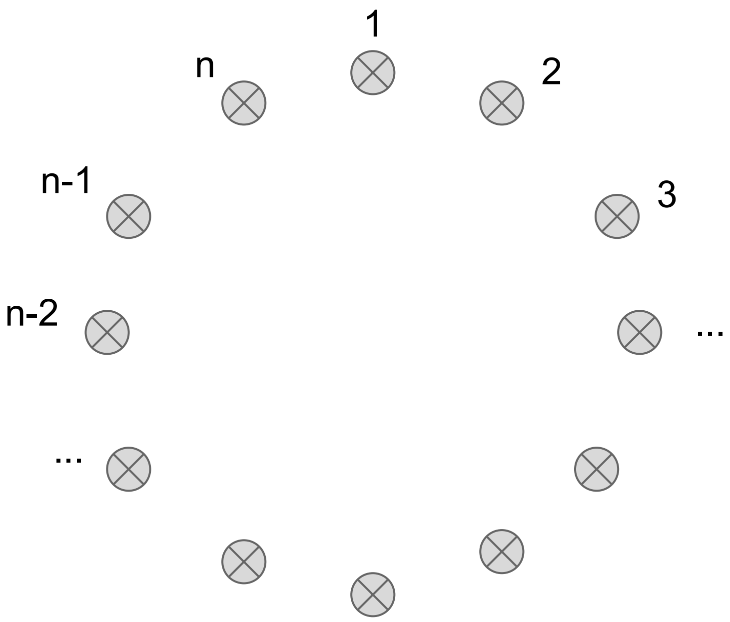
Polygraph

Accountable Byzantine Agreement

Table of Content

Model

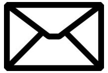
n processes

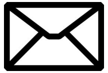


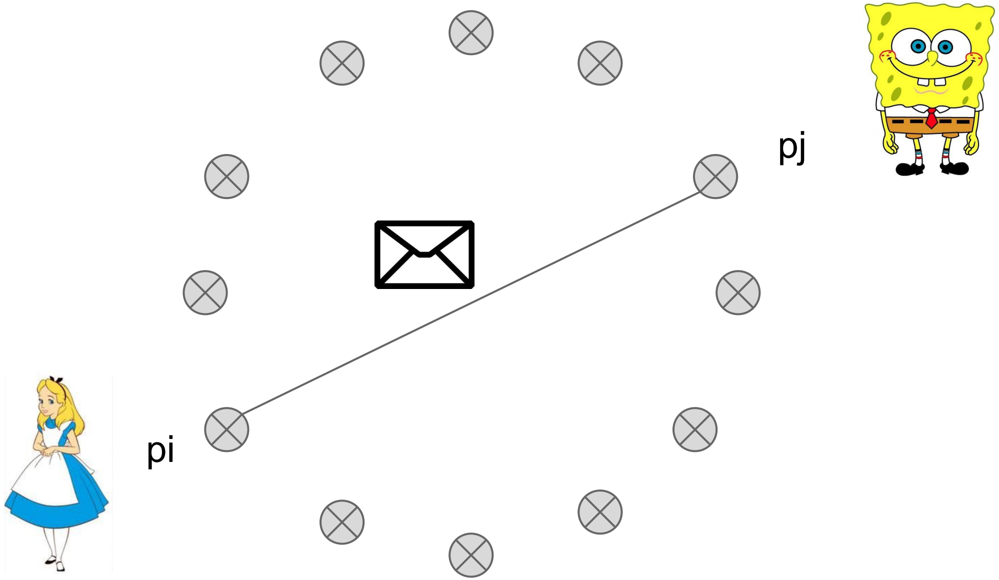
Communication Network

Reliable

Point-to-point

Send()

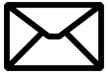
Recv()

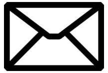


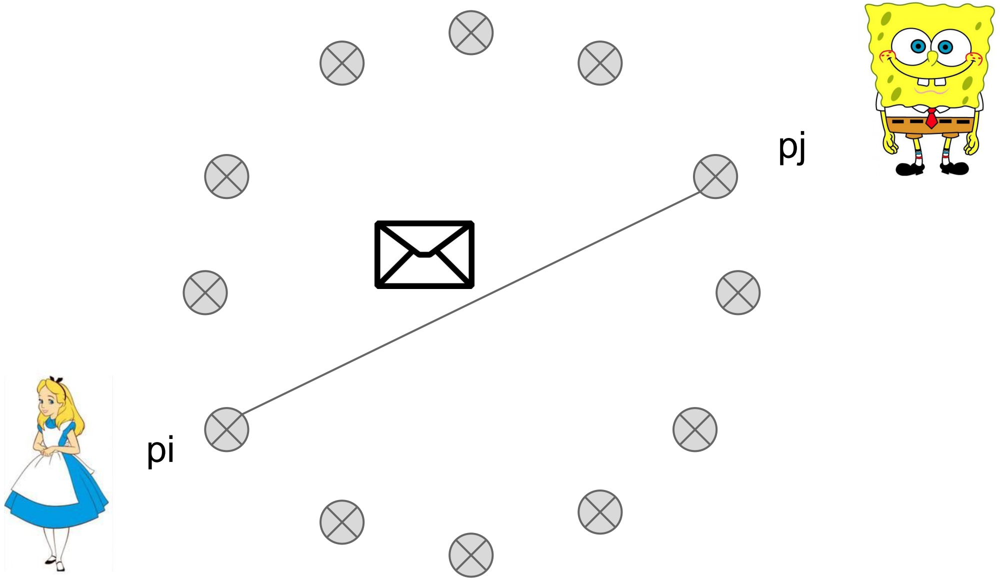
Communication Network

Reliable

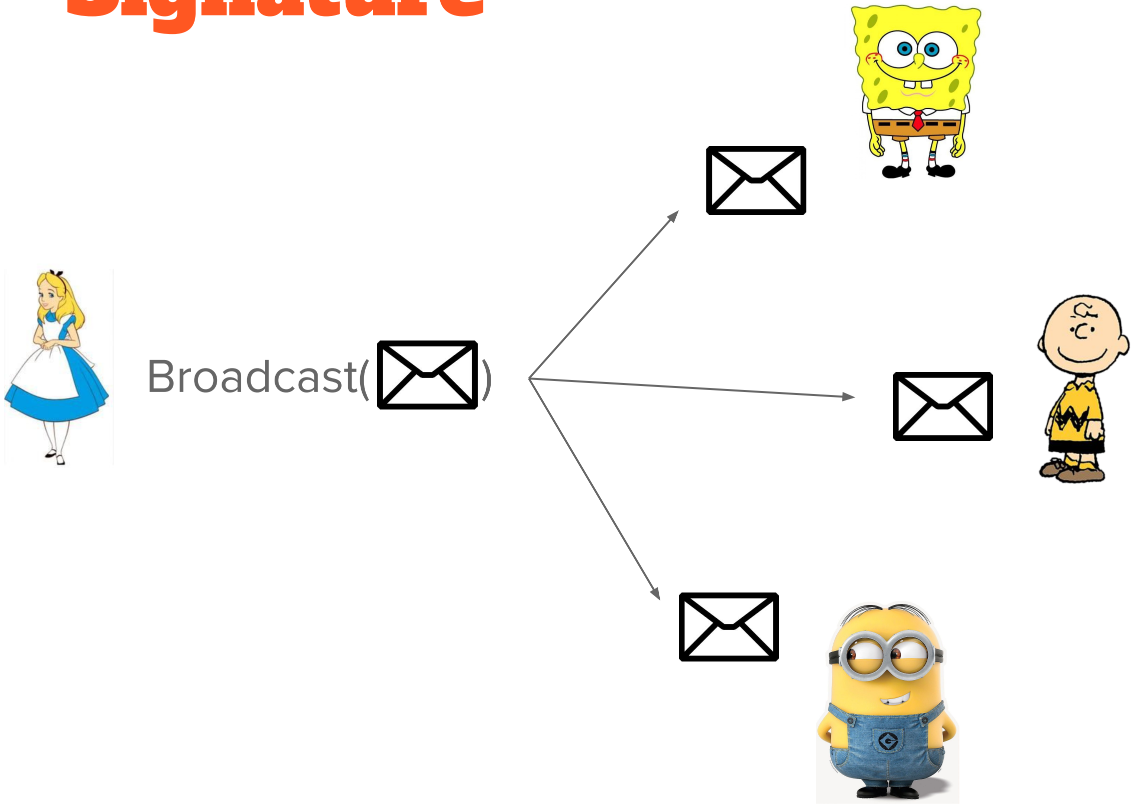
Point-to-point

Send()

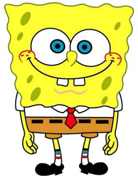
Recv()

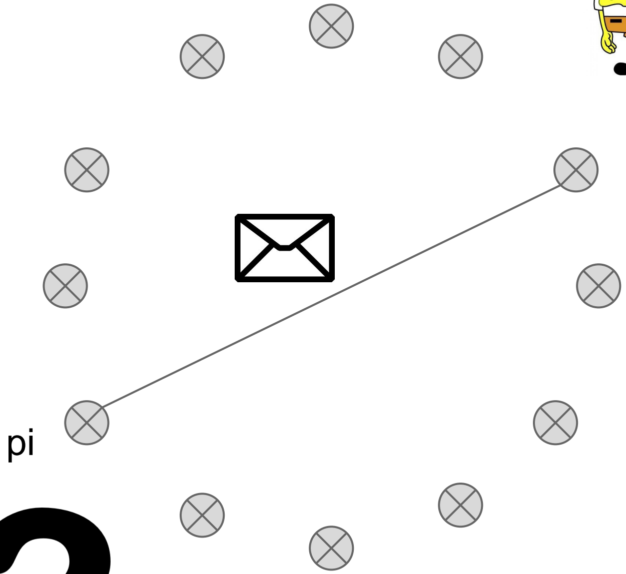


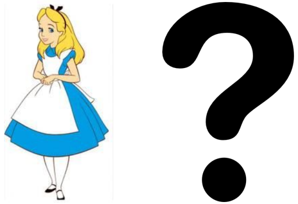
Signature



Partial Synchrony







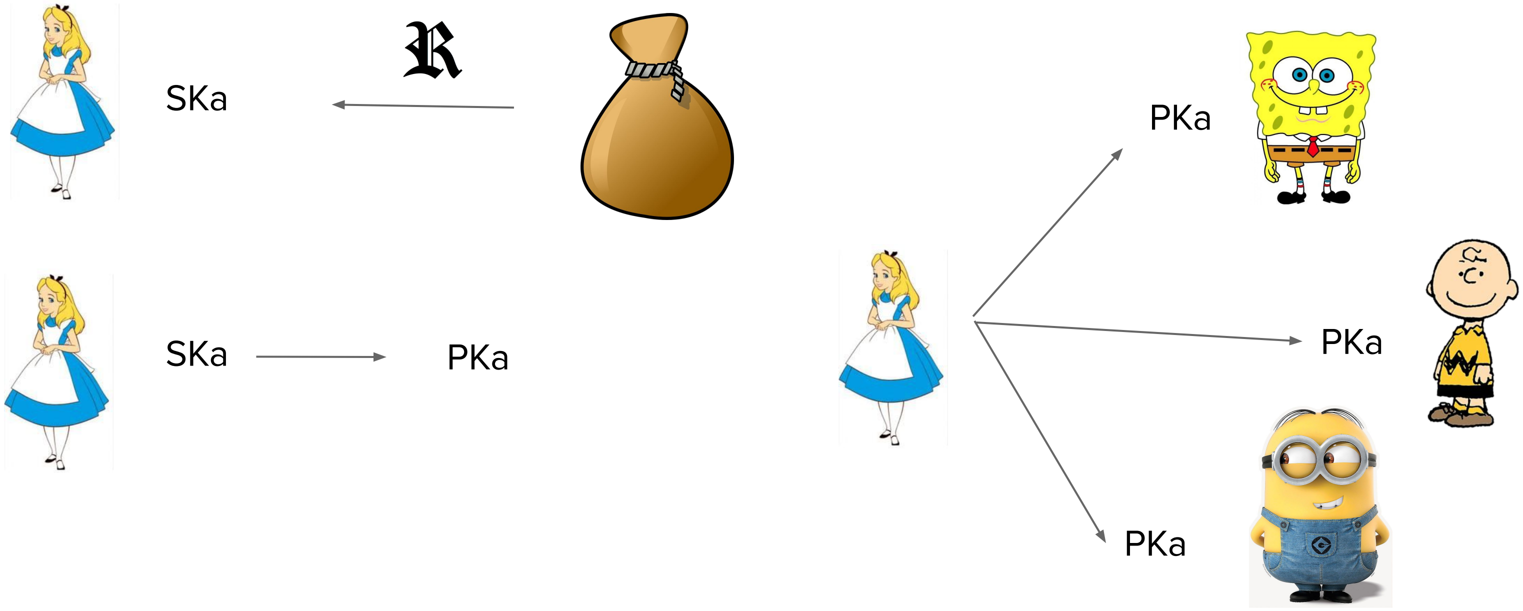
Signature

Authentication

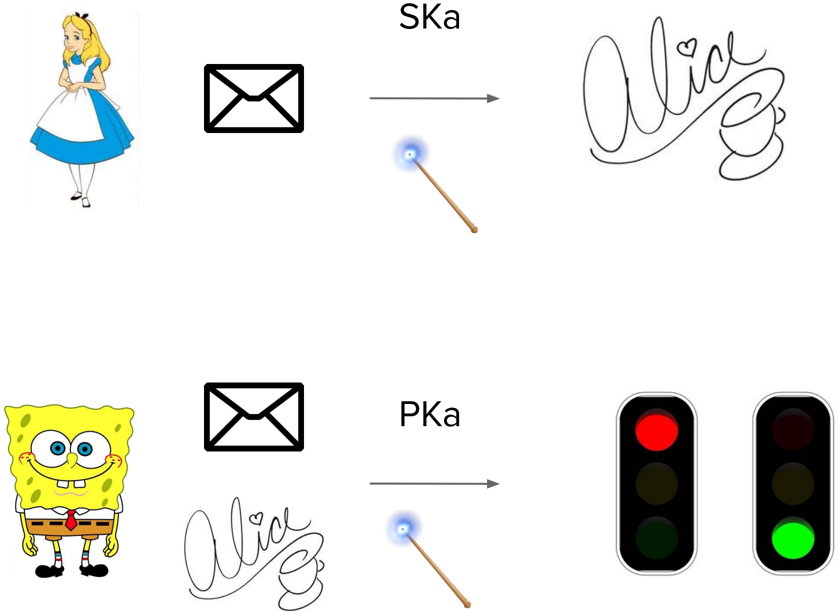
Integrity

Non-Repudiation

Signature



Signature

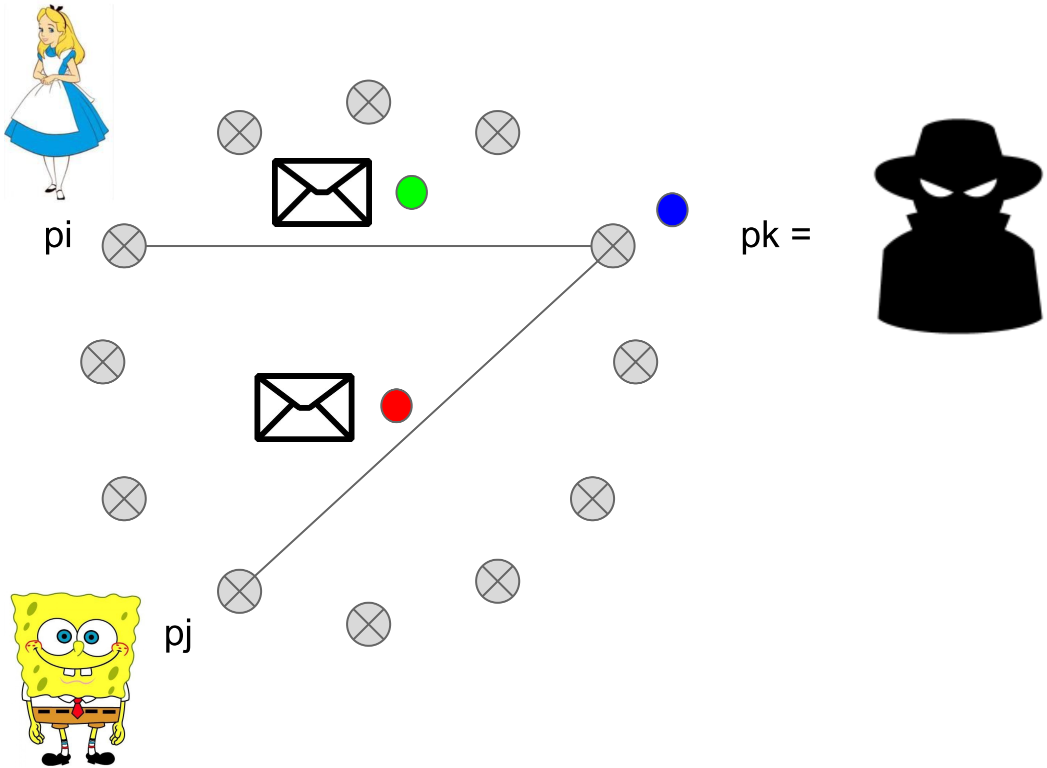


Byzantine processes

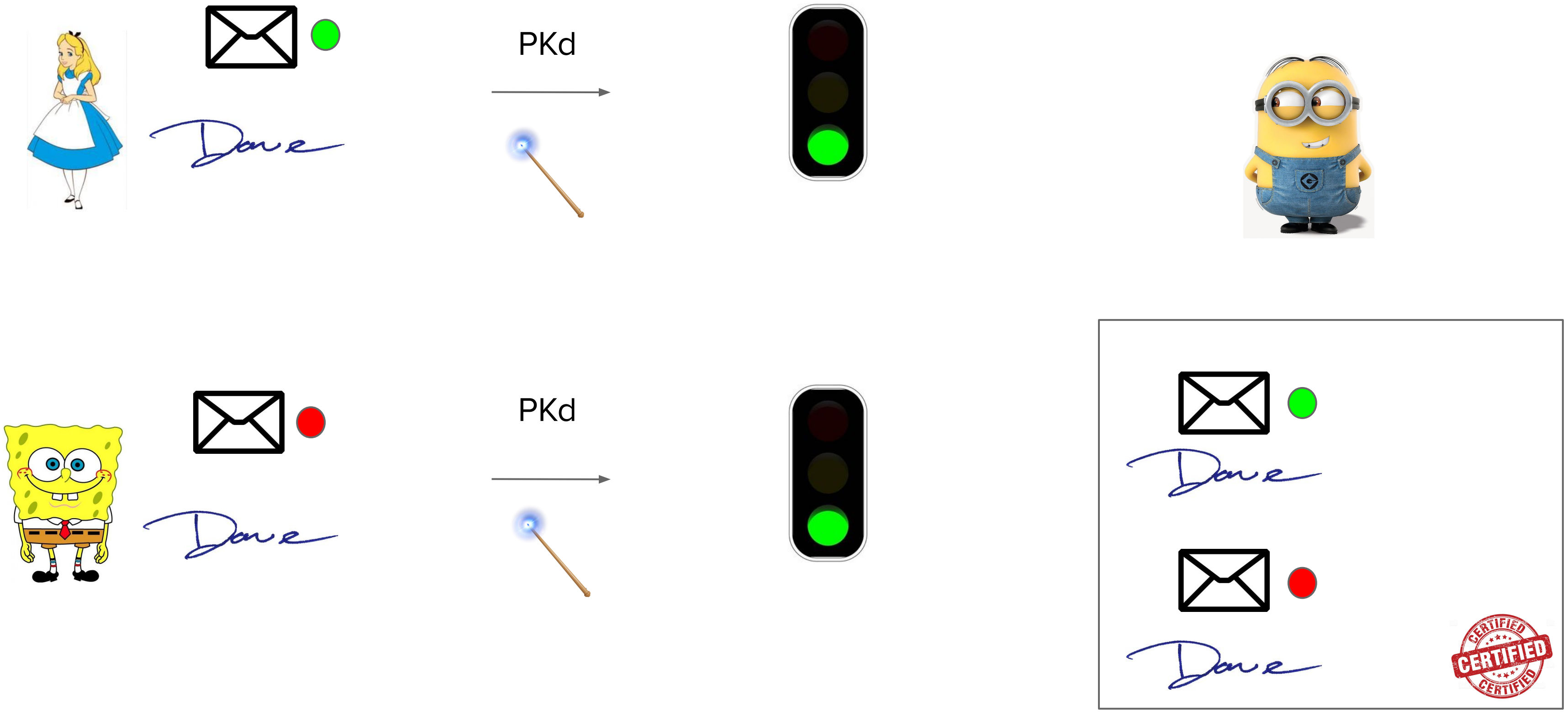
omission

commission

Mutant messages



Conflicting Signed Message

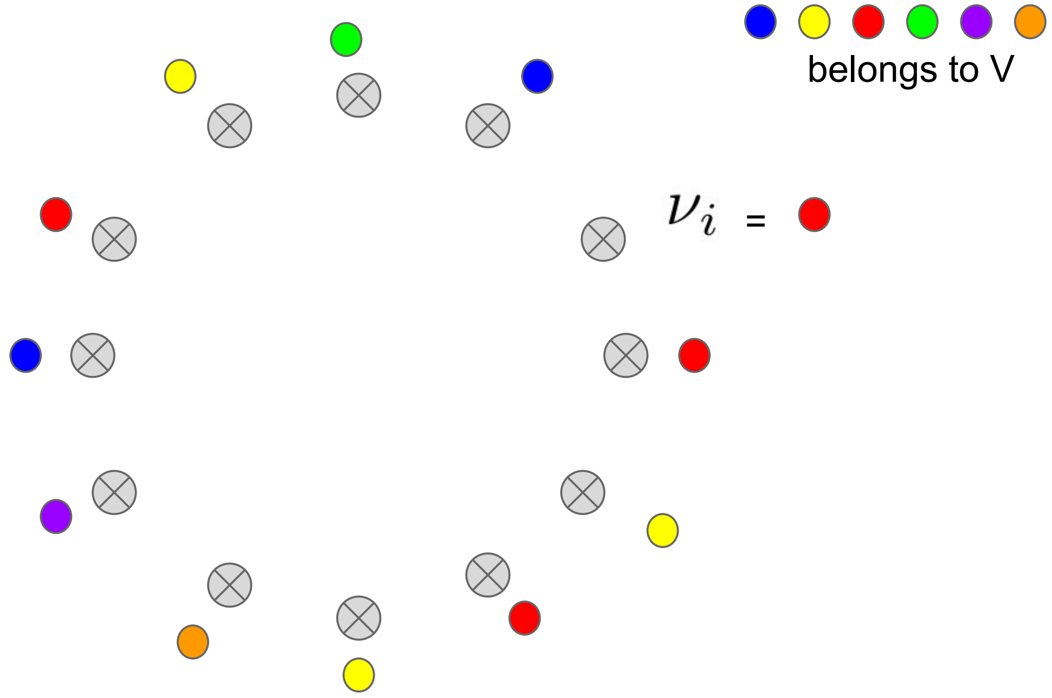


The Adversary

- **anticipate** the intern state of every process
- all his connexion are **infinitely rapid**
- can manage like one person the actions of a **malicious coalition of t processes**
- **can’t forge the signature** of a correct process
- **can’t interfere** with the messages exchanges among honest users.

What is the Consensus Problem ?

Initial value

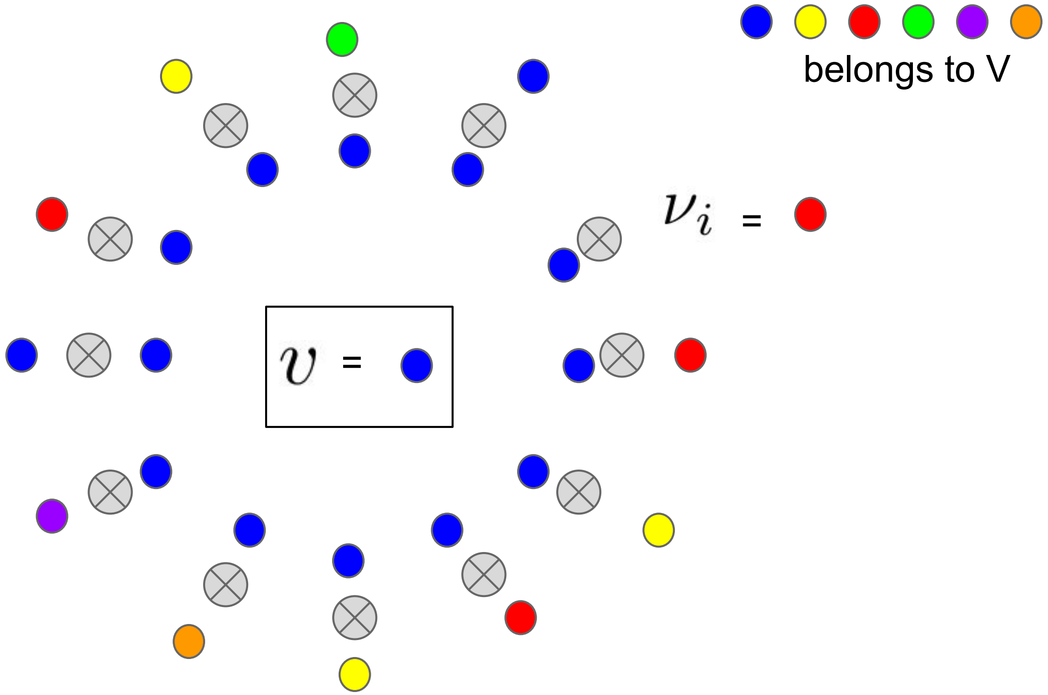


Solving the Consensus

Validity

Agreement

Liveness



Solving the Consensus

Validity

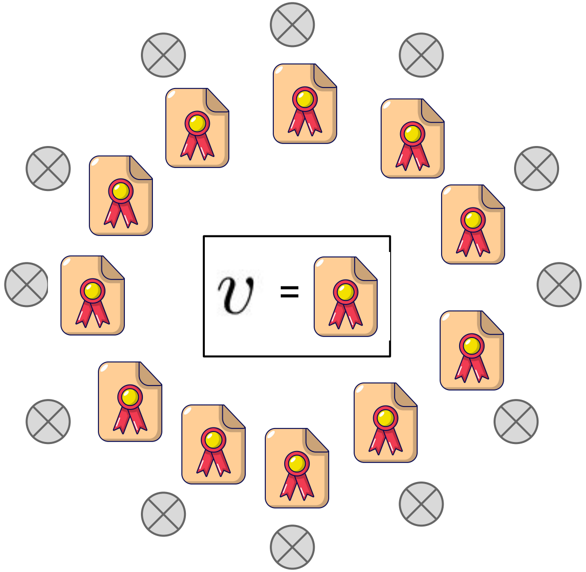
Agreement

➔ no double-spending

Liveness

➔ Some txs

are eventually committed



ν_i =

Valid() :

- No double-spending
- Correct Signatures
- Everyone has the money to pay

How many  can we tolerate ?

1980, M. Pease, R. Shostak and L. Lamport

Reaching Agreement in the Presence of Faults

$$t < n/3$$

GDBC

Gracefully Degrading Byzantine Consensus

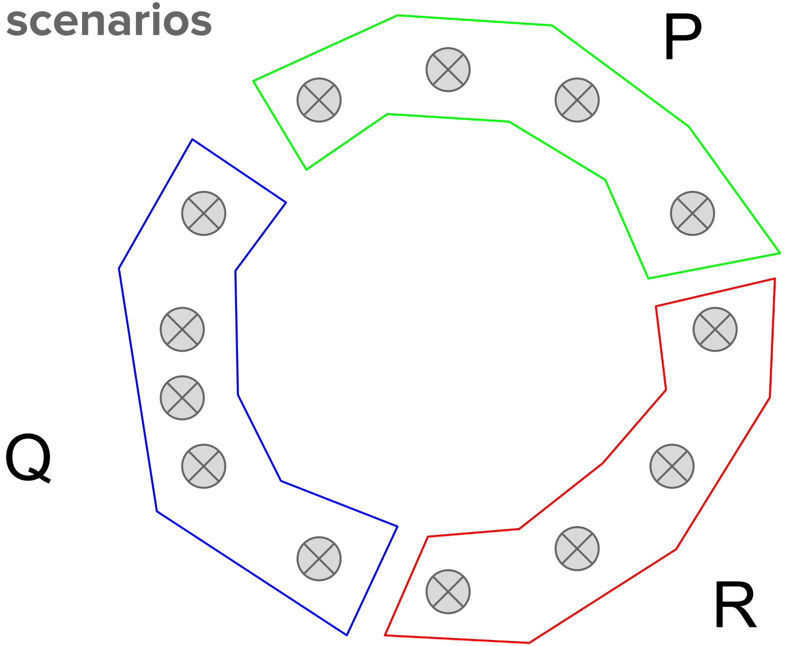
Gracefully Degrading Byzantine Consensus

< n/3 ➔ Consensus (Safety + Liveness)

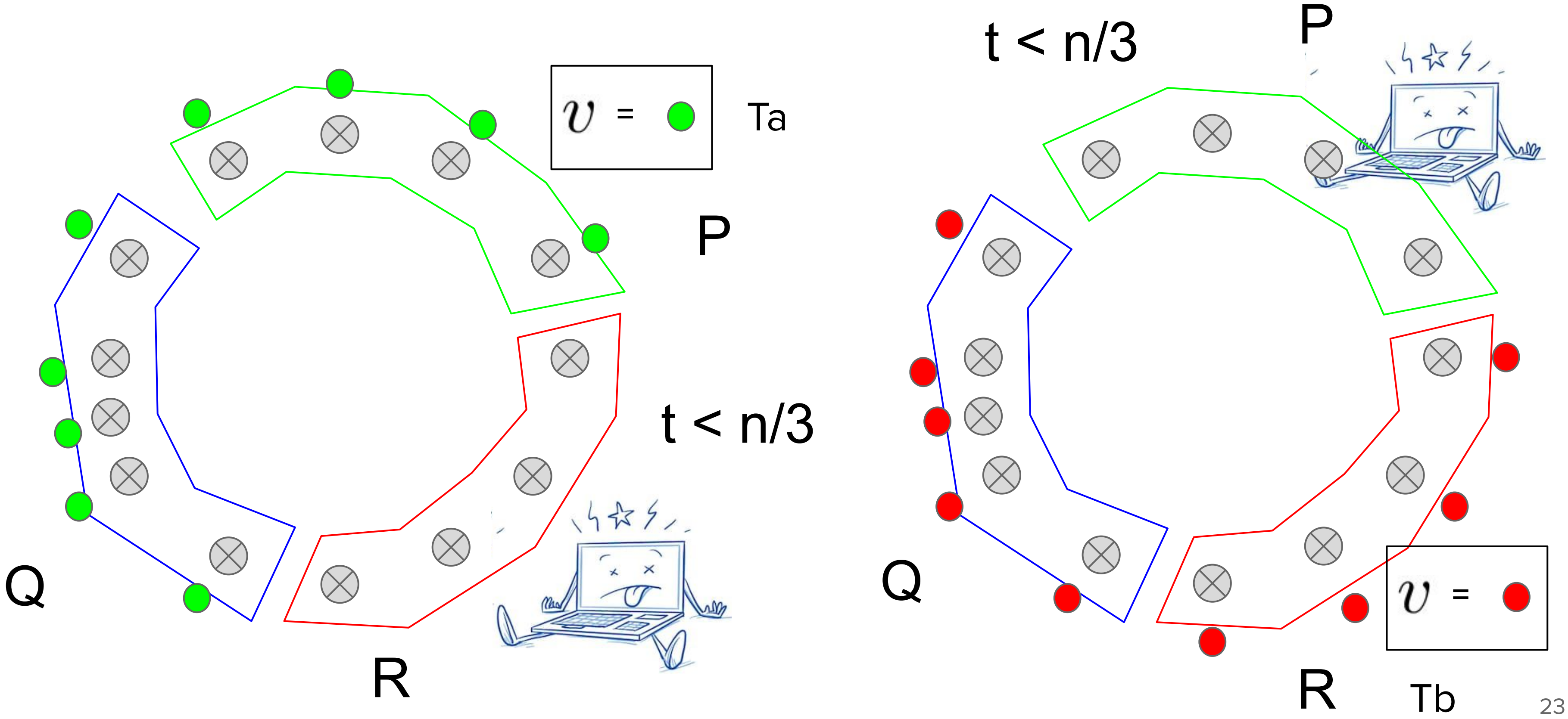
>= n/3 ➔ Safety (~~Liveness~~)

Impossibility of solving GDBC

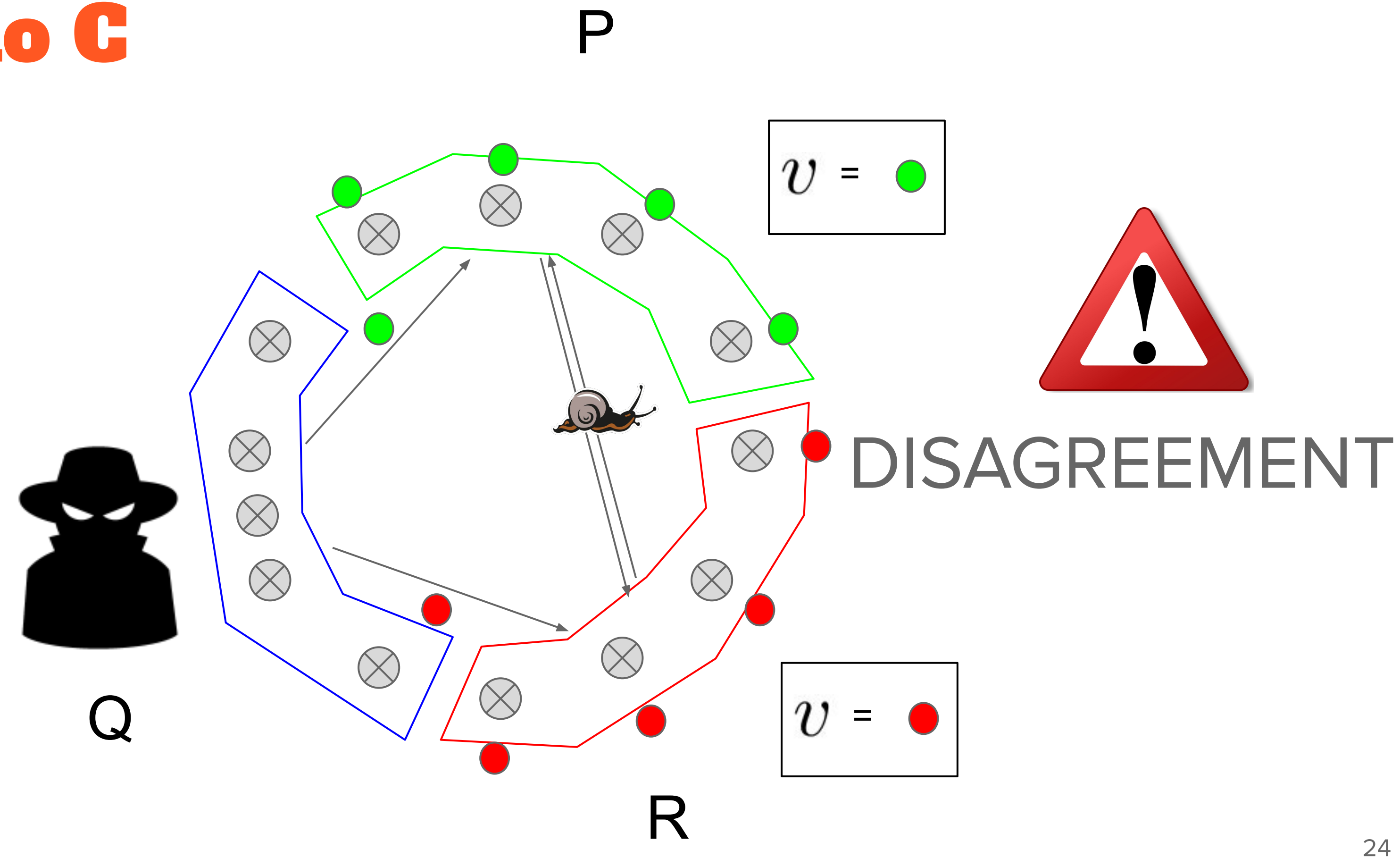
Undistinguishable scenarios



Scenarios A and B



Scenario C



Polygraph

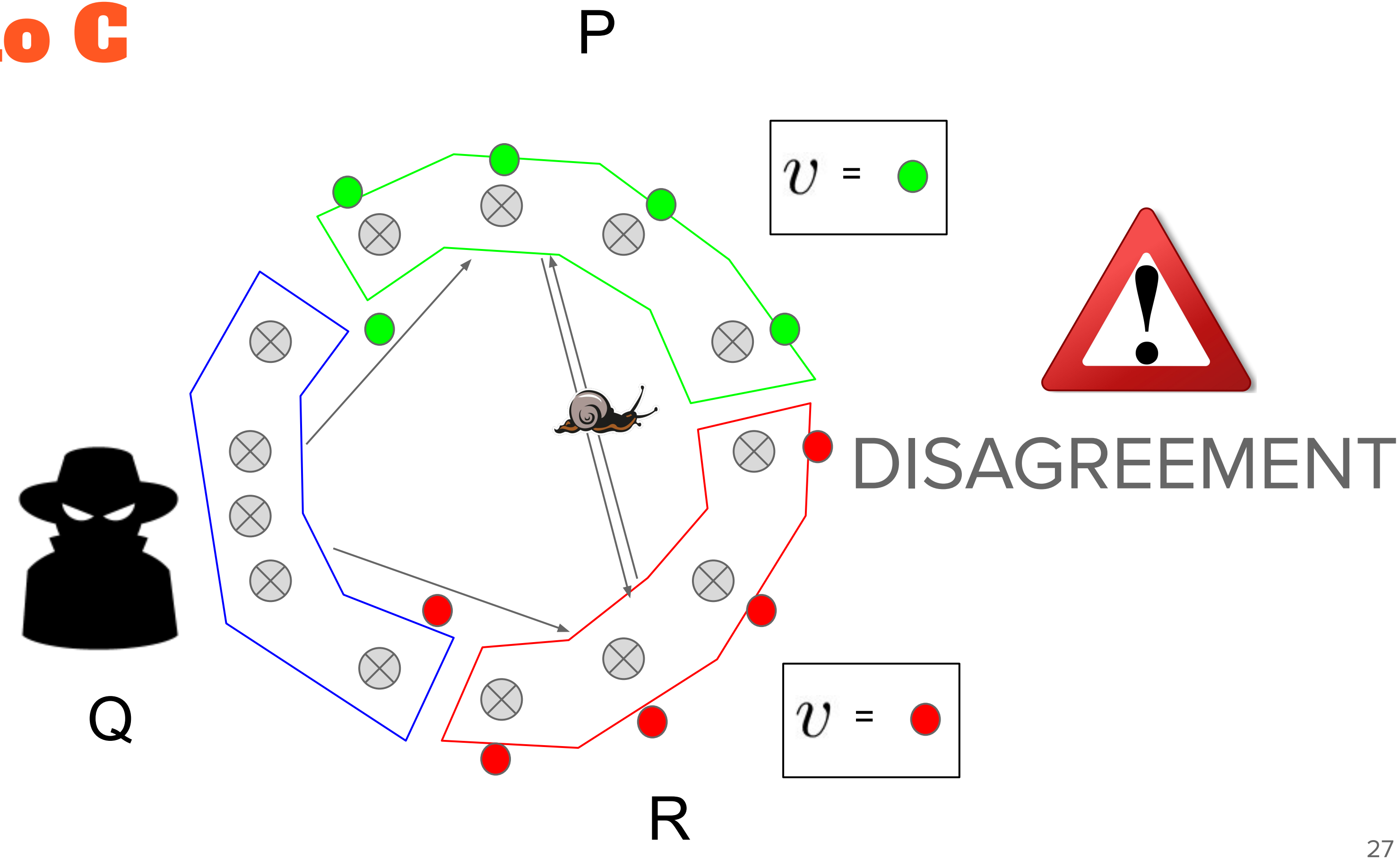
Accountable Byzantine Consensus

Accountable Byzantine Consensus

< n/3 → Consensus (Safety + Liveness)

→ Disagreement → Detection →

Scenario C

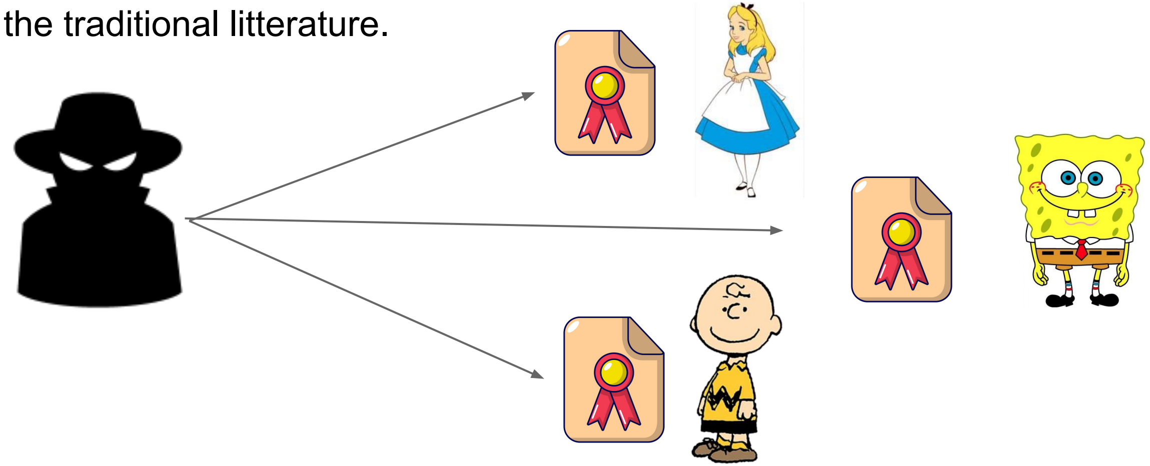


On the validity

Weak Validity : If all processes are correct and if a correct process decides v, then v is the initial value of some process.

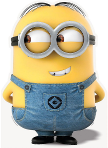
Strong Validity : If all correct processes have the same initial value v and a correct process decides, then it decides v

Here, We always authorize a valid Block proposed by a malicious node, but we still solve weak Validity to avoid trivial solution and follow the traditional litterature.

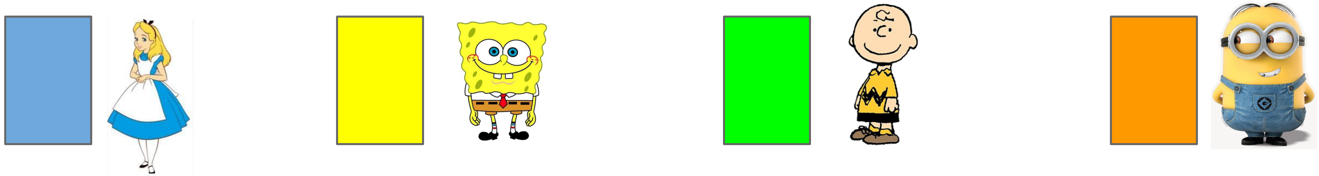


Solution

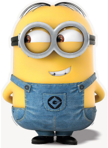
Binary Reduction

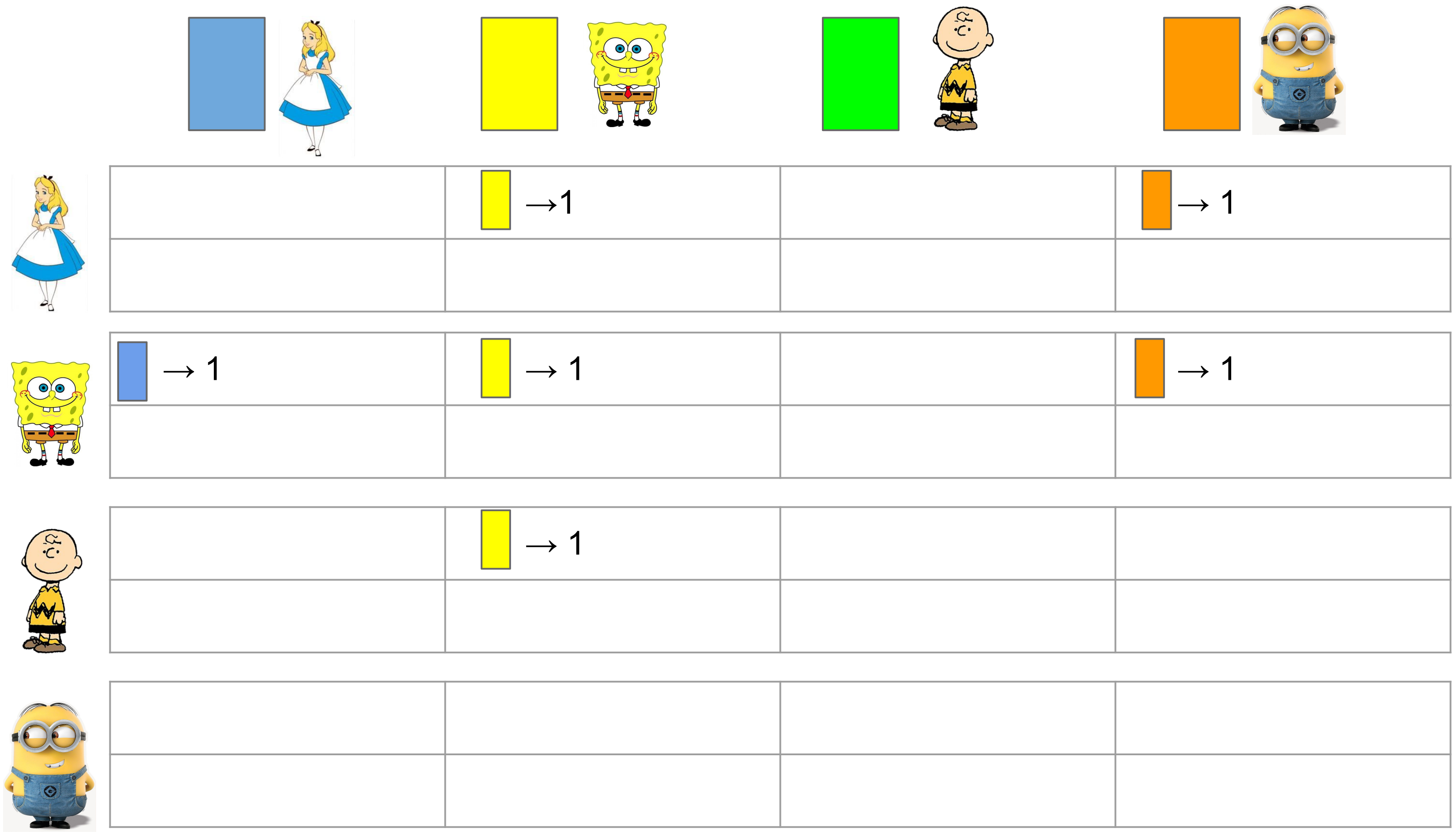
				
				
				
				
				

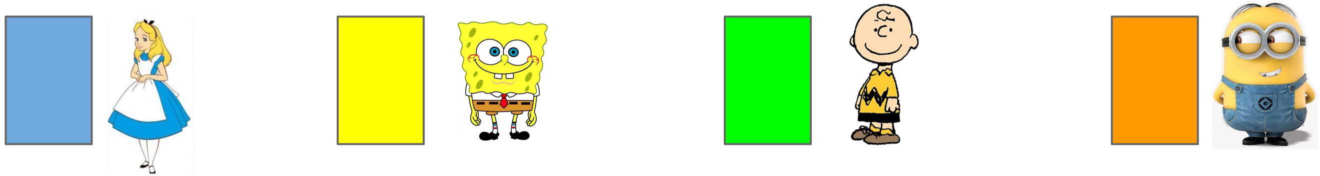
→



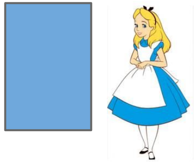
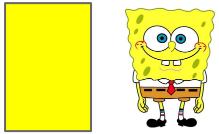
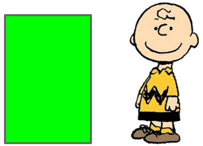
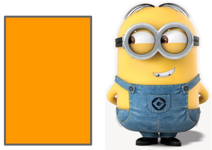
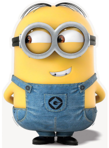
		→1		
				
				
				

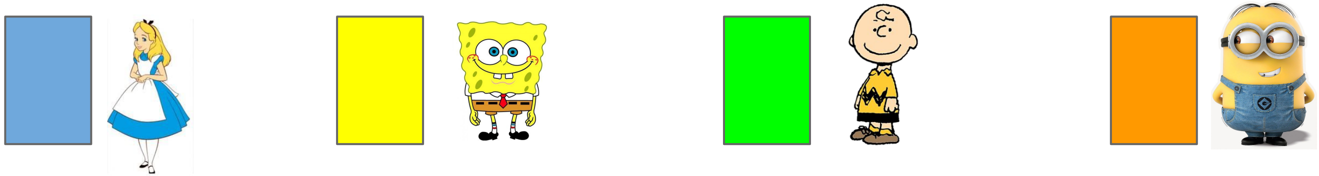
				
		→1		
		→ 1		
		→ 1		
				

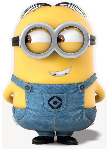

→1
→ 1
→ 1
→ 1
→ 1
→ 1

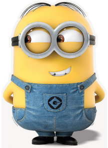


		→1		→ 1
		1		
	→ 1	→ 1		→ 1
		1		
		→ 1		
		1		
				

				
	→ 0	→1	→ 0	→ 1
	→ 1	→ 1	→ 0	→ 1
	→ 0	→ 1	→ 0	→ 0
				



	→ 0	→ 1	→ 0	→ 1
	0	1	0	1
	→ 1	→ 1	→ 0	→ 1
	0	1	0	1
	→ 0	→ 1	→ 0	→ 0
	0	1	0	1
				

				
	→ 0	→ 1	→ 0	→ 1
	0	1	0	1
	→ 1	→ 1	→ 0	→ 1
	0	1	0	1
	→ 0	→ 1	→ 0	→ 0
	0	1	0	1
				

Polygraph

Accountable Binary Byzantine Consensus
With Strong Validity

Accountable Binary Byzantine Consensus

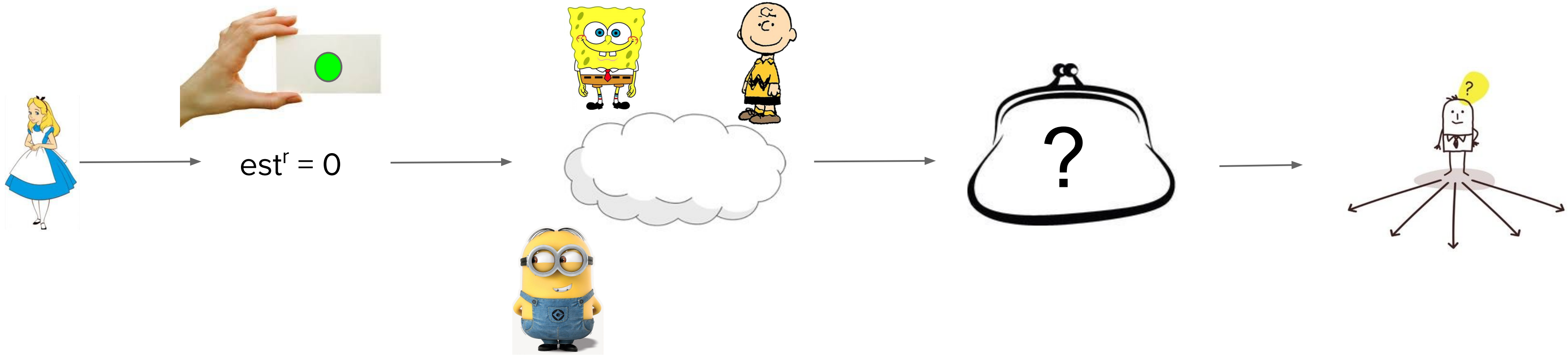
< n/3 → Consensus (Safety + Liveness)

→ Disagreement → Detection →

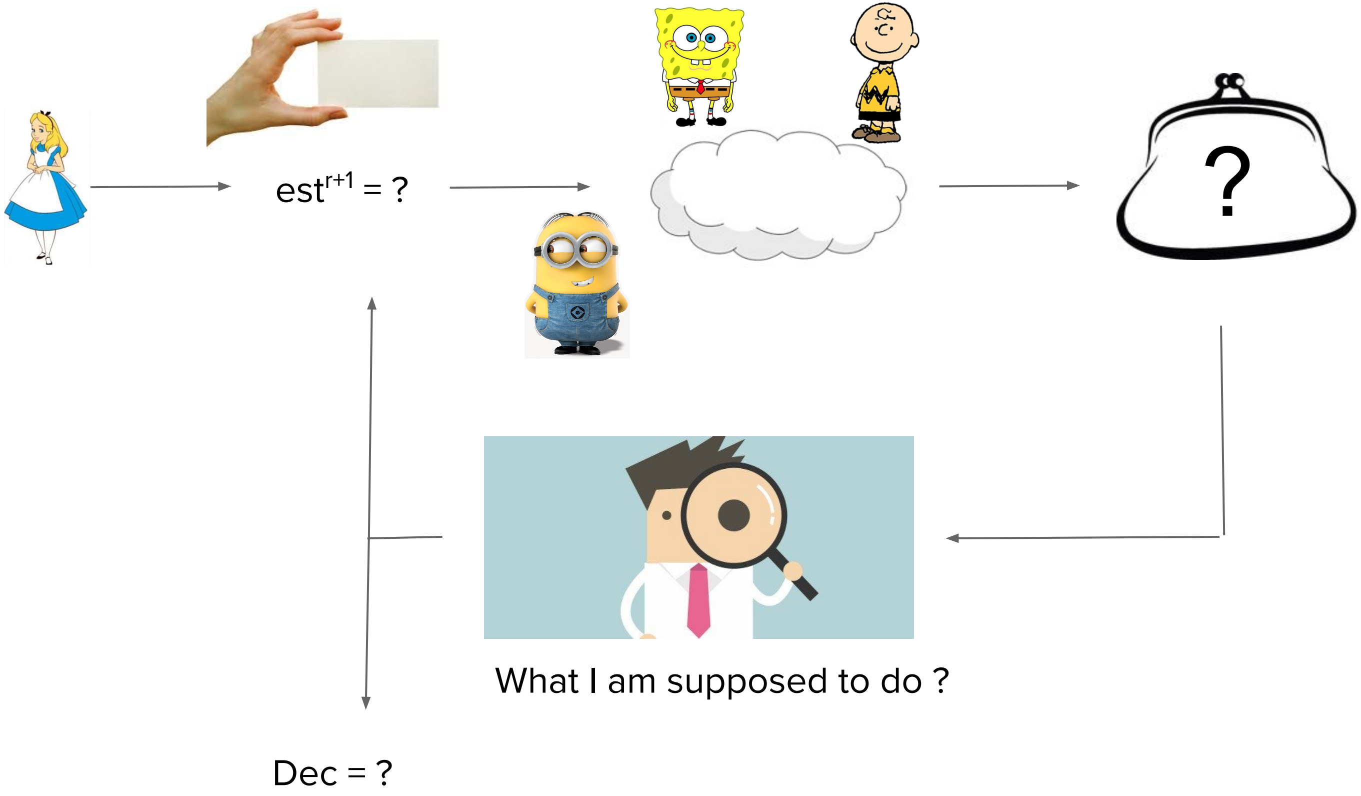
Decision in different round

Give the intuition why two correct nodes can decide in different rounds

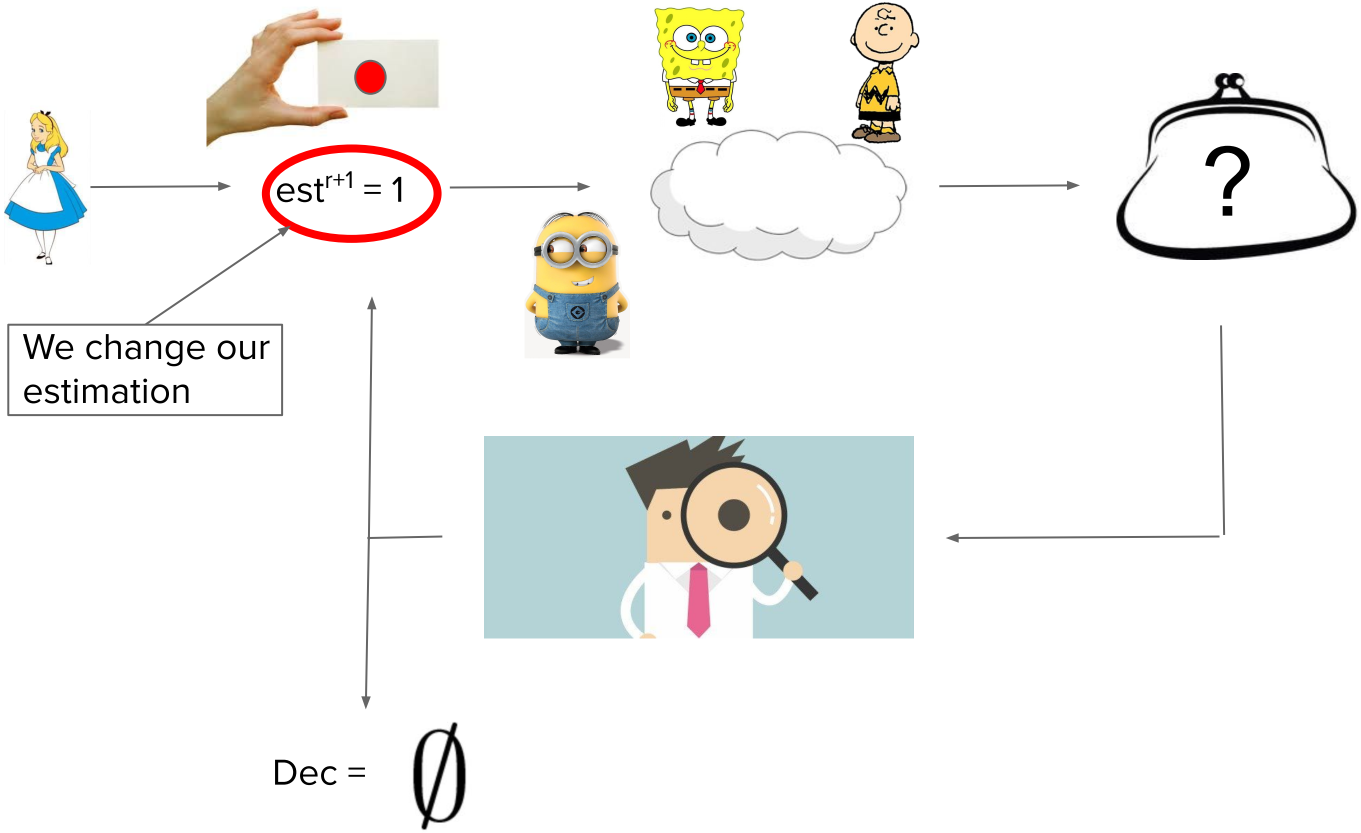
Binary Consensus Architecture



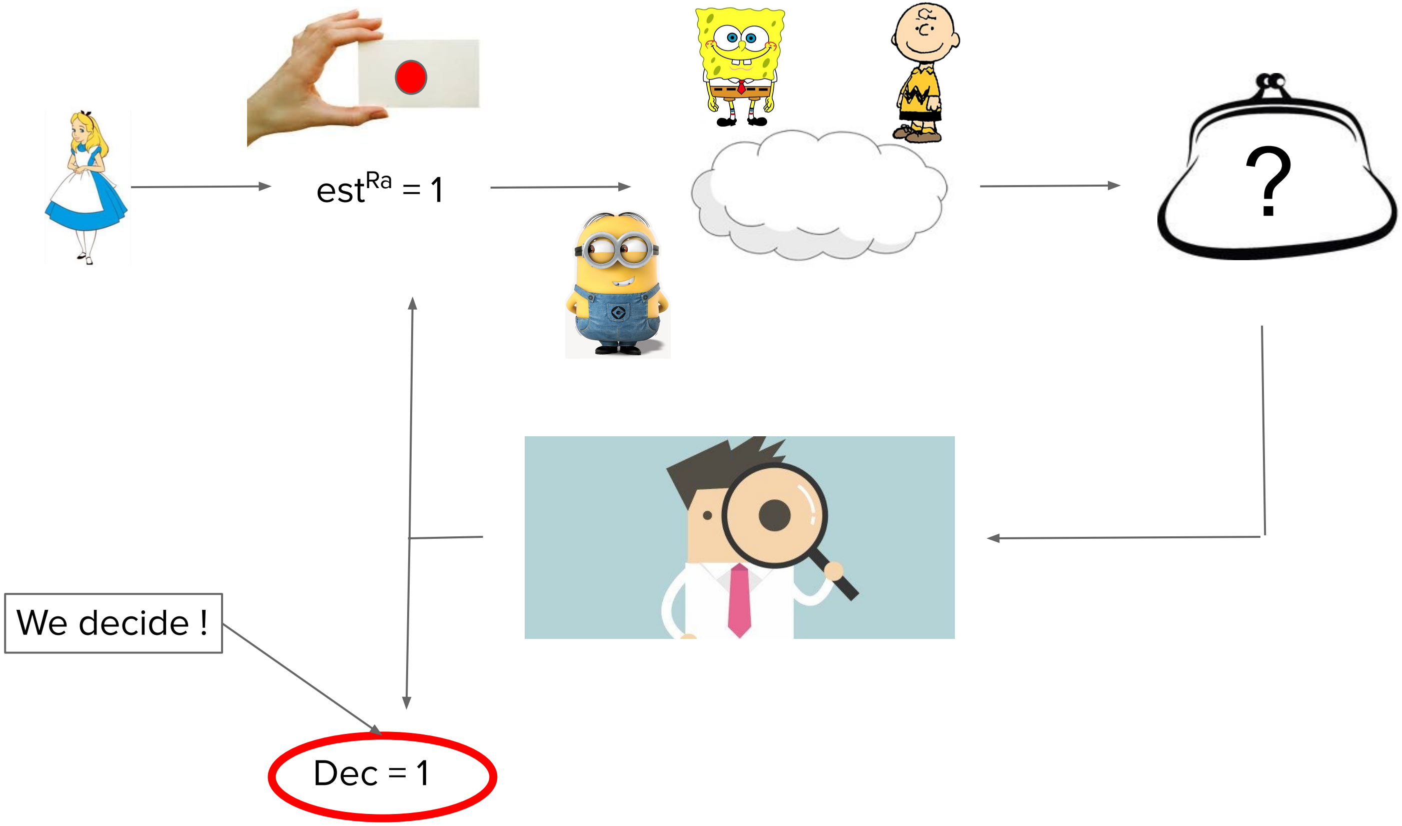
Binary Consensus Architecture



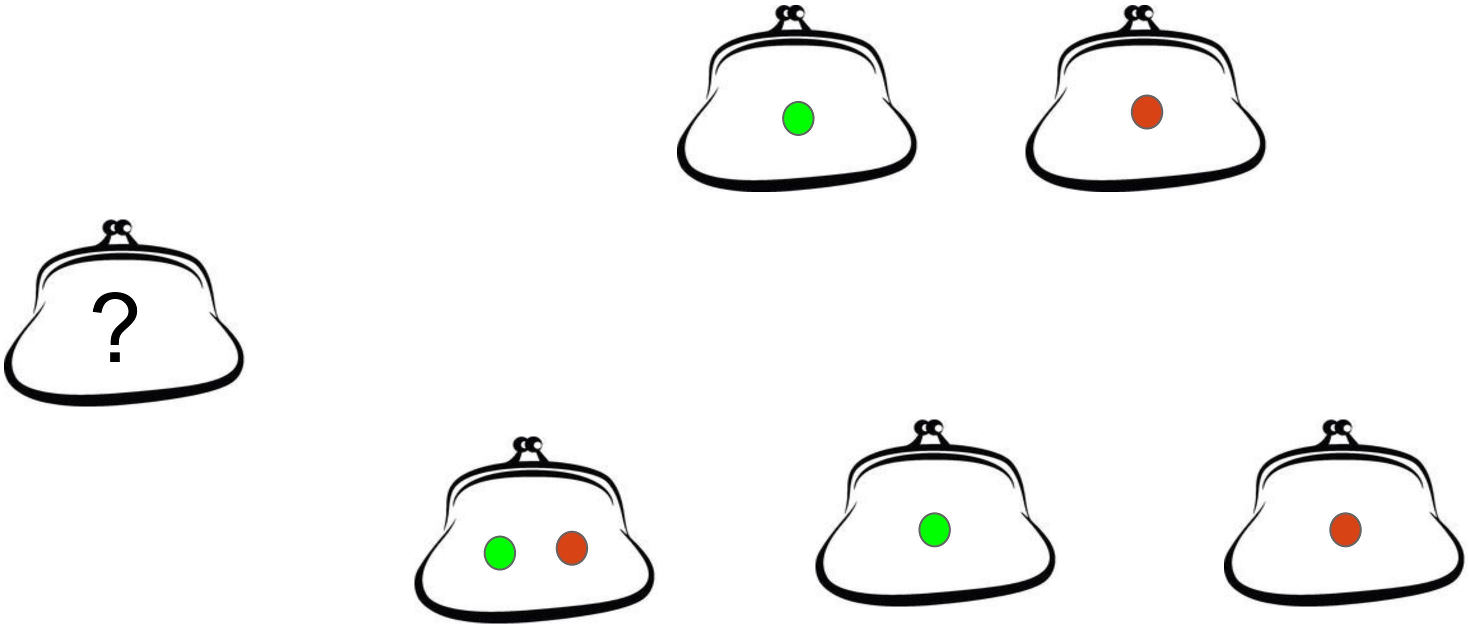
Binary Consensus Architecture



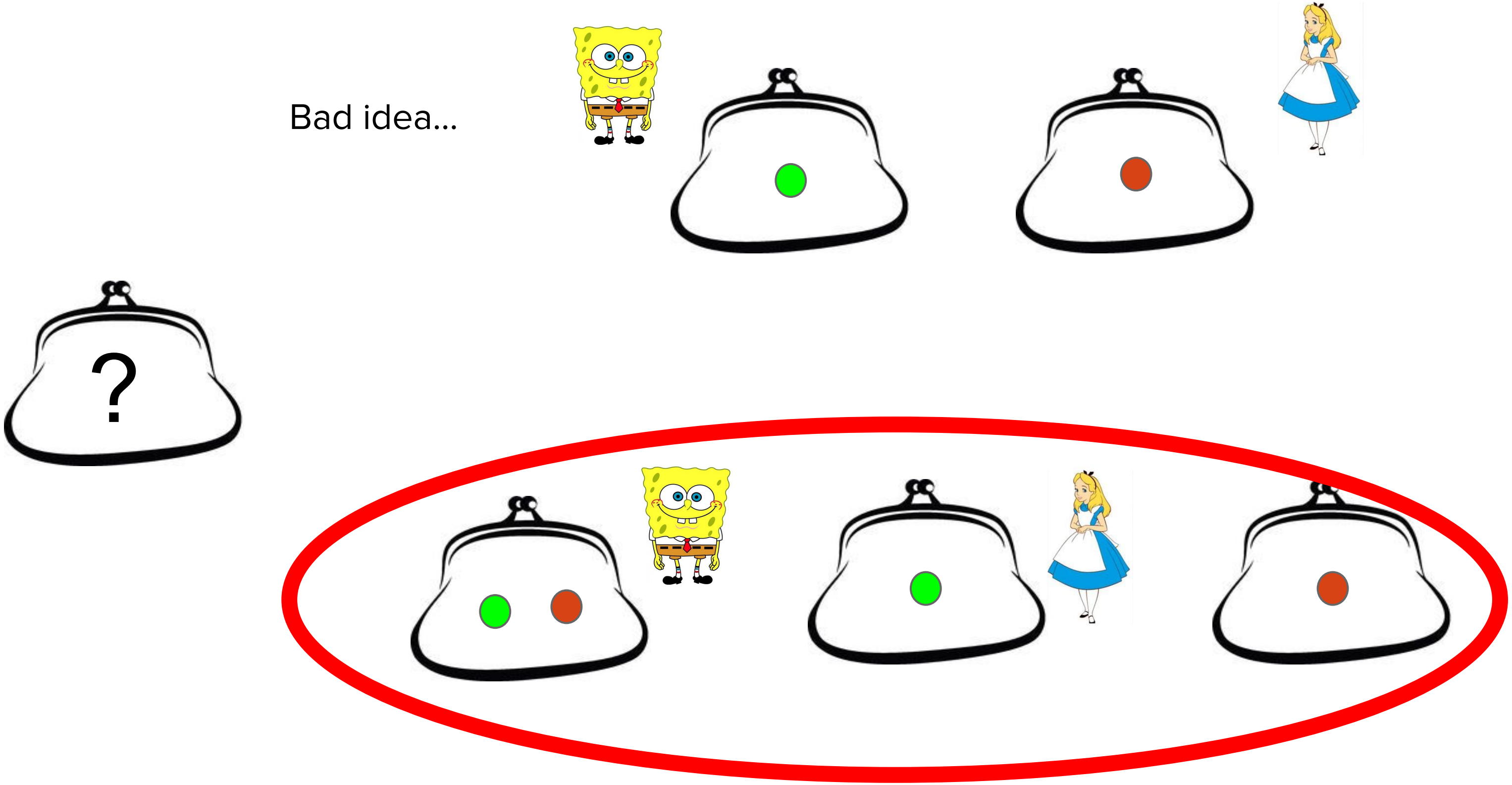
Binary Consensus Architecture



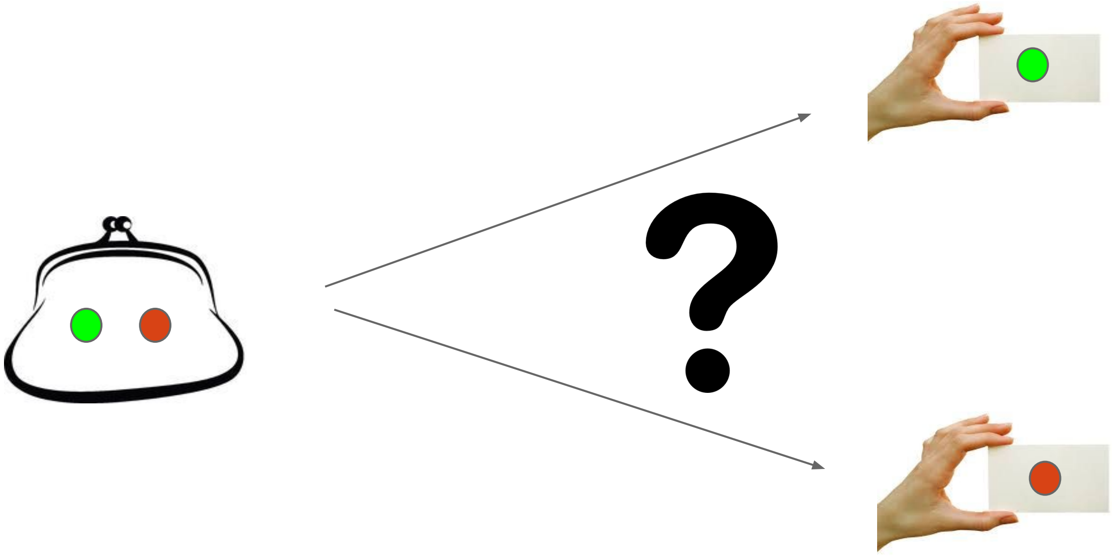
Which kind of output for discussion ?



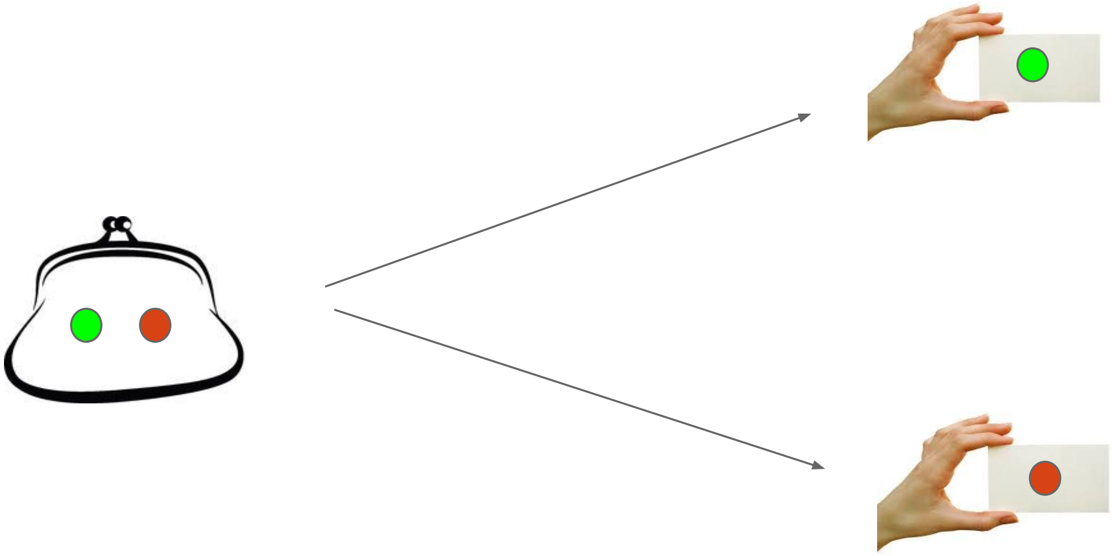
Which kind of output for discussion ?



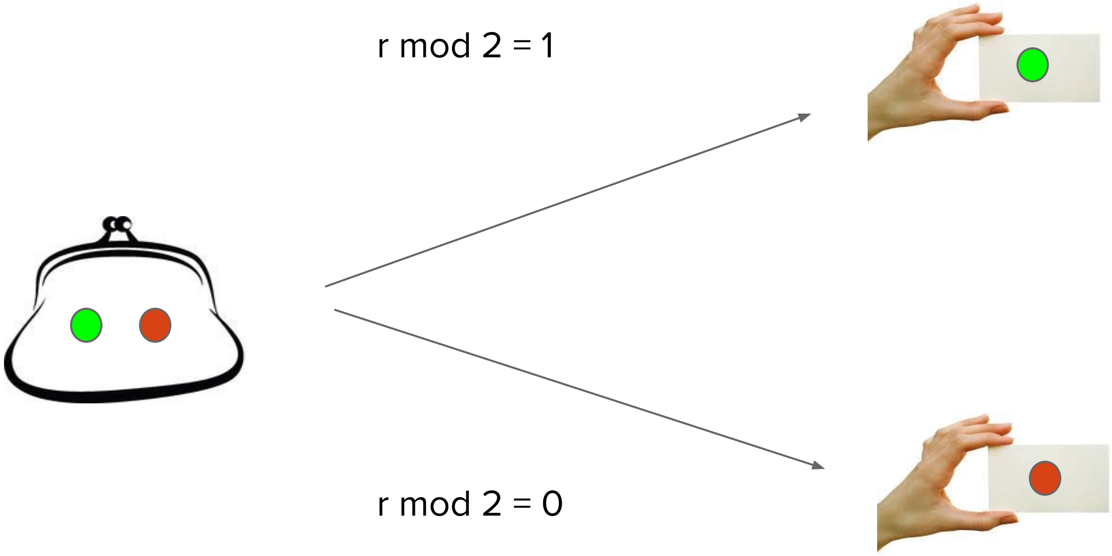
Which estimation in ambiguous case



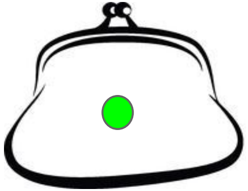
No default value for Validity



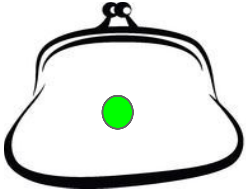
Common re-estimation



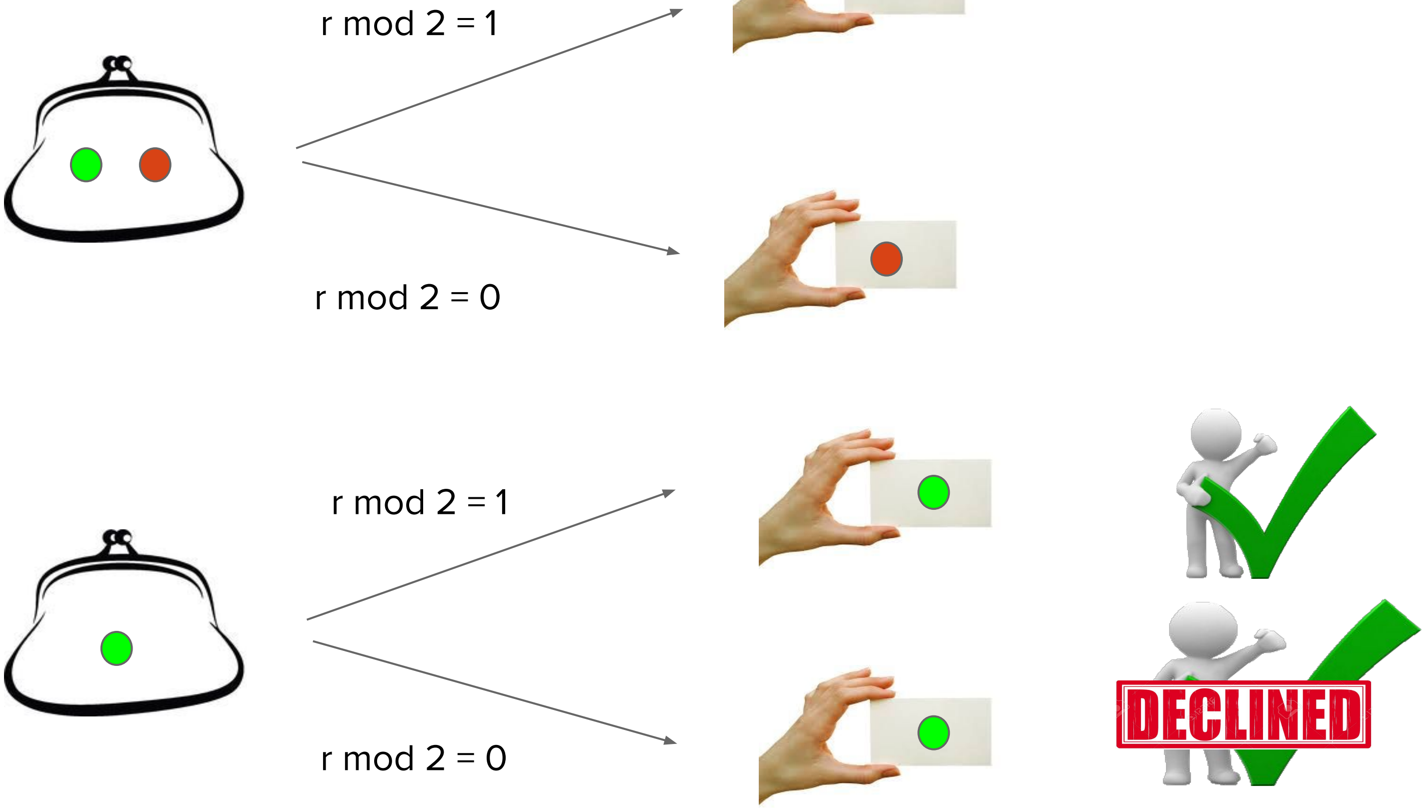
Can we decide when value is unique



Can we decide when value is unique

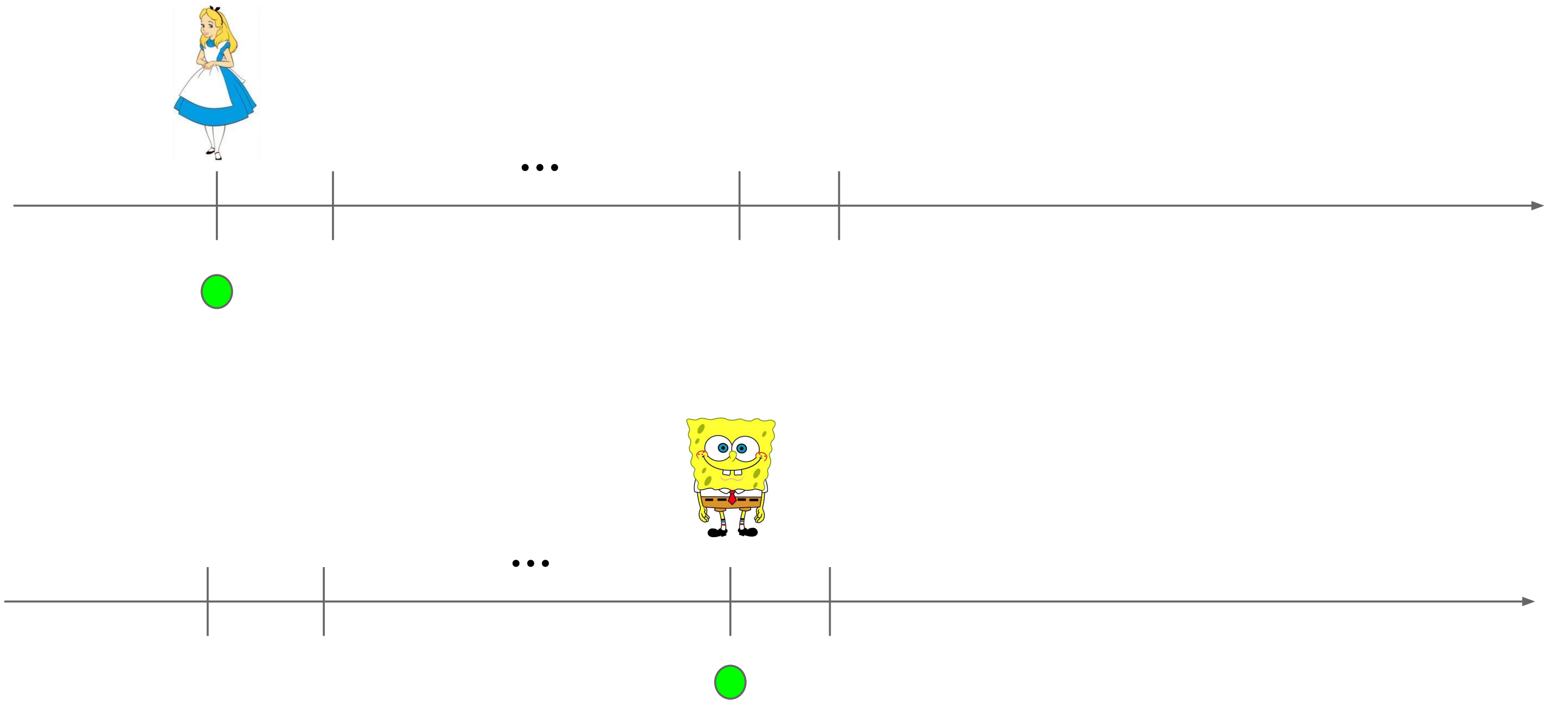


Common decision

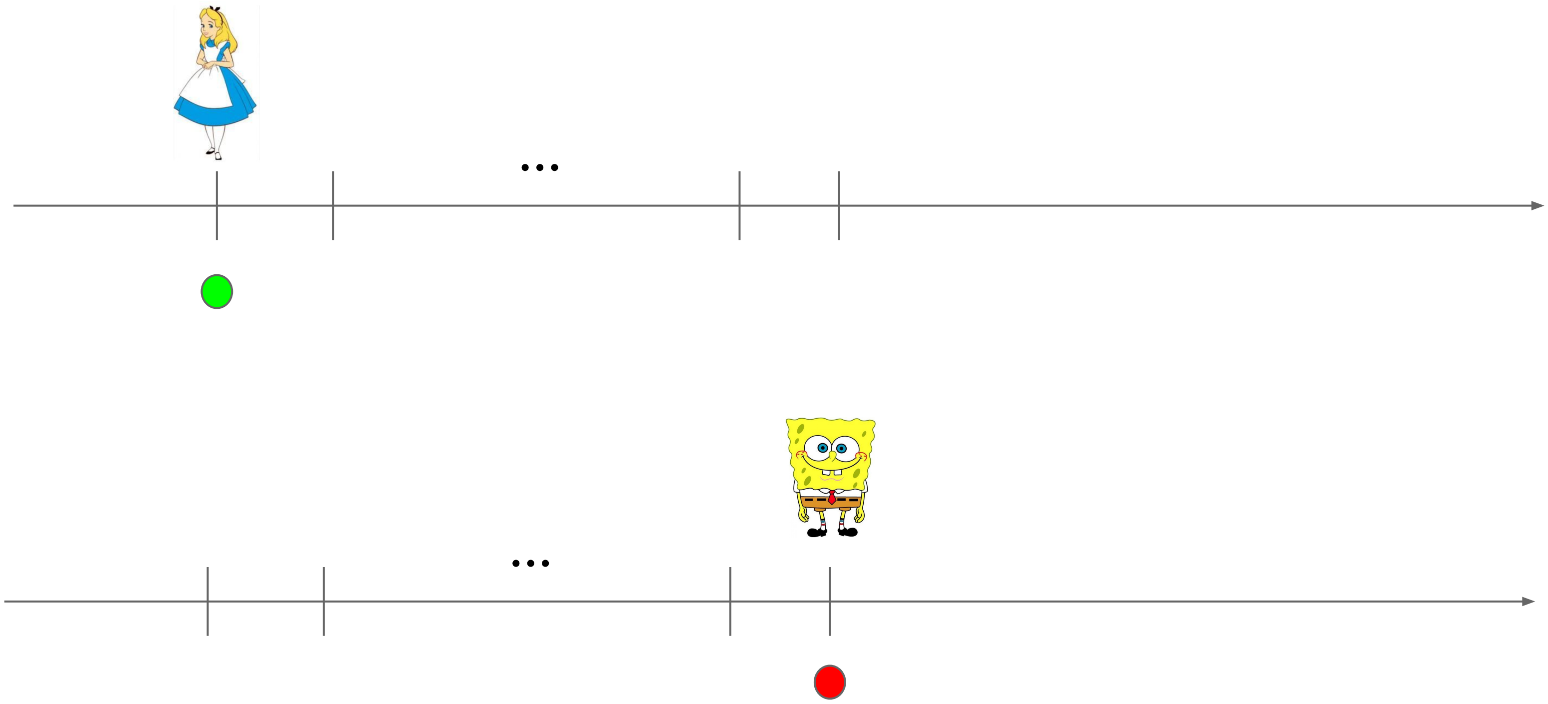


Decision follow the estimation of the other correct group

t<n/3



t>=n/3



Different round decision

Allow attack without mutant messages

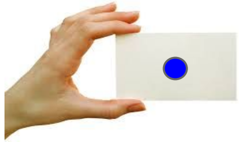
➜ naive forward inefficient (to many messages anyway)

➜ track commission can be non trivial

Extension of the algorithm

The original algorithm

- BV-broadcast (**estimate** value)

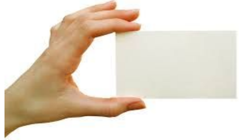


- build a set **bin_value**

- broadcast (bin_value)

- build a set **value**
- check the situation

- compute a new **estimate** value

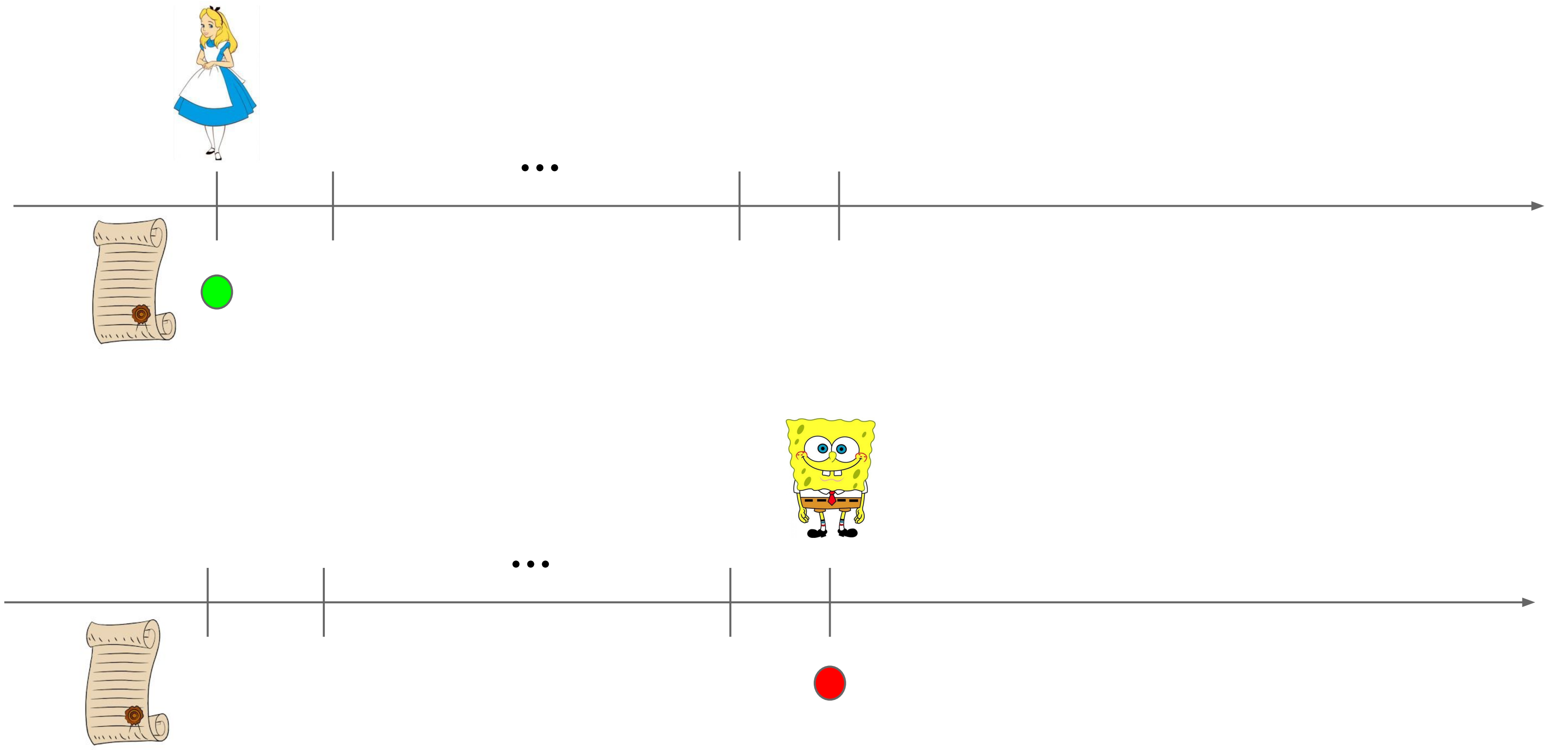


The extension of the algorithm

- **Signature** : Authentication, Integrity, Non-Repudiation

- **Certificate** : Justification of what we send, proof that we did not flip our value

t>=n/3

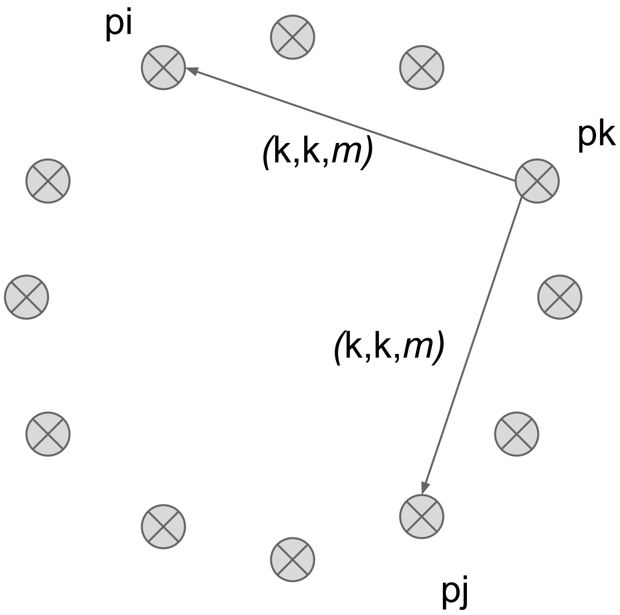


Naive forward

Everybody forwards what he received

naive forward : msg-complexity ++

round r, line l : l[r]



register(l[r],k)

(k,a,*m*)
...
(k,i,*m*)
...
(k,k,*m*)
...
(k,z,*m*)

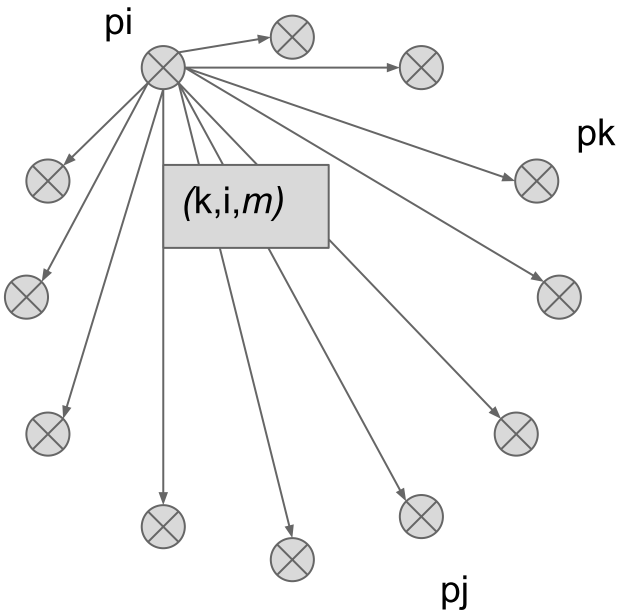
...

register(l[r],q)

(q,a,*m*)
...
(q,i,*m*)
...
(q,k,*m*)
...
(q,z,*m*)

forward_ledger_j(l[r])

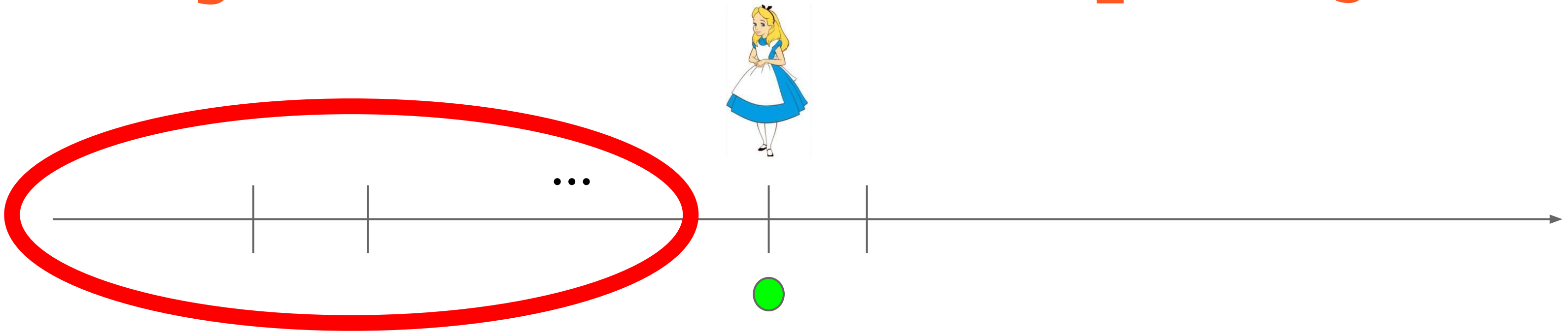
$O(n^2)$ messages



Naive justification

Everybody send all history

Naive justification : bit-complexity ++



Bounded Justification

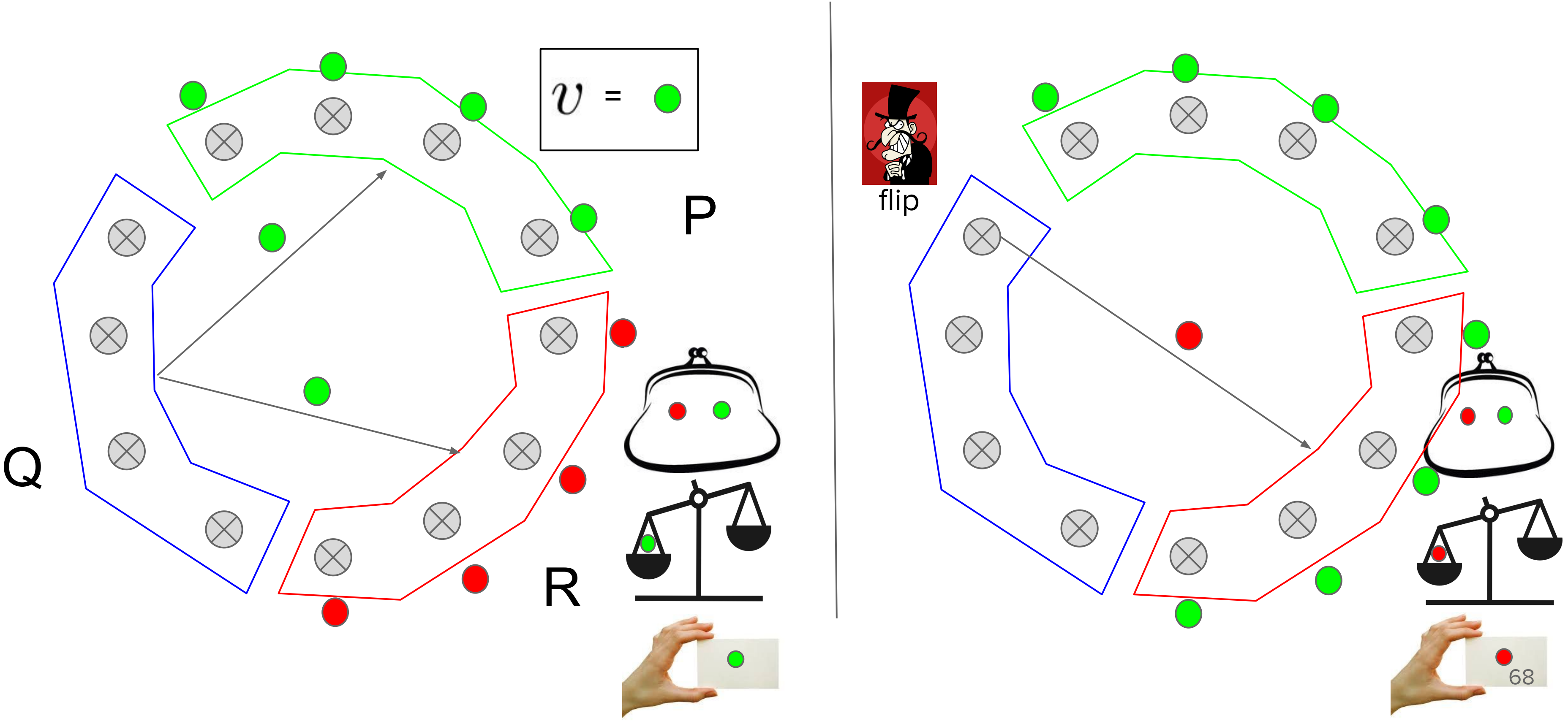
Certificate of a bounded part of the history

Bounded justification

Flip Attack

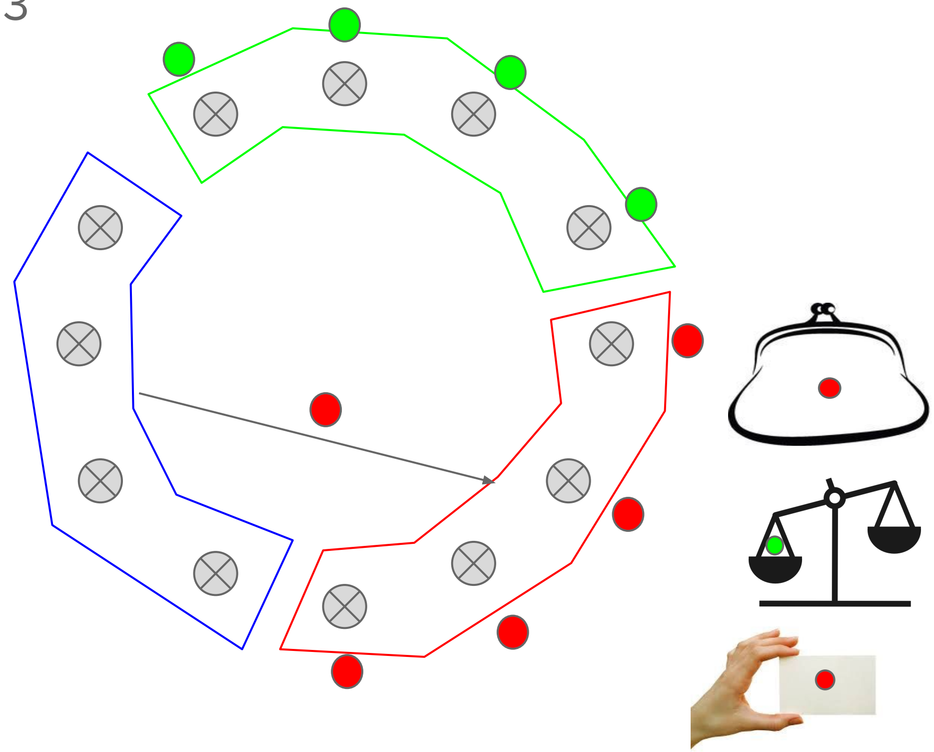
subvert the naive forward strategy

Flip attack

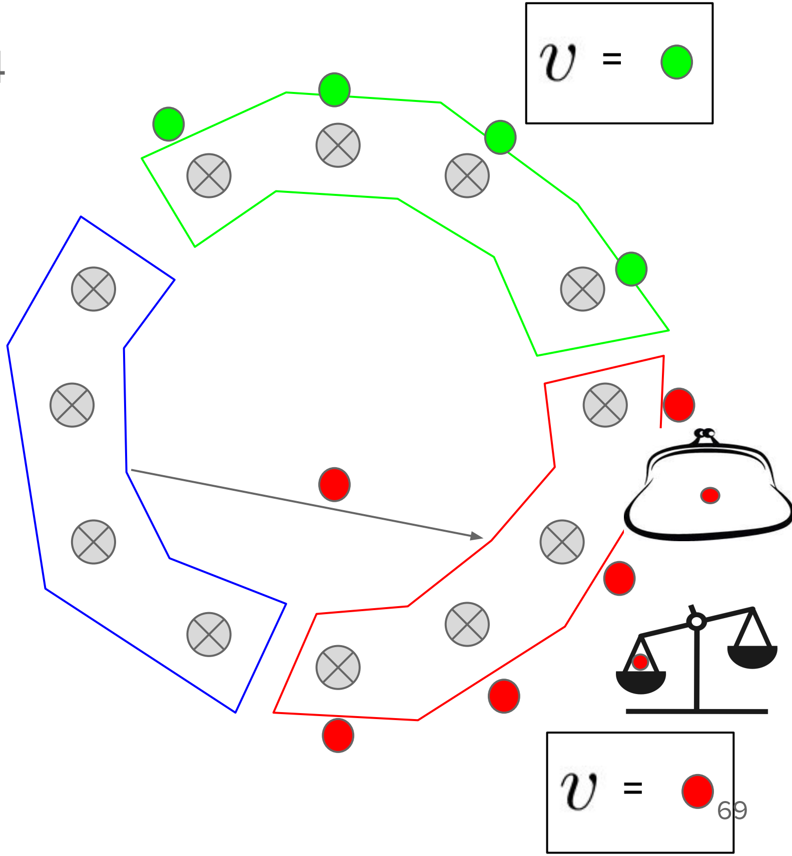


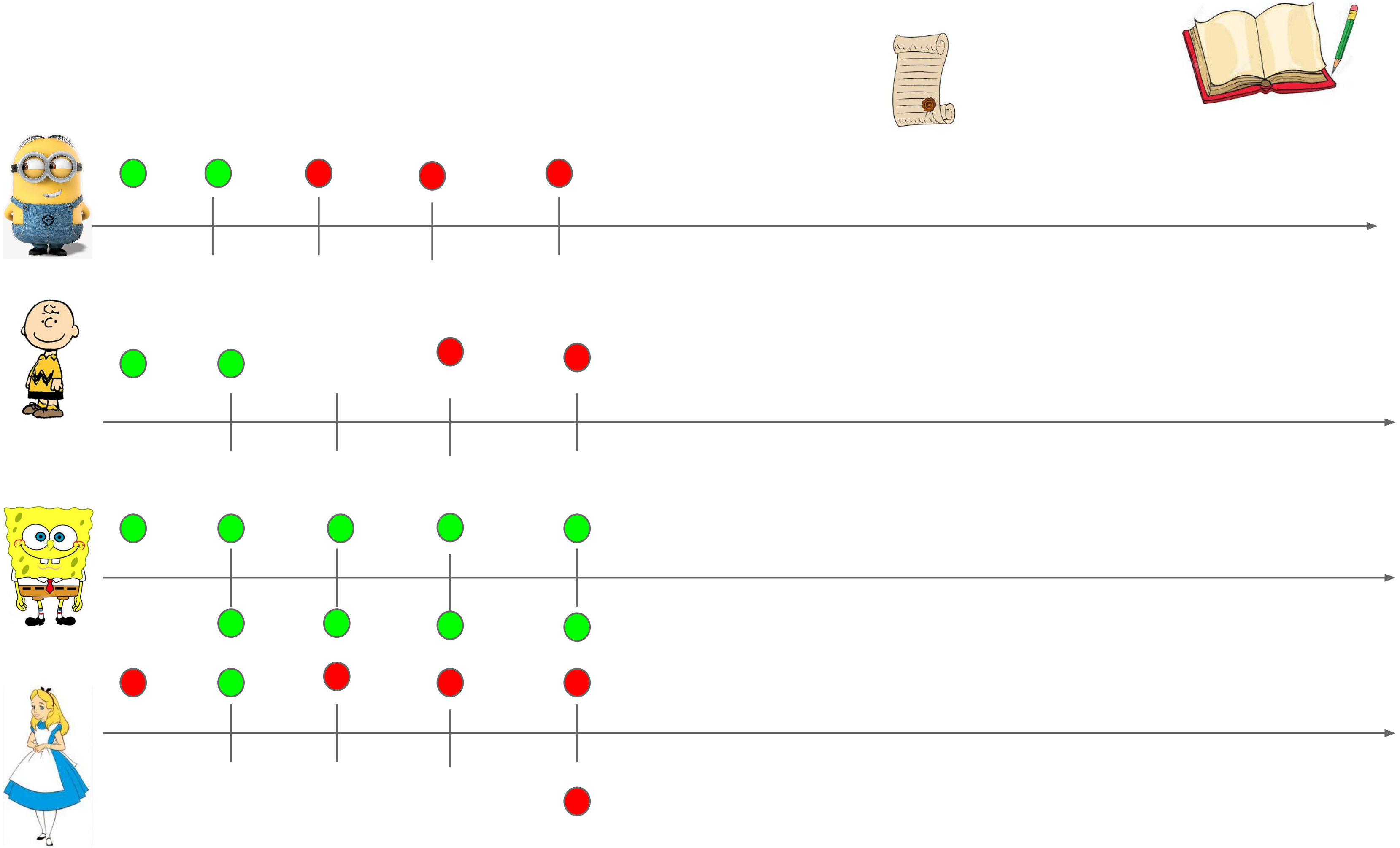
Flip attack

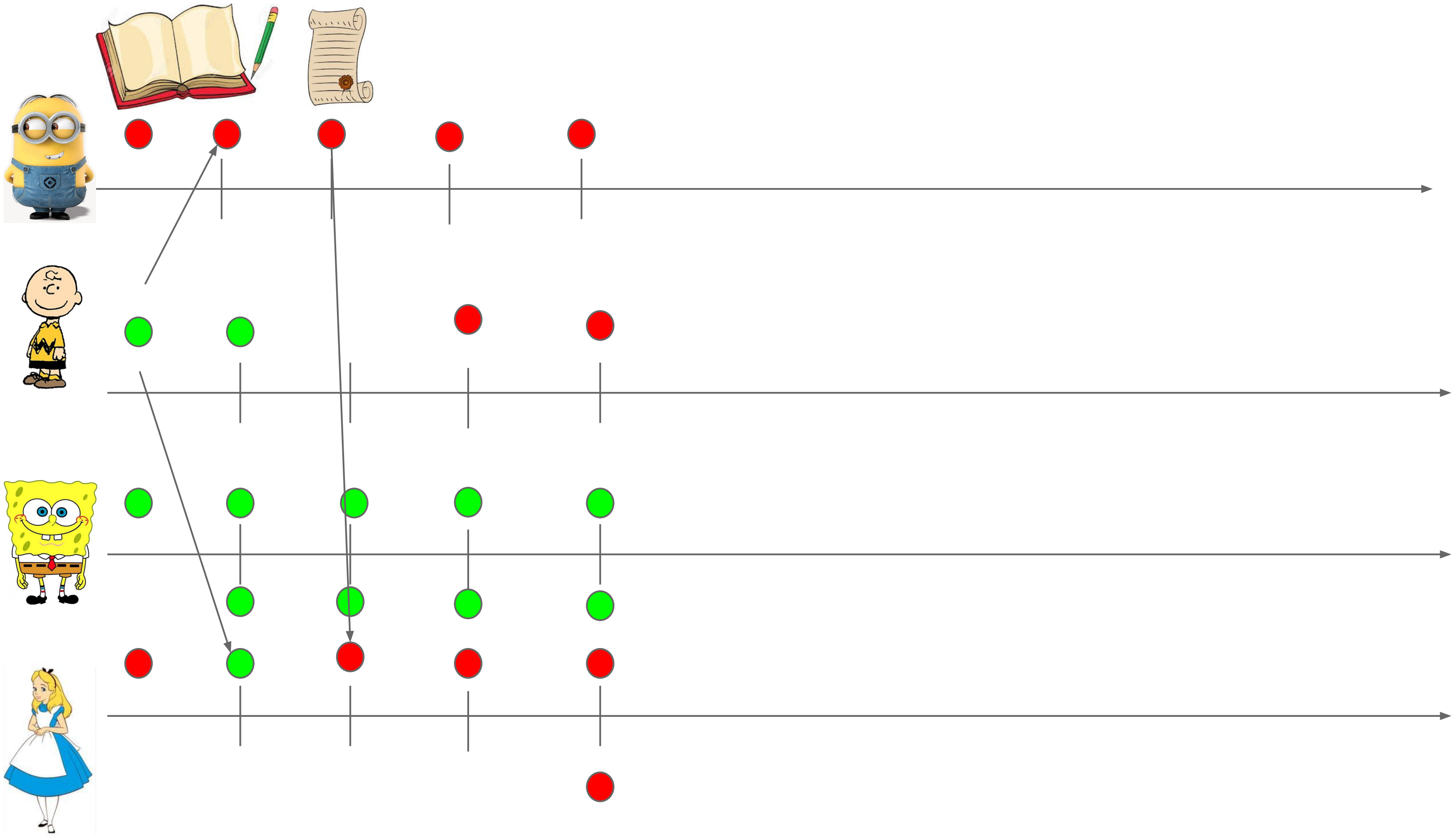
#3

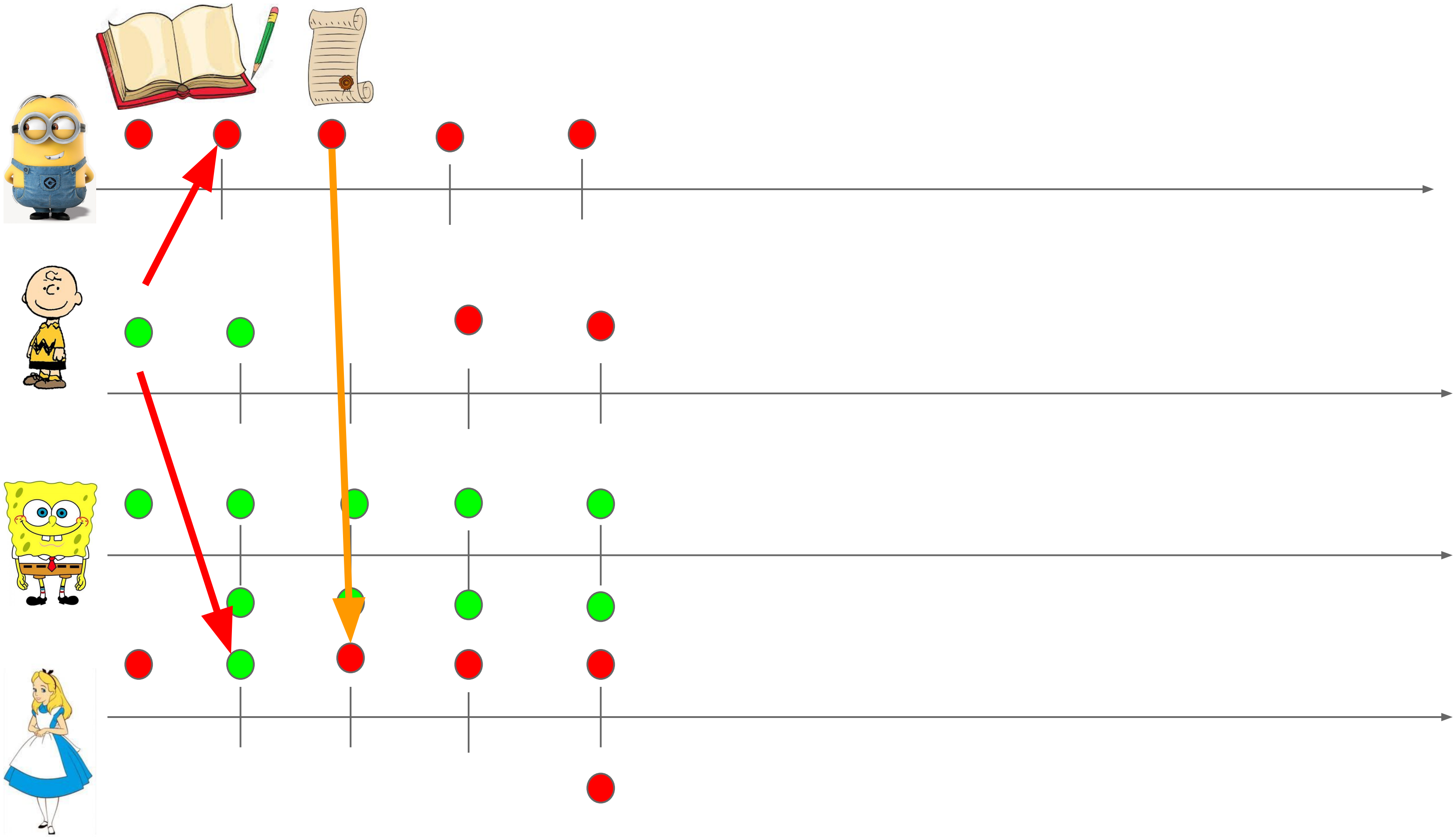


#4









Conclusion

Accountable Byzantine Consensus
With acceptable complexity

< n/3 → Consensus (Safety + Liveness)

→ Disagreement → Detection →

Open Question

Generic accountable transformation

Game theory extension

Noisy Environment (Weaker Adversary)

Complexity Optimization

Suspicion Forever (only put the hash of the justification) and in case of disagreement : challenge the owner of the conflicting message to compute a justification that match the hash)

Appendix

Detection in hindsight

Pay an additional cost only if a disagreement occurred

As Peer-Review (Haeberlen, Petr Kouznetsov, and Peter Druschel)

Detailed Solution

BV-Broadcast

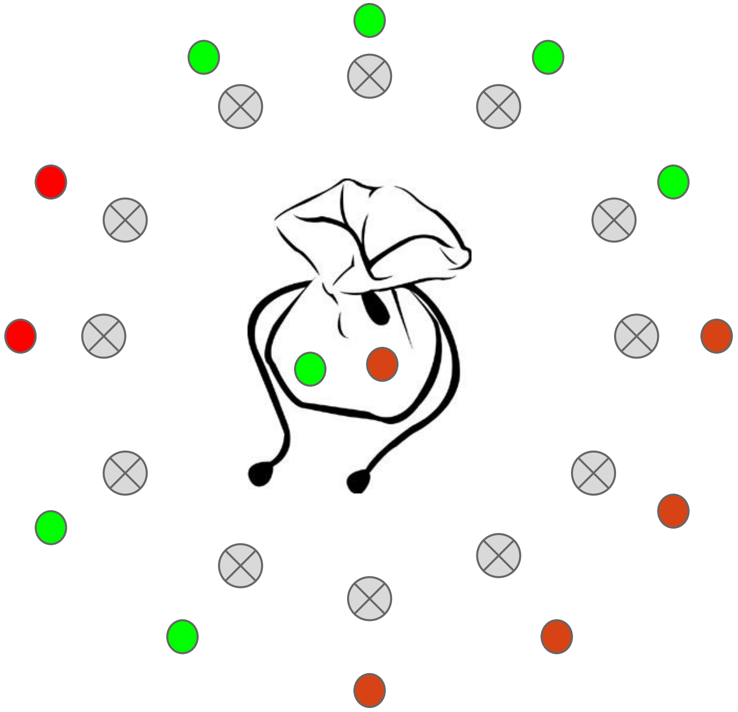
t < n/3 :

BV-Obligation

BV-Justification

BV-Uniformity

BV-Termination



forall t :

BV-Accountability

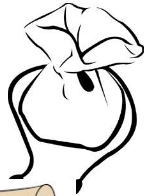
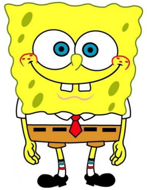
(**BV-Obligation**). If at least (t^0 + 1) correct processes BV-Broadcast the same value v, v is eventually added to the set bin_values_i , of each correct process pi.

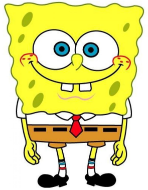
(**BV-Justification**). If pi is non-faulty and v ∈ bin_values_i , v has been BV-broadcast by a non-faulty process.

(**BV-Uniformity**). If a value v is added to the set bin_values i of a correct process p i , eventually v ∈ bin_values j at every non-faulty process pj.

(**BV-Termination**). Eventually, a set bin_values i of a correct process pi is not empty.

BV-Accountability

If belongs to  of , then

 has a valid  send by ,

justifying the posting of by .

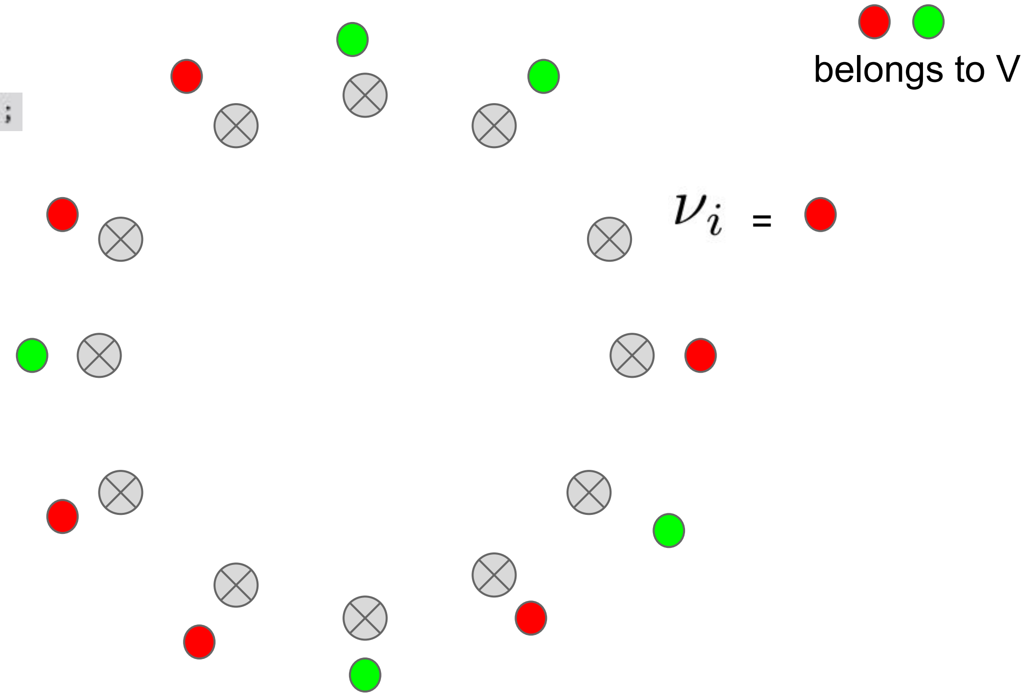
 captures the motivation of 

Binary Byzantine Consensus : CGLR17

operation bin_propose(v_i) **is**
(01) $est_i \leftarrow v_i$; $r_i \leftarrow 0$;
(02) **while** (true) **do**
(03) $r_i \leftarrow r_i + 1$;
(04) BV_Broadcast EST$[ri](est_i)$;
(05) **wait until** ($bin\_values_i[r_i] \neq \emptyset$);
(06) **broadcast** AUX$[r_i]$($bin\_values_i[r_i]$);
(07) **wait until** (messages AUX$[r_i]$($b\_val_{p(1)}$), ..., AUX$[r_i]$($b\_val_{p(n-t_0)}$)
have been received from $(n - t_0)$ different process
$p(x), 1 \leq x \leq n - t_0$, and their contents are such that
$\exists$ a non-empty set $values_i$ such that
(i) $values_i \subseteq bin\_values_i[r_i]$ and
(ii) $values_i = \cup_{1 \leq x \leq n-t_0} b\_val_x$);
(08) $b_i \leftarrow r_i \; mod \; 2$;
(09) **if** $values_i = \{v\}$
(10) **then** $est_i \leftarrow v$; **if** ($v = b_i$) **then** decide(v) **if not yet done end if**;
(11) **else** $est_i \leftarrow b_i$
(12) **end if** ;
(13) **end while** ;

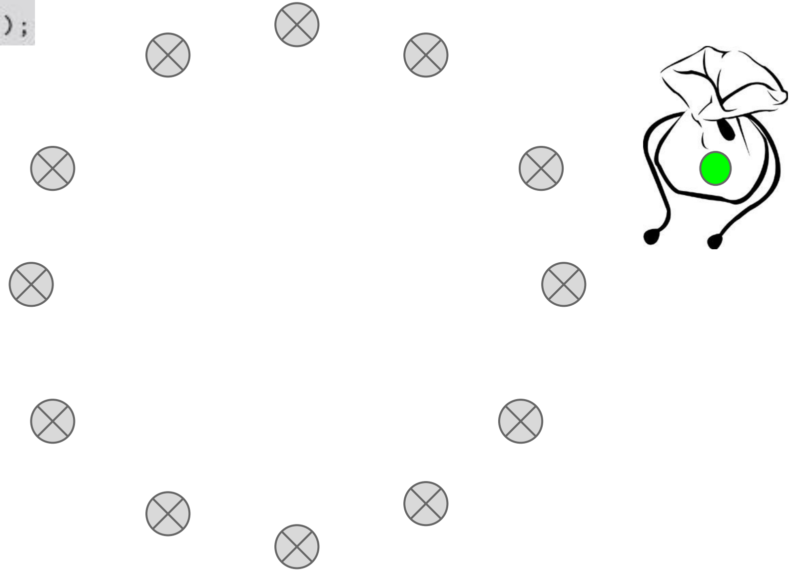
Initial value

```
(04)    BV_Broadcast EST[ri](est_i);
```

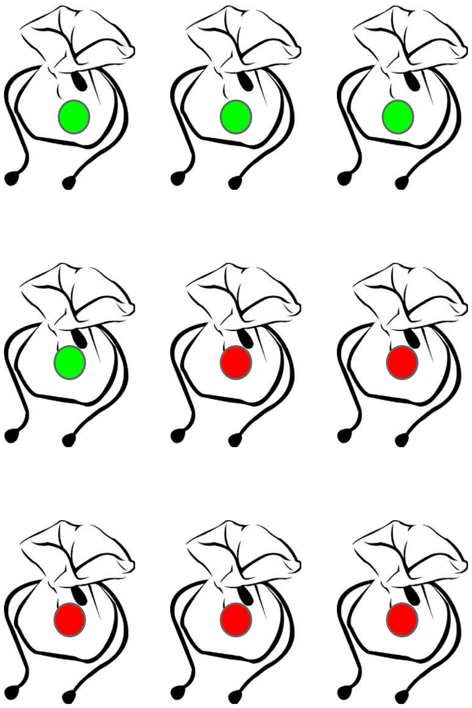


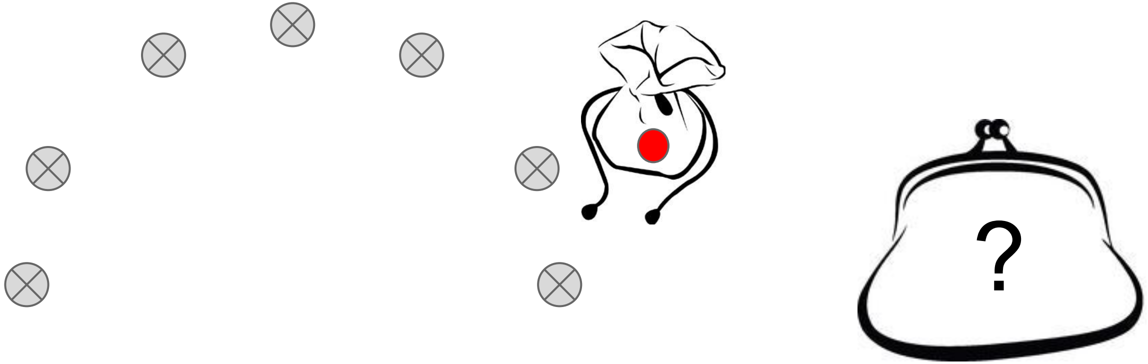
bin values is no more empty

wait until ($bin\_values_i[r_i] \neq \emptyset$);
broadcast AUX[r_i]($bin\_values_i[r_i]$);



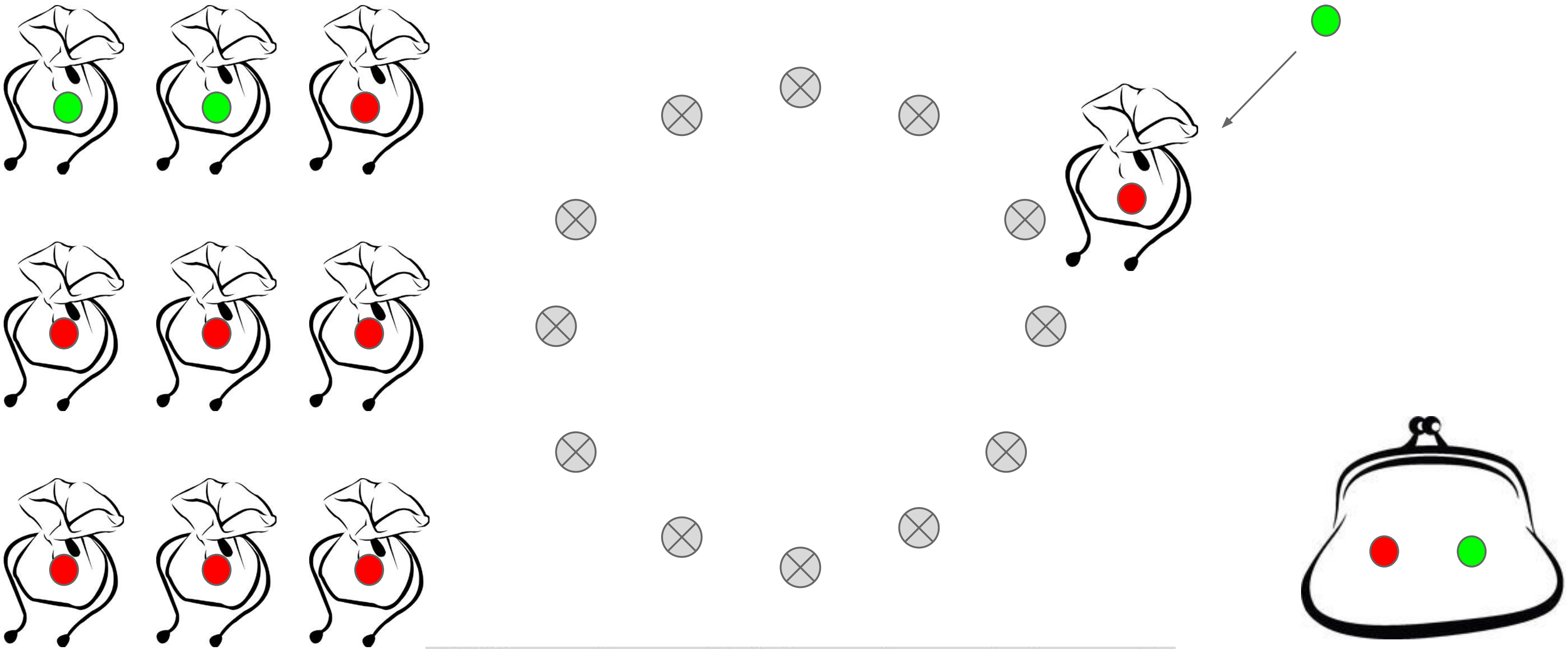
build values





(07) wait until (messages AUX[r_i]($b\_val_{p(1)}$), ..., AUX[r_i]($b\_val_{p(n-t_0)}$)
have been received from $(n - t_0)$ different process
$p(x), 1 \leq x \leq n - t_0$, and their contents are such that
$\exists$ a non-empty set $values_i$ such that
(i) $values_i \subseteq bin\_values_i[r_i]$ and
(ii) $values_i = \cup_{1 \leq x \leq n-t_0} b\_val_x$);

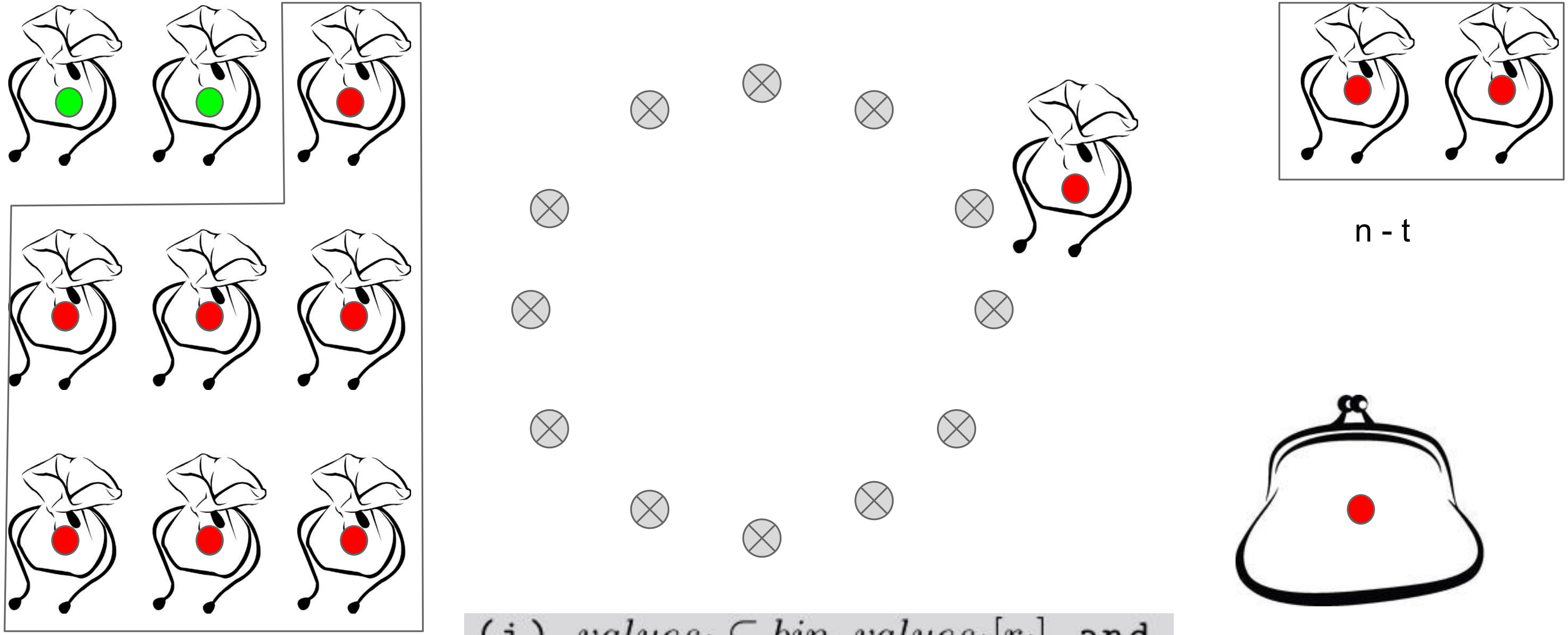
scenario 1



(i) $values_i \subseteq bin\_values_i[r_i]$ and

(ii) $values_i = \cup_{1 \leq x \leq n-t_0} b\_val_x$)

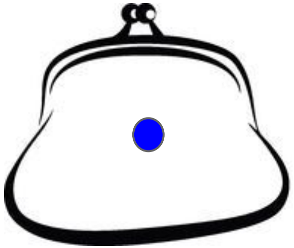
scenario 2

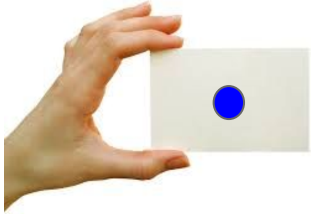


(i) $values_i \subseteq bin\_values_i[r_i]$ **and**

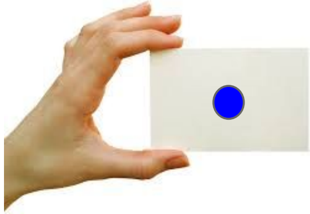
(ii) $values_i = \cup_{1 \leq x \leq n-t_0} b\_val_x$)

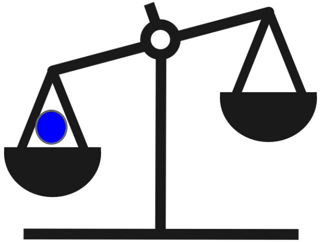
Estimation : case singleton

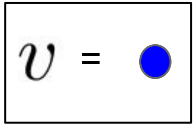




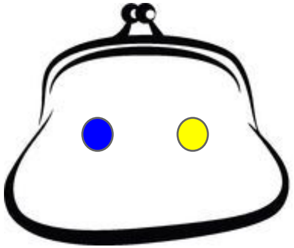
```
b_i ← r_i mod 2;
if values_i = {v}
    then est_i ← v; if (v = b_i) then decide(v) if not yet done end if
    else est_i ← b_i
end if ;
```

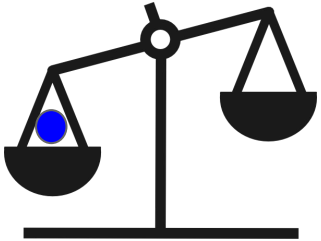


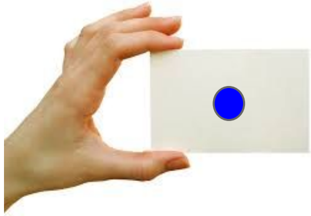




Estimation : case couple







```
b_i ← r_i mod 2;
if values_i = {v}
    then est_i ← v; if (v = b_i) then decide(v) if not yet done end if
    else est_i ← b_i
end if ;
```

Binary Byzantine Consensus : CGLR17

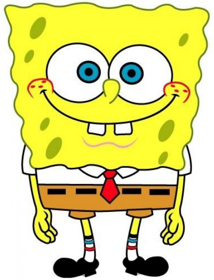
operation bin_propose(v_i) is
(01) $est_i \leftarrow v_i$; $r_i \leftarrow 0$;
(02) while (true) do
(03) $r_i \leftarrow r_i + 1$;
(04) BV_Broadcast EST$[ri](est_i)$;
(05) wait until $(bin\_values_i[r_i] \neq \emptyset)$;
(06) broadcast AUX$[r_i]$$(bin_values_i[r_i])$;
(07) wait until (messages AUX$[r_i]$$(b_val_{p(1)})$, ..., AUX$[r_i]$$(b_val_{p(n-t_0)})$
have been received from $(n - t_0)$ different process
$p(x), 1 \leq x \leq n - t_0$, and their contents are such that
$\exists$ a non-empty set $values_i$ such that
(i) $values_i \subseteq bin\_values_i[r_i]$ and
(ii) $values_i = \cup_{1 \leq x \leq n - t_0} b\_val_x$);
(08) $b_i \leftarrow r_i \bmod 2$;
(09) if $values_i = \{v\}$
(10) then $est_i \leftarrow v$; if $(v = b_i)$ then decide(v) if not yet done end if;
(11) else $est_i \leftarrow b_i$
(12) end if ;
(13) end while ;

The detection of the malicious coalition

What put on the attached certificate ?

The Characters

Alice



Bob

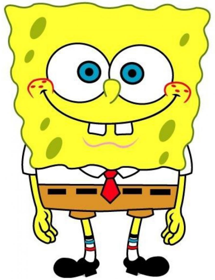
Charlie

Donald

The Inquiry

Certificate

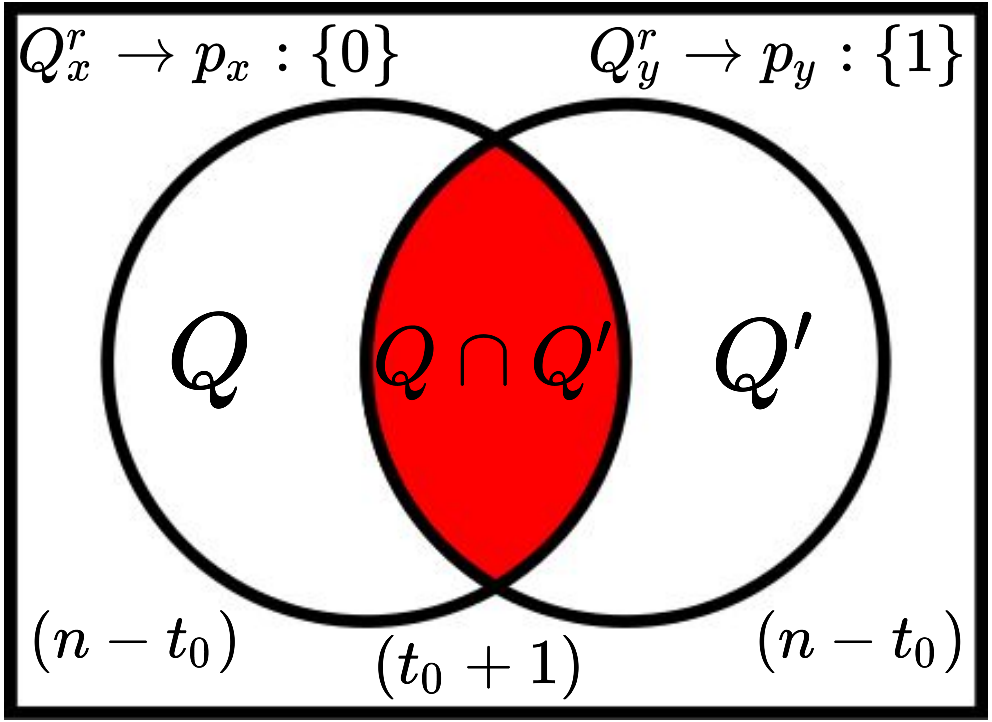
$$Q_A^{r_A} \to A : \{v\}$$



Certificate

$$Q_B^{r_B} \to B : \{w\}$$

Guilty processes



Detection

operation Culpability_Detection$(v_j, r_j, Q_j^{r_j})$

$dec_j = S_j(j, v_j, r_j, \underline{Q_j^{r_j}})$ \\$(\overline{v_j} = r_j \ mod\ 2) \cap (\forall m \in \underline{Q_j^{r_j}},\ m.value = v_j)$

broadcast(dec_j)

when dec_i is received from p_i

    if $(v_i \neq r_i \ mod\ 2) \cup (\exists m \in \underline{Q_i^{r_i}} \mid m.value \neq v_i)$

        $G_{new} \leftarrow G_{old} \cup p_i$

        $proofs \leftarrow proofs \cup dec_i$

        exit

    end if

    if$(v_i \neq v_j) \cap (r_i < r_j)$

        $inquiry_i \leftarrow 1$

    end if

when $inquiry_i = 1$

    if $r_j - r_i = 1$

        pick m from $T1\_list_j^{r_i+1}$ sent from $p_k \mid m.value = v_j$

        pick $\underline{Q_k^{r_i}}$ from m

            \\m exists because of BV-Accountability

        $G_{new} \leftarrow G_{old} \cup (Q_i^{r_i} \cap Q_k^{r_i})$

        $proofs \leftarrow proofs \cup [\underline{Q_i^{r_i}}, \underline{Q_k^{r_i}}]$

        exit

    else

        pick m from $T1\_list_j^{r_i+2}$ sent from $p_k \mid m.value = v_j$

        pick $\underline{Q_q^{r_i}}$ from m

            \\m exists because of BV-Accountability

        $G_{new} \leftarrow G_{old} \cup (Q_i^{r_i} \cap Q_q^{r_i})$

        $proofs \leftarrow proofs \cup [\underline{Q_i^{r_i}}, \underline{Q_q^{r_i}}]$

        exit

    end if

when $G_{old} \neq G_{new}$

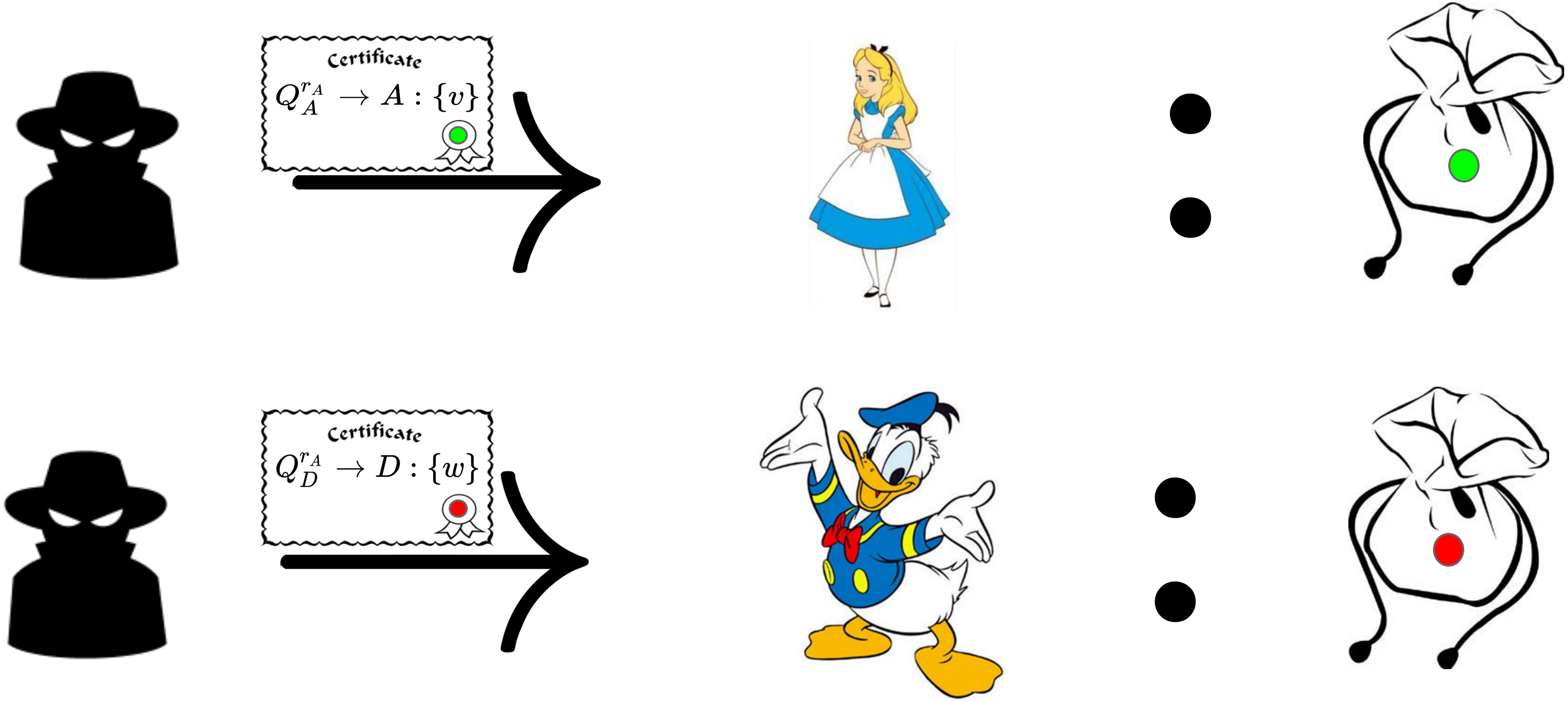
    broadcast$(G_{new},\ proofs)$

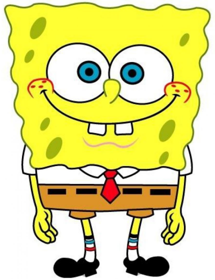
Detection

	A	B	C	D
r	$Q_A^r \xrightarrow{6} A : \{v\}$,	$values_B^r = \{0, 1\}$		$Q_D^r \xrightarrow{6} D : \{w\}$
$r+1$		$values_B^{r+1} = \{v\}$	$D \xrightarrow{4} C : w$ & $(\underline{Q_D}^r, \bullet)$	
$r+2$		$values_B^{r+2} \neq \{v\}$ $C \xrightarrow{4} B : w$ & $(\underline{Q_C}^{r+1}, \underline{Q_D}^r)$		

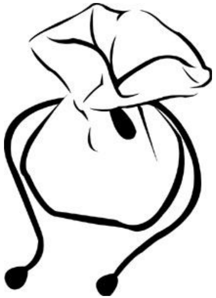
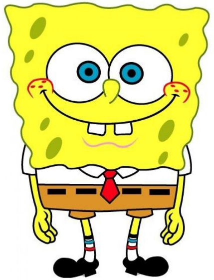
Evidence

At the same round r



- $\in$  of 

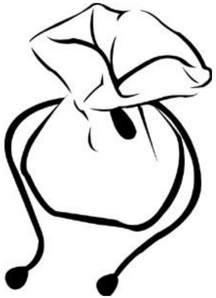
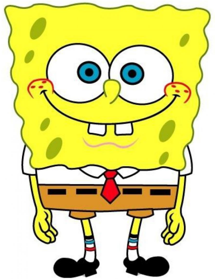
$(r_A + 2)$

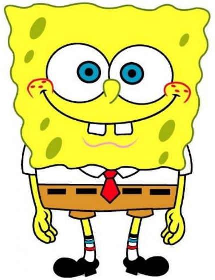
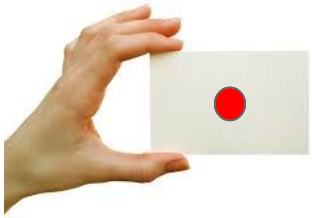
- $\in$  of 

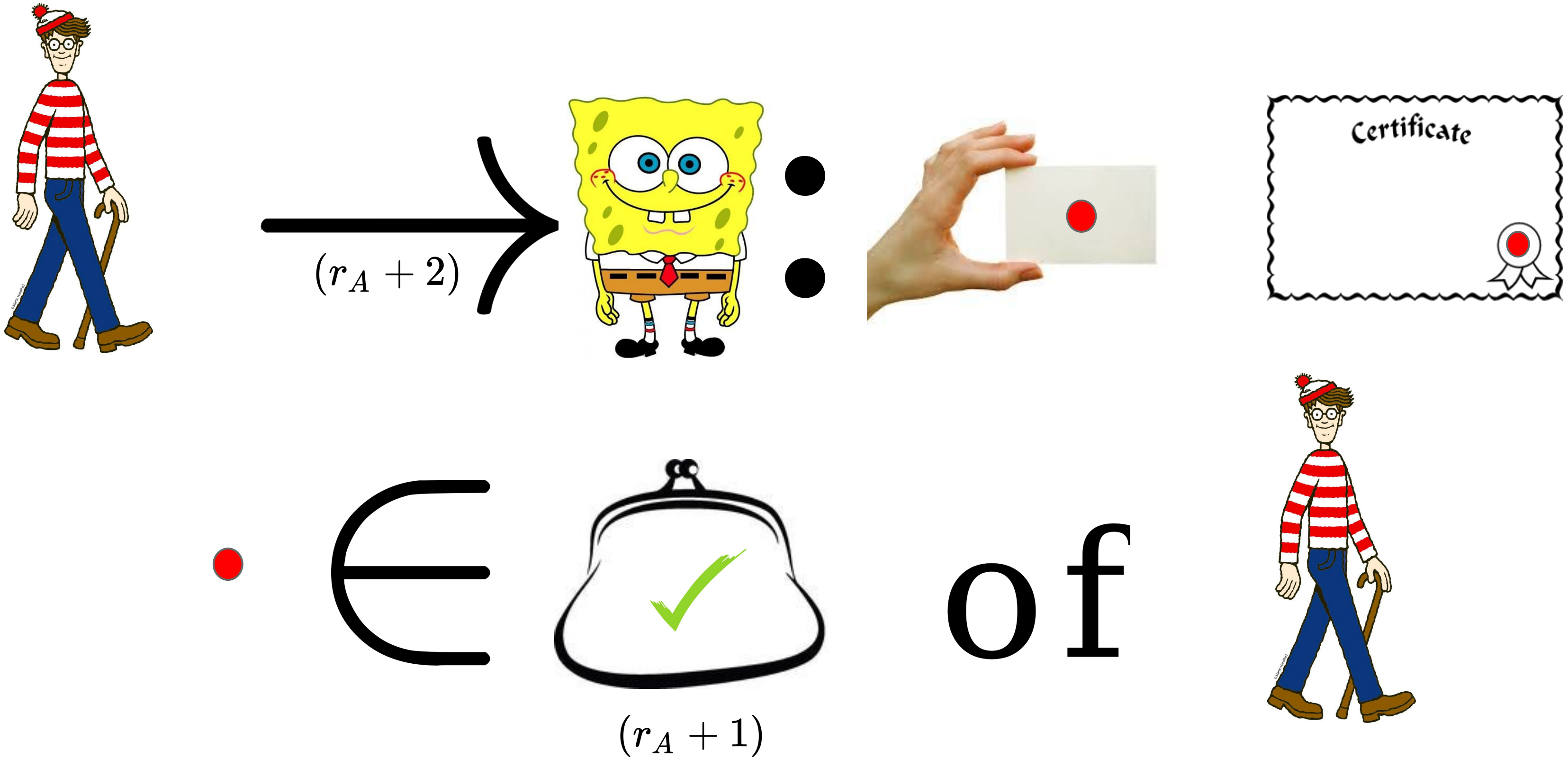
$(r_A + 2)$

(i) $values_i \subseteq bin\_values_i[r_i]$ and

(ii) $values_i = \cup_{1 \leq x \leq n - t_0} b\_val_x$)

 $\in$  $(r_A + 2)$ of 

 $\xrightarrow[(r_A + 2)]{}$  :  


$(r_A + 2)$
Certificate
$\in$
of
$(r_A + 1)$

- $\in$  of 

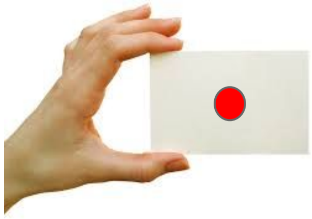
  $(r_A + 1)$

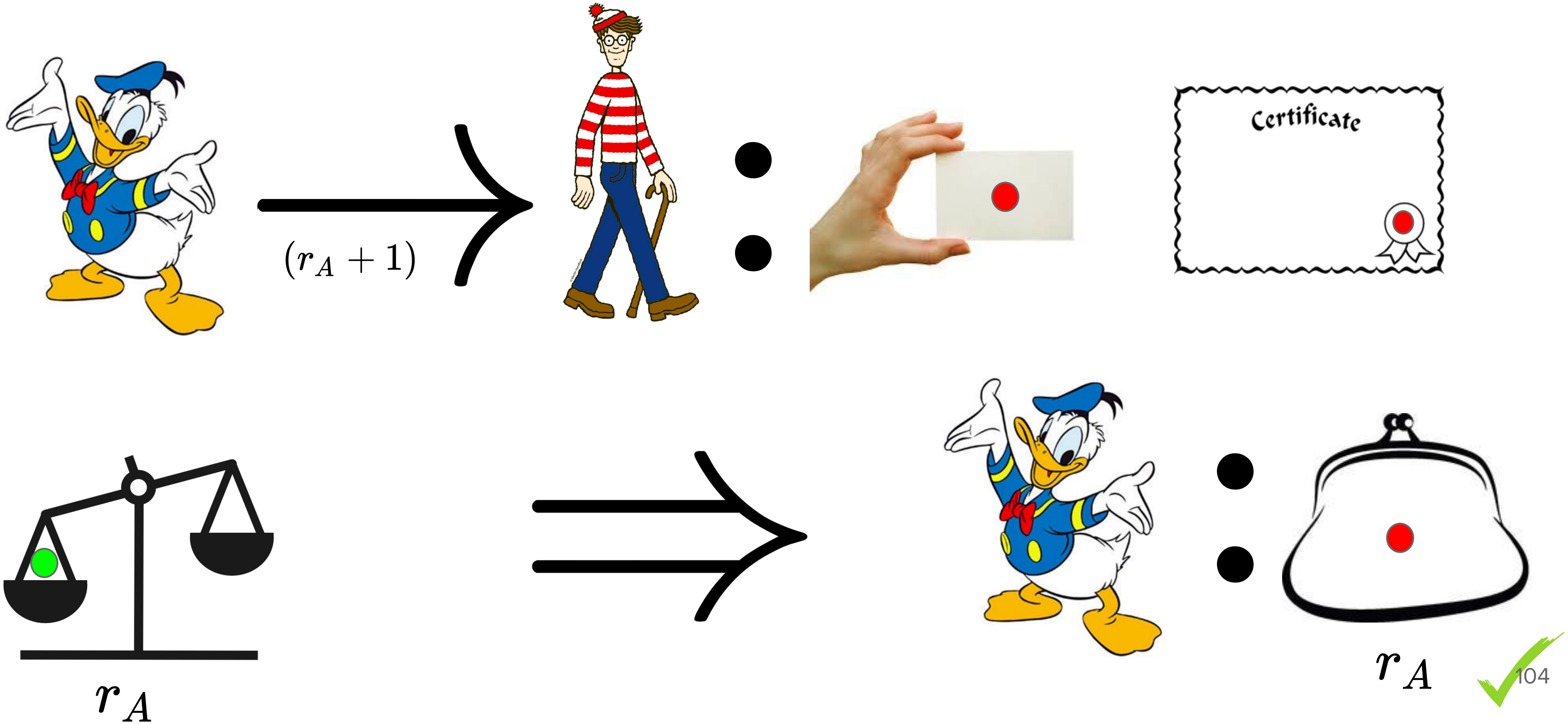
- $\in$  of 

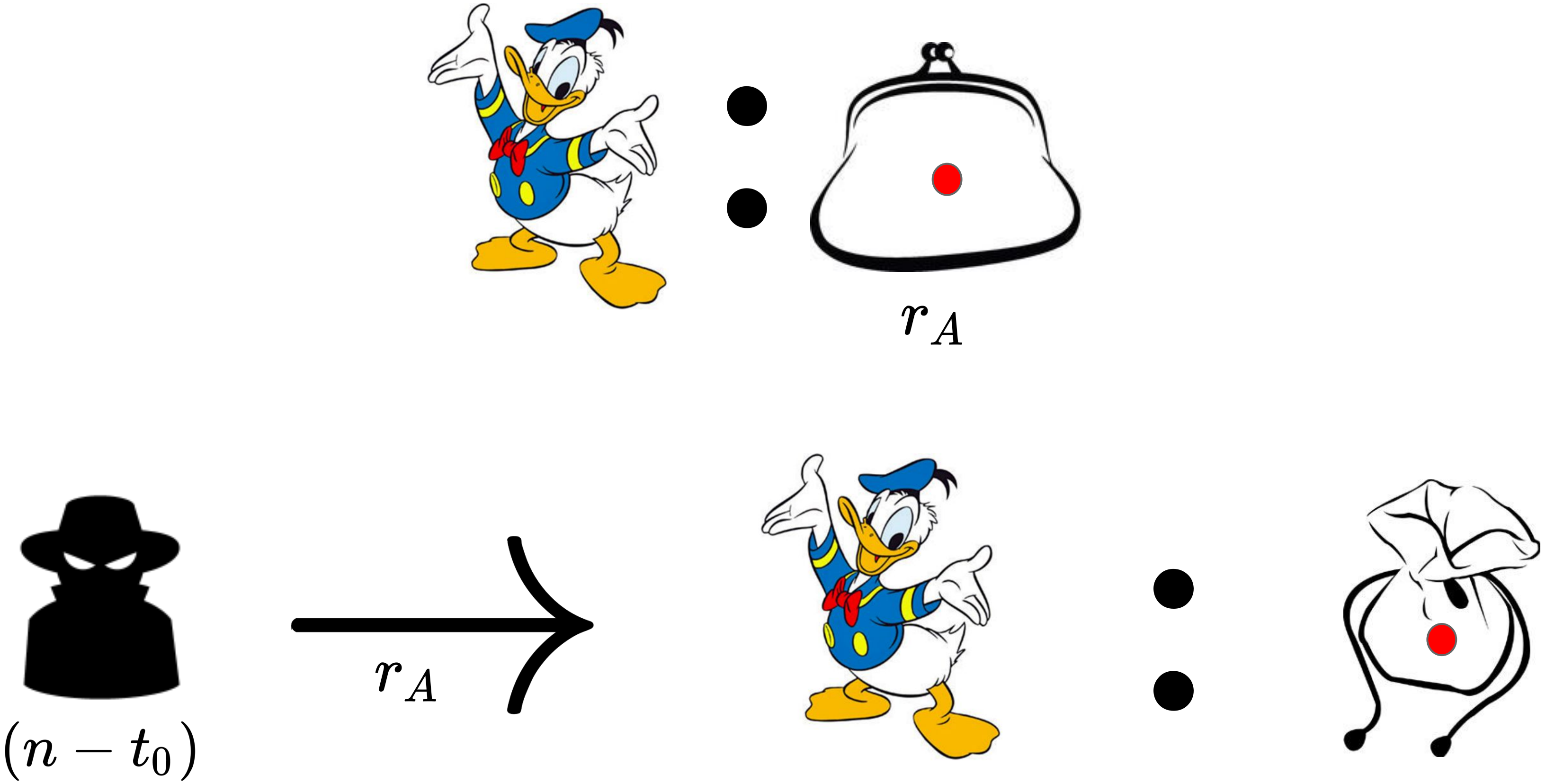
  $(r_A + 1)$

 $\in$  of 

$(r_A + 1)$

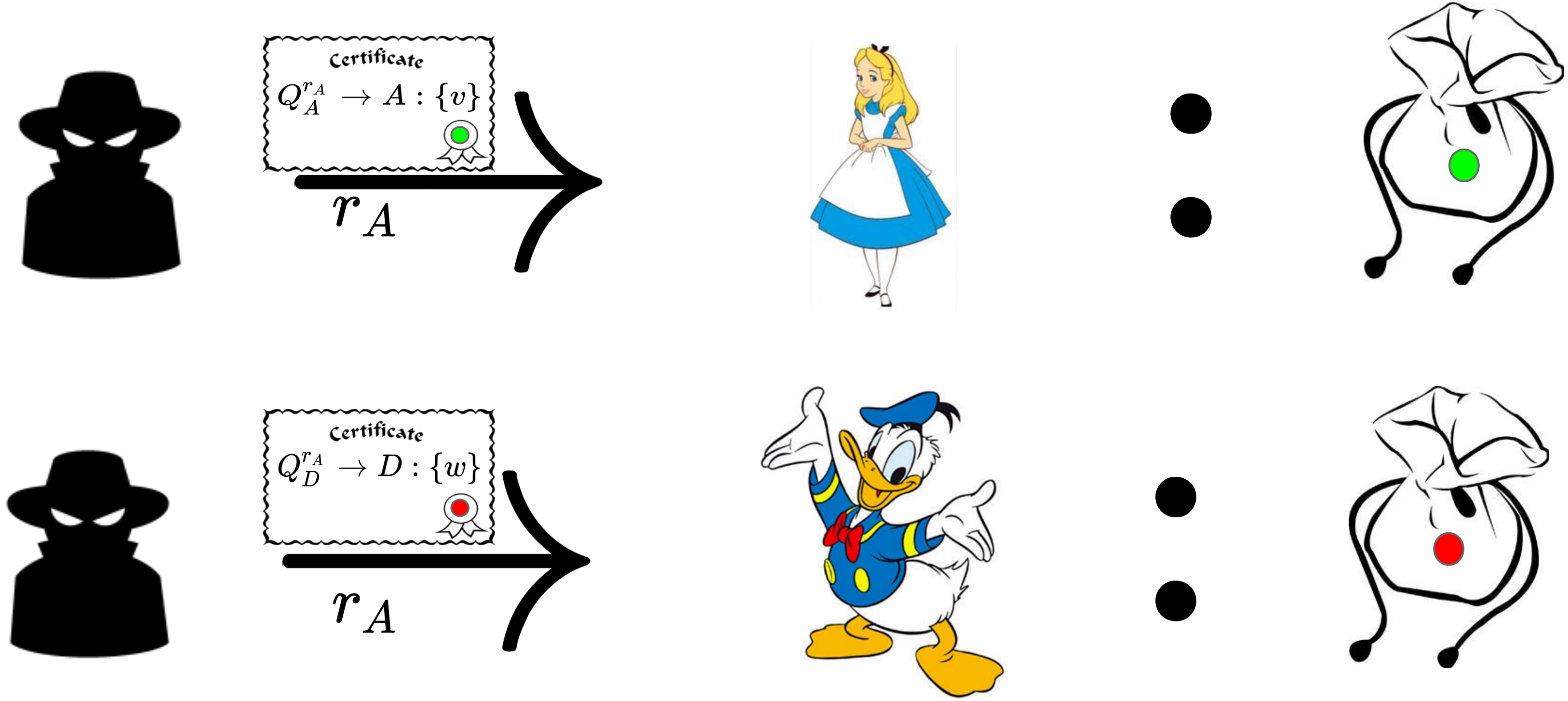
 $\xrightarrow[(r_A + 1)]{}$  :  


$(r_A + 1)$
Certificate
r_A
r_A

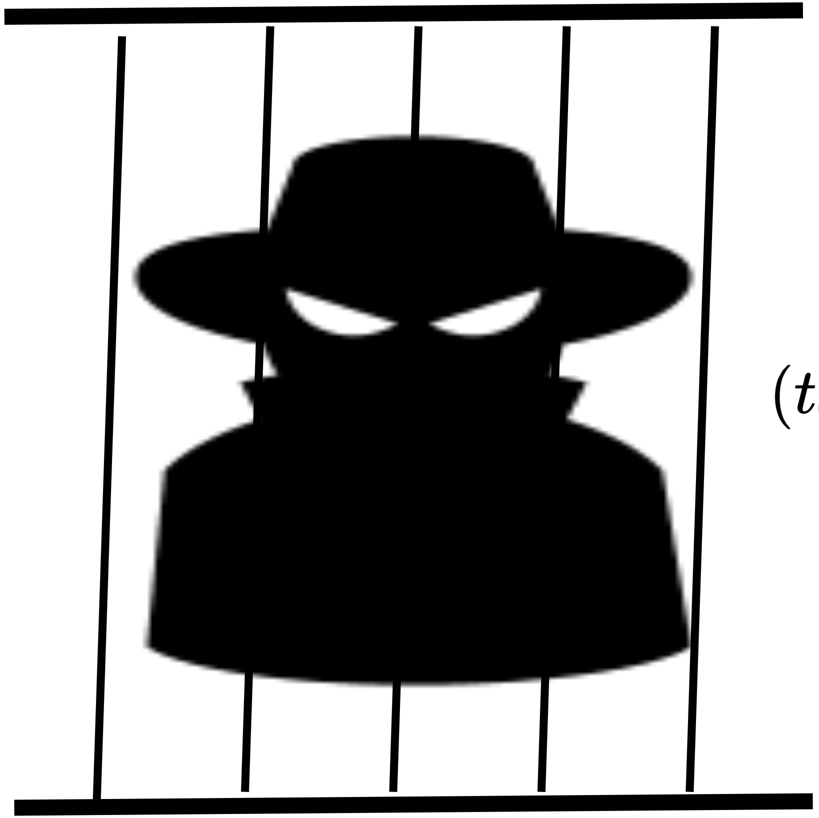

r_A
$(n - t_0)$
r_A

Evidence

At the same round r



Penalty



$(t_0 + 1)$

Future Work

- Bound the bit-complexity Concession
- Bound the probability of the success of an attack
- Propose a generic transformation for any BBC algorithm
- Implement it in the RedBellyBlockchain

Questions ?

Bonus

Probability of Success of an Attack

The Adversary

- **anticipate** the estimate value of every process
- all his connexion are **infinitely rapid**
- can manage like one person the actions of a malicious coalition of **$t < (n-t^0-1)$ processes**
- **can’t forge the signature** of a correct process
- **can’t interfere** with the messages exchanges among honest users.

Assumption on the Network Uniformity

Let a correct process broadcast a message m in a specific line.

The probability distribution of the **interleaving of the reception** of m among honest follows a **uniform law**. That is every interleaving has the same probability to occur.

Algorithm extension

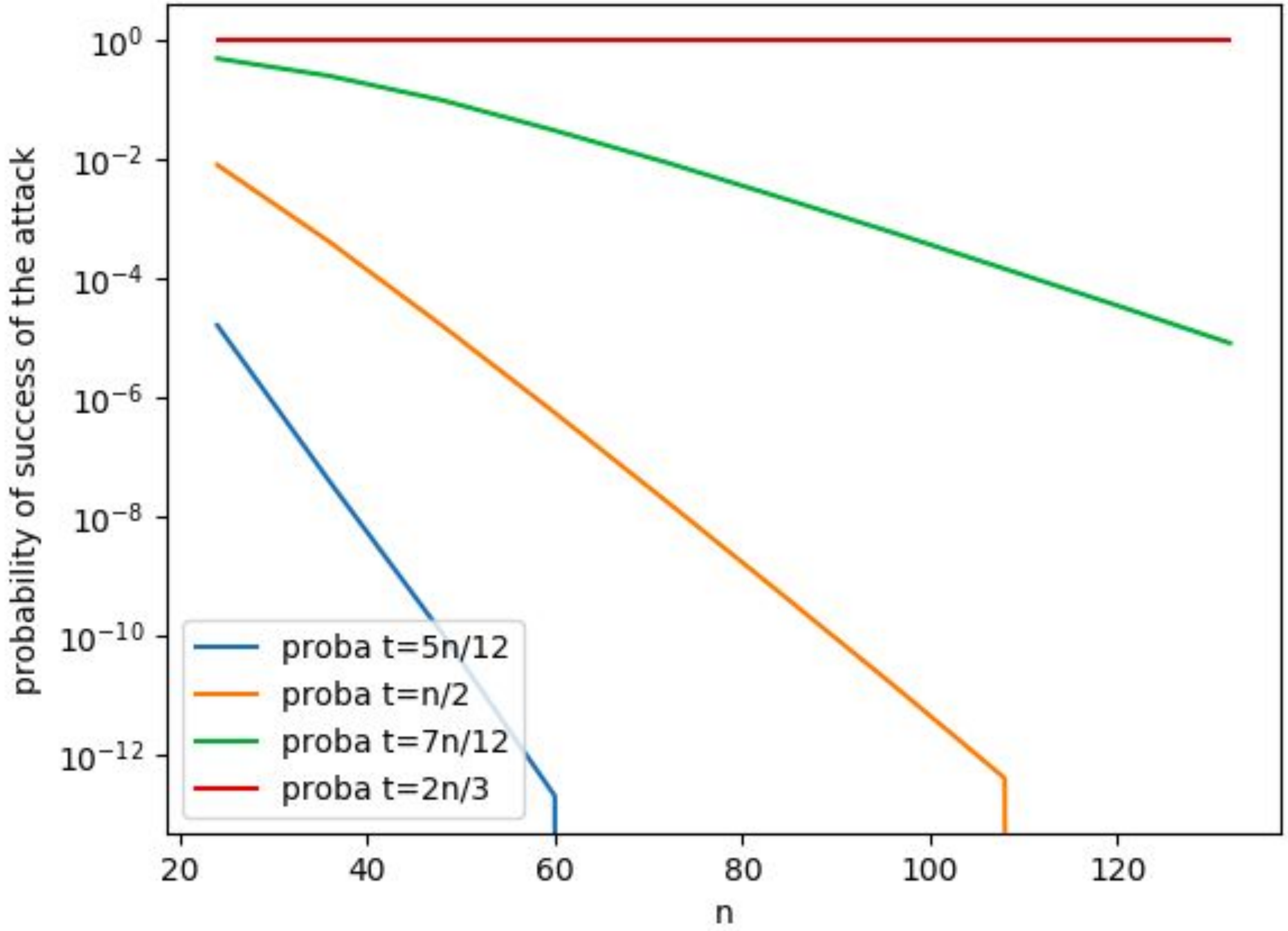
decision ➔ **pre-decision** + **special round** and then **decision**

special round :

- broadcast his **m*** = {pre-decision + ledger containing some proofs}
- wait for $(n\text{-}t^0)$ messages m*

To decide, i (resp. j) needs **(1)** : **$(n\text{-}t\text{-}t^0\text{-}1)$ messages m*** from other correct processes from a set P (resp. R) confirming his own pre-decision.

(2) the messages m* from P (resp. R) to i (resp. j) has to be delivered before those from R (resp. P). (because 2 != pre-decision ➔ detection)



10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
probability of success of the attack
20
40
60
80
100
120
n
proba t=5n/12
proba t=n/2
proba t=7n/12
proba t=2n/3

Decreasing the probability of the attack

repeat the special round k times :

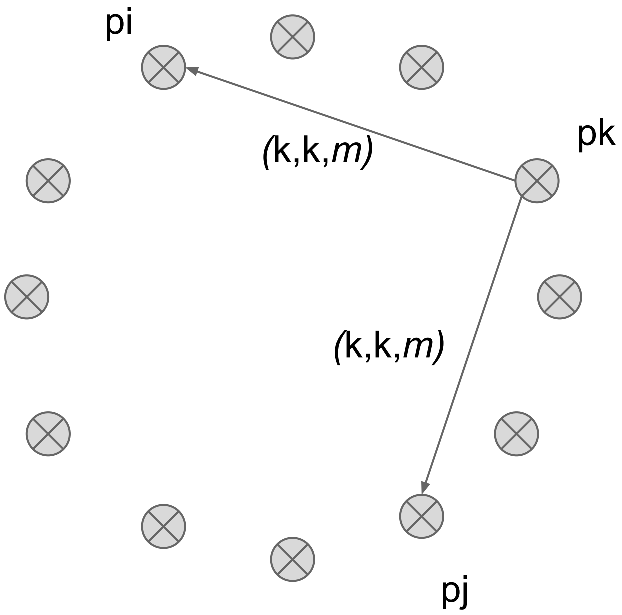
- the cost in complexity is multiplied by k
- the probability is raised to the power of k

Naive forward

Everybody forwards what he received

naive forward

round r, line l : l[r]



register(l[r],k)

(k,a,*m*)
...
(k,i,*m*)
...
(k,k,*m*)
...
(k,z,*m*)

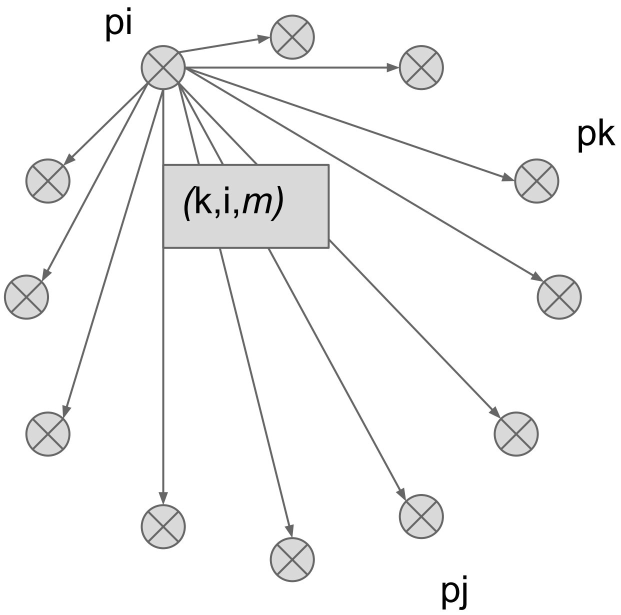
...

register(l[r],q)

(q,a,*m*)
...
(q,i,*m*)
...
(q,k,*m*)
...
(q,z,*m*)

forward_ledger_j(l[r])

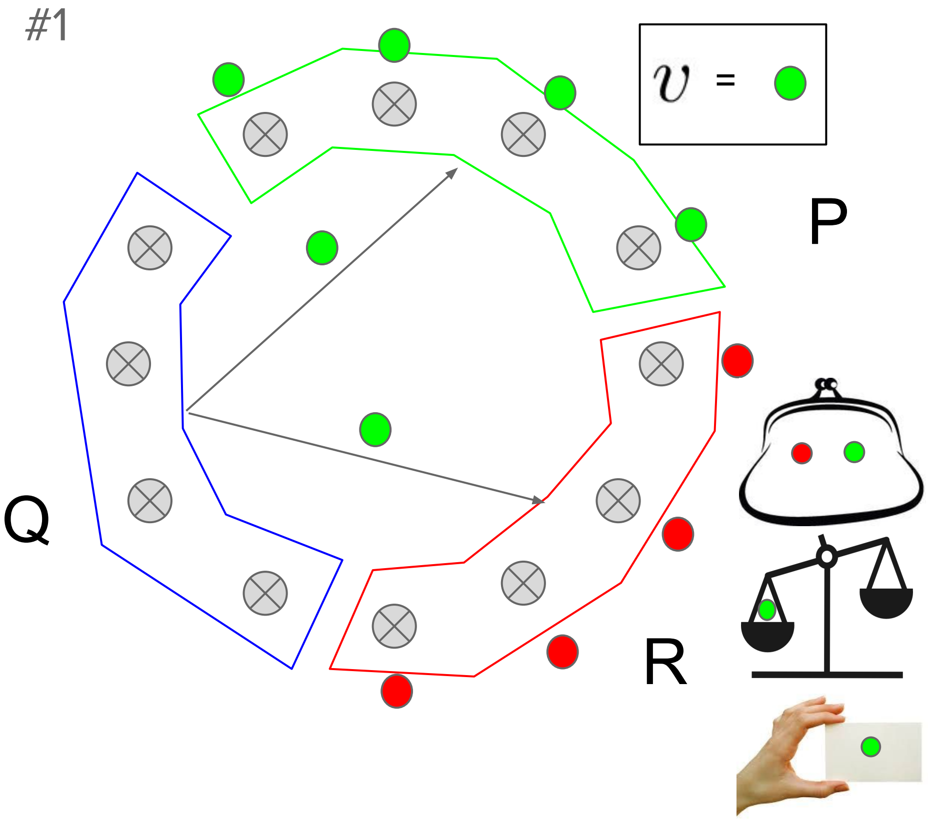
$O(n^2)$ messages

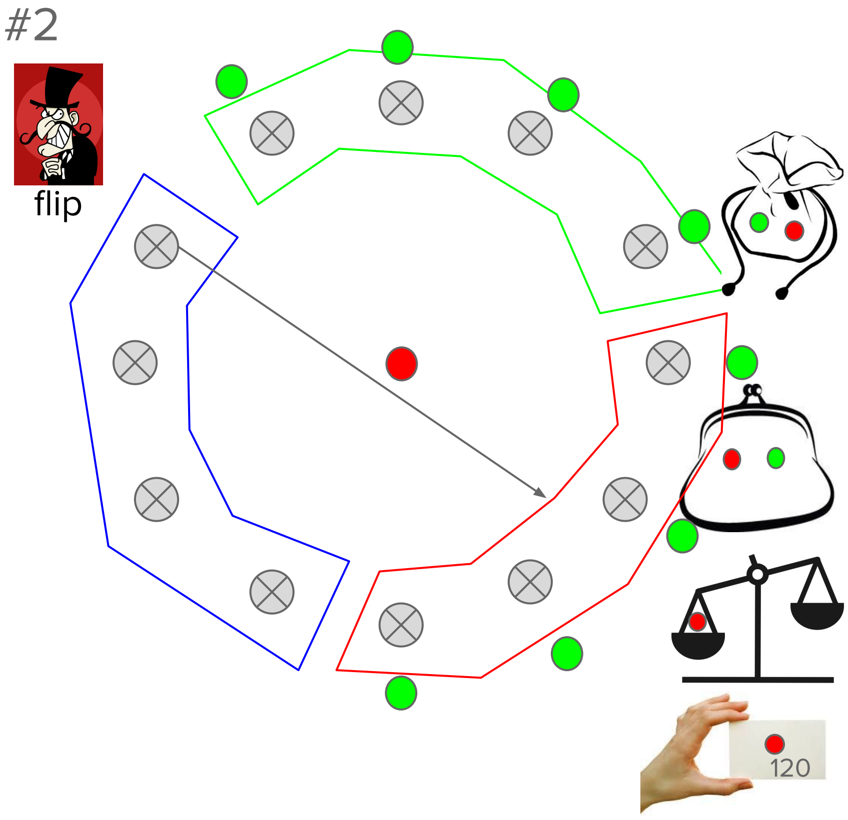


Flip Attack

subvert the naive forward strategy

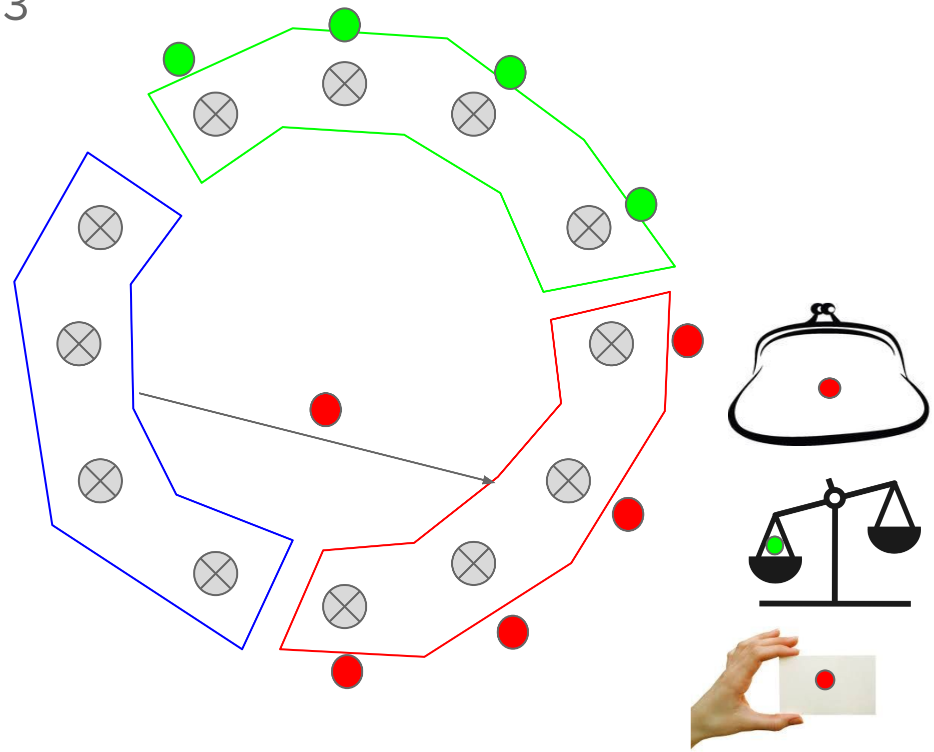
Flip attack



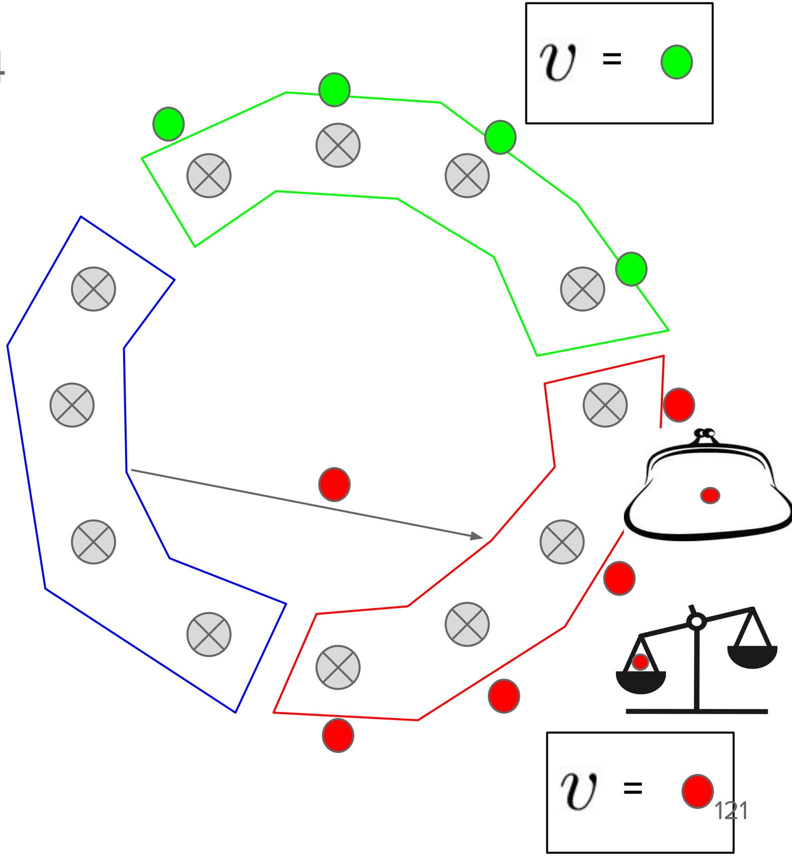


Flip attack

#3

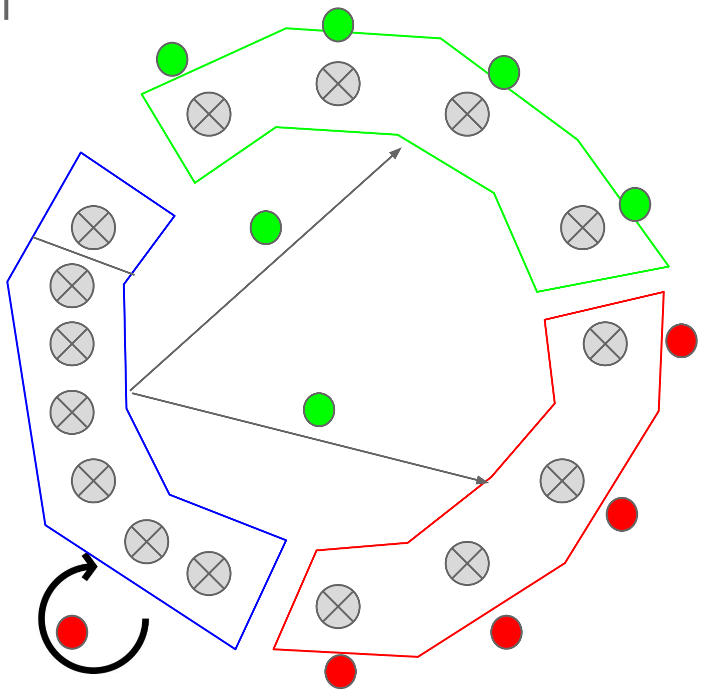


#4

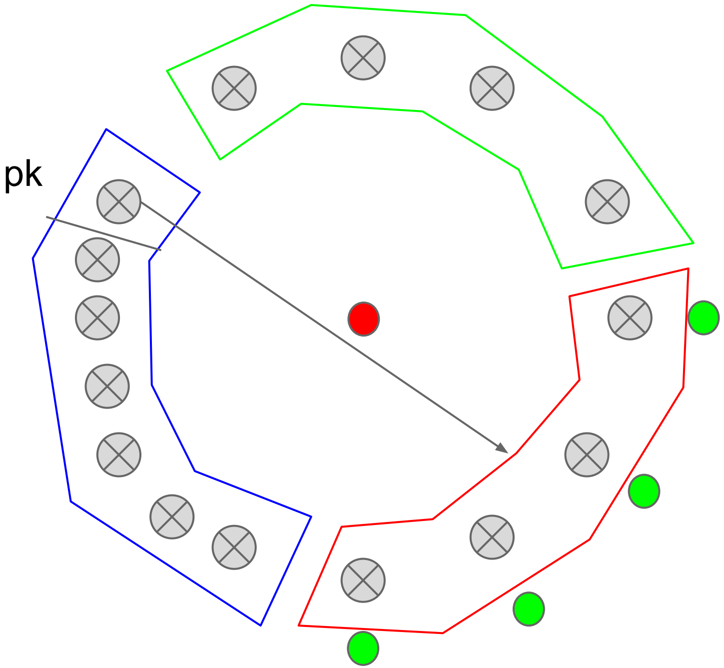


Flip attack

#1



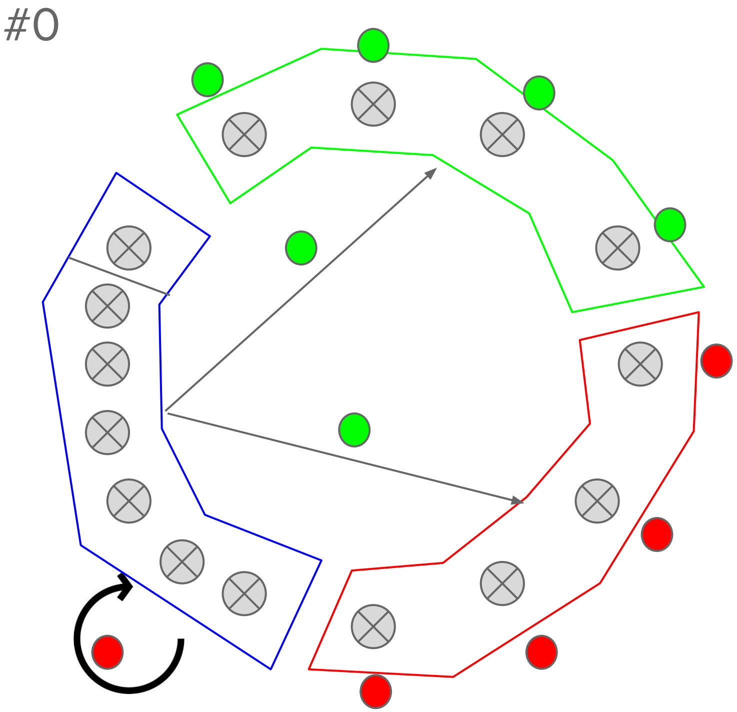
#2

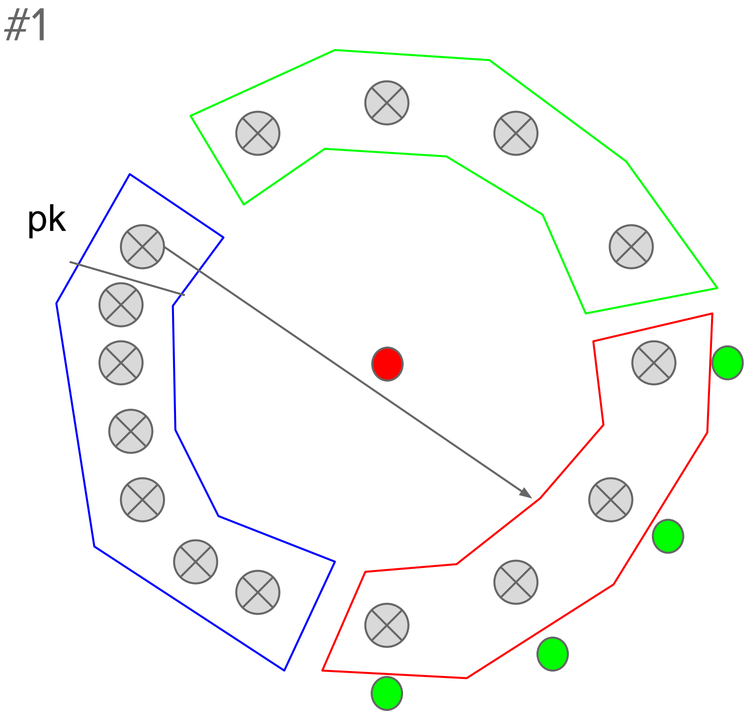


Detection in hindsight

Pay an additional cost only if a disagreement occurred

Flip attack





Generic Solution

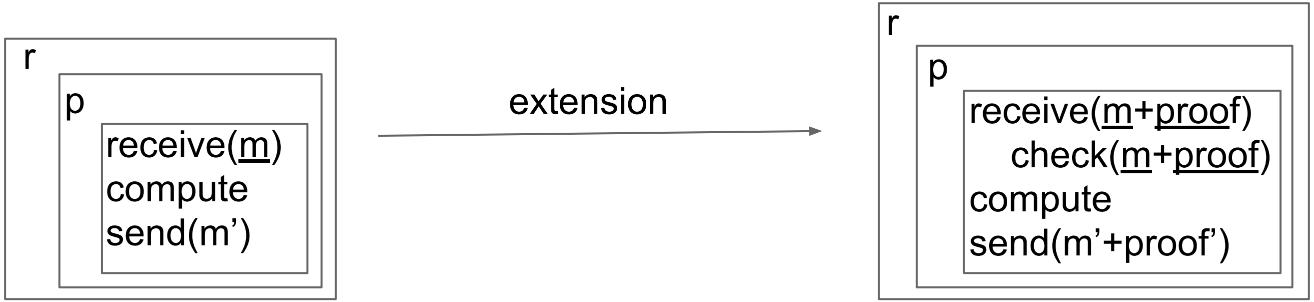
Justify what you send !

Class of Algorithm : C^0

Such an algorithm can be seen as a succession of instruction which can be divided into **rounds** which can be then divided into **pads** which can be divided into **lines**.

r^0	r^1	r^2	
p^0	p^0	p^0	
l^0	l^0	l^0	
l^1	l^1	l^1	
l^2	l^2	l^2	
…	…	…	…
p^1	p^1	p^1	
…	…	…	
p^2	p^2	p^2	
…	…	…	

Every pad can be divided in specific lines



where m holds $(n-t^0)$ message

where proof holds $(n-t^0)$ proofs holding $(n-t^0)$ messages

proof' = m

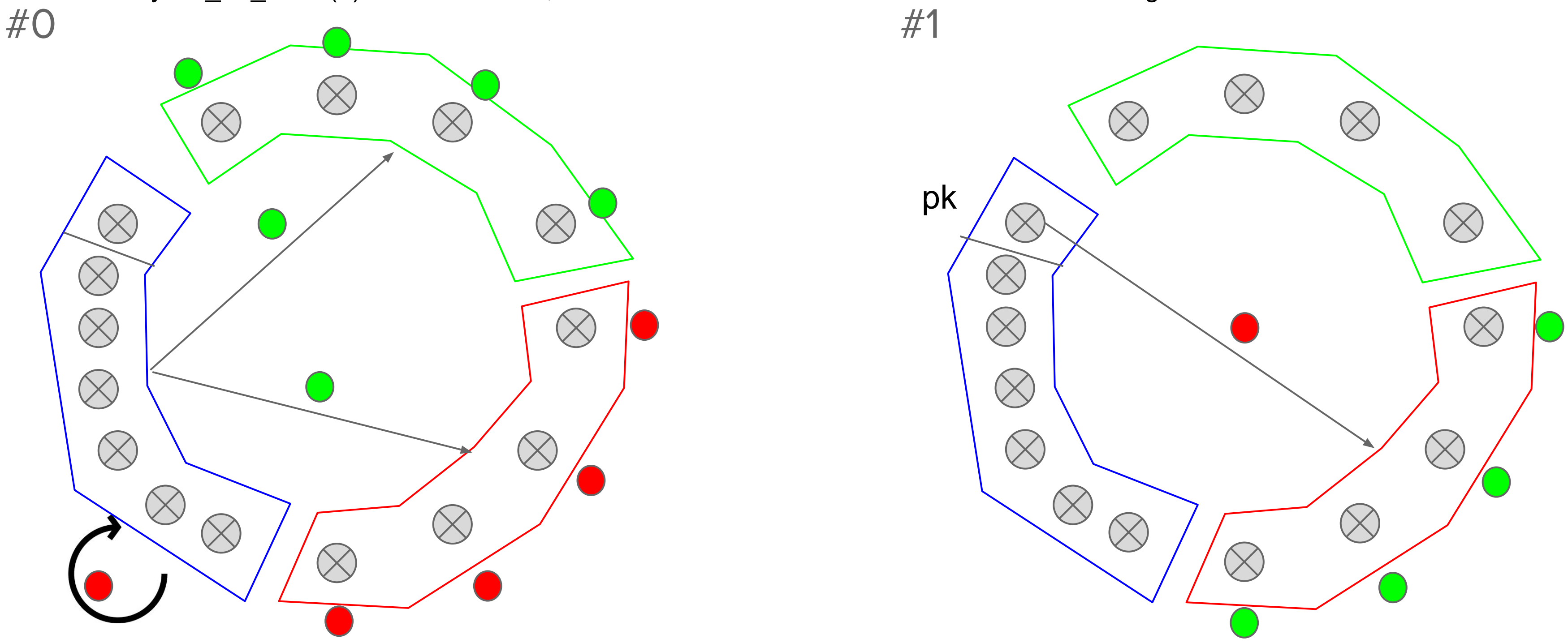
proof' is enough to justify m'

If i and j disagree, they eventually build a proof of culpability

- If pk BV_T1_bdcst(1) in round 1, he built an associated justification PROOF with (n-t^0) messages : AUX[#0](1) If a correct node pi from P decided 0 at round 0, he will BV_T1_bdcst(0) in round 1 with an associated justification PROOF^2 with (n-t^0) messages : AUX[#0](0)

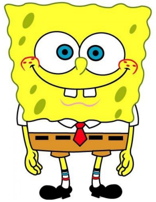
The intersection of the two set has a size of at least (t^0+1) members who cheated and pj will get PROOF and PROOF^2 that he will broadcast to everybody.

- The same reasoning can be applied at the round 2.
- If nobody BV_T1_bdcst(1) at round 1 or 2 , the consensus liveness can be tackled but no disagreement will occur



We want to generalize this idea

- decided in round r. This decision is reasoned by a set of message M and a set of proof P.



- decided in round r'. This decision is reasoned by a set of message M' and a set of proof P'.

<u>Question</u> : Are M, M', P and P' enough to always proof guilty a malicious coalition ?

Can we give a generic proof of it ?

Possible approach

Divide the type of Byzantine behaviour : **mute**, **mutant messages**, **commission**.

The mute behaviour can only tackle the liveness.

The mutant messages, will be always detected.

A **commission** will need a justification, that is a **proof** in P, holding $(n-t^0)$ messages.

We would like to show that for any algorithm in C^0, those $(n-t^0)$ messages in P will generate a collision with other $(n-t^0)$ messages in P'.

Accountable-BV-Broadcast

operation ACC-BV-broadcast EST[r_i]($v_i, cert_i^{r_i}$)
(01) $msg1_i = (T1, r_i, i, B\_VAL(v_i), cert_i^{r_i})$
(02) broadcast($msg1_i$)

(03) when a T1-message $msg1_j$ is received from p_j
(04) if ($valid(cert_j, v_j)$) and no message from p_j has been added)
(05) add $msg1_j$ to $T1\_list_i$;
end if;

(06) when a T2-message $msg2_j$=($T2, v_j$) is received from p_j
(07) add $msg2_j$ in $T2\_list_i$

(08) when $\exists(m^1, (m^2, ..., m^{t_0+1})) \in T1\_list_i \times (T1\_list_i \cup T2\_list_i)^{t_0}$ |
$\forall(p, q) \in [1, t_0 + 1]^2$, $(m^p.value = m^q.value = v) \cap (m^p.id \neq m^q.id) \cap$
(no message with value v has been sent)
(09) broadcast $(T2, v)$

(10) when $\exists(m^1, (m^2, ..., m^{2t_0+1})) \in T1\_list_i \times (T1\_list_i \cup T2\_list_i)^{2t_0}$ |
$\forall(p, q) \in [1, 2t_0 + 1]^2$, $(m^p.value = m^q.value = v) \cap (m^p.id \neq m^q.id)$
(11) $bin\_values_i \leftarrow bin\_values_i \cup \{v\}$

Accountable-BV-Broadcast

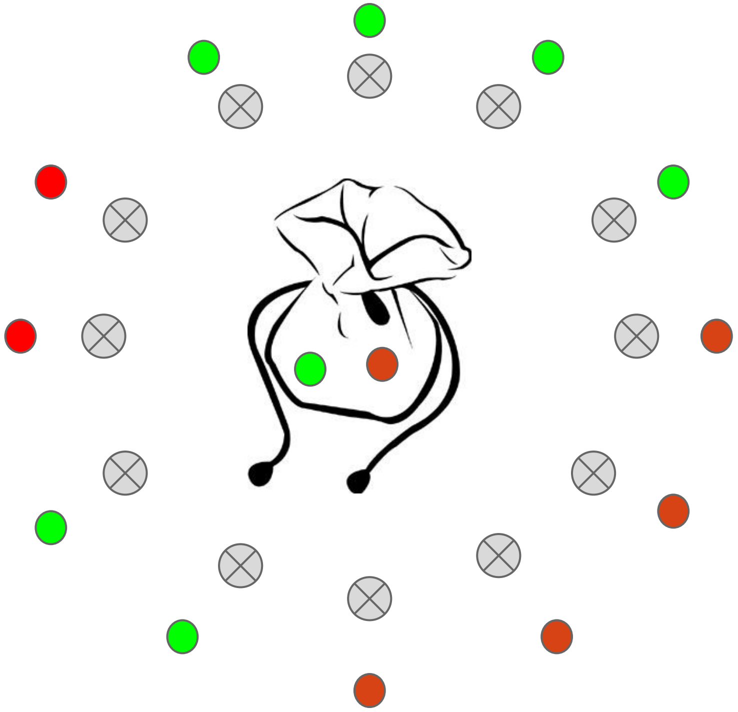
t < n/3 :

BV-Obligation

BV-Justification

BV-Uniformity

BV-Termination



forall t :

BV-Accountability

Questions ?