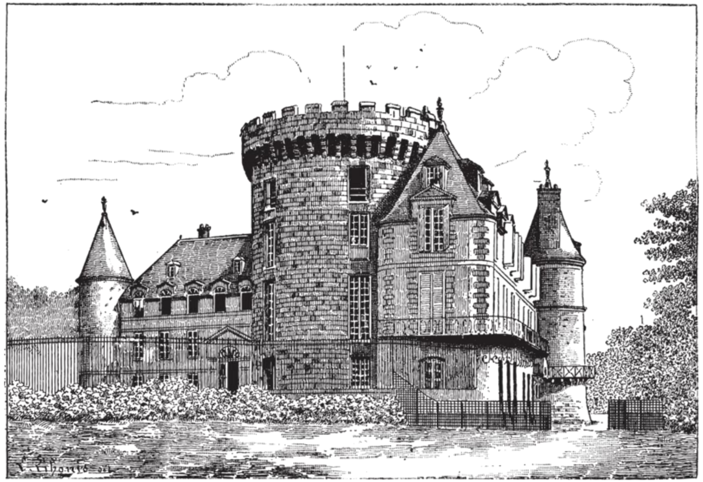
Blockchains & Distributed Ledgers

Lecture 05

Petros Wallden

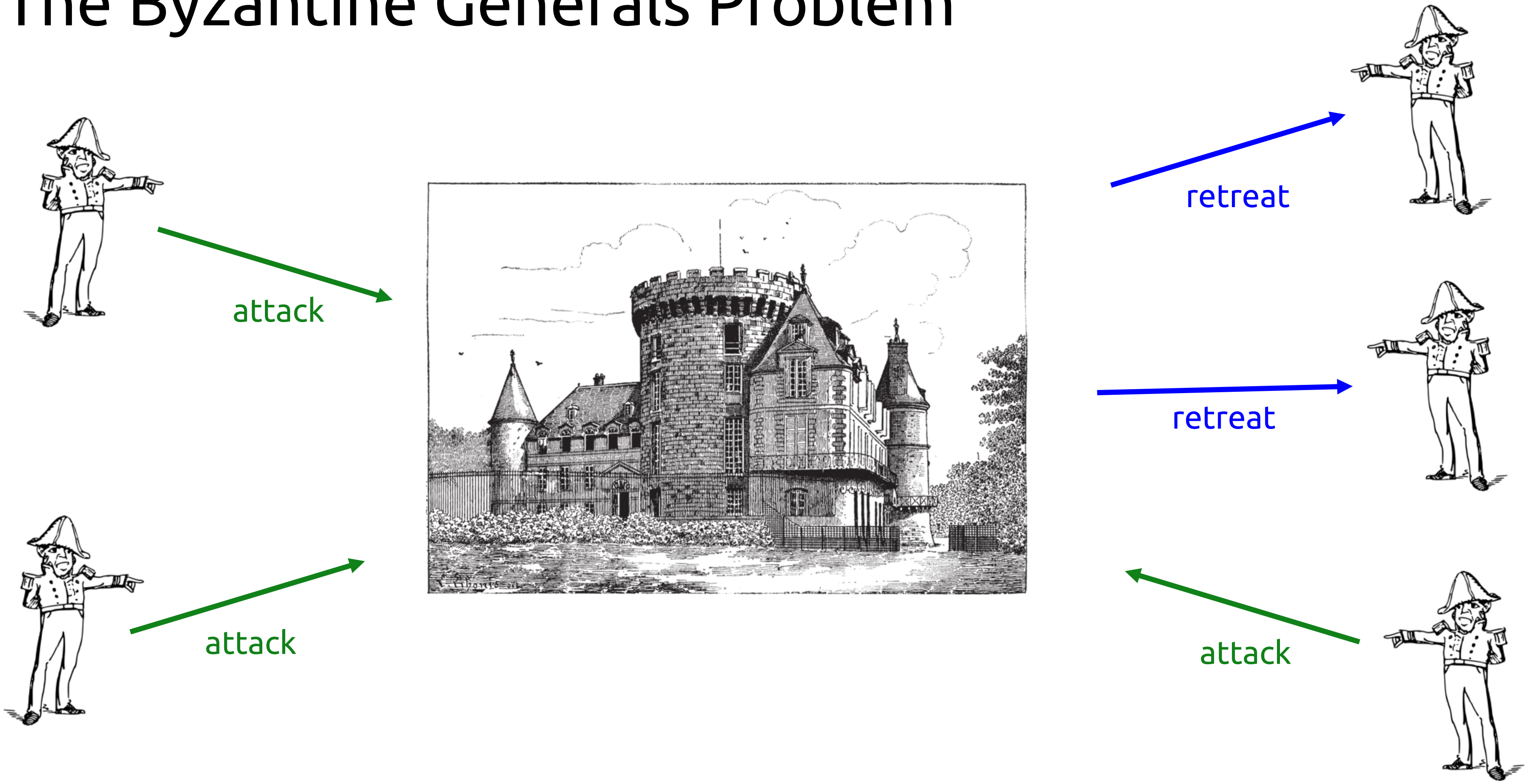
Slide credits: PW, Dimitris Karakostas, Aggelos Kiayias, Dionysis Zindros, Christos Nasikas

The Byzantine Generals Problem

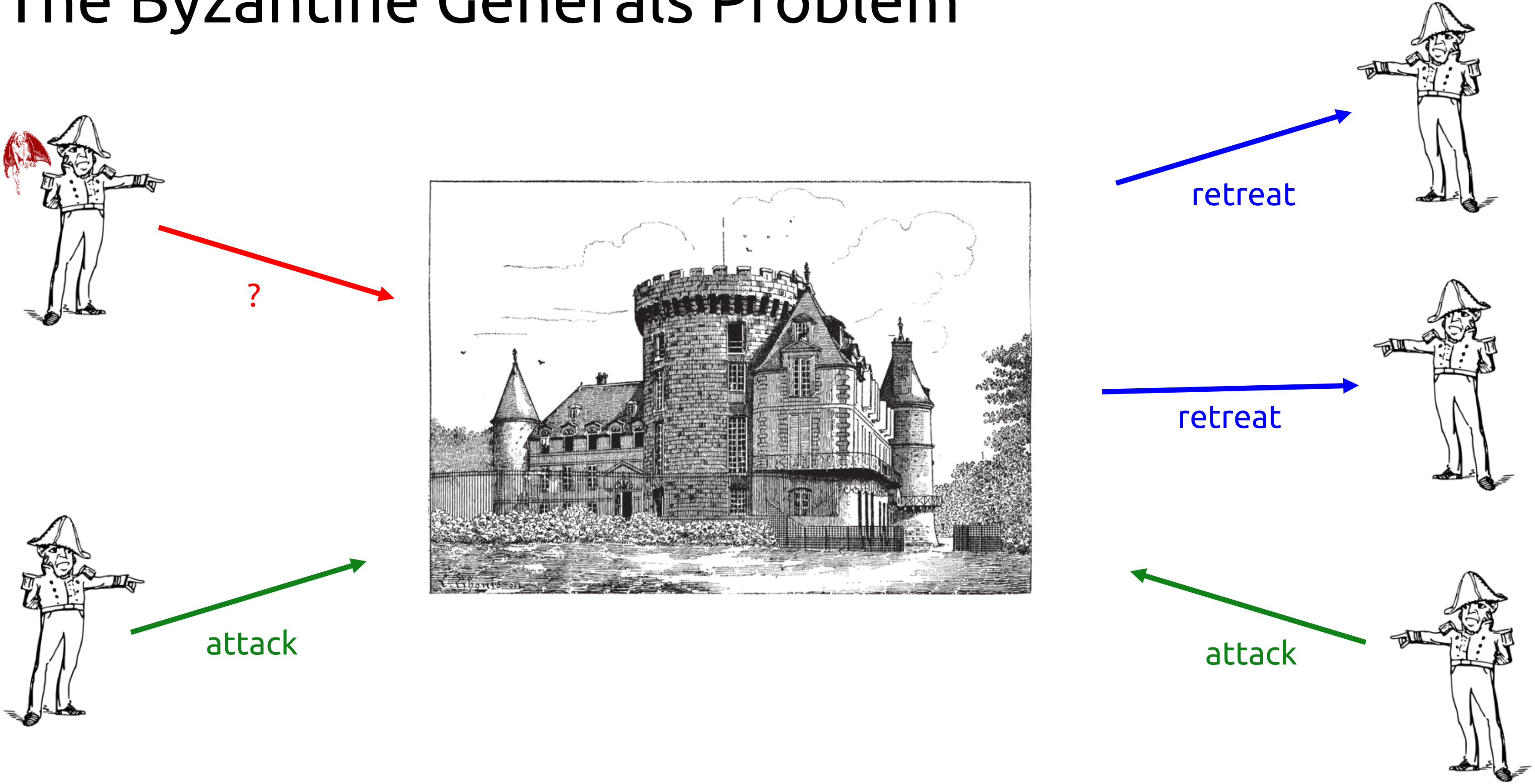


[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease.
The byzantine generals problem

The Byzantine Generals Problem



The Byzantine Generals Problem

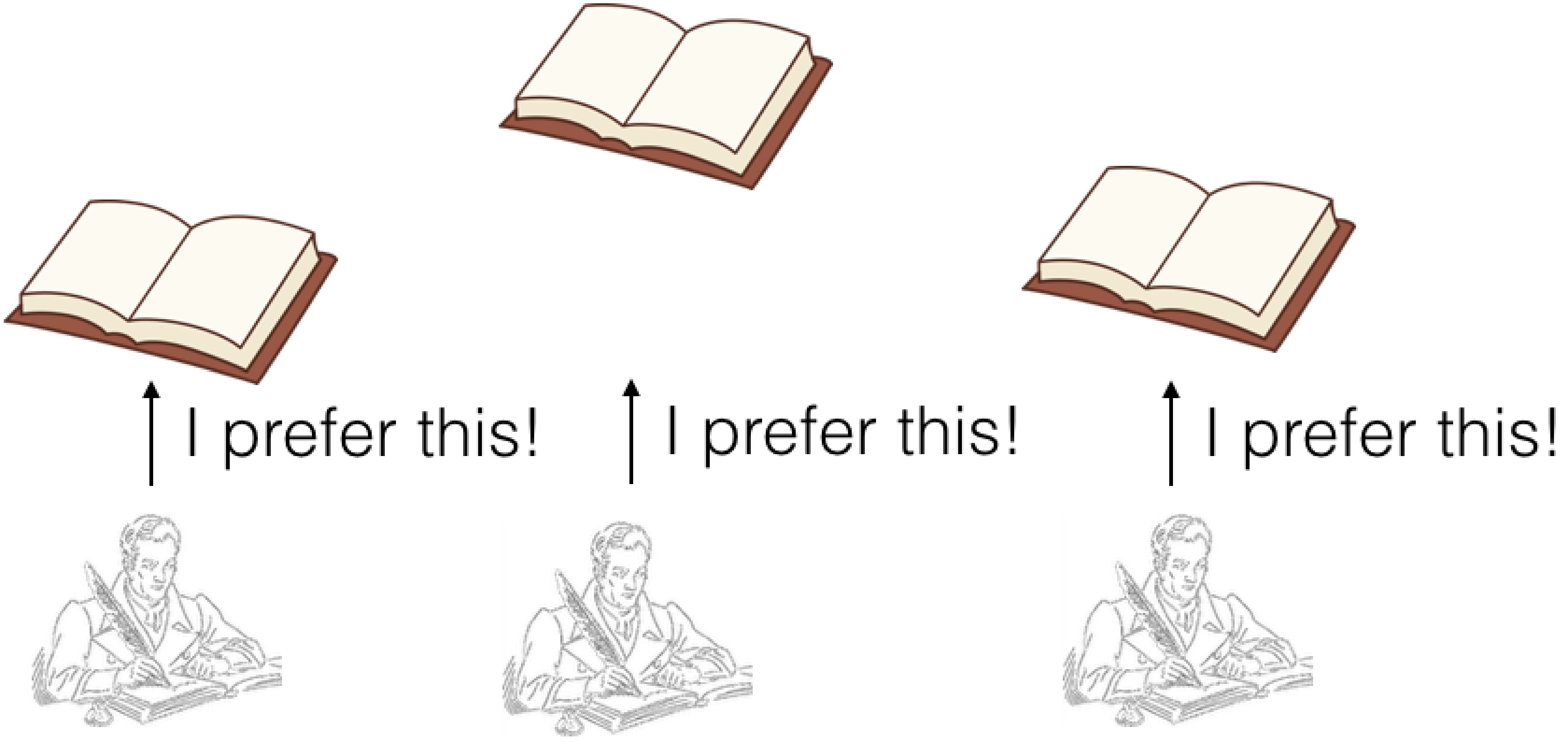


The Consensus Problem

Motivation for the Consensus Layer, I

- A transaction history and/or state of the service needs to be **agreed** by all servers.
- Servers may be operated by participants with **diverging interests**, in terms of the history of transactions and/or state of the service.

Motivation for the Consensus Layer, II



Consensus : Problem Statement

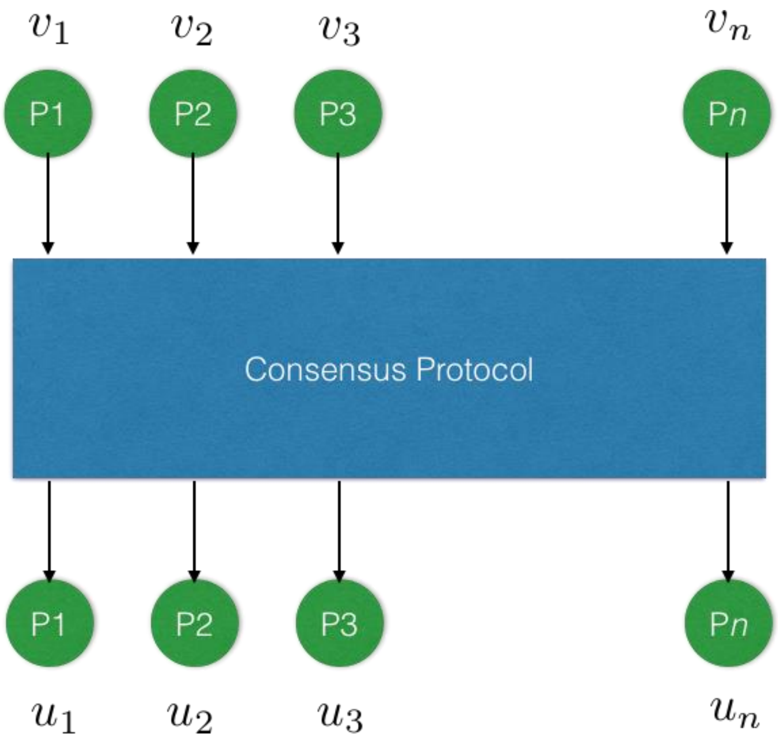
- A number (t) of the participating entities can diverge from the protocol.
- This has been called Byzantine behaviour in the literature.
- The properties of the protocol are defined in the presence of this “malicious” coalition of parties that attempts to disrupt the process for the “honest” parties.

$$\mathsf{H}, |\mathsf{H}| = n - t$$

The consensus problem

High level requirement

agree on the output



Study initiated by Lamport, Pease, Shostak 1982

Consensus Properties

- Termination $\forall i \in \mathsf{H}(u_i \text{ is defined})$

Consensus Properties

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

Consensus Properties

- Termination $$\forall i \in \mathsf{H}(u_i \text{ is defined})$$
- Agreement $$\forall i, j \in \mathsf{H}\,(u_i = u_j)$$
- Validity $$\exists v(\forall i \in \mathsf{H}\,(v_i = v)) \implies (\forall i \in \mathsf{H}\,(u_i = v))$$

Consensus Properties

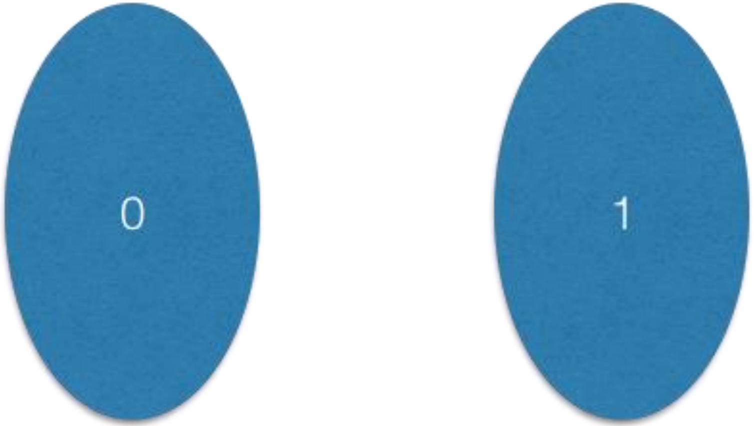
- Termination $\forall i \in \mathsf{H}(u_i \text{ is defined})$
- Agreement $\forall i, j \in \mathsf{H}\,(u_i = u_j)$
- Validity $\exists v(\forall i \in \mathsf{H}\,(v_i = v)) \implies (\forall i \in \mathsf{H}\,(u_i = v))$
  - Strong Validity $\forall i \in \mathsf{H}\,\exists j \in \mathsf{H}\,(u_i = v_j)$

Honest Majority is Necessary, I

Consider an adversary that performs one of the following with probability ⅓

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

$\emptyset$



$\forall i \in A_0 : (v_i = 0)$ $\forall i \in A_1 : (v_i = 1)$

Honest Majority is Necessary, II

- If consensus protocol secure:
  - Adversary corrupts A_0: output of honest parties (that belong to A_1) should be 1.
  - Adversary corrupts A_1: output of honest parties (that belong to A_0) should be 0.
  - Adversary corrupts no-one: output of all parties should be the same.
- Adversary corrupts each set with prob. ⅓ and instructs corrupted parties to follow the protocol
  - honest parties cannot distinguish between honest/corrupted parties
- If all parties output same value: validity is violated with prob. at least ⅓
- If all parties output different value: consistency is violated with prob. at least ⅓

Is Honest Majority Sufficient?

- Two important scenarios have been considered in the consensus literature.
  - Point to point channels. **No setup.**
  - Point to point channels. **With setup.**

- The setup provides a correlated private initialization string to each participant; *it is assumed* to be honestly produced.

Setup and Network

Setup/Network	Synchrony	Partial Synchrony
No Setup	$t < n/3$	$t < n/3$
With Setup	$t < n/2$	$t < n/3$

We know consensus can be achieved, assuming the above bounds on adversarial parties.

The typical setup and network configuration in classical consensus protocols

- Setup: a public-key directory
  - Parties have signing and verification keys for a digital signature scheme.
  - Each party knows every other party’s verification key.
- Network: point-to-point channels
  - Synchronous, partially synchronous, or asynchronous

Bitcoin Consensus

Enter Bitcoin (2008-09)

- Important concepts used by Bitcoin
  - blockchain data structure
  - proof of work (POW)
- Both known and studied earlier, but combined for a novel application

The setup and network configuration in Bitcoin

- Setup: a random (unpredictable) string
  - The blockchain protocol runs without relying on public-key crypto
- Network: peer-to-peer diffusion
  - Synchronous for at least a small subset of the participants (that may be evolving over time).

The Bitcoin Setting for Consensus

- Also referred to as the "permissionless" setting.
- The bitcoin setting is 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: a genesis block

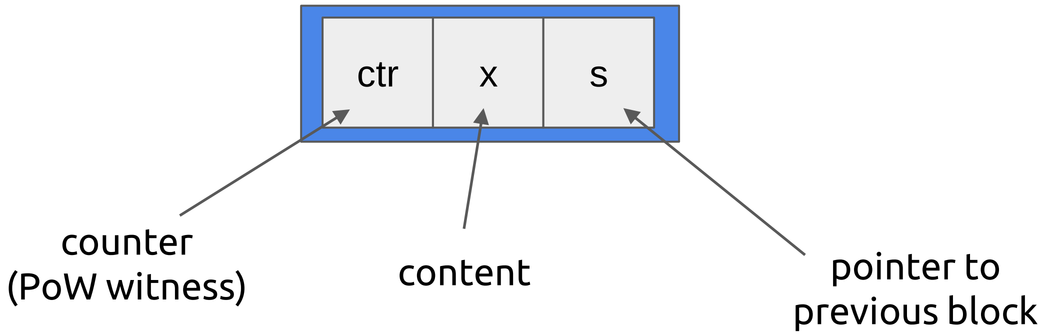
The Bitcoin “backbone”

- 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”

[GKL2015] Garay, Kiayias, Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications.

Model

- Assume there are *n* parties running the protocol
- Synchronous
- Each party has a quota of *q* queries to the function H(.) in each round
- A number of *t* parties are controlled by an adversary (a malicious coalition)
  - Security arguments are *for any adversary*


ctr
x
s
counter
(PoW witness)
content
pointer to
previous block

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

```
function pow(x, C)
    if C = ε then                              ▷ Determine proof of work instance
        s ← 0
    else
        ⟨s′, x′, ctr′⟩ ← head(C)
        s ← H(ctr′, G(s′, x′))
    end if
    ctr ← 1
    B ← ε
    h ← G(s, x)
    while (ctr ≤ q) do
        if (H(ctr, h) < T) then                ▷ This H(·) invocation subject to the q-bound
            B ← ⟨s, x, ctr⟩
            break
        end if
        ctr ← ctr + 1
    end while
    C ← CB                                     ▷ Extend chain
    return C
end function
```

Blockchain

$B_0 = \langle \bot, x_0, ctr_0 \rangle$ ← genesis block

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

…

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

$\mathcal{C} = \langle B_0, B_1, \ldots, B_n \rangle$

chain

head

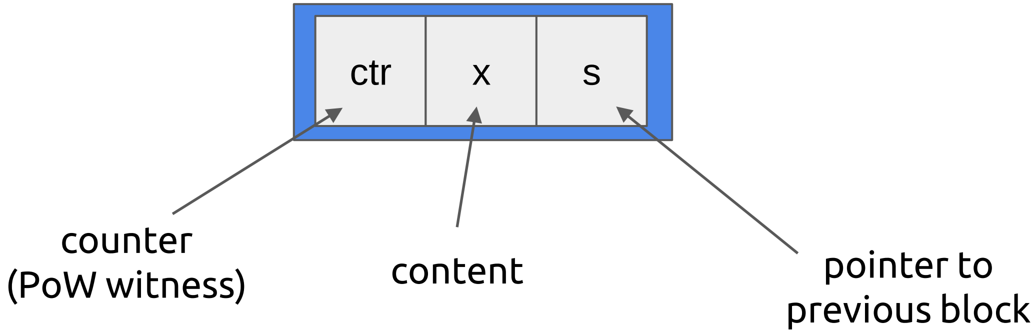
$$s_i = H(ctr_{i-1}, G(x_{i-1}, s_{i-1}))$$

$$x_{\mathcal{C}} = \langle x_0, x_1, \ldots, x_n \rangle$$

$$\mathcal{C}^{\lceil k} = \langle B_0, B_1, \ldots, B_{n-k} \rangle$$

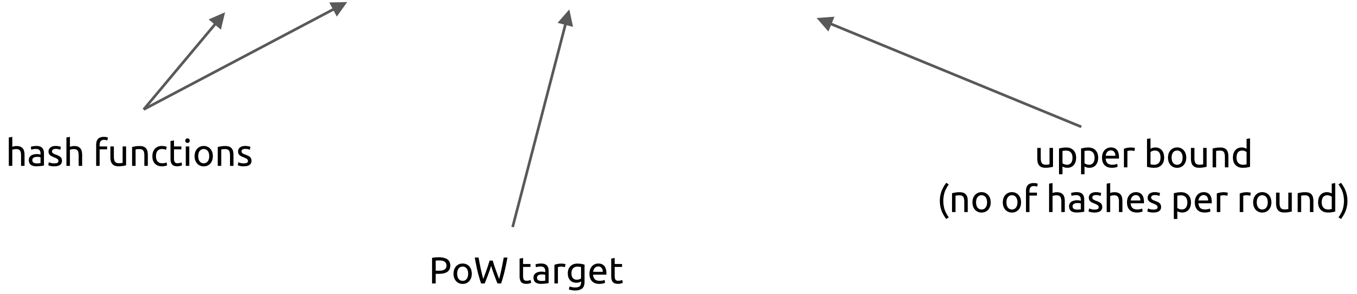
Algorithm 1 The *chain validation predicate*, parameterized by q, T, the hash functions $G(\cdot), H(\cdot)$, and the *content validation predicate* $V(\cdot)$. The input is $\mathcal{C}$.

function validate($\mathcal{C}$)
    $b \leftarrow V(\mathbf{x}_{\mathcal{C}})$
    if $b \wedge (\mathcal{C} \neq \varepsilon)$ **then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
        $\langle s, x, ctr \rangle \leftarrow \mathrm{head}(\mathcal{C})$
        $s' \leftarrow H(ctr, G(s, x))$
        repeat
            $\langle s, x, ctr \rangle \leftarrow \mathrm{head}(\mathcal{C})$
            if $\mathsf{validblock}_q^T(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$ **then**
                $s' \leftarrow s$ ▷ Retain hash value
                $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove the head from $\mathcal{C}$
            else
                $b \leftarrow$ False
            end if
        until $(\mathcal{C} = \varepsilon) \vee (b = \text{False})$
    end if
    return (b)
end function



validblock predicate:

$$(H(ctr, G(x, s)) < T) \wedge (ctr \leq q)$$



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

function maxvalid($\mathcal{C}_1, \ldots, \mathcal{C}_k$)
    $temp \leftarrow \varepsilon$
    for $i = 1$ to k **do**
        if validate($\mathcal{C}_i$) **then**
            $temp \leftarrow \max(\mathcal{C}_i, temp)$
        end if
    end for
    return $temp$
end function

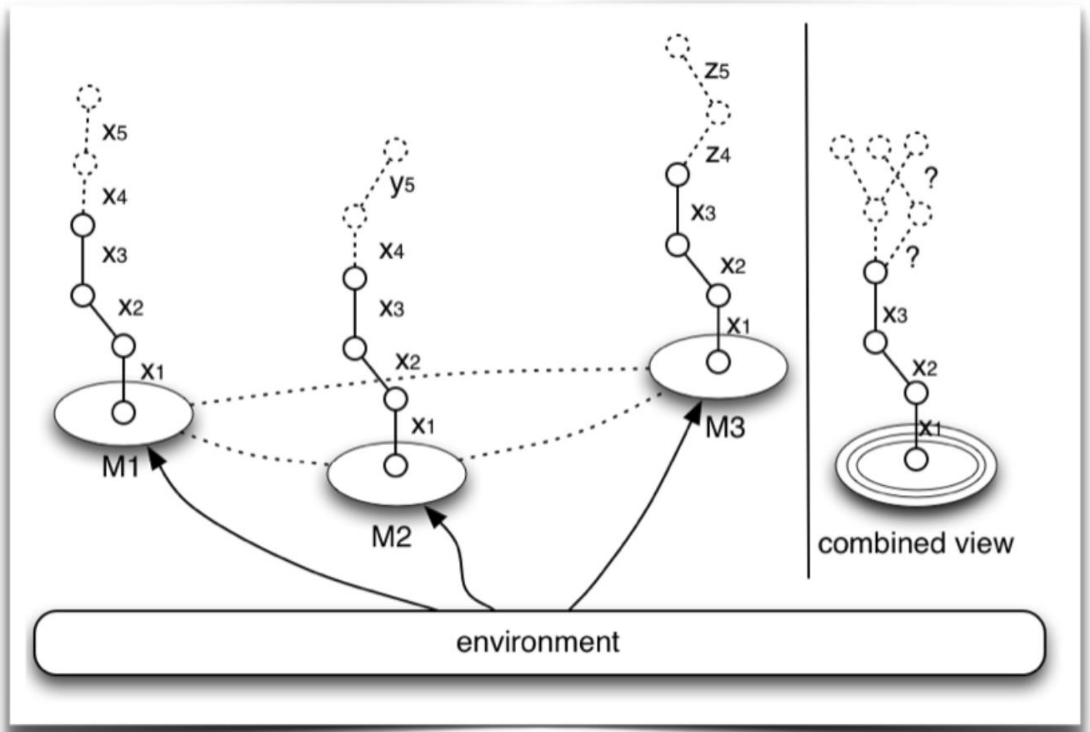
Algorithm 4 The Bitcoin backbone protocol, parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$. At the onset it is assumed "init= True".

if (init) **then**
    $\mathcal{C} \leftarrow \varepsilon$
    $st \leftarrow \varepsilon$
    $round \leftarrow 1$
    $init \leftarrow$ False
else
    $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}$, any chain $\mathcal{C}'$ found in $\textsc{Receive}())$
    if $\textsc{Input}()$ contains $\textsc{Read}$ **then**
        write $R(\tilde{\mathcal{C}})$ to $\textsc{Output}()$ ▷ Produce necessary output before the POW stage.
    end if
    $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \textsc{Input}(), \textsc{Receive}())$ ▷ Determine the x-value.
    $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
    if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
        $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
        $\textsc{Diffuse}(\mathcal{C})$ ▷ Broadcast the chain in case of adoption/extension.
    else
        $\textsc{Diffuse}(\perp)$ ▷ Signals the end of the round to the diffuse functionality.
    end if
    $round \leftarrow round + 1$
end if

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***

Racing Attacks

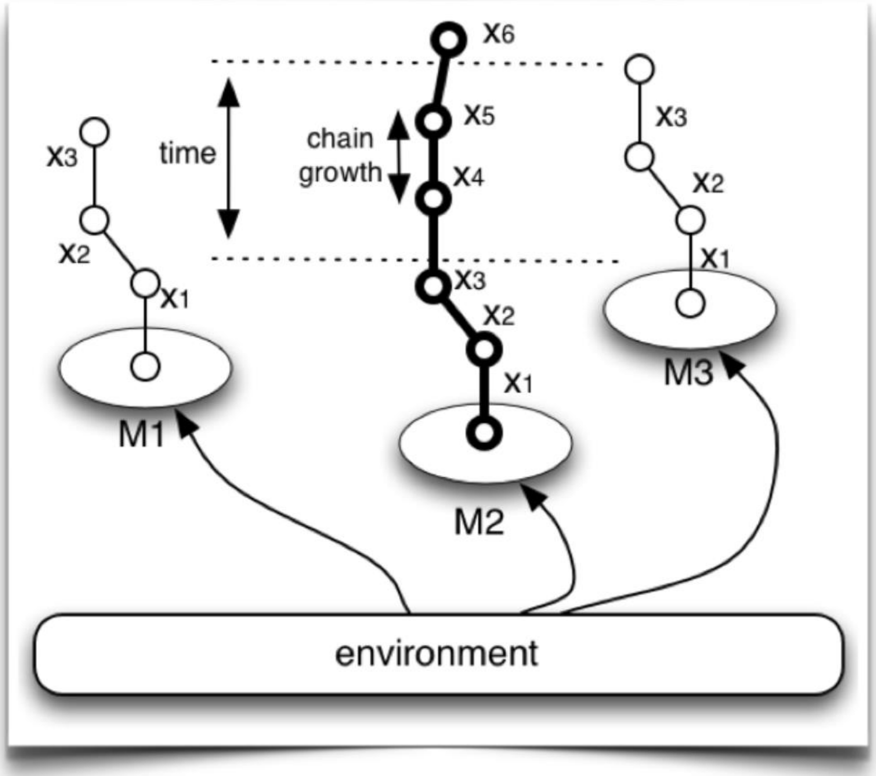
- Attacker splits from the main chain and tries to overtake the “honest chain”

  => Common prefix breaks

- Intuition why the attack is a small probability event:

  concentration bounds help honest parties

Chain Growth, I



Chain Growth, II

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

In any period of *s* rounds at least τs blocks are added to the chain of an honest party P.

- The property holds true in a probabilistic sense with an error probability that exponentially decays in ***s***

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

Abstention Attacks

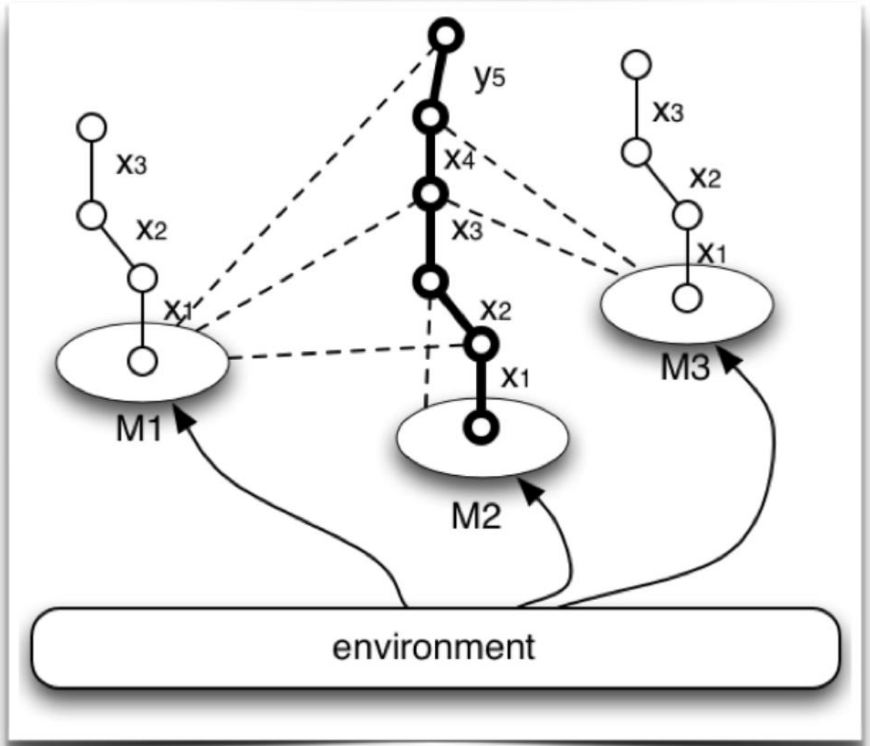
- Attacker stops producing blocks

  => Chain growth stops

- Intuition why the attack is a small probability event:

  honest parties will eventually issue blocks

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 $(1 - \mu)\ell$

- The property holds true probabilistically with an error that exponentially decays in $\boldsymbol{\ell}$

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

Block Withholding Attacks

- Attacker mines privately and releases their block at the same time an honest party releases its own block
- Assuming honest propagation favours the adversary, the honest block is dropped, reducing chain quality

- <u>Intuition why the attack is a small probability event:</u>

  over time the adversary cannot produce blocks at the same rate as honest parties (to compete with them)

Robust Transaction Ledger (RTL) - Ledger Consensus

- It can be shown that the three properties can provide a ledger with two core characteristics
- **Persistence**: Transactions are organized in a *“log”* and honest nodes agree on it.
- **Liveness**: New transactions are included in the log, after a suitable (upper-bounded) period of time.

Establishing a RTL from a Blockchain

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

Ledger Consensus vs. Consensus

- What is the connection?
  - ledger is an ever-going protocol with inputs (e.g., transactions) continuously coming from also external sources
  - consensus is a one-shot execution
- Is it possible to reduce consensus to the ledger? Is it possible to reduce the ledger to consensus?
  - (See the GKL paper for more details)

Hash operations

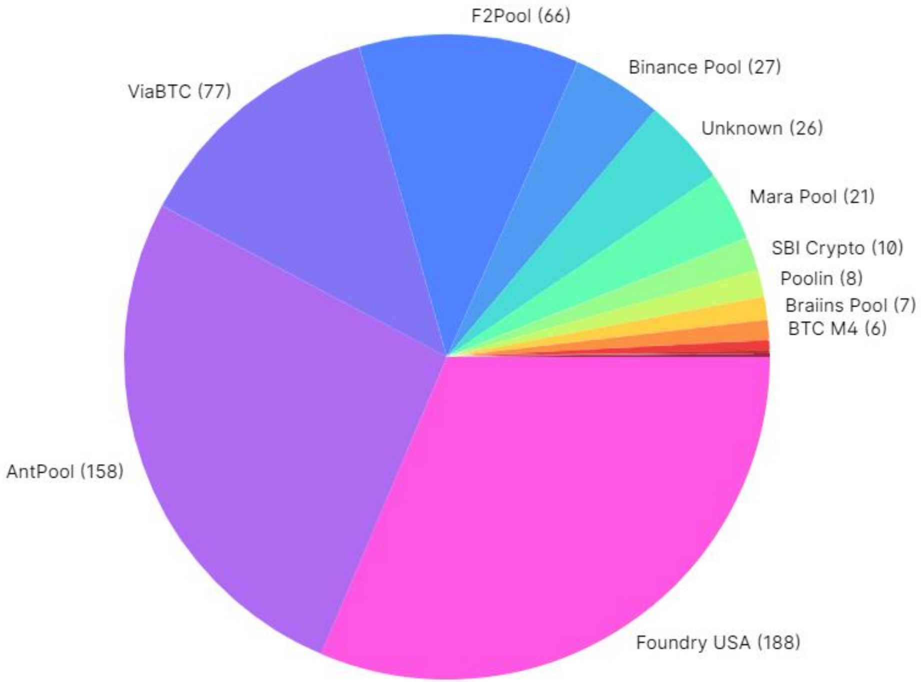
- Consider a regular PC (30 MHash / sec)
- With expectation of 2^{74} hashing operations, mining a block will require ~ 20 *million* years.

https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison

Parallelising mining

- Bitcoin’s Proof of Work can be parallelized
- Parties tend to form mining *pools*
    - Instead of working separately, work **together** to solve PoW for the same block.
    - By collecting **“shares”** (small hashes of the block that are not quite as small as needed) one can prove how much they contributed.

Bitcoin mining pools



https://www.blockchain.com/pools

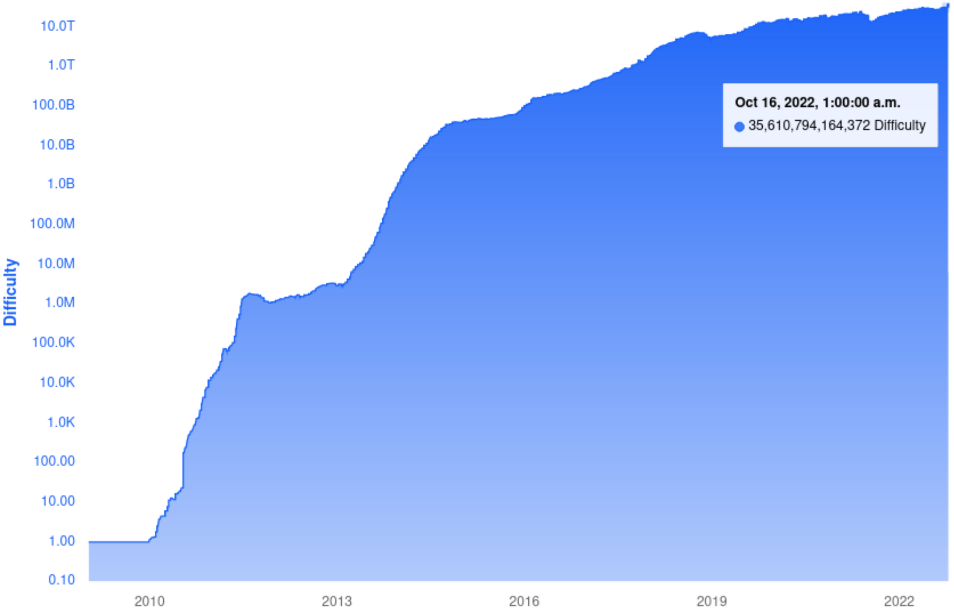
Recall: PoW algorithm

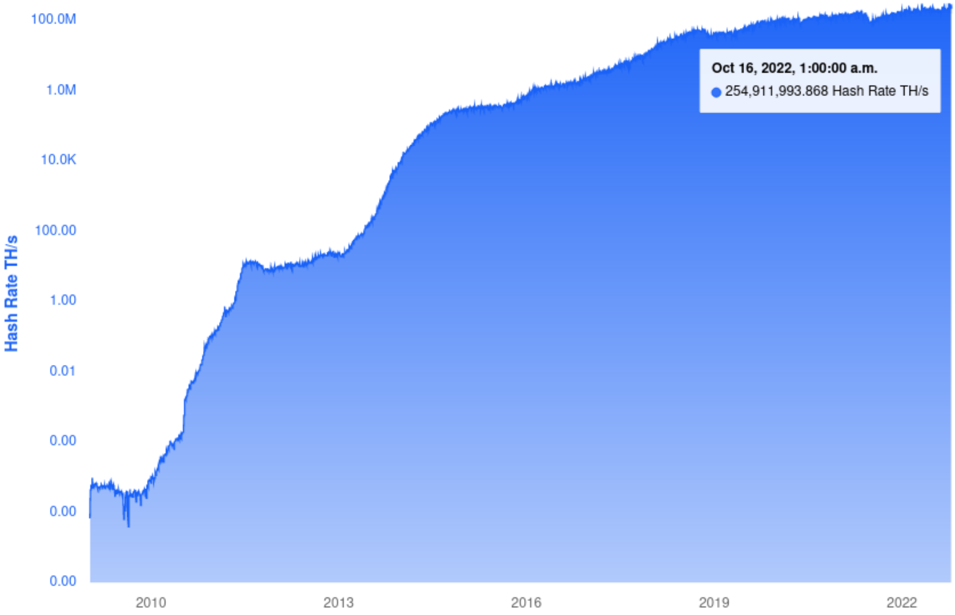
```
int counter;
counter = 0
while Hash(data, counter) > Target
    increment counter
return counter
```

Dynamic Availability

- So far: *n* nodes maintain the blockchain
- This number may change over time:
  - new users enter the system
  - existing users leave
- The change over time can be dramatic
- The Bitcoin blockchain handles this, by adjusting the target (difficulty) of the Proof of Work algorithm

Target difficulty / Total hash rate over time





Adjusting the difficulty

"maxvalid" rule is changed

s.t. parties adopt **chain with highest difficulty** linearly related to:

$$\sum_i \frac{1}{T_i}$$

The *f* parameter [GKL15]

f = probability of producing a block in a round of interaction

- f depends on:
  - target T
  - number of miners
  - duration of round
- If f becomes too small, parties do not progress
  - Chain growth slows
  - Liveness is hurt
- If f becomes too large, parties “collide” often
  - Attacker can exploit network scheduling of message delivery to create forks
  - Persistence is hurt
- To resolve this dynamically, Bitcoin **recalculates** T to keep f constant

Target recalculation

$$\text{next target} = \begin{cases} \frac{1}{\tau} \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 < \frac{1}{\tau} \cdot T; \\ \tau \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 > \tau \cdot T; \\ \frac{n_0}{n} \cdot T_0 & \text{otherwise} \end{cases}$$

- Recalculation occurs at the end of every “epoch”
  - m: epoch length in blocks (in Bitcoin: 2016)
- n_0: estimation of number of ready parties at the system’s onset (party=CPU)
- T_0: initial target
- τ: recalculation threshold parameter (in Bitcoin: 4)
- T: target in effect

- $n = m/(pT\Delta)$: the “effective” number of parties in the epoch
  - Δ: last epoch’s duration based on block timestamps
  - pT: probability of a single party being successful in PoW in a round

Clay pigeon shooting game

Clay pigeon shooting game

- Suppose you shoot on targets successively against an opponent
  - your success probability: 0.3
  - your opponent's success probability: 0.4
  - you shoot in sequence 1000 targets
  - winner is the one that got the most hits
- What is your probability of winning?

Chernoff Bounds

Let: $\delta > 0, \mathbf{Prob}[X_i = 1] = p_i, \mu = \sum_{i=1}^{n} p_i$

Then:

$$\mathbf{Prob}[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/(2+\delta))$$

$$\mathbf{Prob}[\sum_{i=1}^{n} X_i \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2), \delta \in (0,1)$$

Analysis, I

- You have an expectation of 300 hits
- Your opponent has an expectation of 400 hits
- What is *your* probability of winning?
  - Denote by X whether you hit a target, similarly Y for your opponent
  - From Chernoff bounds:

$$\mathbf{Pr}[\sum_{i=1}^{1000} X_i \geq 345] \leq \exp(-(0.15)^2 300/2.15) < 4.3\%$$

$$\mathbf{Pr}[\sum_{i=1}^{1000} Y_i \leq 348] \leq \exp(-(0.13)^2 400/2) < 3.5\%$$

Analysis, II

- You have an expectation of 300 hits
- Your opponent has an expectation of 400 hits
- What is *your* probability of winning?
  - Denote by X whether you hit a target, similarly Y for your opponent
  - From Chernoff bounds:

$$\mathbf{Pr}[\sum_{i=1}^{1000} X_i \geq 345] \leq \exp(-(0.15)^2 300/2.15) < 4.3\%$$

$$\mathbf{Pr}[\sum_{i=1}^{1000} Y_i \leq 348] \leq \exp(-(0.13)^2 400/2) < 3.5\%$$

- If the negation of both events happens, you will certainly lose:
  - Thus, probability of you winning is less than 8%

$$\mathbf{Pr}[X_{<345} \wedge Y_{>348}] = (1 - \mathbf{Pr}[X_{\geq 345}])(1 - \mathbf{Pr}[Y_{\geq 348}]) \geq \ 92.3\%$$

Analysis, III

- Now you are given a choice:
  - decrease the size of the clay pigeon target by a ratio β
  - augment your “kills” by multiplying with $1/\beta$
  - your accuracy is linear with β
  - your opponent will keep playing in the same way as before
- Do you accept to play like this?

Analysis, IV

- Now you are given a choice:
  - decrease the size of the clay pigeon target by a ratio *β*
  - augment your "kills" by multiplying with 1/β
  - your accuracy is linear with *β*
  - your opponent will keep playing in the same way as before
- Do you accept to play like this?
- Each shot has success probability: $\mathbf{Pr}[X_i' = 1] = \beta \cdot \mathbf{Pr}[X_i = 1]$
- The score expectation of each shot remains: $E[(1/\beta)X_i'] = (1/\beta)\beta E[X_i] = E[X_i]$
- But decreasing *β* results in increased variance → previous argument fails

$$\mathbf{Pr}[\sum_{i=1}^{1000} X_i' \geq 345\beta] \leq \exp(-(0.15)^2 300\beta/2.15)$$

β	bound
1,	~4.3%
0.5,	~20.8%
0.25,	~45.6%
0.10,	~73.1%

The Difficulty Raising Attack

- The recalculation threshold (τ) is essential
- Without it, an adversary that has a minority of hashing power:
    - Creates a private, artificially difficult chain
    - Similar to clay pigeon shooting game, this increases the variance in its block production rate
    - Overcoming the chain of the honest parties becomes a non-negligible event

[B13] Lear Bahack. Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft)