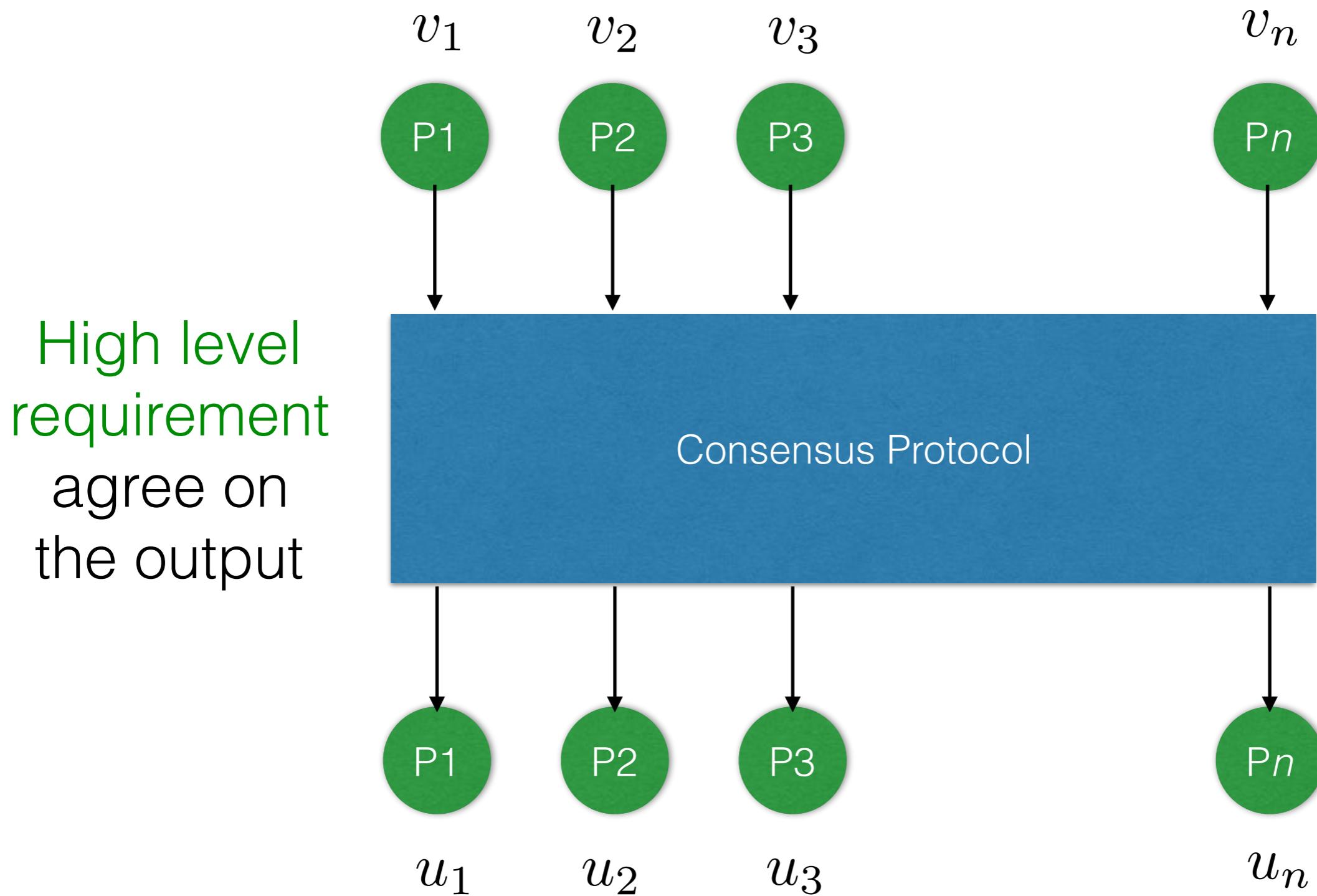


Motivation for the Consensus Layer, I

- A transaction history needs to be **agreed** by all participants.
- Participants may have diverging interests in terms of the history of transactions.

The Consensus Problem



Study initiated by Lamport, Pease, Shostak 1982

Problem Statement

- A number (say, t) of the participating entities can diverge from the protocol.
- This has been called **Byzantine behaviour** in the literature.
- The properties of the protocol are defined in the presence of this “malicious” coalition of parties that attempts to disrupt the process for the “honest” parties. $H, |H| = n - t$

Consensus Properties

- Termination $\forall i \in \mathbb{H} (u_i \text{ is defined})$
- Agreement $\forall i, j \in \mathbb{H} (u_i = u_j)$

Strong Validity : $\forall i \in \mathbb{H} \exists j \in \mathbb{H} (u_i = v_j)$

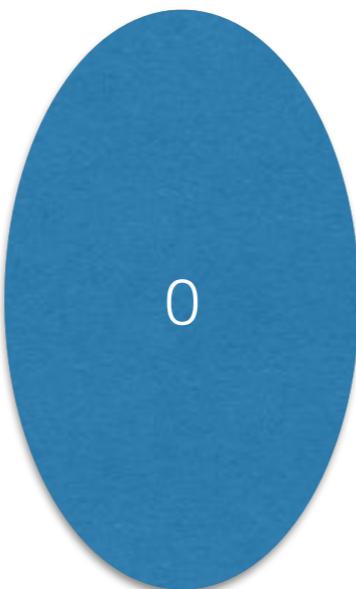
Honest Majority is Necessary, I

Consider an adversary that performs one of the following with probability 1/3

$$\frac{n}{2}$$

$$\frac{n}{2}$$

\emptyset



$\forall i \in A_0 : (v_i = 0)$



$\forall i \in A_1 : (v_i = 1)$

Honest Majority is Necessary, II

- If the adversary corrupts A_0 , then output of honest parties (that belong to A_1) should be 1.
- If the adversary corrupts A_1 , then output of honest parties (that belong to A_0) should be 0.
- If the adversary corrupts no-one, then the output of all parties should be equal.

Is Honest Majority Sufficient?

- Two important scenarios have been considered in the consensus literature.
 - Point to point channels. **No setup.** $t < \frac{n}{3}$
 - Point to point channels. **With setup.** $t < \frac{n}{2}$

The setup enables provides a correlated private initialization string to each participant. It is assumed to be honestly produced.

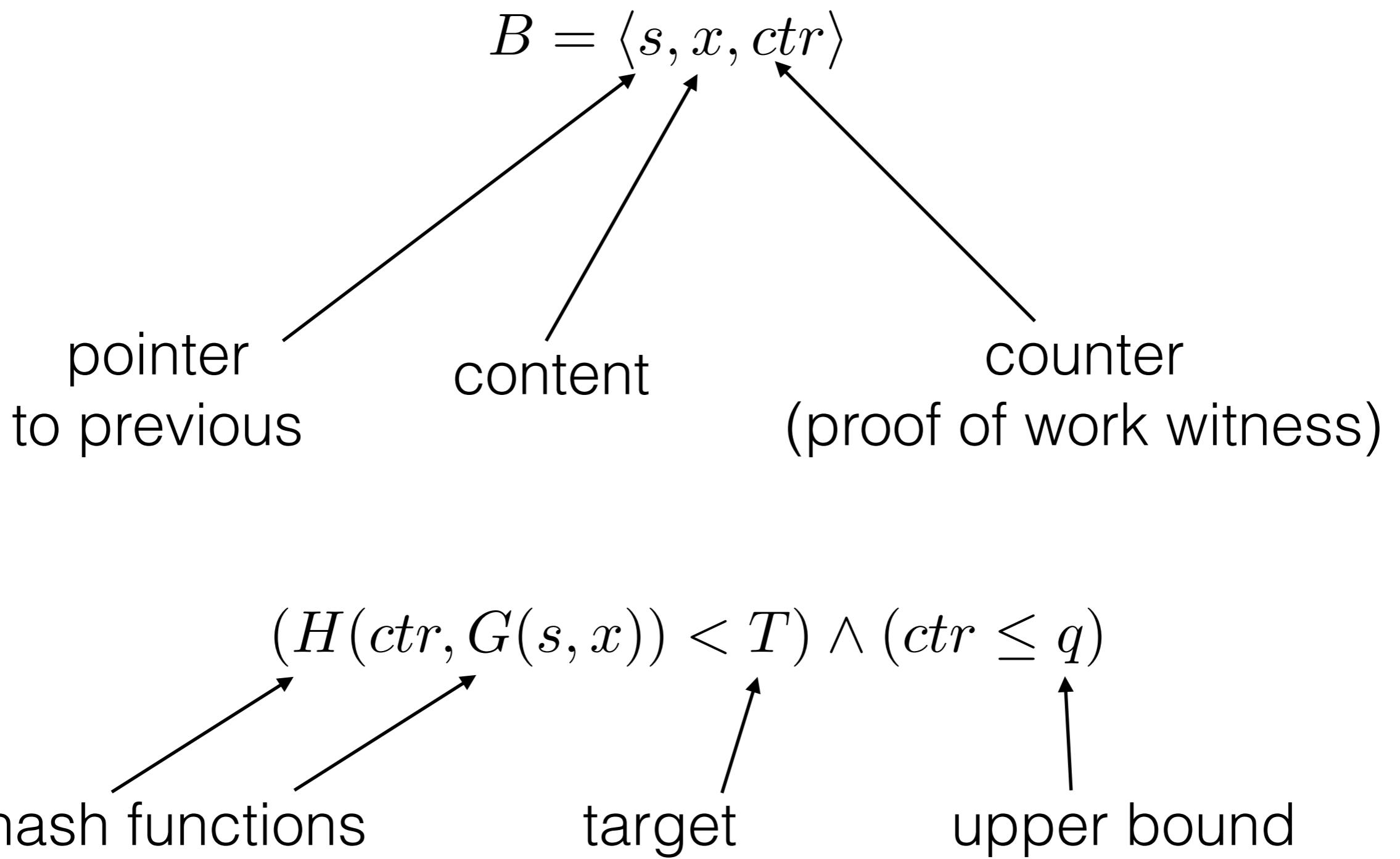
Setup in Consensus

- A "public-key infrastructure" (PKI).
 - parties have signing and verification keys for a digital signature scheme.
 - Each party knows every other party's verification key.

Enter Bitcoin

- (2008-2009)
- Important concepts used
 - blockchain data structure.
 - proof of work (POW).
- both known and studied earlier, but put in combination for a novel application.

Blocks



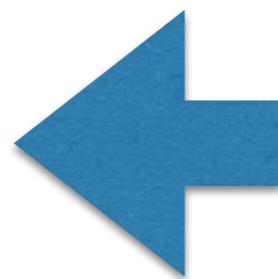
Blockchain

$B_0 = \langle \perp, x_0, ctr_0 \rangle$

$B_1 = \langle s_1, x_1, ctr_1 \rangle$

\vdots

$B_n = \langle s_n, x_n, ctr_n \rangle$



genesis block

$$s_i = H(ctr_{i-1}, G(s_{i-1}, x_{i-1}))$$

$$\mathcal{C} = \langle B_0, \dots, B_n \rangle$$

head

$$\mathbf{x}_{\mathcal{C}} = \langle x_0, x_1, \dots, x_n \rangle$$

$$\mathcal{C}^{\lceil k } = \langle B_0, \dots, B_{n-k} \rangle$$

The bitcoin “backbone”

- The core of the bitcoin protocol
 - The chain validation predicate.
 - The chain selection rule (max-valid)
 - The proof of work function.
 - The main protocol loop
- Protocol is executed by “**miners**.”

Algorithm 1 The *chain validation predicate*, parameterized by q, T , the hash functions $G(\cdot), H(\cdot)$, and the *content validation predicate* $V(\cdot)$. The input is \mathcal{C} .

```
1: function validate( $\mathcal{C}$ )
2:    $b \leftarrow V(\mathbf{x}_{\mathcal{C}})$ 
3:   if  $b \wedge (\mathcal{C} \neq \epsilon)$  then                                 $\triangleright$  The chain is non-empty and meaningful w.r.t.  $V(\cdot)$ 
4:      $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$ 
5:      $s' \leftarrow H(ctr, G(s, x))$ 
6:     repeat
7:        $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$ 
8:       if  $\text{validblock}_q^T(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$  then
9:          $s' \leftarrow s$                                           $\triangleright$  Retain hash value
10:         $\mathcal{C} \leftarrow \mathcal{C} \lceil^1$                                  $\triangleright$  Remove the head from  $\mathcal{C}$ 
11:       else
12:          $b \leftarrow \text{False}$ 
13:       end if
14:     until  $(\mathcal{C} = \epsilon) \vee (b = \text{False})$ 
15:   end if
16:   return ( $b$ )
17: end function
```

Algorithm 2 The function that finds the “best” chain, parameterized by function $\max(\cdot)$. The input is $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$.

```
1: function maxvalid( $\mathcal{C}_1, \dots, \mathcal{C}_k$ )
2:    $temp \leftarrow \varepsilon$ 
3:   for  $i = 1$  to  $k$  do
4:     if validate( $\mathcal{C}_i$ ) then
5:        $temp \leftarrow \max(\mathcal{C}_i, temp)$ 
6:     end if
7:   end for
8:   return  $temp$ 
9: end function
```

Algorithm 3 The *proof of work* function, parameterized by q , T and hash functions $H(\cdot)$, $G(\cdot)$. The input is (x, \mathcal{C}) .

```
1: function pow( $x, \mathcal{C}$ )
2:   if  $\mathcal{C} = \varepsilon$  then                                 $\triangleright$  Determine proof of work instance
3:      $s \leftarrow 0$ 
4:   else
5:      $\langle s', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$ 
6:      $s \leftarrow H(ctr', G(s', x'))$ 
7:   end if
8:    $ctr \leftarrow 1$ 
9:    $B \leftarrow \varepsilon$ 
10:   $h \leftarrow G(s, x)$ 
11:  while ( $ctr \leq q$ ) do
12:    if ( $H(ctr, h) < T$ ) then                   $\triangleright$  This  $H(\cdot)$  invocation subject to the  $q$ -bound
13:       $B \leftarrow \langle s, x, ctr \rangle$ 
14:      break
15:    end if
16:     $ctr \leftarrow ctr + 1$ 
17:  end while
18:   $\mathcal{C} \leftarrow CB$                                  $\triangleright$  Extend chain
19:  return  $\mathcal{C}$ 
20: end function
```

Algorithm 4 The Bitcoin backbone protocol, parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$.

```
1:  $\mathcal{C} \leftarrow \varepsilon$ 
2:  $st \leftarrow \varepsilon$ 
3:  $round \leftarrow 1$ 
4: while TRUE do
5:    $\tilde{\mathcal{C}} \leftarrow \text{maxvalid}(\mathcal{C}, \text{any chain } \mathcal{C}' \text{ found in RECEIVE()})$ 
6:   if INPUT() contains READ then
7:     write  $R(\tilde{\mathcal{C}})$  to OUTPUT()            $\triangleright$  Produce necessary output before the POW stage.
8:   end if
9:    $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \text{INPUT}(), \text{RECEIVE}())$            $\triangleright$  Determine the  $x$ -value.
10:   $\mathcal{C}_{\text{new}} \leftarrow \text{pow}(x, \tilde{\mathcal{C}})$ 
11:  if  $\mathcal{C} \neq \mathcal{C}_{\text{new}}$  then
12:     $\mathcal{C} \leftarrow \mathcal{C}_{\text{new}}$ 
13:    DIFFUSE( $\mathcal{C}$ )                   $\triangleright$  Broadcast the chain in case of adoption/extension.
14:  else
15:    DIFFUSE( $\perp$ )                 $\triangleright$  Signals the end of the round to the diffuse functionality.
16:  end if
17:   $round \leftarrow round + 1$ 
18: end while
```

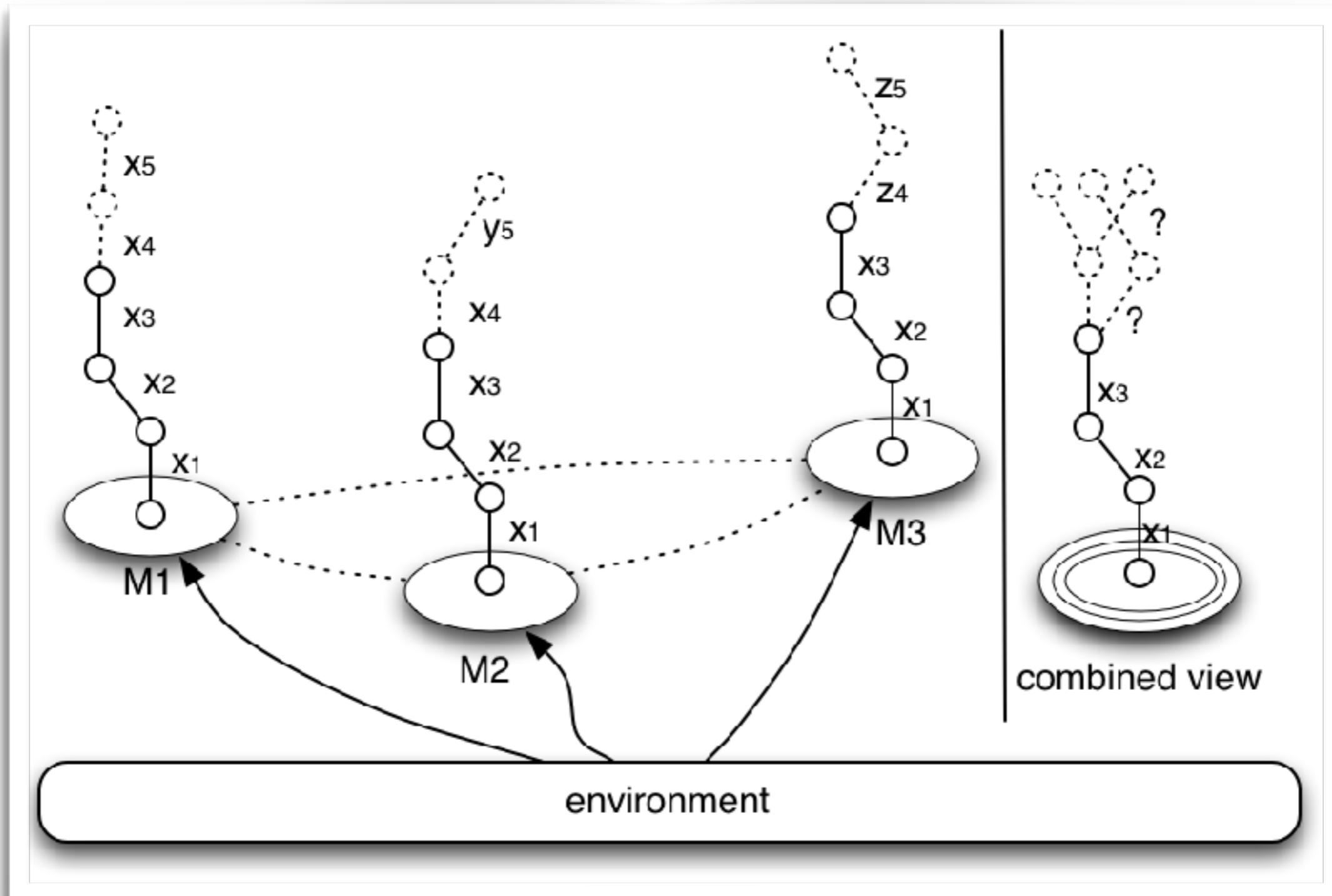
Adversary Model

- Assume there are n parties running of the protocol
 - synchronously.
 - each one has a quota of q queries to the function $H(\cdot)$ in each round.
- A number of t parties are controlled by an adversary (a malicious coalition).

Basic Properties

- Common Prefix
- Chain Quality
- Chain Growth

Common Prefix, I



Common Prefix, II

(strong common prefix / consistency)

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

The property holds true in a probabilistic sense with an error that decays exponentially in k
If we take different chains of parties at different rounds, if we remove k blocks from one, we will be inside other chain. The more blocks remove, the higher is the likelihood you will be in the prefix of other chain that will be adopted by honest party in future round.
The probab is close to one, but never always 1.

(1-epsilon). The honest parties maintain a main chain with small diverging forks. There will be no deep forks. Fork means divergence from the main chain.

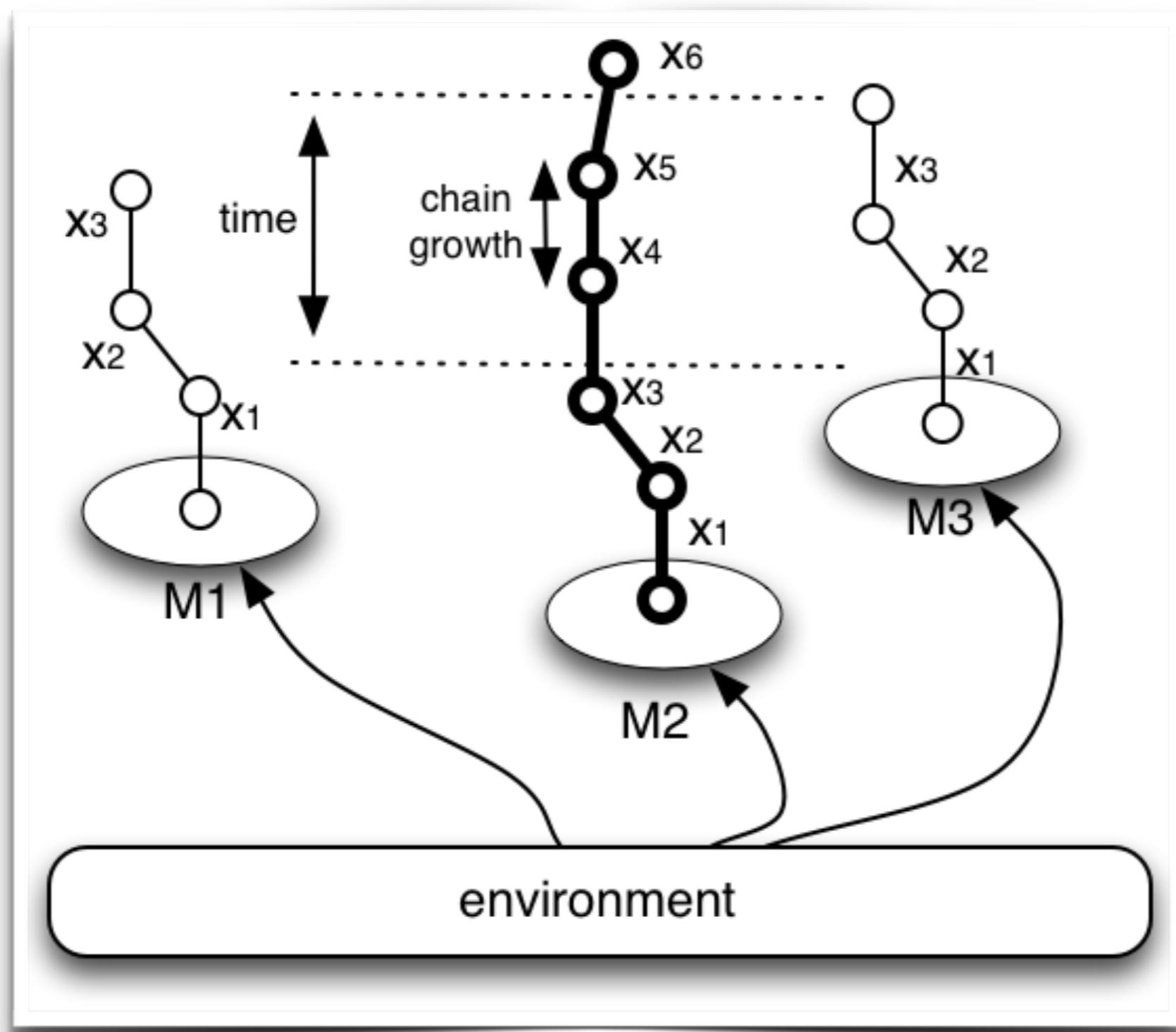
Common Prefix, III

(strong common prefix / consistency)

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

- The property goes in future. Am I looking at the right chain? You remove k blocks ,and with high probability you are sure that future chains will adopt prefix.

Chain Growth, I



Chain Growth, II

Parameters $\tau \in (0, 1), s \in \mathbb{N}$

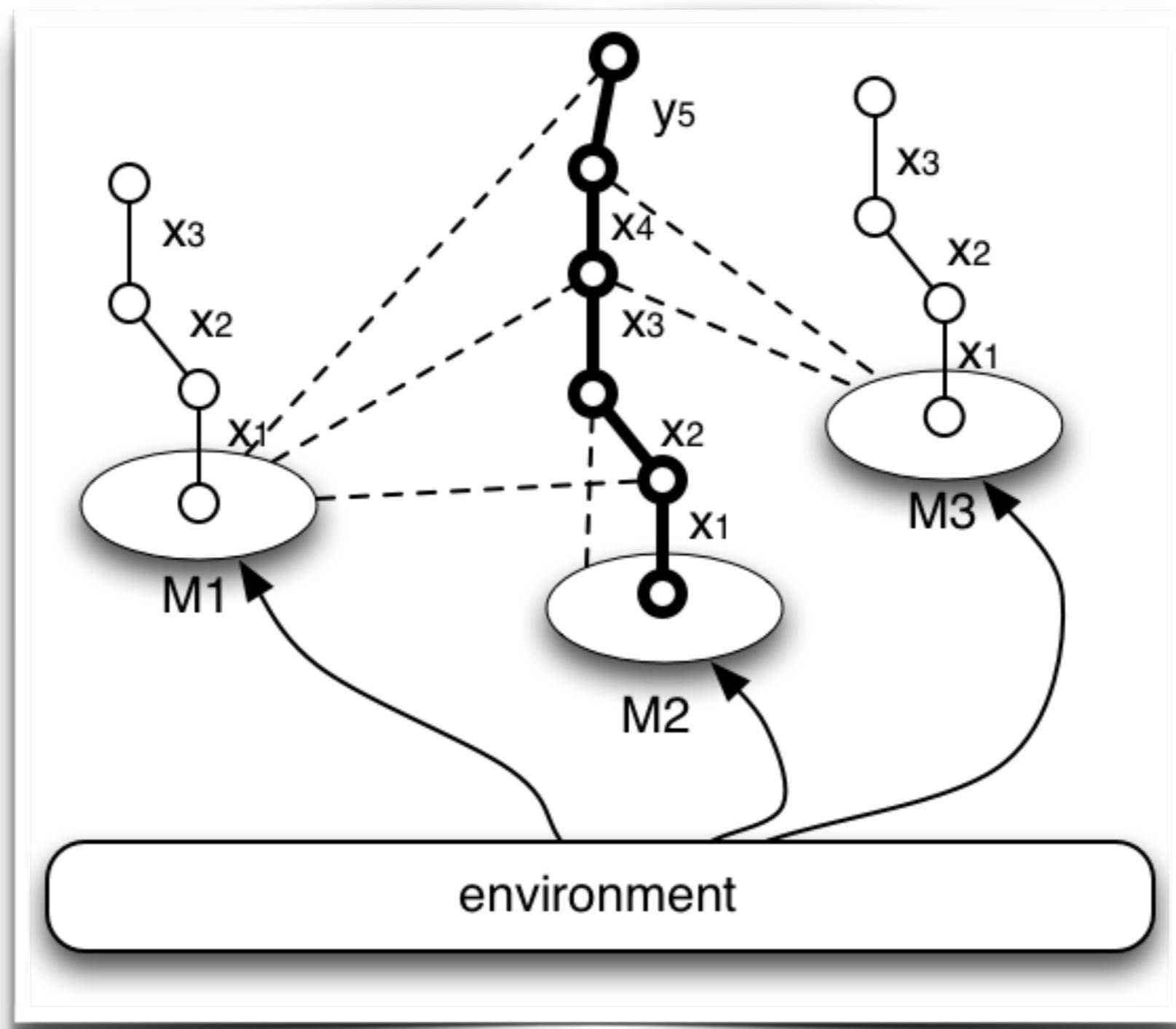
$\forall r_1, r_2$ honest player P with chains $\mathcal{C}_1, \mathcal{C}_2$

$$r_2 - r_1 \geq s \implies |\mathcal{C}_2| - |\mathcal{C}_1| \geq \tau s$$

The property holds true in a probabilistic sense with an error probability that exponentially decays in s . *The chain will grow at honest node after period of time s . Tau is close to zero, but bounded away.*

$\tau \approx$ probability at least one honest party finds a POW in a round

Chain Quality, I



Chain Quality, II

Parameters $\mu \in \{0, 1\}$, $\ell \in \mathbb{N}$

The ratio of blocks of an ℓ -long segment of an honest chain produced by the adversary is bounded by $(1 - \mu)\ell/l$

The probability holds true probabilistically with an error that exponentially decays in ℓ where the blocks in chain coming from? consider when $t=n/2$

$$\mu \approx \frac{n - 2t}{n - t}$$

Robust Transaction Ledger

- The three properties can provide a ledger with the following characteristics.
 - persistence: Transactions are organized in a “log” and honest nodes agree on it.
 - liveness: New transactions are included in the log nodes, after a suitable period of time.

Establishing the ledger

- Persistence follows from strong Common Prefix.
 - (need to exclude k most recent blocks)
- Liveness from Chain Growth and Chain Quality.
 - (leave sufficient time for chain to grow and then apply chain quality to ensure that at least one honest block is included).

The Bitcoin Setting for Consensus

- The bitcoin setting is a different compared to what has been considered classically for the consensus problem.
 - Communication is by **diffusion**. (no point-to-point channels).
 - Message delivery is assumed but message origins and recipient list are not specified.
 - The protocol setup is **not** a private correlated setup (digital signatures are **not** used to authenticate miners)
 - A public setup is assumed (in the form of the genesis block)

Natoshii Email.

Re: Bitcoin P2P e-cash paper

Satoshi Nakamoto | Thu, 13 Nov 2008 19:34:25 -0800

James A. Donald wrote:

> It is not sufficient that everyone knows X. We also
> need everyone to know that everyone knows X, and that
> everyone knows that everyone knows that everyone knows X
> - which, as in the Byzantine Generals problem, is the
> classic hard problem of distributed data processing.

The proof-of-work chain is a solution to the Byzantine Generals' Problem. I'll try to rephrase it in that context.

A number of Byzantine Generals each have a computer and want to attack the King's wi-fi by brute forcing the password, which they've learned is a certain number of characters in length. Once they stimulate the network to generate a packet, they must crack the password within a limited time to break in and erase the logs, otherwise they will be discovered and get in trouble. They only have enough CPU power to crack it fast enough if a majority of them attack at the same time.

They don't particularly care when the attack will be, just that they all agree.

It has been decided that anyone who feels like it will announce a time, and whatever time is heard first will be the official attack time. The problem is