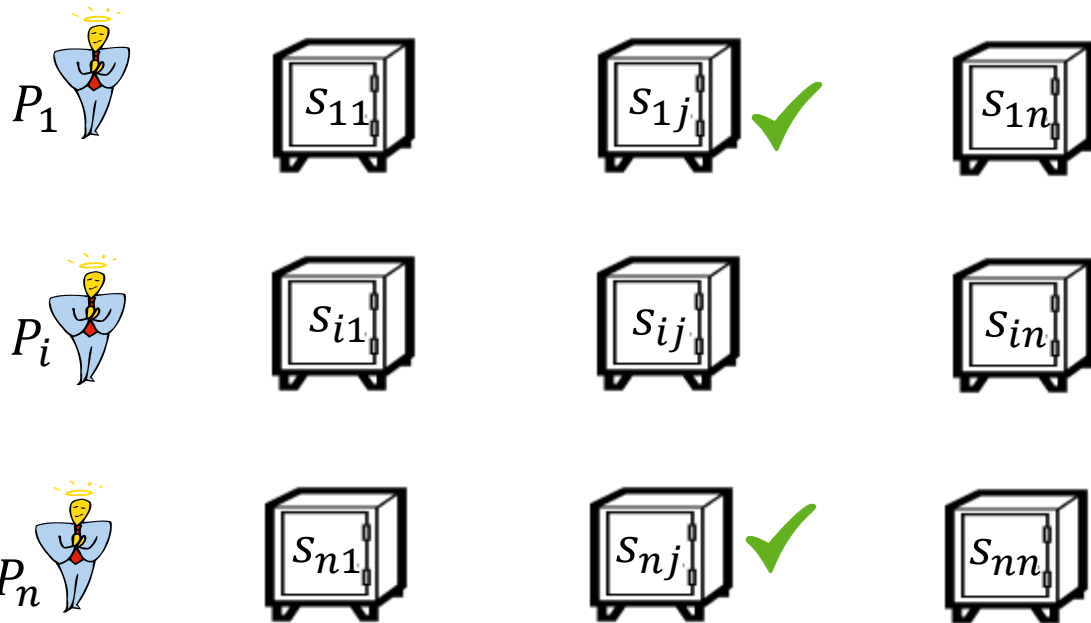
- **Correctness:** if honest parties **eventually terminate Sh**, then D has **some secret** s^* and later only s^* is reconstructed during Rec -- $s^* = s$ holds for an **honest D**
 - ❖ Some **honest P_i terminates Sh** \Rightarrow all honest parties **eventually terminate Sh**
- **Privacy:** **Honest D** \Rightarrow secret s is hidden till Rec
 - ❖ **Honest parties** participate in Rec \Rightarrow all honest parties **eventually terminate Rec**

Instantiating CC Using AVSS (Sh, Rec)

- Each P_i shares a **random and private element** $s_{ij} \in \mathbb{F}$ on the behalf of P_j --- sharing instance Sh_{ij}



- Claim:** a valid C_j defines a **random and unknown value** $\text{coin}_j \in \mathbb{F}$ with P_j

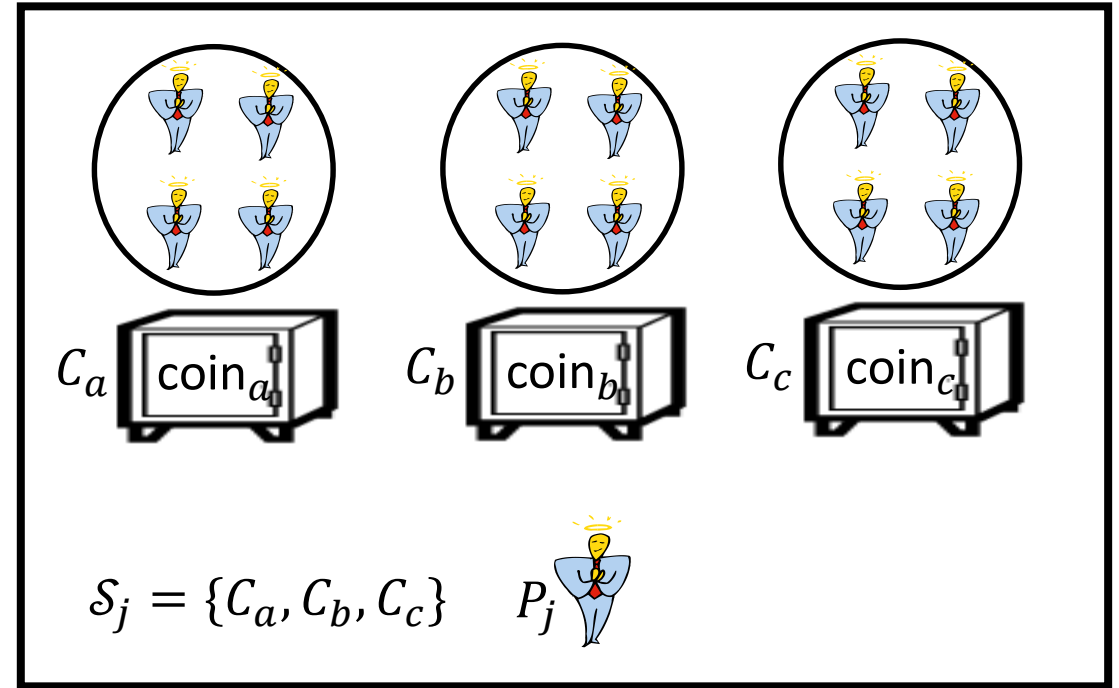
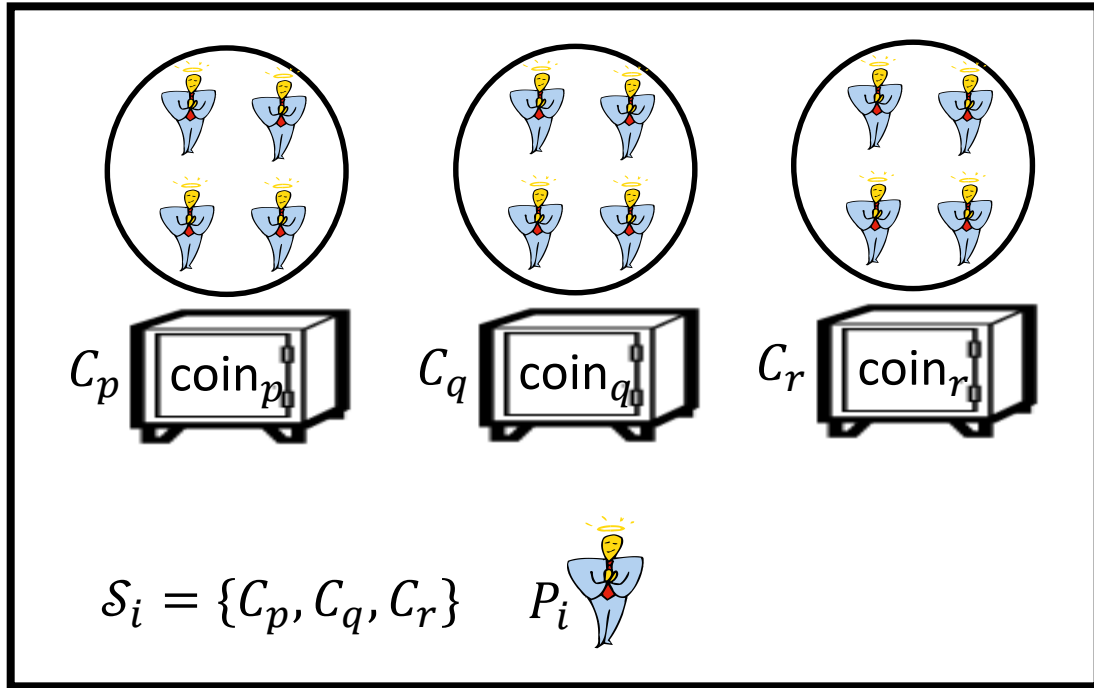


$$\text{coin}_j \stackrel{\text{def}}{=} \sum_{P_i \in C_j} s_{ij}$$

- Each P_j **publicly announces** a set C_j of $(t + 1)$ dealers from its dealer list, after terminating their Sh_{ij} instances
 - Parties **verify** if C_j is a valid list by themselves waiting to terminate those Sh_{ij} instances

Instantiating CC Using AVSS (Sh, Rec)

- Each P_i **interacts** and maintains a list of **valid unknown coin values** \mathcal{S}_i , such that:



- Eventually each $|\mathcal{S}_i| \geq n - t$
- There exists a **common (unknown) subset** M of size at least $n/3$ across all \mathcal{S}_i sets

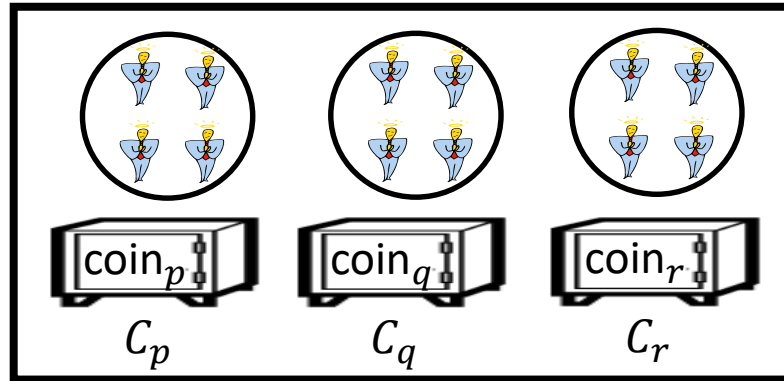
$$|\mathcal{S}_1 \cap \mathcal{S}_2 \cdots \cap \mathcal{S}_n| \geq n/3$$

- The above property is the crux to maintain **common-ness property** across final outputs

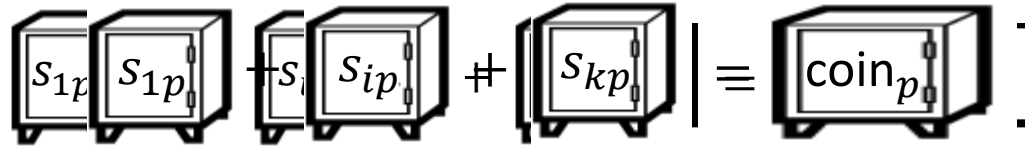
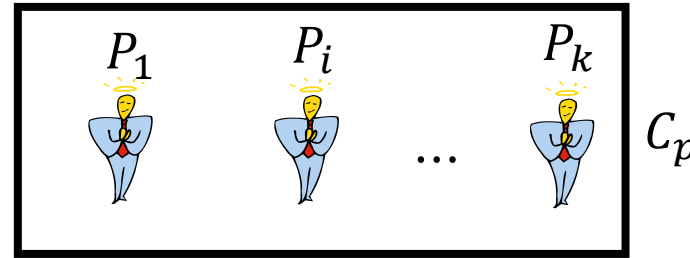
Instantiating CC Using AVSS (Sh, Rec)

□ Once \mathcal{S}_i is of size $n - t$, party P_i publicly announces the same and **parties reconstruct the coins** in \mathcal{S}_i

❖ The required **AVSS-Rec instances** are invoked



$\mathcal{S}_i = \{C_p, C_q, C_r\}$ P_i



$$\text{coin}_p \stackrel{\text{def}}{=} \sum_{P_i \in C_j} S_{ip}$$



$$\text{val}_p \stackrel{\text{def}}{=} (\text{coin}_p \bmod u)$$

❖ Each reconstructed coin-value is **reduced modulo** $u \stackrel{\text{def}}{=} 0.87n$ --- Each $\text{val}_p \in_r [0, \dots, u - 1]$

□ Deciding **final output bit** b_i for party P_i :

$$b_i = \begin{cases} 0, & \text{if some } \text{val}_p = 0 \text{ in set } \mathcal{S}_i \\ 1, & \text{otherwise} \end{cases}$$

□ Probability that all honest parties output $b = 0$

❖ **Favorable event:** ~~some~~ $\text{val}_p \neq 0$ in the **common subset** M

$$\left(1 - \left(1 - \frac{1}{u}\right)^n\right) \frac{1}{u} \geq 0.25$$

Roadmap

□ **Part I** : Byzantine agreement

- ❖ Problem definition and practical applications
- ❖ Known results



□ **Part II** : Randomized Byzantine agreement

- ❖ Framework of Rabin and BenOr
- ❖ Instantiating common-coin using verifiable secret-sharing (VSS)



Conclusion

- We discussed about asynchronous Byzantine agreement (ABA)
 - ❖ A fundamental problem in secure distributed computing

- Plenty of **challenging** open problems
 - ❖ Stay tuned for my second talk in the workshop

Advertisement : NPTEL MOOC on Cryptography

❑ 12-week (32 hours) online course titled “Foundations of Cryptography”

❖ Jan – May 2020 session

❖ Provision for a **certificate from IISc/NPTEL**



Week 1: Historical Ciphers, Perfect Security and Limitations

Week 2: Computational Security, Semantic Security and Pseudorandom Generators (PRGs)

Week 3: Stream Ciphers, Provably-secure Instantiation of PRG, Practical Instantiation of PRG, CPA-security and Pseudo-random Functions (PRFs)

Week 4: CPA-Secure Ciphers, Modes of Operations, Theoretical and practical constructions of Block Ciphers

Week 5: DES, AES and MAC

Week 6: Information-theoretic Secure MAC, Cryptographic Hash Functions, Ideal-Cipher Model, Davies-Meyer construction and Merkle-Damgård Paradigm

Week 7: Birthday Attacks, Applications of Hash Functions, Random Oracle Model and Authenticated Encryption

Week 8: Generic Constructions of Authenticated Encryption Schemes, Key-exchange Problem, One-way Trapdoor Functions and Cyclic Groups

Week 9: Discrete-Logarithm Problem, Computational Diffie-Hellman Problem, Decisional Diffie-Hellman Problem, Elliptic-Curve Based Cryptography and Public-Key Encryption

Week 10: El Gamal Encryption Scheme, RSA Assumption, RSA Public-key Cryptosystem, KEM-DEM Paradigm

Week 11: CCA-secure Public-key Hybrid Ciphers Based on Diffie-Hellman Problems and RSA-assumption, Digital Signatures

Week 12: Schnorr Signature, Overview of TLS/SSL, Number Theory, Interactive Protocols and Farewell

