Network-Agnostic Security Comes (Almost) For Free in DKG & MPC

Crypto 2023, Santa Barbara, USA

Renas Bacho, CISPA Helmholtz Center for Information Security & Saarland University

Daniel Collins, EPFL

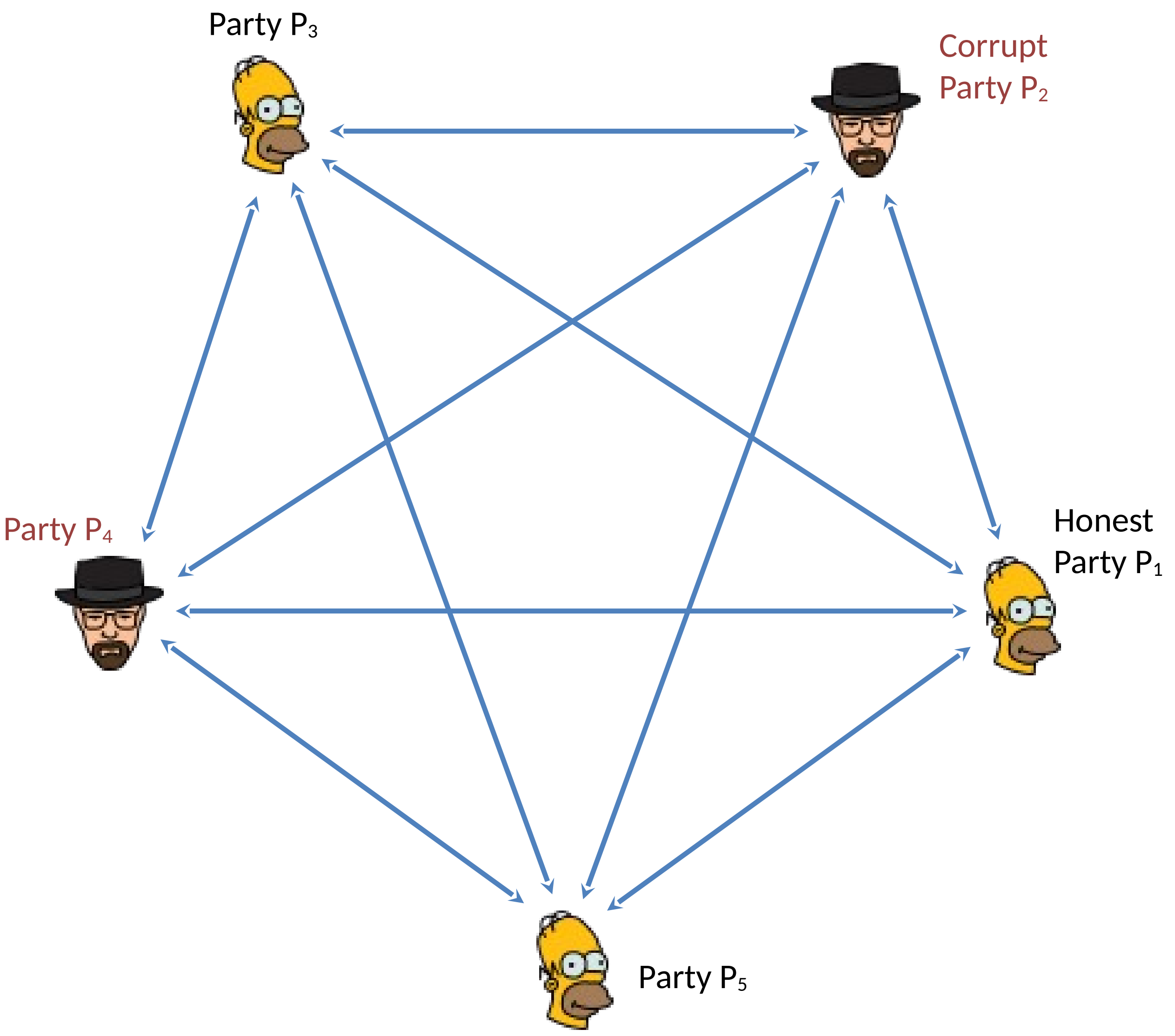
Chen-Da Liu-Zhang, HSLU & Web3 Foundation

Julian Loss, CISPA Helmholtz Center for Information Security

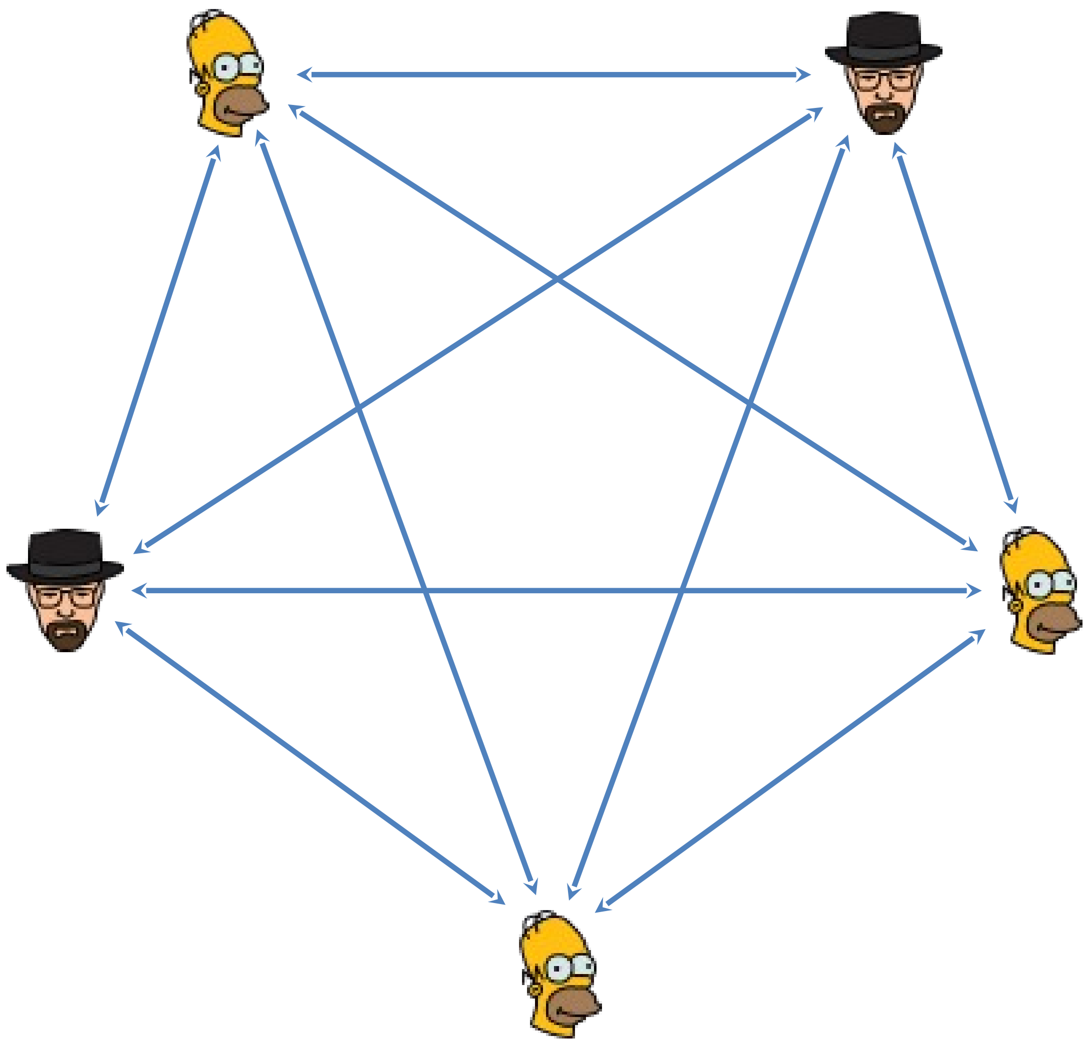
Why Threshold Cryptography?

- Avoid any single point of failure.
- Distribute a task or secret among a set of fault-tolerant parties (or servers).
- Set of n parties $P_1, \ldots, P_n$, up to t of which are malicious, want to perform a task:
  - Threshold signatures
  - Threshold encryption
  - Distributed coin flipping

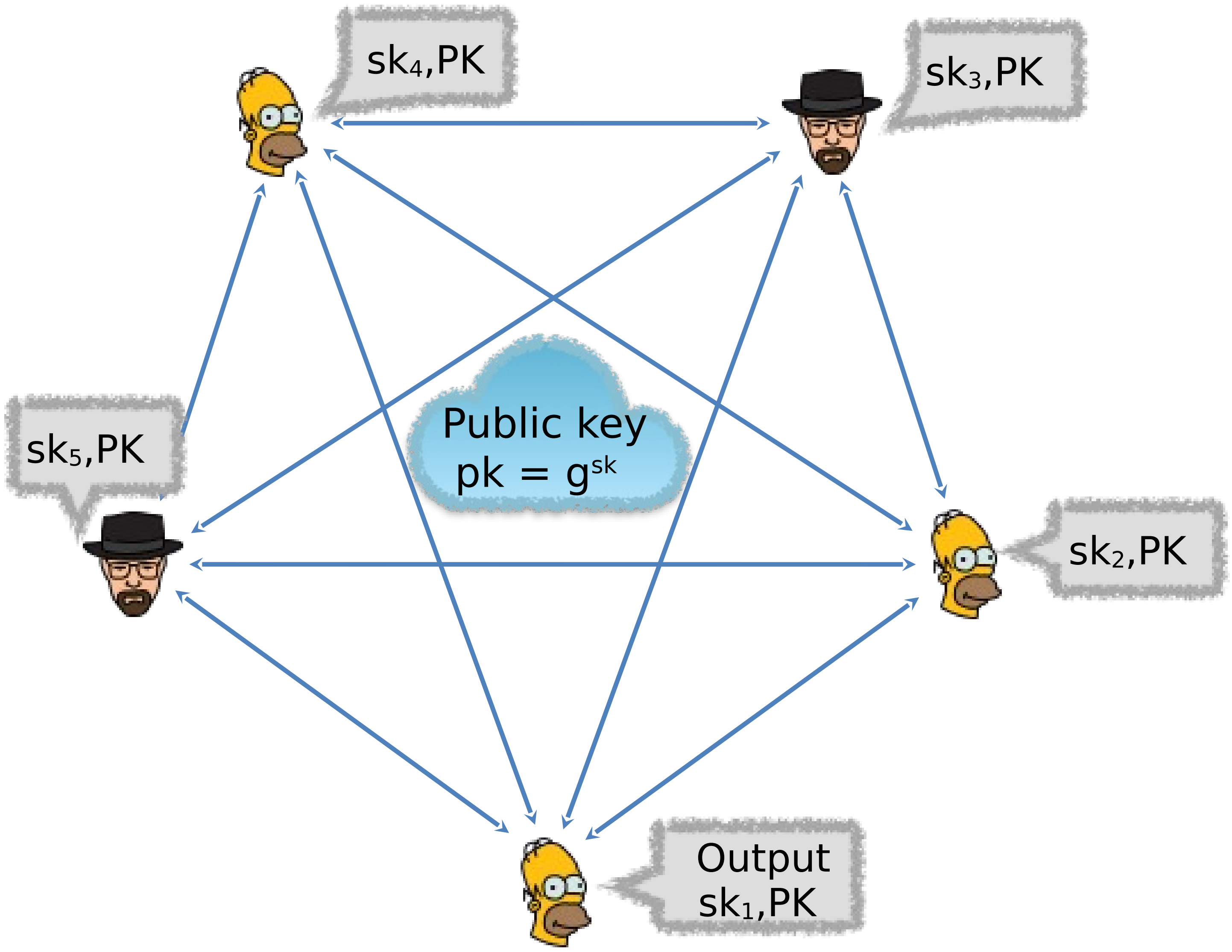
Distributed Key Generation!



Distributed Key Generation (DKG) Protocol



Distributed Key Generation (DKG) Protocol

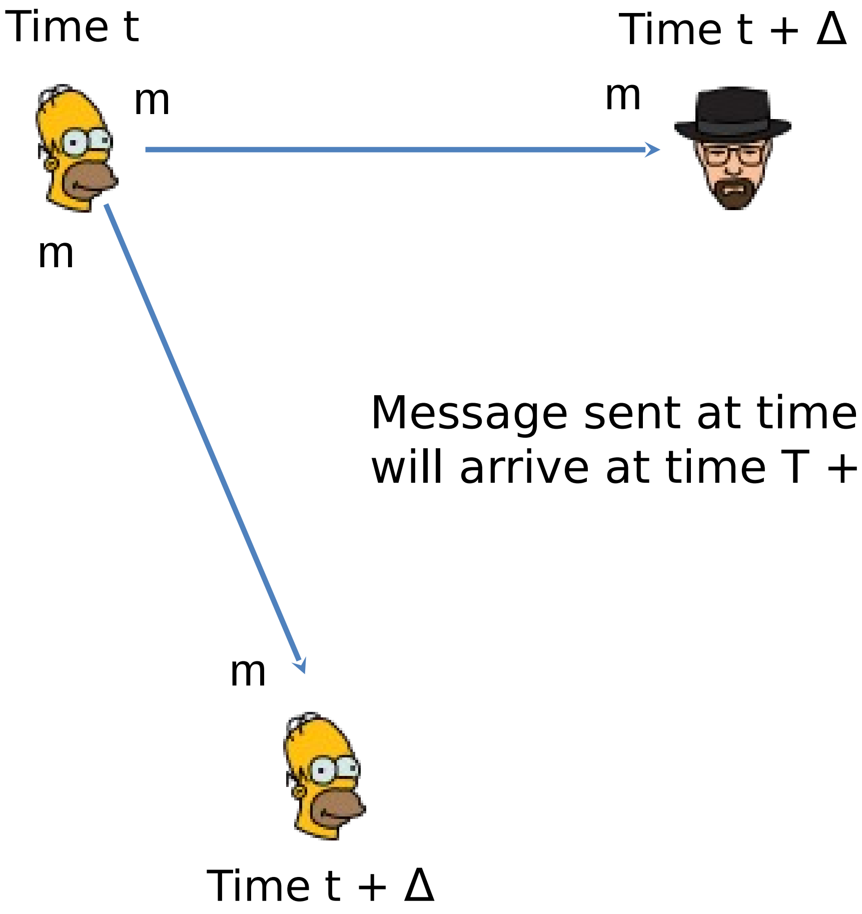


Properties:

- **Consistency:** All honest parties output the same public key and the same vector of public key shares PK = $(pk_1, \ldots, pk_n)$.
- **Correctness:** There is a polynomial f in $Z_p[X]$ of degree t s.t. $sk_i = f(i)$ and $pk_i = g^{sk_i}$. In addition, $pk = g^{f(0)}$.
- **Secrecy:** No information on x can be learned by the adversary except of what is implied by the public key.
- **Uniformity:** The public key output is uniformly distributed.

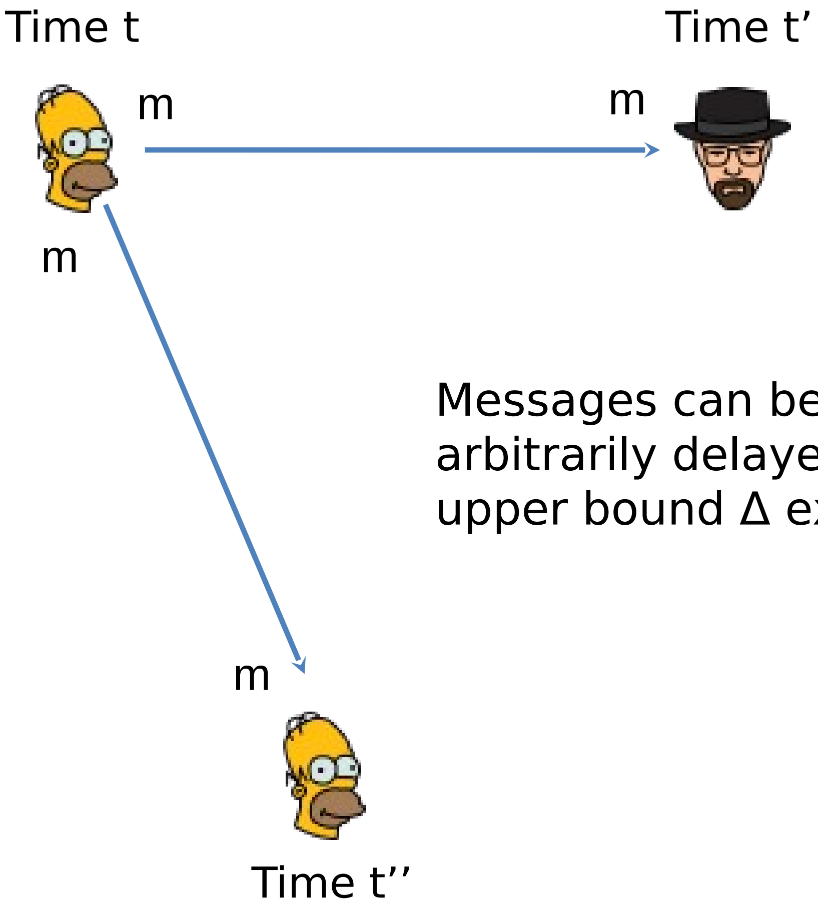
Synchronous and Asynchronous Networks

Synchronous Network



Message sent at time T
will arrive at time T + Δ

Asynchronous Network



Messages can be
arbitrarily delayed. No
upper bound Δ exists.

Authenticated
point-to-point channels

Network-Agnostic Protocols

- Network is either synchronous *or* asynchronous throughout execution.

→ **Problem**: Parties do not know which world they are in.

- Synchronous protocols: tolerate $t_s < n/2$ corruptions, but are insecure in asynchrony!

- Asynchronous protocols: tolerates $t_a < n/3$ corruptions, but only $t_s < n/3$ in synchrony!

- In this work, we consider $t_a + 2t_s < n$ (which we show is necessary and sufficient for DKG)

Network-Agnostic Protocols are More Versatile

- In this work, we consider $t_a + 2t_s < n$ (necessary and sufficient!)

- Consider t_a and t_s such that $t_a < n/3 \leq t_s < n/2$.
- Let f(t) be the probability that there are more than t faults.
- Let p be the probability that the network delay exceeds Δ.

Network-Agnostic Protocols are More Versatile

- In this work, we consider $t_a + 2t_s < n$ (necessary and sufficient!)

- Consider t_a and t_s such that $t_a < n/3 \leq t_s < n/2$.
- Let f(t) be the probability that there are more than t faults.
- Let p be the probability that the network delay exceeds Δ.

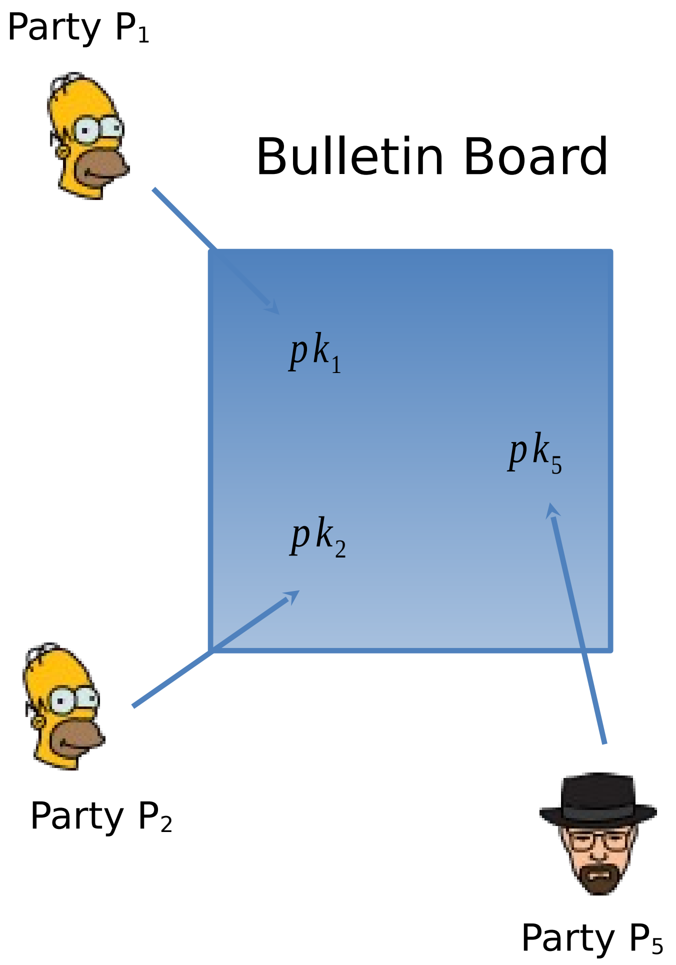
- Suppose that $p = f(t_a) = 1/10$, $f((n-1)/3) = 1/20$ and $f(t_s) = 0$. Then:
  - A *synchronous* protocol fails with probability 1/10;
  - An *asynchronous* protocol fails with probability 1/10;
  - A *network-agnostic* protocol fails with probability $f(t_s) + p*f(t_a) = 1/100$.

Motivation and Model

- Network is either synchronous *<u>or</u>* asynchronous throughout.

→ **Problem**: Parties do not know which world they are in.

- *Plain PKI* model:
  - At setup, parties each upload a public key.
  - No trusted setup (except possibly a CRS).



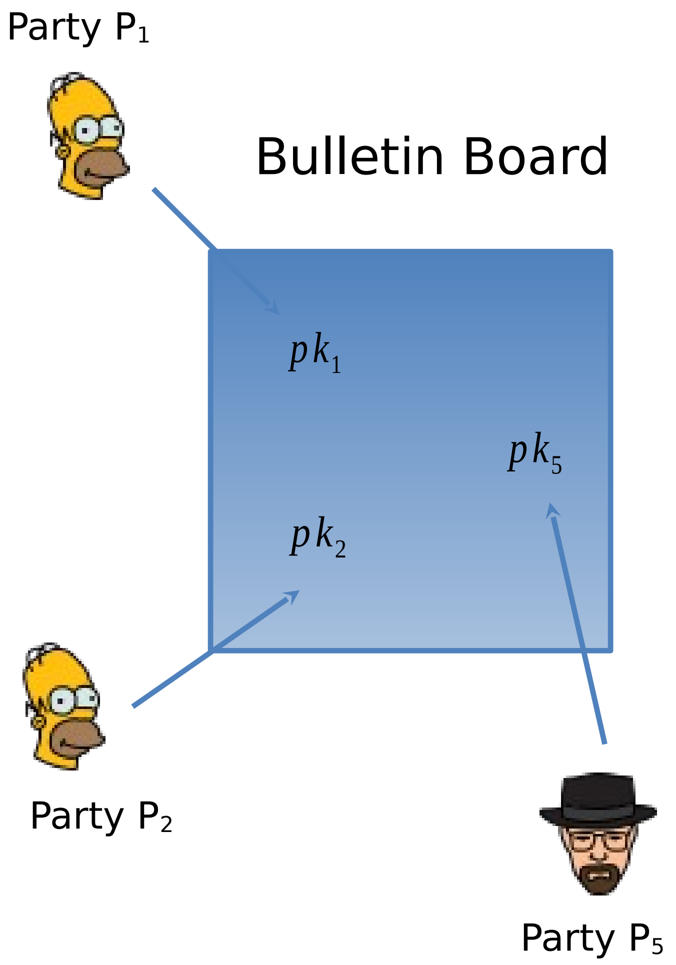
Motivation and Model

- Network is either synchronous *or* asynchronous throughout.

  → **Problem**: Parties do not know which world they are in.

- *Plain PKI* model:
  - At setup, parties each upload a public key.
  - No trusted setup (except possibly a CRS).

- *Static security*: adversary corrupts parties before the protocol begins.
- Can corrupt up to t_s parties in synchrony and t_a parties in asynchrony.
- A protocol is t_s/t_a-secure in synchrony/asynchrony if it satisfies its properties in synchrony/asynchrony respectively.



Our Main Result

- **Theorem (Network-agnostic DKG):**
  - Let $t_s < n/2$ and $t_a < n/3$ be such that $t_a + 2t_s < n$. Then, in the plain PKI setting there is a DKG that is t_s-secure in synchrony and t_a-secure in asynchrony with $O(\lambda n^3)$ communication complexity.
  - Application: We get efficient network-agnostic MPC without trusted setup!

DKG Protocol	Network	Adv.	Comm.	Rounds	Setup
Shrestha et al. [SBKN21]	sync	Static	$O(\lambda n^3)$	$O(n)/O(1)$	PKI, RO, CRS
Das et al. [DYX+22]	async	Static	$O(\lambda n^3)$	$O(\log n)$	PKI, RO
Abraham et al. [AJM+21]	async	Static	$\tilde{O}(\lambda n^3)$	$O(1)$	PKI, CRS
Zhang et al. [ZDL+22]	async	Static	$O(\lambda n^4)$	$O(1)$	-
Abraham et al. [AJM+22]	async	Adaptive	$\tilde{O}(\lambda n^3)$	$O(1)$	PKI, CRS
This work (Section 5)	fallback	Static	$O(\lambda n^3)$	$O(n)$	PKI, RO, CRS

Along the way: Efficient Synchronous Broadcast

Protocol	**Resil.**	**Adaptive**	**Comm.**	**Rounds**	**Len.**	**Setup**
Abraham et al. [ACD+19]	$1/2-\epsilon$	Yes	$\tilde{O}(\lambda n + \ell n)$	$O(1)$	MV	Trusted
Momose-Ren [MR21b]	$1/2$	Yes	$O(\lambda n^2)$	$O(n)$	Bin.	Trusted
Chan et al. [CPS20]	$1-\epsilon$	Yes	$O(\lambda^2 n^2)$	$O(\lambda)$	Bin.	Trusted
Dolev-Strong [DS83]	1	Yes	$O(\lambda n^3 + \ell n)$	$O(n)$	MV	Plain
Momose-Ren [MR21b]	$1/2-\epsilon$	Yes	$O(\lambda n^2)$	$O(n)$	Bin.	Plain
Tsimos et al. [TLP22]	$1-\epsilon$	No	$O(\lambda^2 n^2)$	$O(n)$	Bin.	Plain
Our Protocol	$1-\epsilon$	No	$O(n\ell + \lambda n^2)$	$O(n)$	MV	Plain

Related Work

- Network-agnostic protocols:
  - ‘Generation 1’: feasibility results for Byzantine agreement [BKL19], MPC [BLL20, ACC22], state machine replication [BKL21], approximate agreement [GLW22]…
  - ‘Generation 2’ protocols: more efficient protocols [DHL21, ABKL22, this work]

Related Work

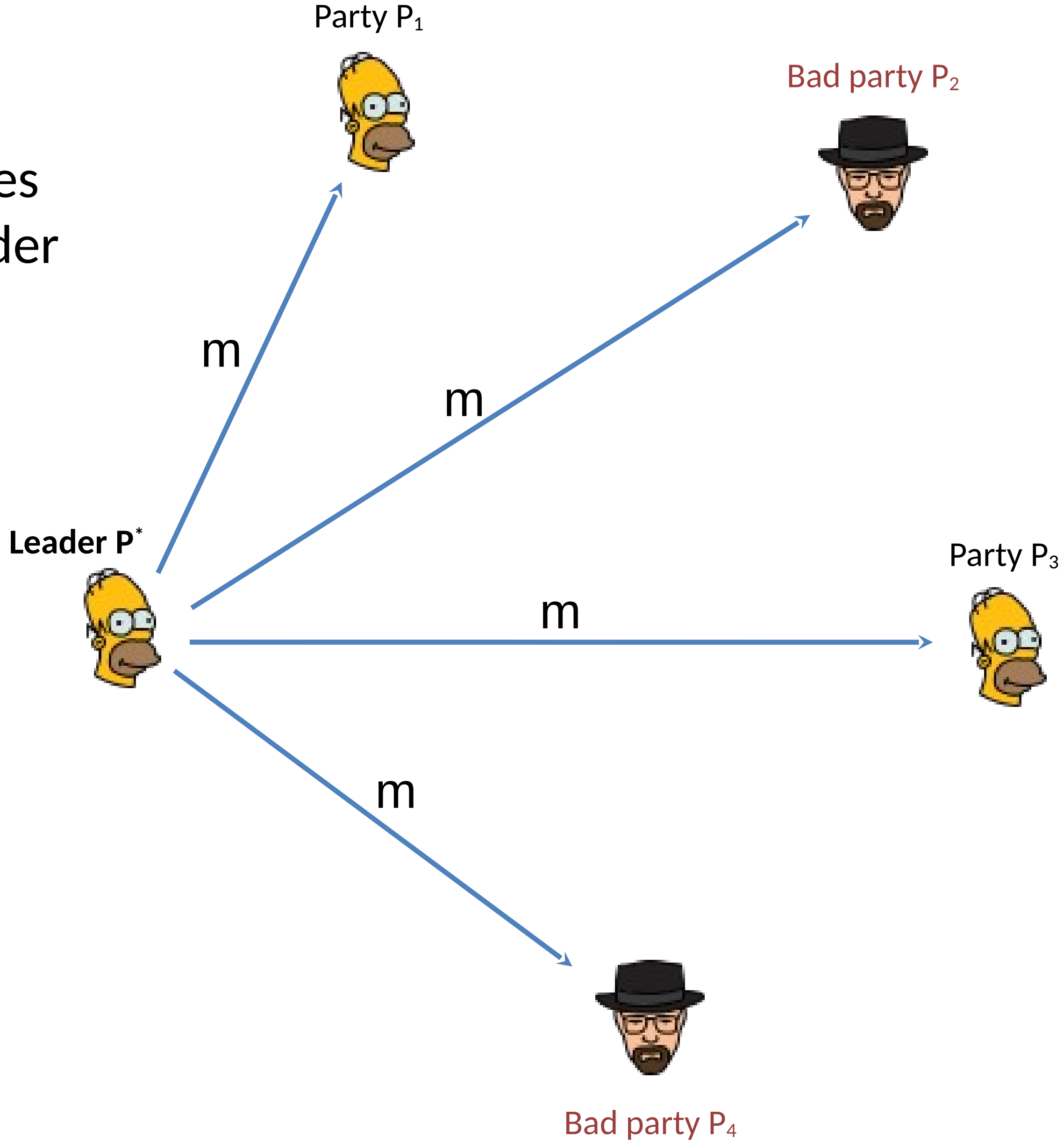
- Network-agnostic protocols:
  - 'Generation 1': feasibility results for Byzantine agreement [BKL19], MPC [BLL20, ACC22], state machine replication [BKL21], approximate agreement [GLW22]…
  - 'Generation 2' protocols: more efficient protocols [DHL21, ABKL22, this work]

- Distributed key generation:
  - Synchrony: classic protocols assume a broadcast channel [GJKR07]; recently got $O(\lambda n^3)$ communication without one [SBKN21]
  - Asynchrony: recent line of work, many now which achieve $O(\lambda n^3)$ communication [DYX+22], [AJM+22], plus one with adaptive security [AJM+23]

Synchronous Broadcast

- System of n parties $P_1, \ldots, P_n$ malicious parties with $t < (1 - \varepsilon)n$ malicious parties and a leader P^*.

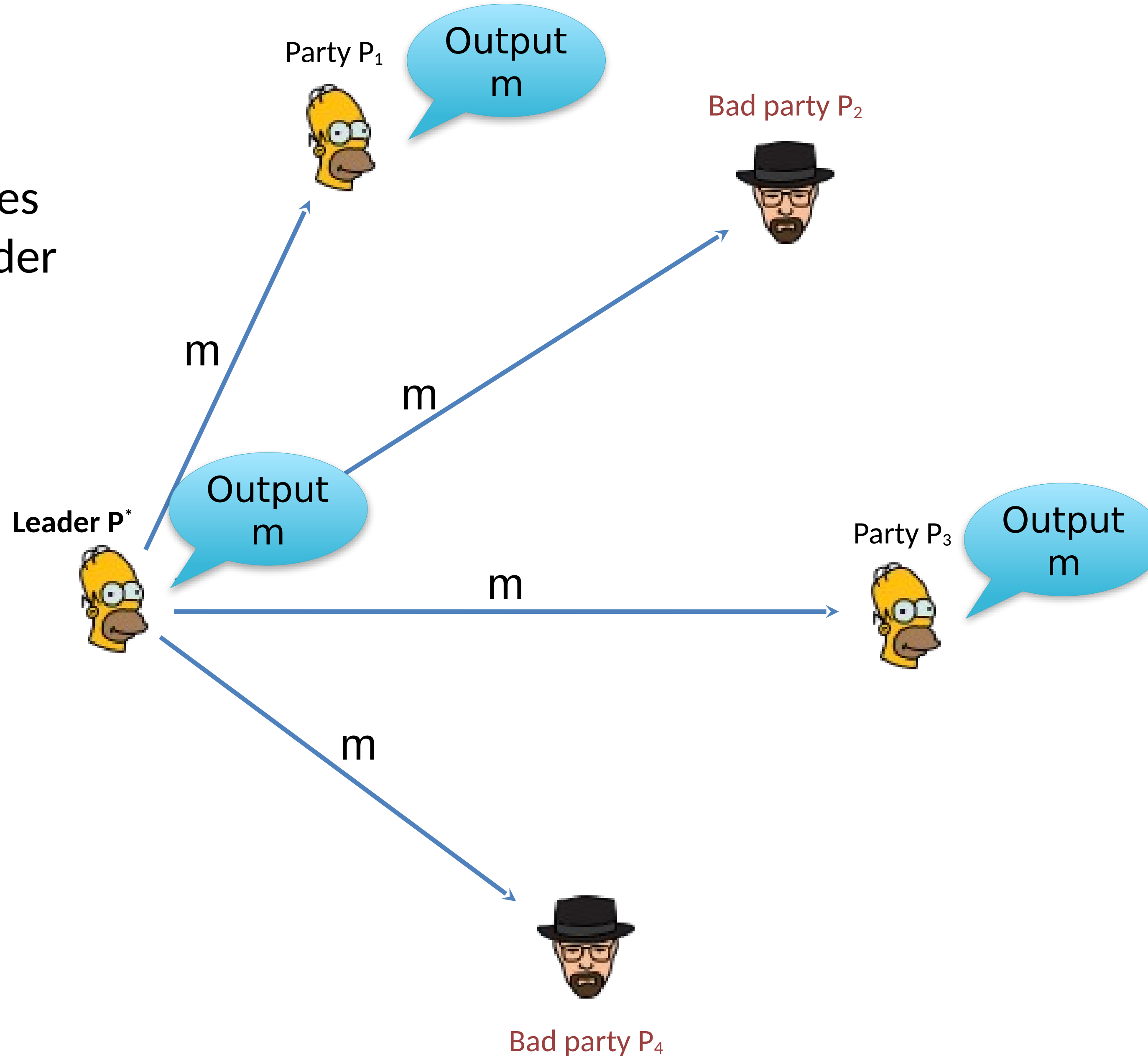
- P^* wants to propagate a message m with:
  - *Validity*: If P^* is honest, all honest parties output m.



Synchronous Broadcast

- System of n parties $P_1, \ldots, P_n$ malicious parties with $t < (1 - \varepsilon)n$ malicious parties and a leader P^*.

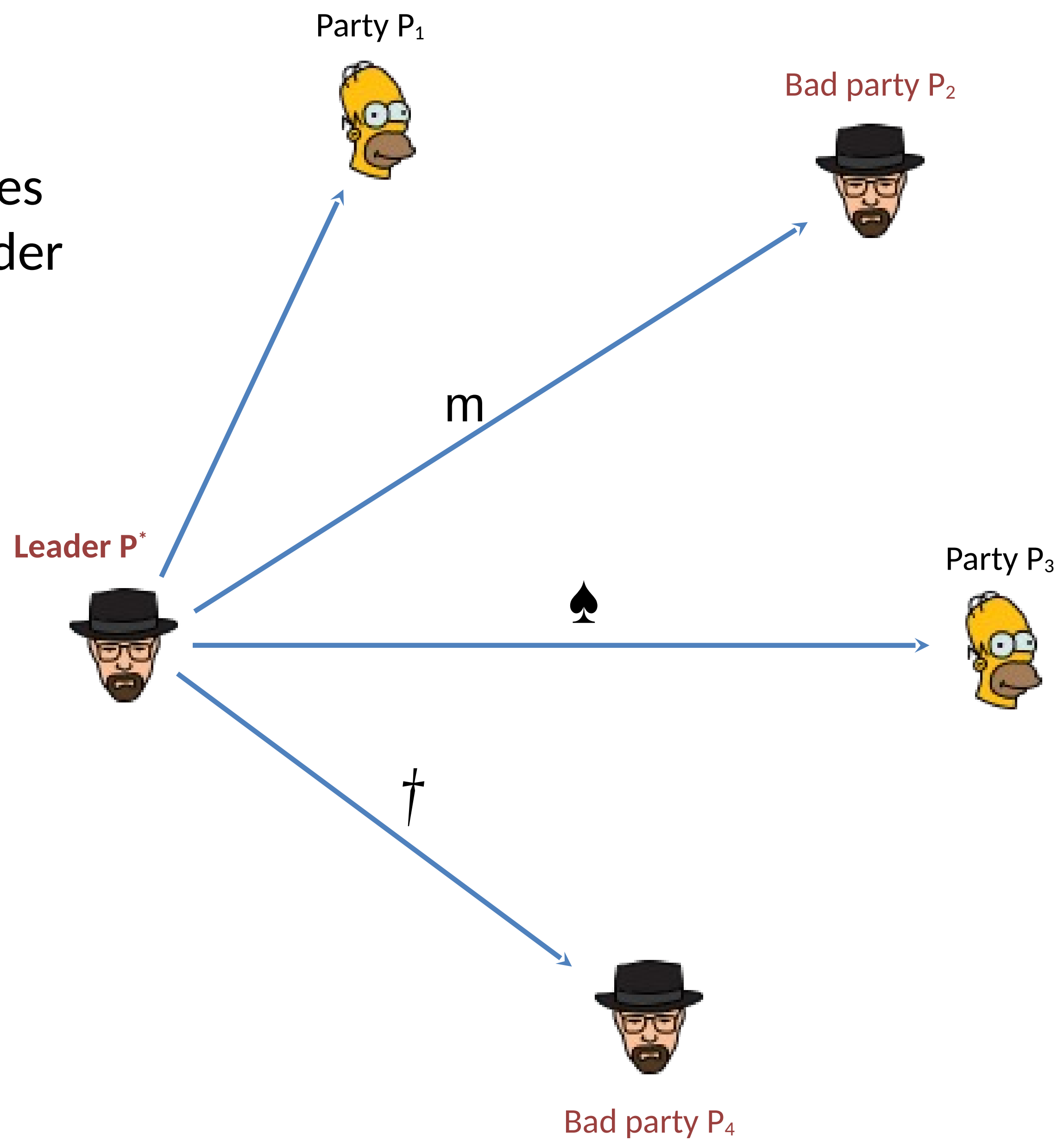
- P^* wants to propagate a message m with:
  - *Validity*: If P^* is honest, all honest parties output m.



Synchronous Broadcast

- System of n parties $P_1, \ldots, P_n$ malicious parties with $t < (1 - \varepsilon)n$ malicious parties and a leader P^*.

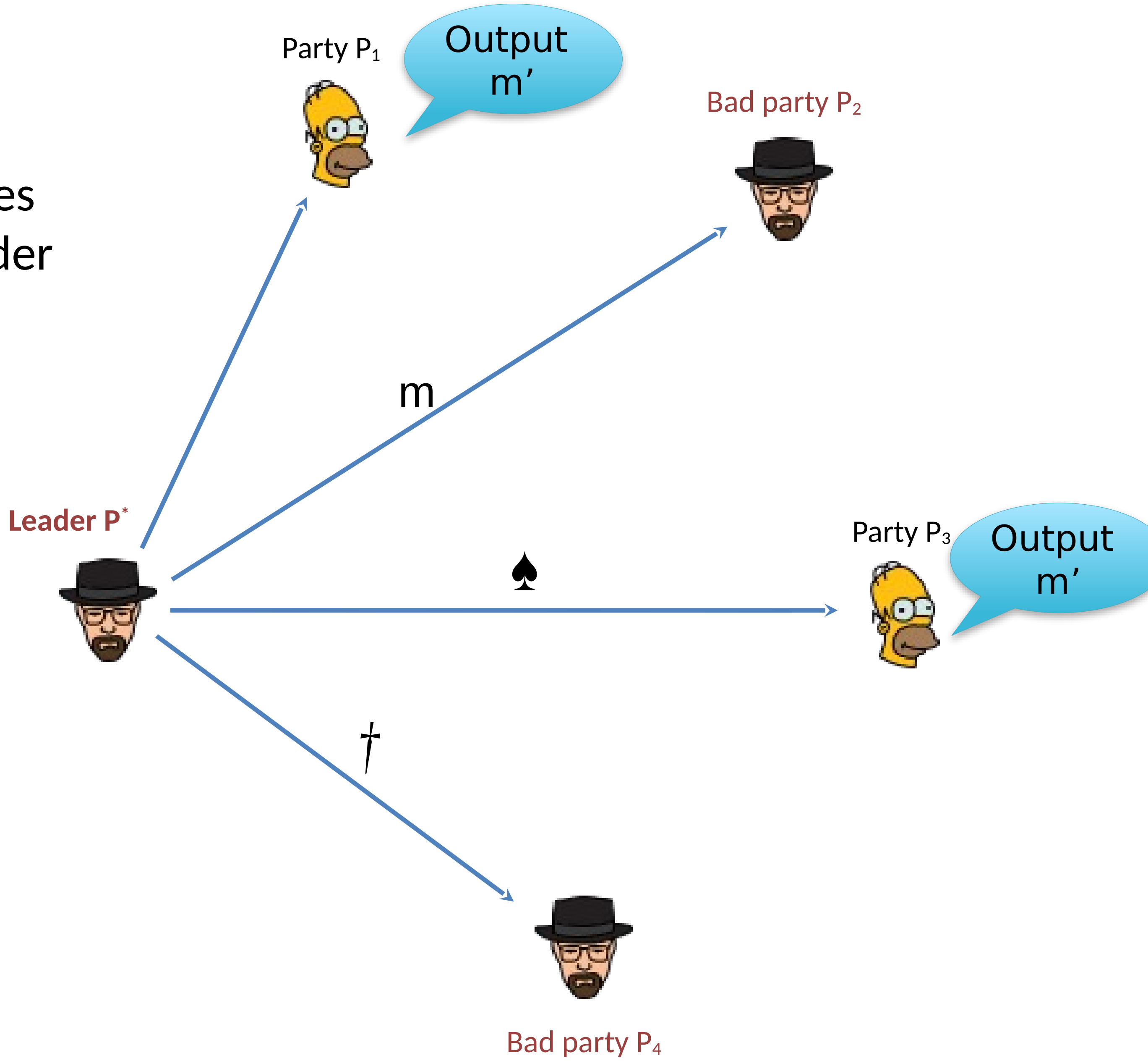
- P^* wants to propagate a message m with:
  - *Validity*: If P^* is honest, all honest parties output m.



Synchronous Broadcast

- System of n parties $P_1, \ldots, P_n$ malicious parties with $t < (1 - \varepsilon)n$ malicious parties and a leader P^*.

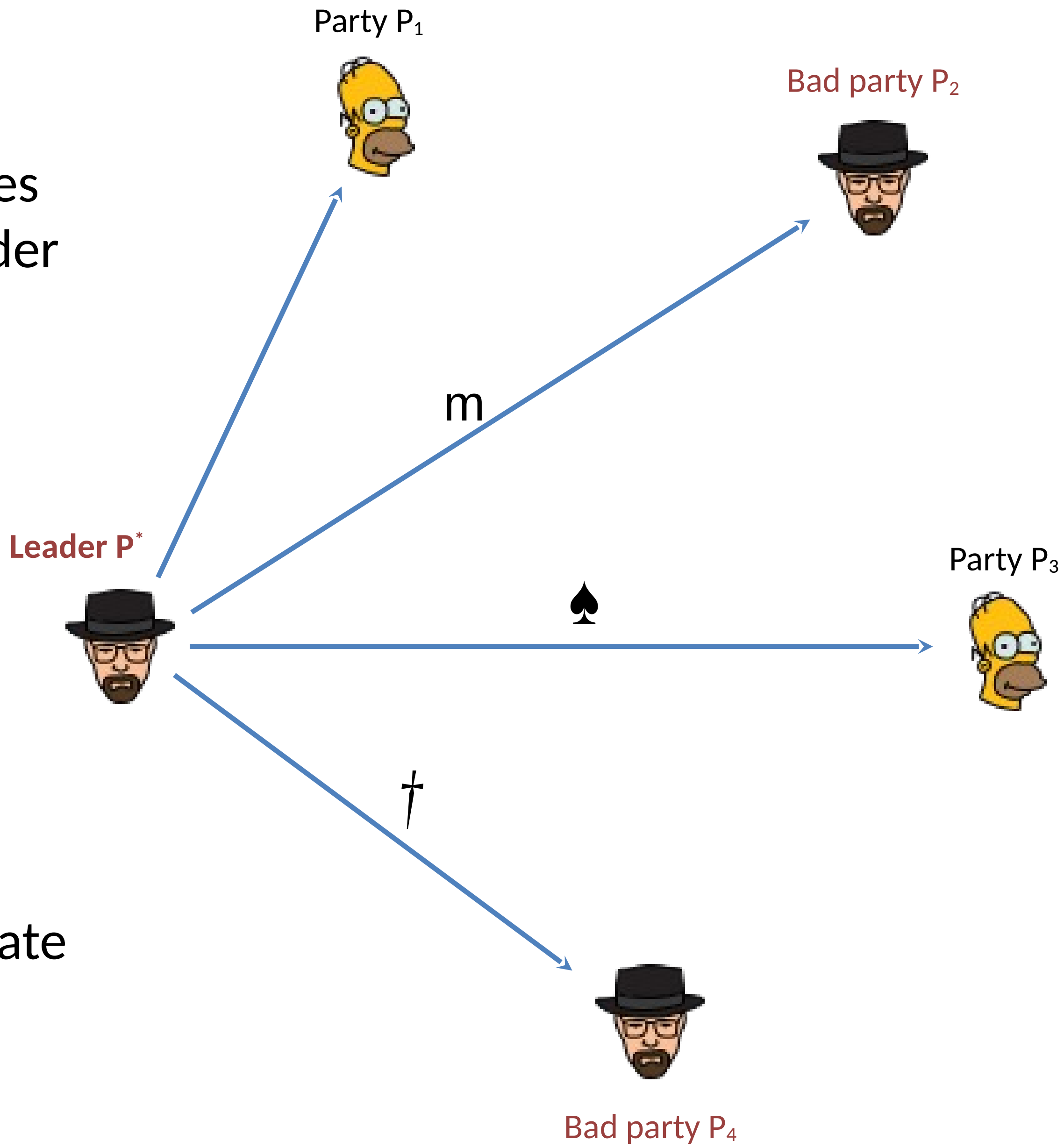
- P^* wants to propagate a message m with:
  - *Validity*: If P^* is honest, all honest parties output m.
  - *Consistency*: All honest parties output the same message m' (possibly ⊥).



Synchronous Broadcast

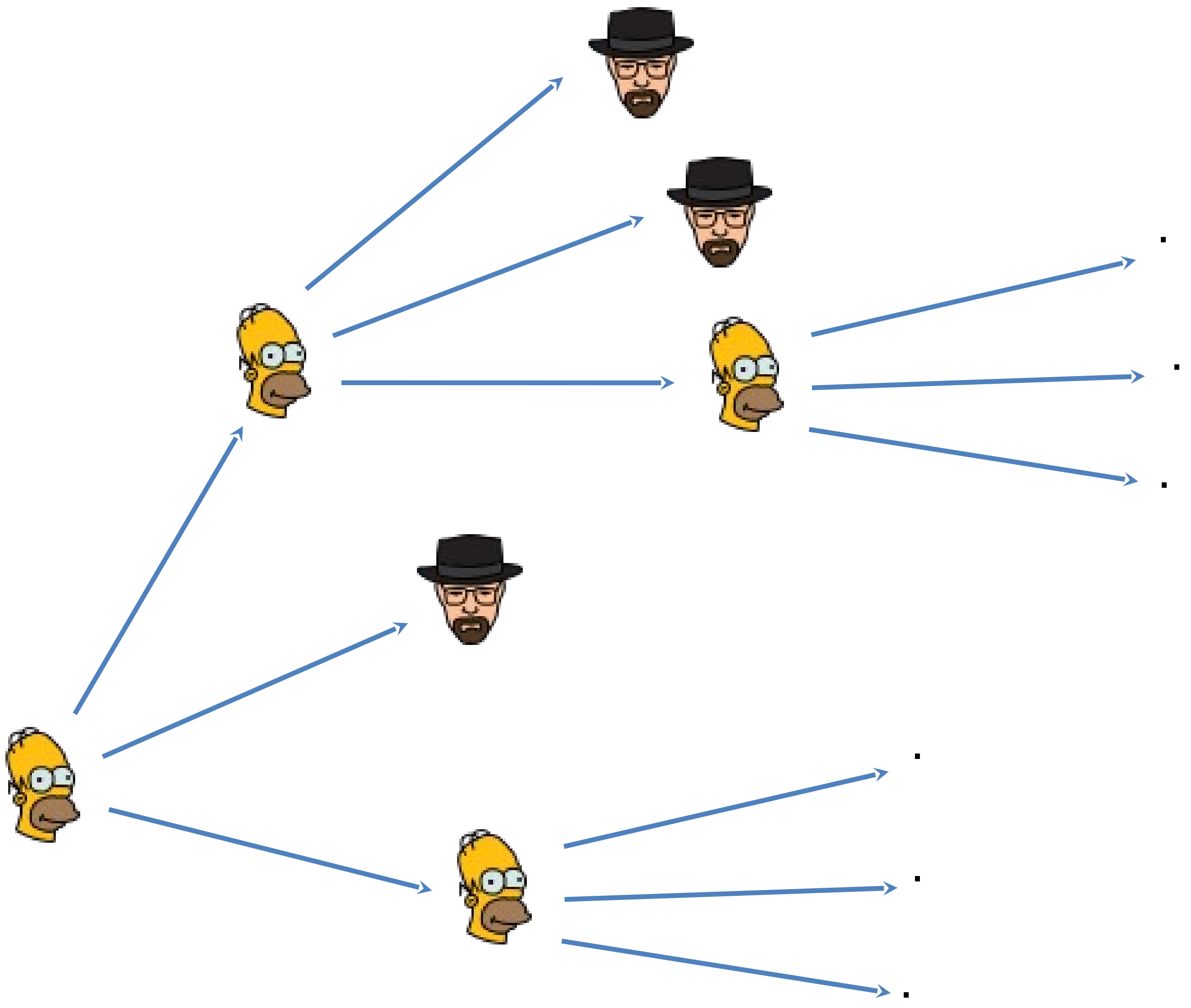
- System of n parties $P_1, \ldots, P_n$ malicious parties with $t < (1 - \varepsilon)n$ malicious parties and a leader P^*.

- P^* wants to propagate a message m with:
  - *Validity*: If P^* is honest, all honest parties output m.
  - *Consistency*: All honest parties output the same message m' (possibly $\perp$).
  - *Termination*: All honest parties terminate with some output.



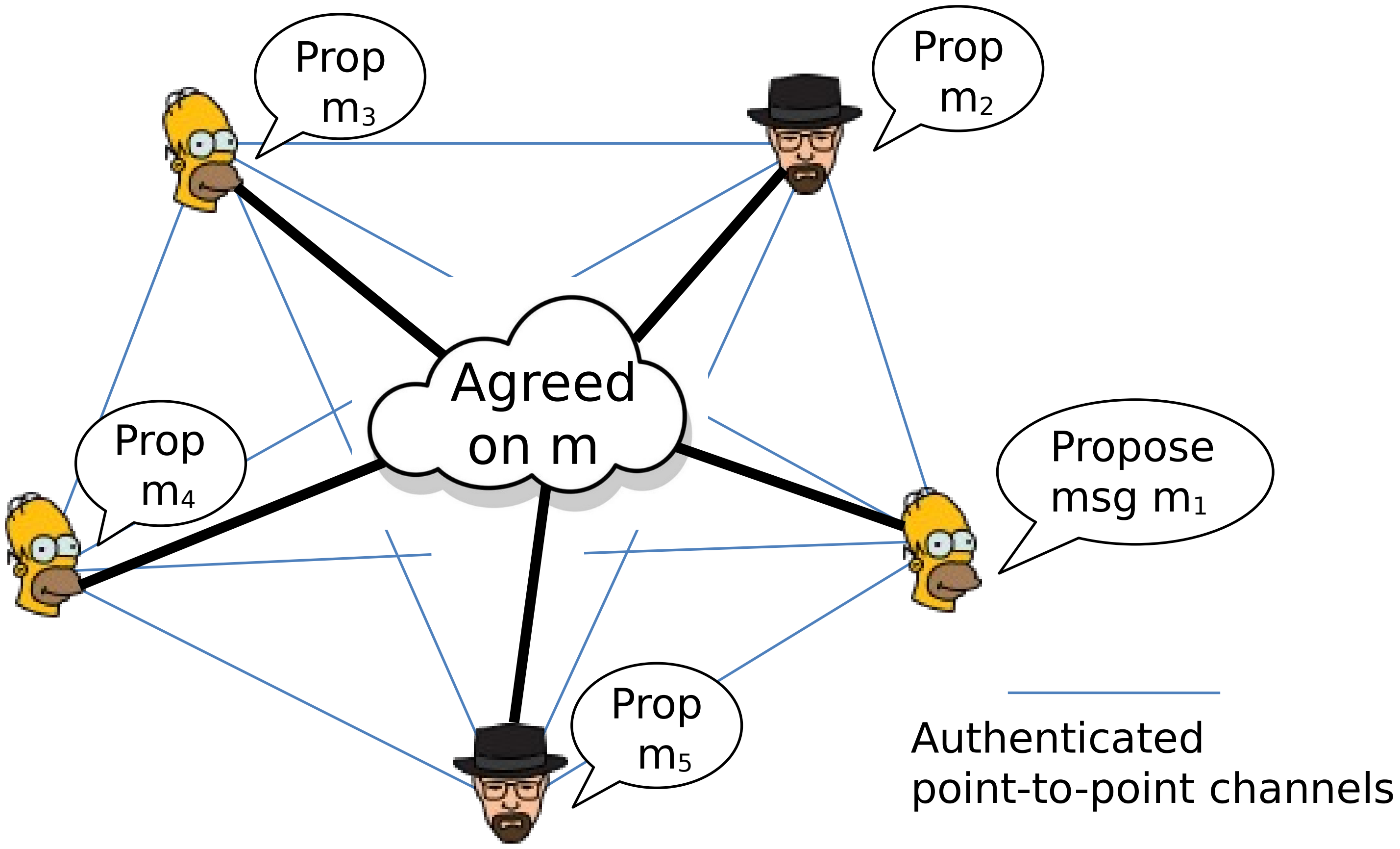
Techniques for Broadcast

- Combination of *gossiping* [TLP22] and *extension protocol* [NRS+20] techniques.
- Idea: Replace signature multicast step in [NRS+20] with gossip.
- **Gossip**: forward the message to O(λ) parties (one is honest with high probability).
- Guarantee: everyone learns the message in log(n) rounds and O(λn) communication.



Intrusion-tolerant Consensus (ITC)

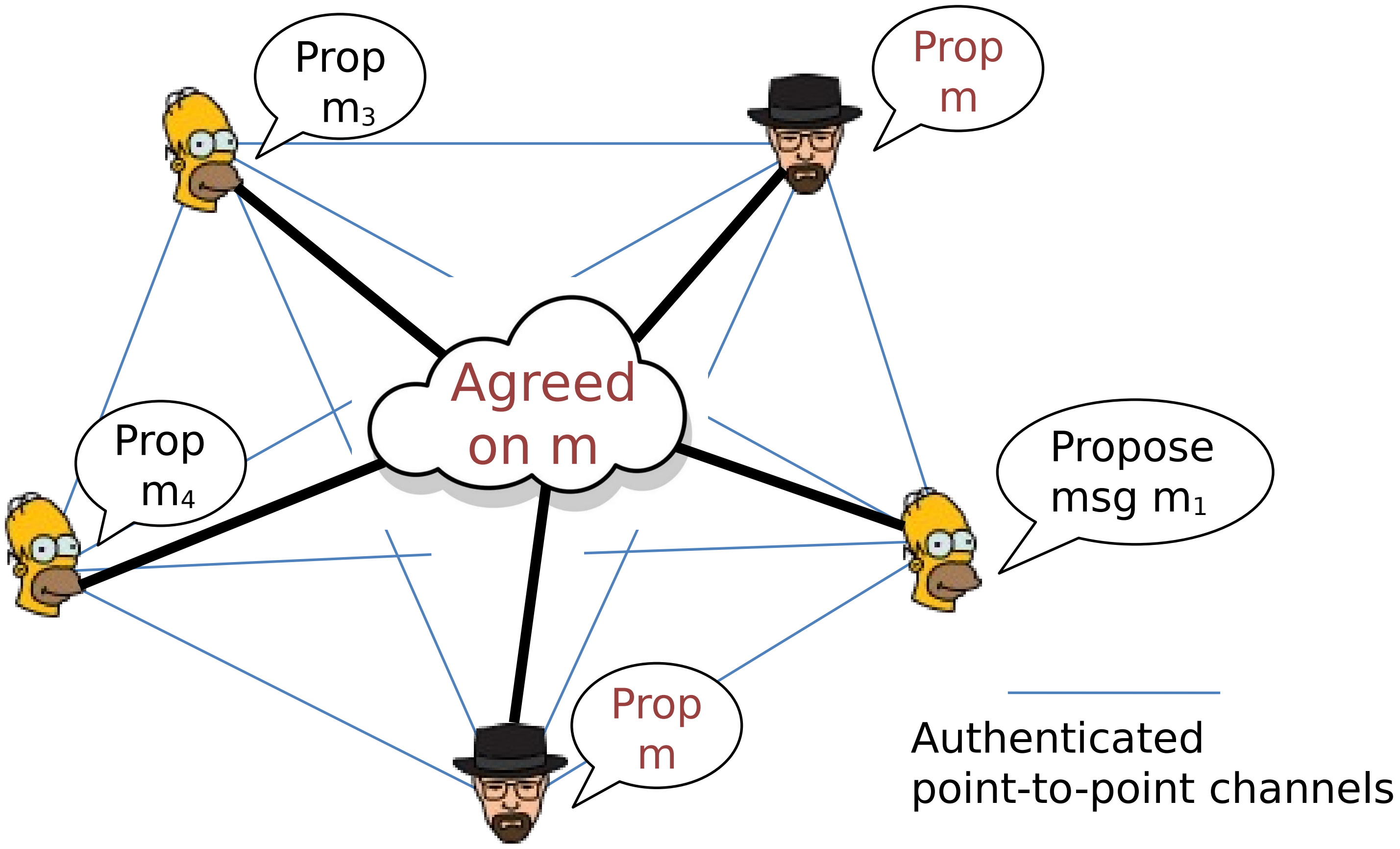
- Consensus: parties propose and *agree* on a message m.



Intrusion-tolerant Consensus (ITC)

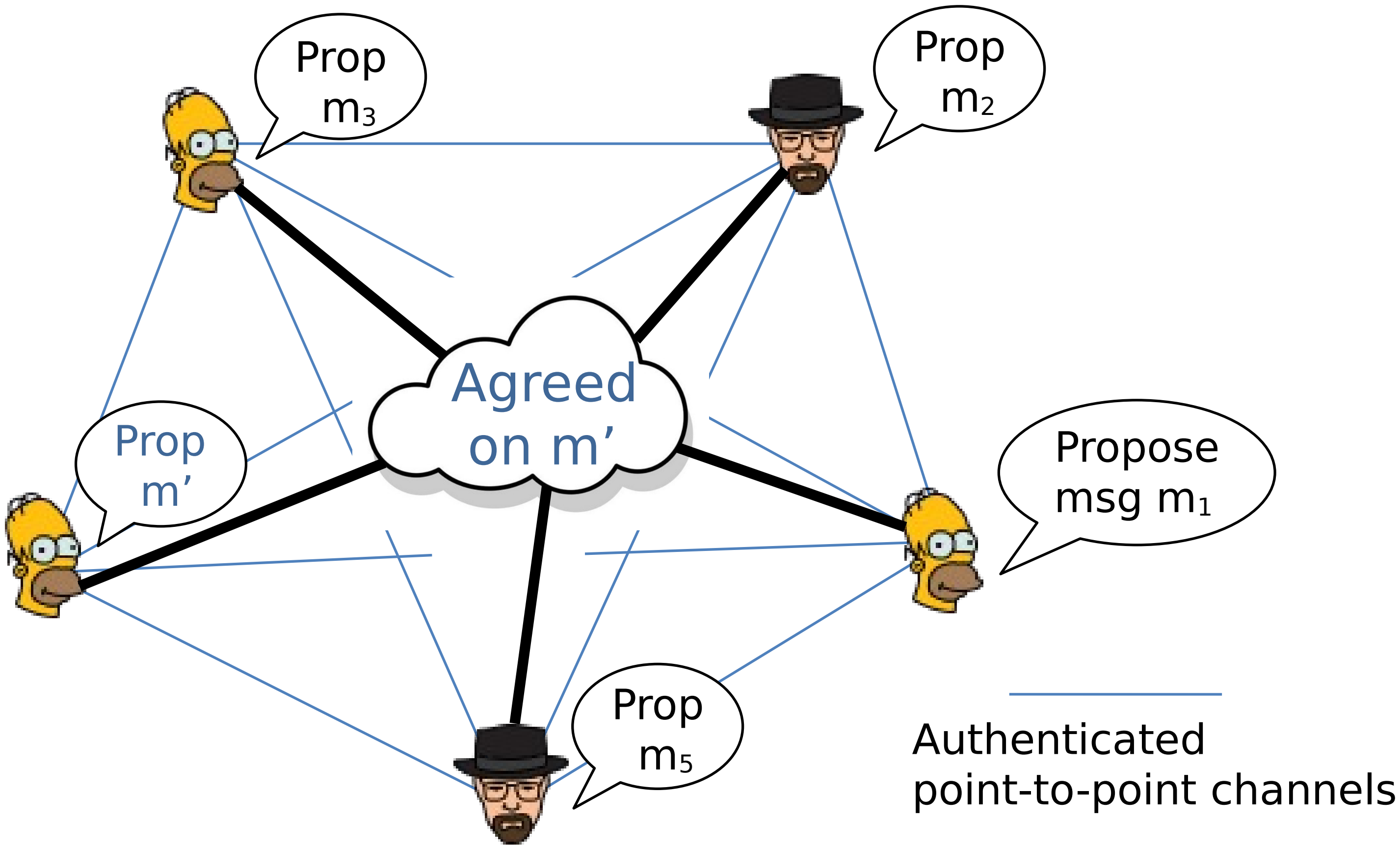
- Consensus: parties propose and *agree* on a message m.

- **Problem**: m could come from a dishonest party.



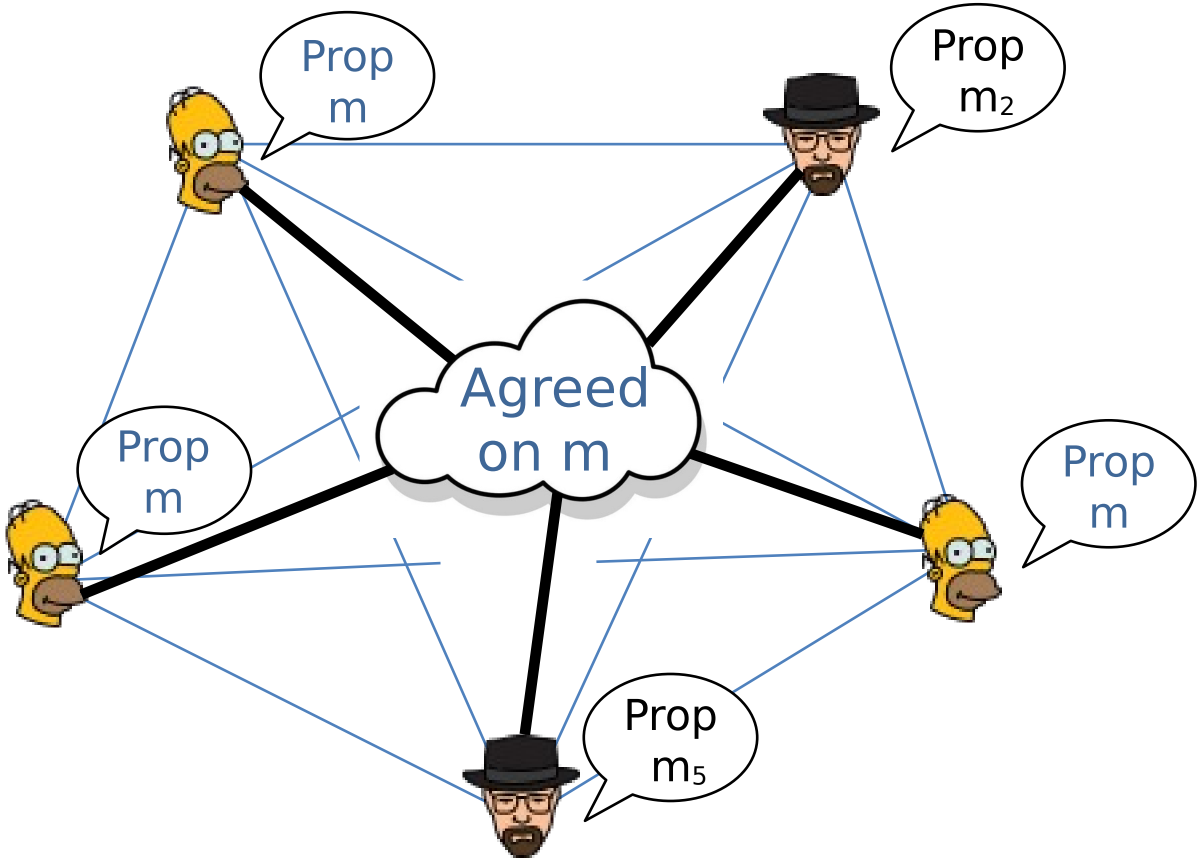
Intrusion-tolerant Consensus (ITC)

- Consensus: parties propose and *agree* on a message m.
- **Problem**: m could come from a dishonest party.
- **Solution**: *intrusion-tolerant* consensus
- Intrusion-tolerance: an honest party can output either an honest party's input or ⊥.



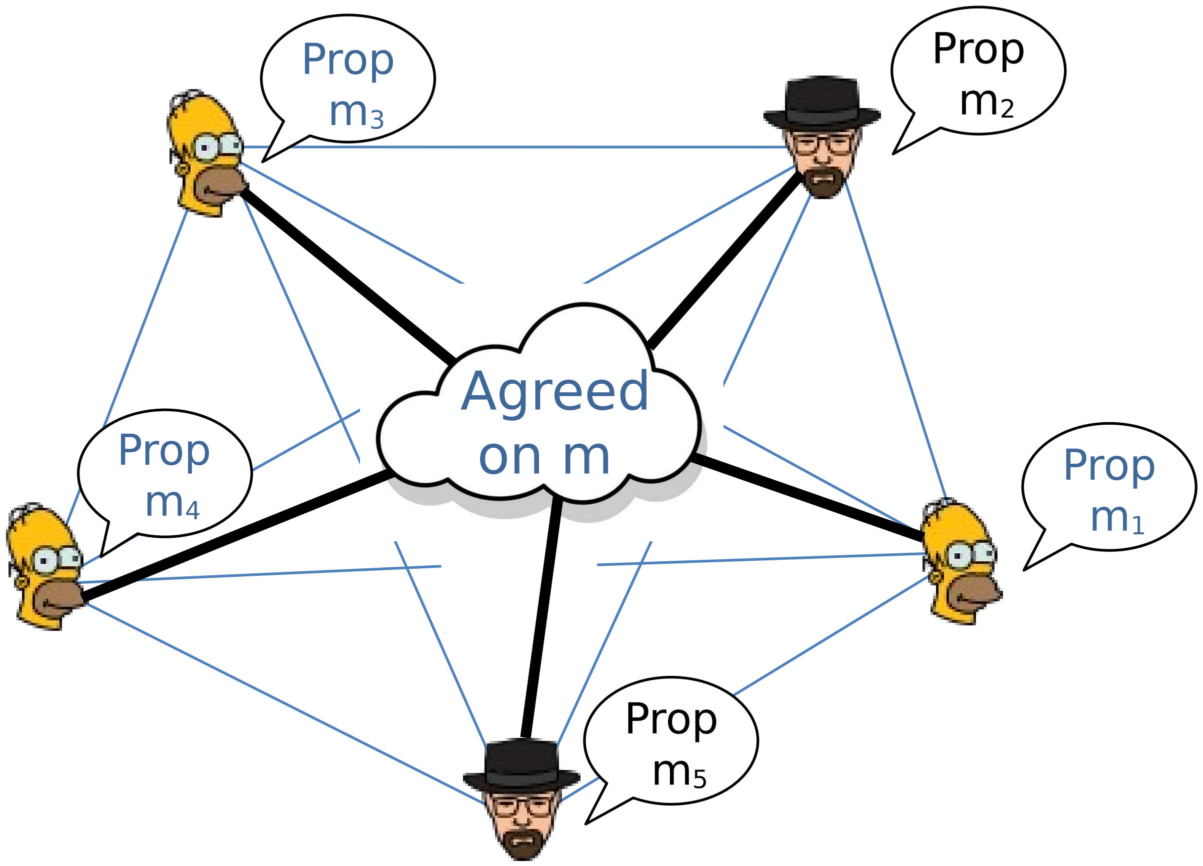
More on Intrusion-Tolerant Consensus

- Validity: if every honest party inputs the same m, they also output m.



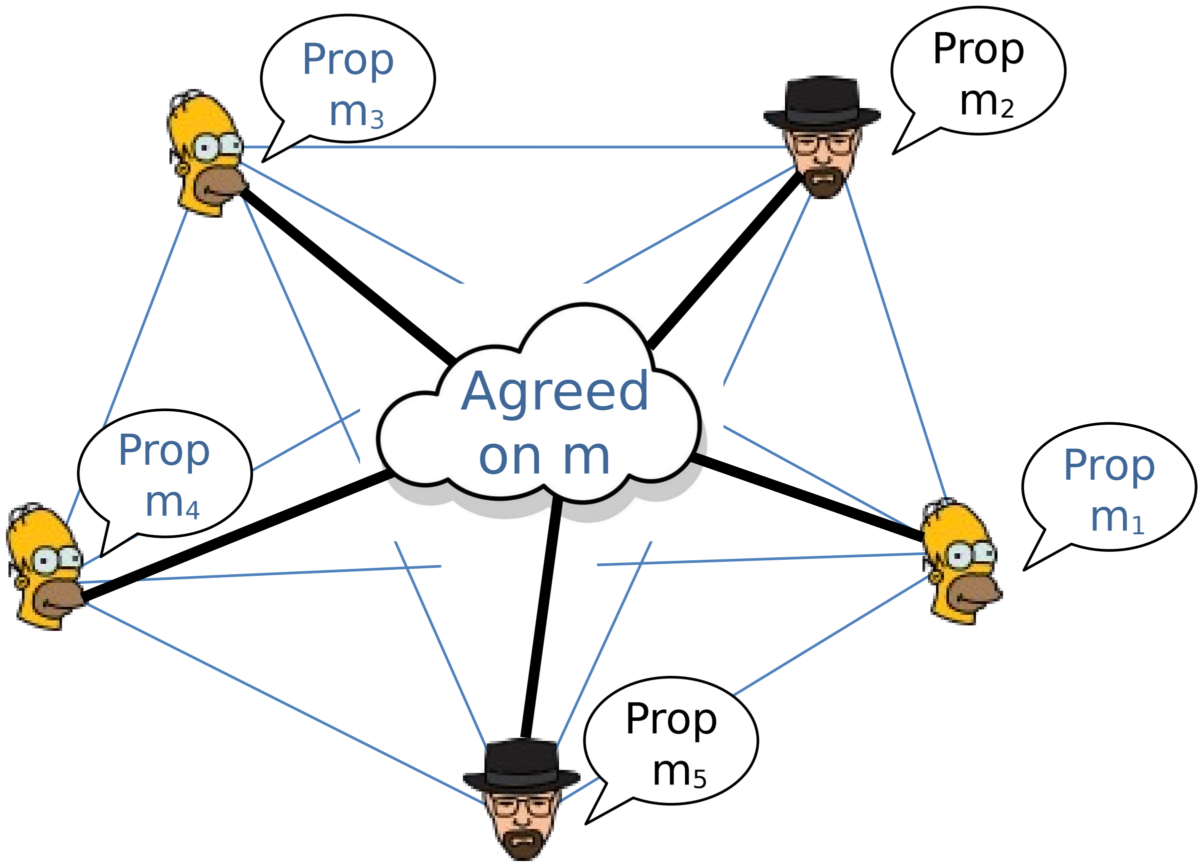
More on Intrusion-Tolerant Consensus

- Validity: if every honest party inputs the same m, they also output m.
- Consistency: every honest party outputs the same message m.



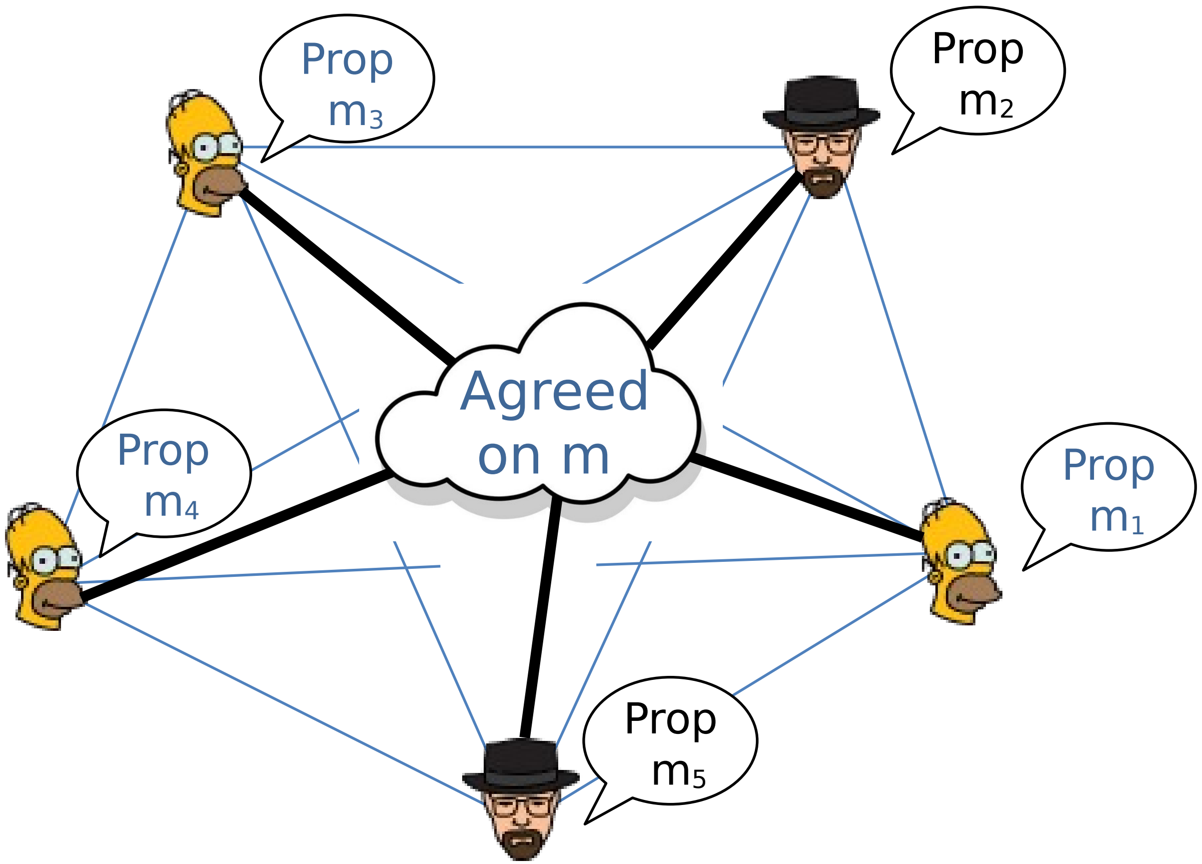
More on Intrusion-Tolerant Consensus

- Validity: if every honest party inputs the same m, they also output m.
- Consistency: every honest party outputs the same message m.
- Liveness: every honest party outputs some message m.



More on Intrusion-Tolerant Consensus

- Validity: if every honest party inputs the same m, they also output m.
- Consistency: every honest party outputs the same message m.
- Liveness: every honest party outputs some message m.
- Asynchronous protocol from [MR17] adapted to the network-agnostic setting.
  - Ensures t_s-validity in synchrony.
  - $O((L + \lambda)n^3)$ communication complexity.



DKG Protocol Overview

- Consists of a synchronous and asynchronous component.

- Goal: agree on $t_s + 1$ or more PVSS sharings to jointly combine.
  - PVSS: publicly verifiable secret sharing: non-interactive sharing from a dealer.

DKG Protocol Overview

- Consists of a synchronous and asynchronous component.

- Goal: agree on $t_s + 1$ or more PVSS sharings to jointly combine.
  - PVSS: publicly verifiable secret sharing: non-interactive sharing from a dealer.

- First, all parties synchronously broadcast a $O(\lambda n)$-sized PVSS sharing.

DKG Protocol Overview

- Consists of a synchronous and asynchronous component.

- Goal: agree on $t_s + 1$ or more PVSS sharings to jointly combine.
  - PVSS: publicly verifiable secret sharing: non-interactive sharing from a dealer.

- First, all parties synchronously broadcast a $O(\lambda n)$-sized PVSS sharing.

- Then, all parties run (multivalued) intrusion-tolerant consensus on the output of all broadcasts.
  - t_s-validity in synchrony, so all parties agree.

DKG Protocol Overview

- Consists of a synchronous and asynchronous component.

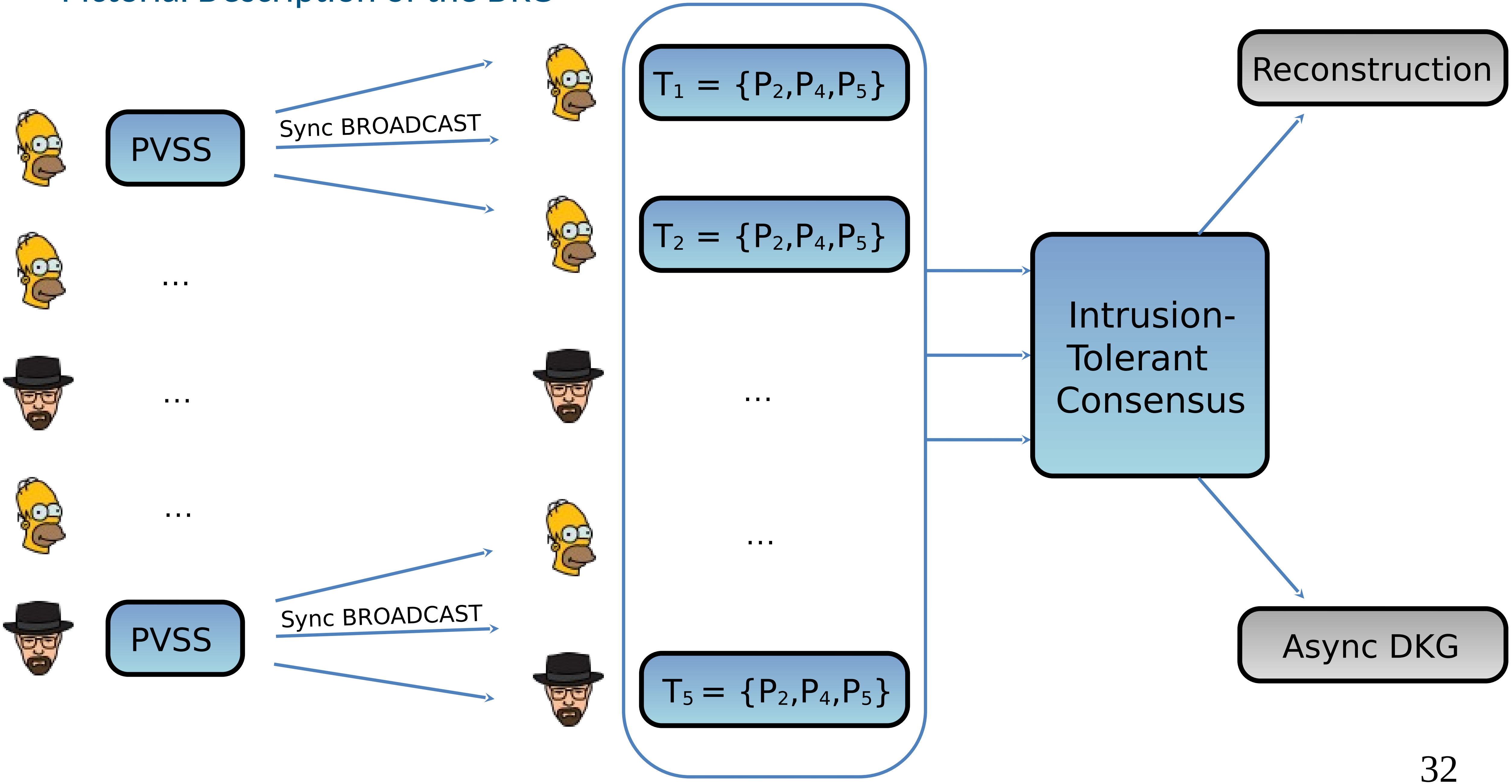
- Goal: agree on $t_s + 1$ or more PVSS sharings to jointly combine.
  - PVSS: publicly verifiable secret sharing: non-interactive sharing from a dealer.

- First, all parties synchronously broadcast a $O(\lambda n)$-sized PVSS sharing.

- Then, all parties run (multivalued) intrusion-tolerant consensus on the output of all broadcasts.
  - t_s-validity in synchrony, so all parties agree.

- In asynchrony, the intrusion-tolerant consensus may return $\perp$.
  - Fallback to an existing ADKG protocol with $O(\lambda n^3)$ communication complexity!

Pictorial Description of the DKG



Additional protocol details

- Running our ITC protocol (with CC $O((L + \lambda)n^3)$) on all PVSS sharings would be *too expensive*
  - Thus, parties run consensus on an *accumulated* value z.
  - z contains n accumulated values: value i is a set of O(n) values P_i needs to reconstruct the public key.
  - Then if ITC terminates with $T \neq \perp$, the honest proposers can forward these values to each P_i.

Additional protocol details

- Running our ITC protocol (with CC $O((L + \lambda)n^3)$) on all PVSS sharings would be *too expensive*.
  - Thus, parties run consensus on an *accumulated* value z.
  - z contains n accumulated values: value i is a set of O(n) values P_i needs to reconstruct the public key.
  - Then if ITC terminates with $T \neq \bot$, the honest proposers can forward these values to each P_i.

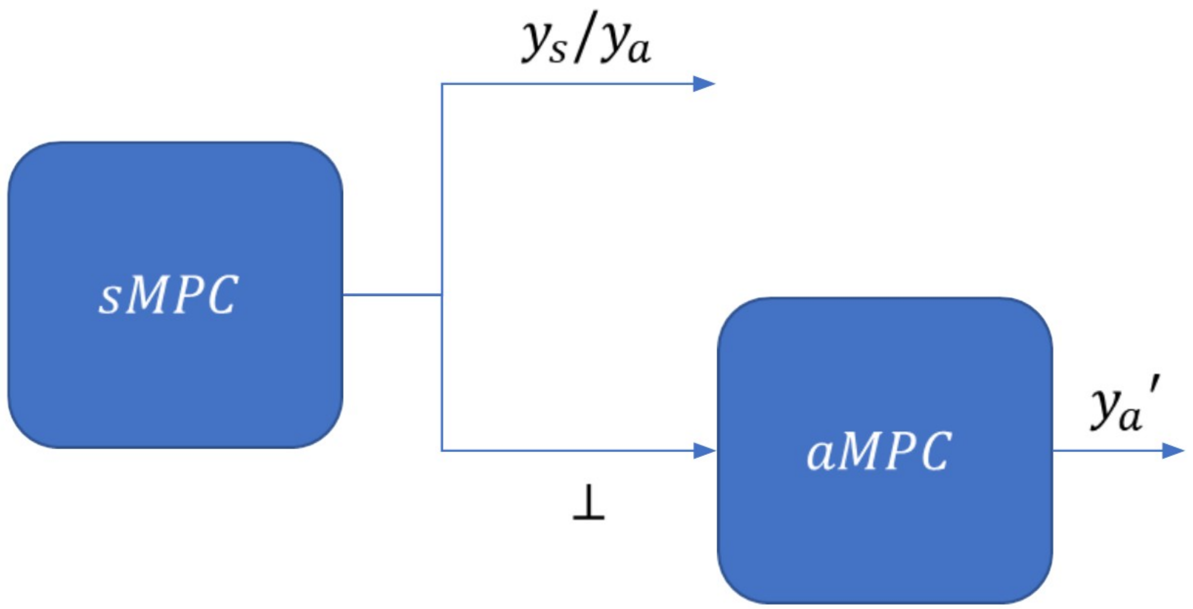
- In asynchrony, synchronous broadcast can result in arbitrary disagreement.
  - Thus, parties propose an accumulated value z to ITC only if they receive enough valid PVSS sharings.
  - Otherwise, they simply propose $\bot$ to ITC.

Application to MPC and Improving Complexity

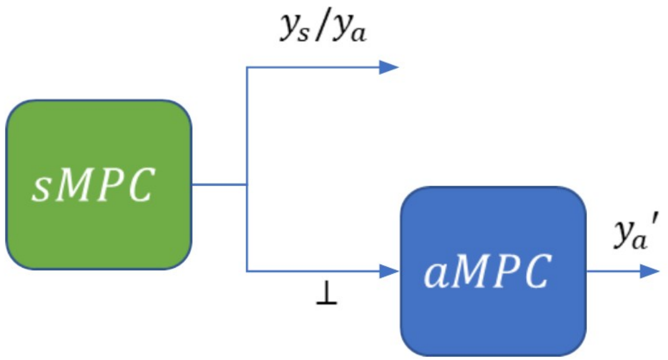
- Our DKG can be used to bootstrap network-agnostic MPC without trusted setup [BLL20, …].

- We also improve complexity over [BLL20]: either a linear improvement in communication or go from trusted setup to plain PKI for free!
  - $O(\lambda n^2)$ complexity per multiplication gate with trusted setup, matching the asynchronous state-of-the-art with additively-homomorphic threshold encryption [HNP08].

- The BLL20 protocol:

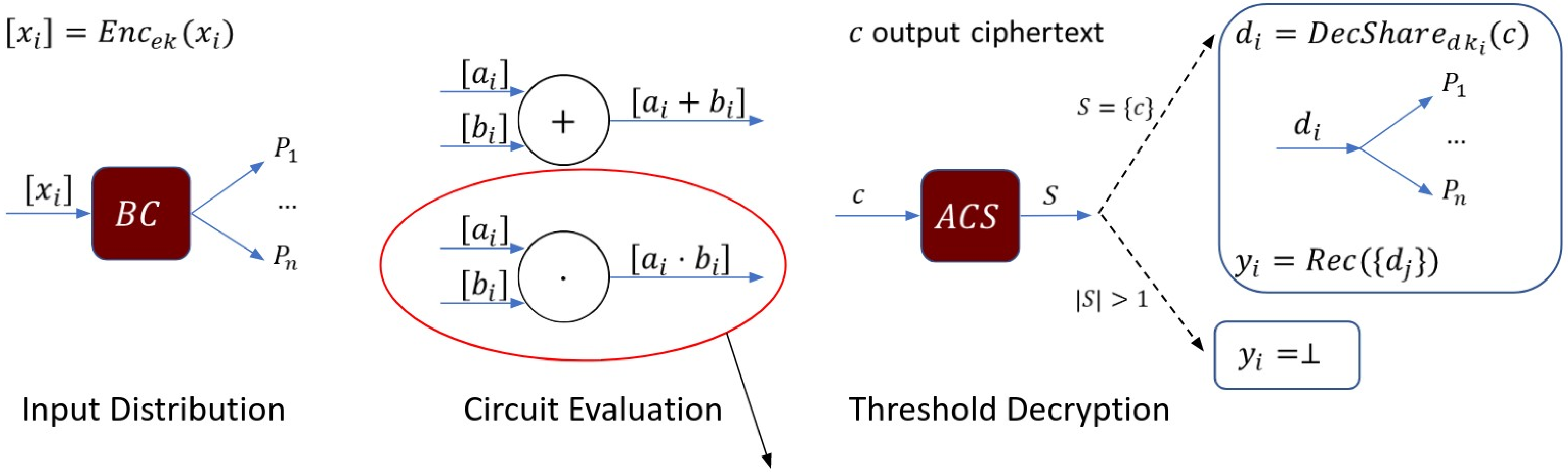


On BLL20



- ACS = agreement on a core set
- BC = broadcast

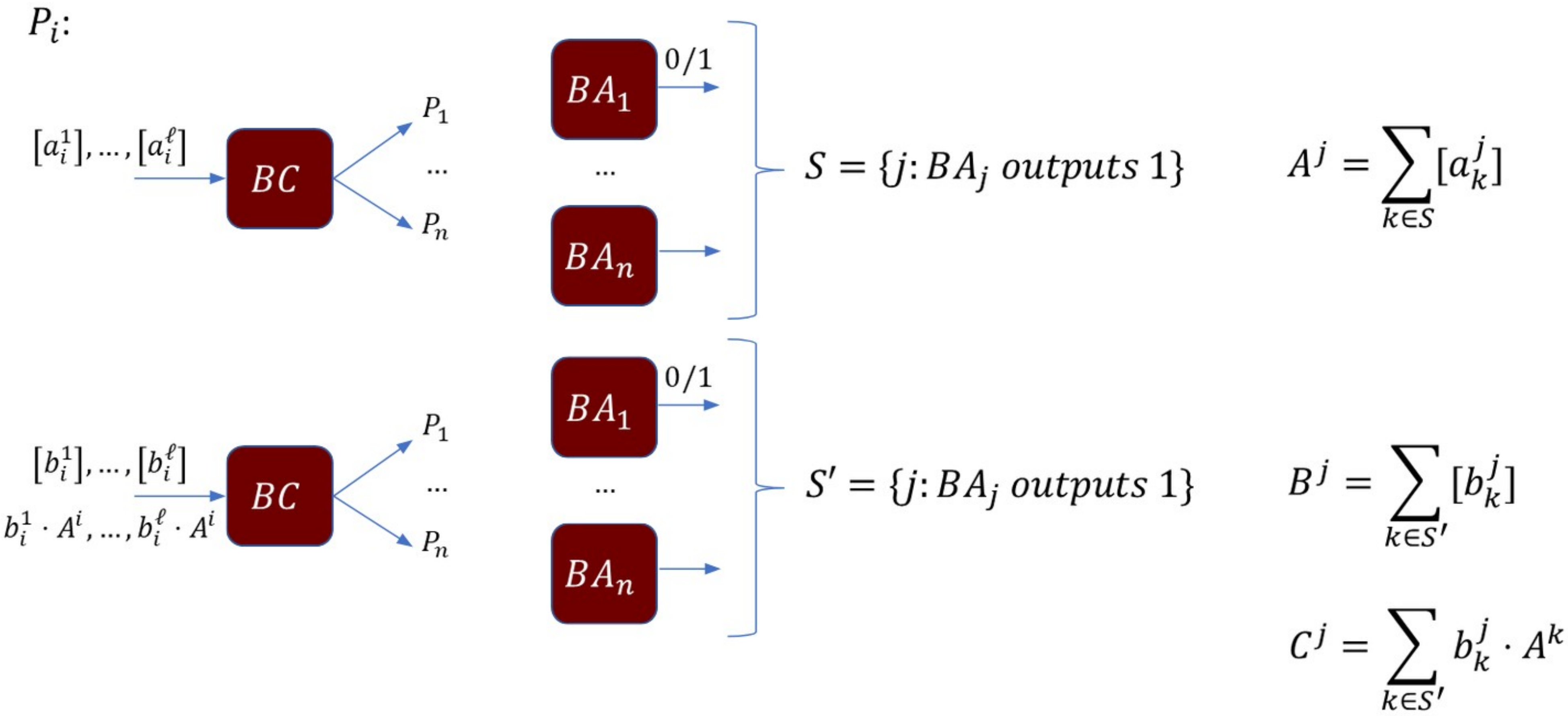
P_i:



Bottleneck!
Uses n network-agnostic BA per gate!
With [DHL21], each BA costs $O(\lambda n^2)$
=> $O(\lambda n^3|C|)$ cost for circuit C!

Efficient Network-Agnostic MPC: Amortisation with Beaver Triples

- Number of BA instances now independent of |C|.
- Uses our O(nL + λn^2) CC broadcast protocol.



Output (A^j, B^j, C^j), for $j = 1 \ldots \ell$

Conclusion and Future Work

- We obtain network-agnostic DKG (almost) for free!

- To make it even freer:
  - Can we obtain O(1) round complexity?
  - Can we obtain adaptive security?
- Additional future work: implementation!

- Thank you for funding us:
  - Renas Bacho: DFG
  - Chen-Da Liu-Zhang: Web3 Foundation, NSF, DARPA, Ripple, DoE, JP Morgan, PNC, Cylab

- Thank you!

Full Paper

Bibliography 1/2

- [GJKR07] Gennaro, Jarecki, Krawczyk, Rabin. Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology
- [HNP08] Hirt, Nielsen, Przydatek. Asynchronous multi-party computation with quadratic communication. ICALP 2008
- [MR17] Mostéfaoui, Raynal. Signature-free asynchronous byzantine systems: from multivalued to binary consensus with $t < n/3$, $O(n^2)$ messages, and constant time. Acta Informatica
- [BKL19] Blum, Katz, Loss. Synchronous consensus with optimal asynchronous fallback guarantees. TCC'19
- [BLL20] Blum, Liu-Zhang, Loss. Always have a backup plan: Fully secure synchronous mpc with asynchronous fallback. CRYPTO'20
- [NRS+20] Nayak, Ren, Shi, Vaidya, Xiang. Improved extension protocols for byzantine broadcast and agreement. DISC'20
- [AJM+21] Abraham, Jovanovic, Maller, Meiklejohn, Stern, Tomescu. Reaching consensus for asynchronous distributed key generation. PODC'21
- [BKL21] Blum, Katz, Loss. Tardigrade: An atomic broadcast protocol for arbitrary network conditions. ASIACRYPT'21

Bibliography 2/2

- [DHL21] Deligios, Hirt, Liu-Zhang. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. TCC'21
- [SBKN21] Shrestha, Bhat, Kate, Nayak. Synchronous distributed key generation without broadcasts. Preprint
- [ABKL22] Alexandru, Blum, Katz, Loss. State machine replication under changing network conditions. ASIACRYPT'22
- [ACC22] Appa, Chandramouli, Choudhury. Perfectly-secure synchronous mpc with asynchronous fallback guarantees. PODC'22
- [DYX+22] Das, Yurek, Xiang, Miller, Kokoris-Kogias, Ren. Practical asynchronous distributed key generation. S&P'22
- [GLW22] Ghinea, Liu-Zhang, Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. PODC'22
- [TLP22] Tsimos, Loss, Papamanthou. Gossiping for communication-efficient broadcast. CRYPTO'22
- [AJM+23] Abraham, Jovanovic, Maller, Meiklejohn, Stern. Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation. CRYPTO'23