DATA PRIVACY AND SECURITY

Prof. Daniele Venturi

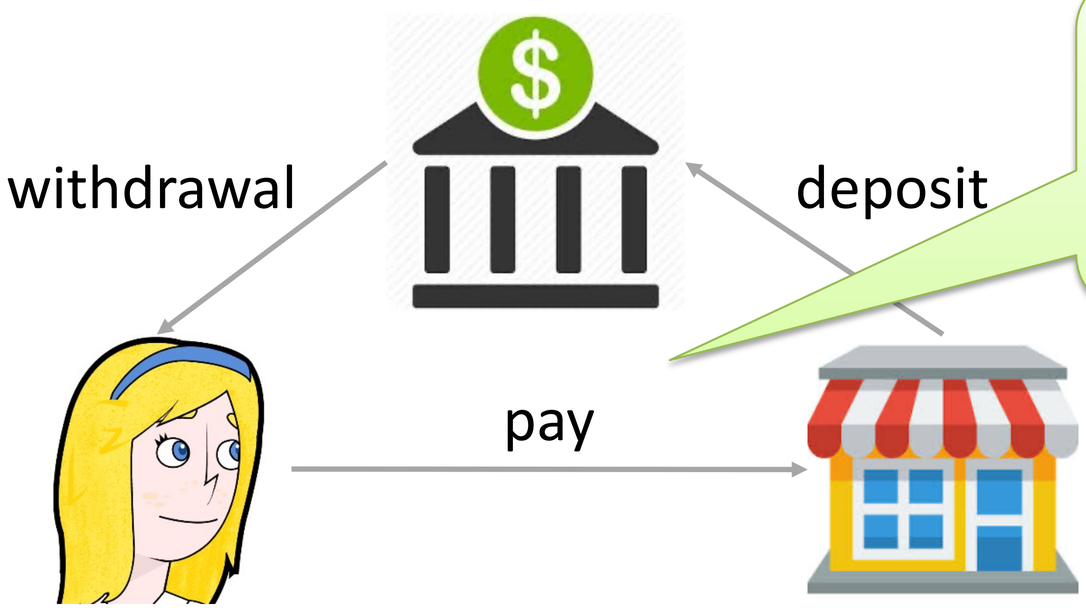
Master's Degree in Data Science
Sapienza University of Rome

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

CHAPTER 6: Bitcoin

History of Digital Cash

- 1990: Chaum's anonymous eCash
  - Uses sophisticated crypto to achieve **security** and user **anonimity**

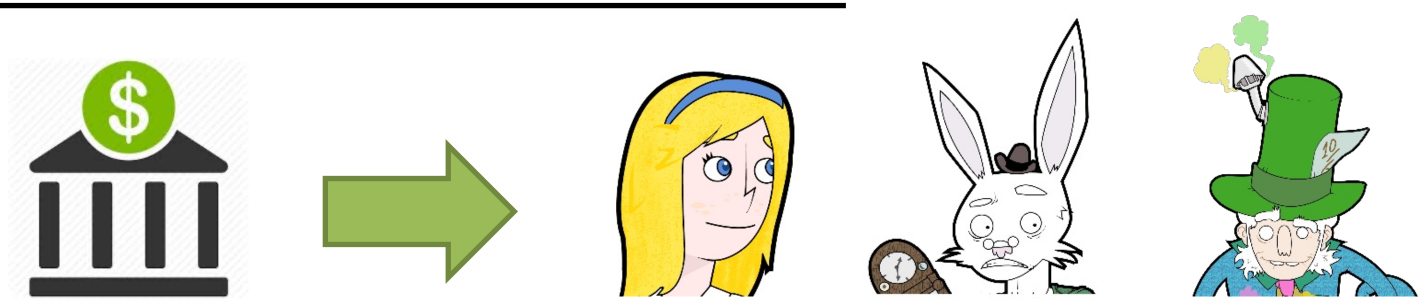


Company founded in 1990… Went **bankrupt** in 1998

History of Digital Cash

- 2008: Bitcoin announced by Satoshi Nakamoto
- 2011-2013: Popular for buying **illegal goods**
  - E.g., Silk Road anonymous marketplace
- End of 2013: Market price skyrockets and the world notices

Main difference with eCash:

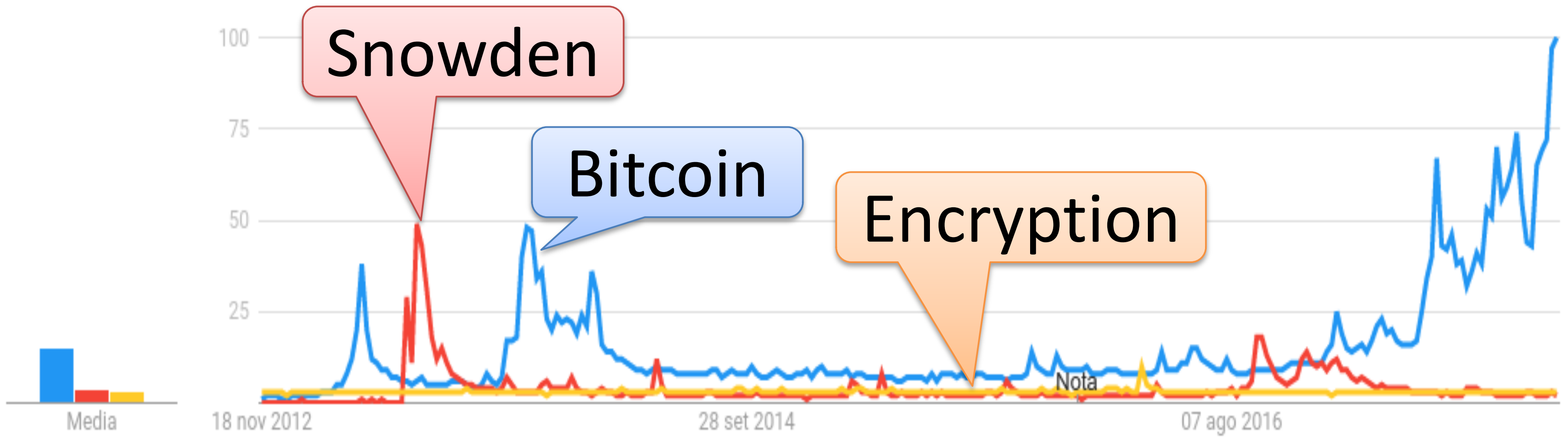


The Bitcoin Revolution

- Problems of earlier ecash systems
  - Need **trusted center** (money does not circulate)
  - **High** transaction fees
- Solutions in Bitcoin ecosystem
  - **Decentralized** system (money circulates)
  - **Variable** transaction fees

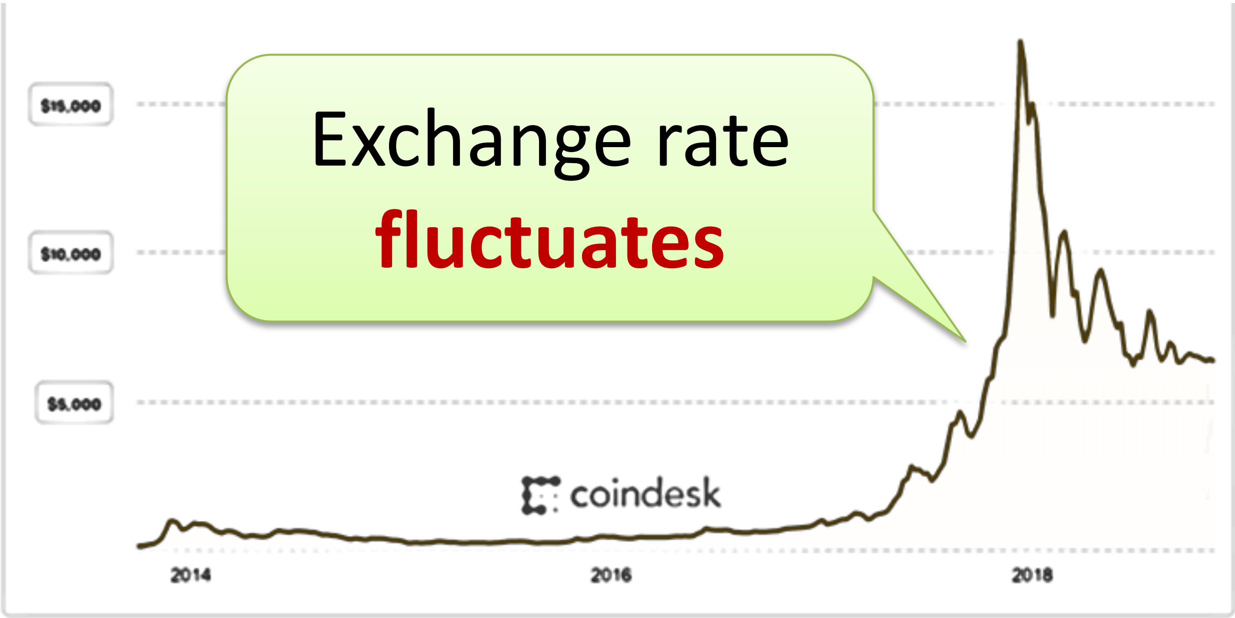
Bitcoin's Success

- Probably one of the most discussed cryptographic technologies ever!



No Trusted Servers!

- Nobody controls the money
  - The amount of money that will ever be created is fixed to around 21 mln Bitcoin (**no inflation**)

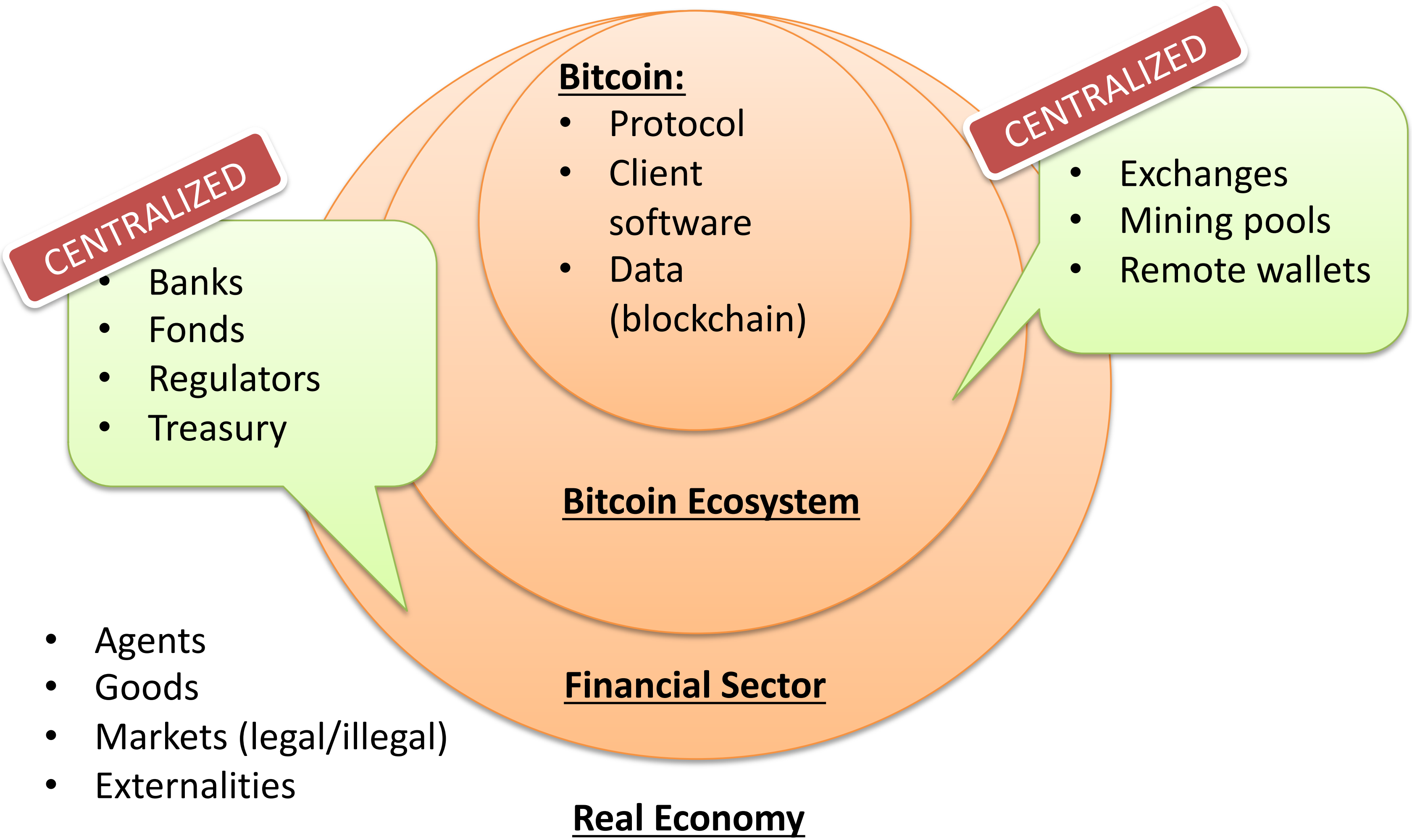


Really No Trusted Server?

- The client software is written by people who are in charge to **change** the system
- Software contains so-called **checkpoints** (more on this later)
- Popular clients:

The people behind the software are **not anonymous**

Bitcoin in Context



Bitcoin ≈ Real Money?

- Bitcoin values comes from the fact that: "People **expect** that other people will accept it in the future."

Enthusiasts:

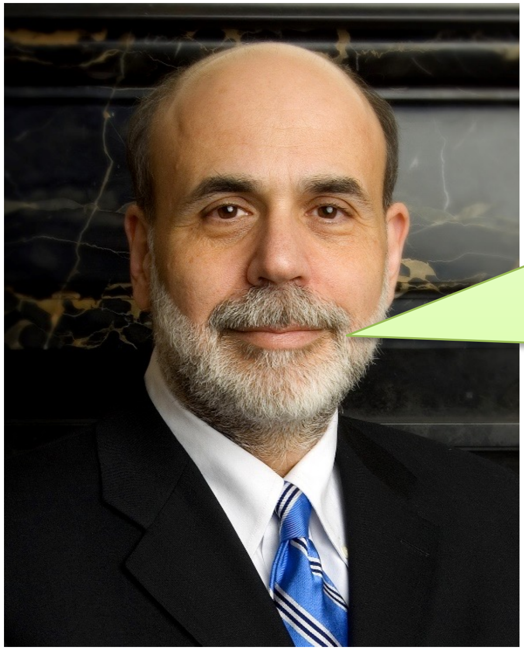
It's like **real money**

Sceptics:

It's a **"Ponzi scheme"**

Some Economist Are More Positive

Ben Bernanke



While these types of innovations may pose risks related to law enforcement and supervisory matters, there are also areas in which they may hold **long-term promise**, particularly if the innovations promote a faster, **more secure** and **more efficient** payment system

- **Billions** of VC funding, many major banks and companies are interested

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Why Bitcoin Became So Popular?

- Ideological reasons
  - Crypto anarchy (**nobody** controls the money)
- Good timing due to financial crisis in 2008
  - **No money printing** in Bitcoin
- Trading of illegal goods due to seeming **anonymity** (pseudonimity)
- Payments can be cheap
  - Almost **no fees** for long time (PayPal 2-10%)
- Novel technology for **distributed systems**

Illegal Market Places

- What is sold?

Category	# of items	% of total
Weed	3338	13.7
Prescriptions	1784	7,3
Books	955	3,9
Cannabis	877	3,6
Cocaine	630	2,6
LSD	440	1,8

- Mostly non-professional sellers
  - Most items only listed for few days
- All markets value: **600.000 USD** per day

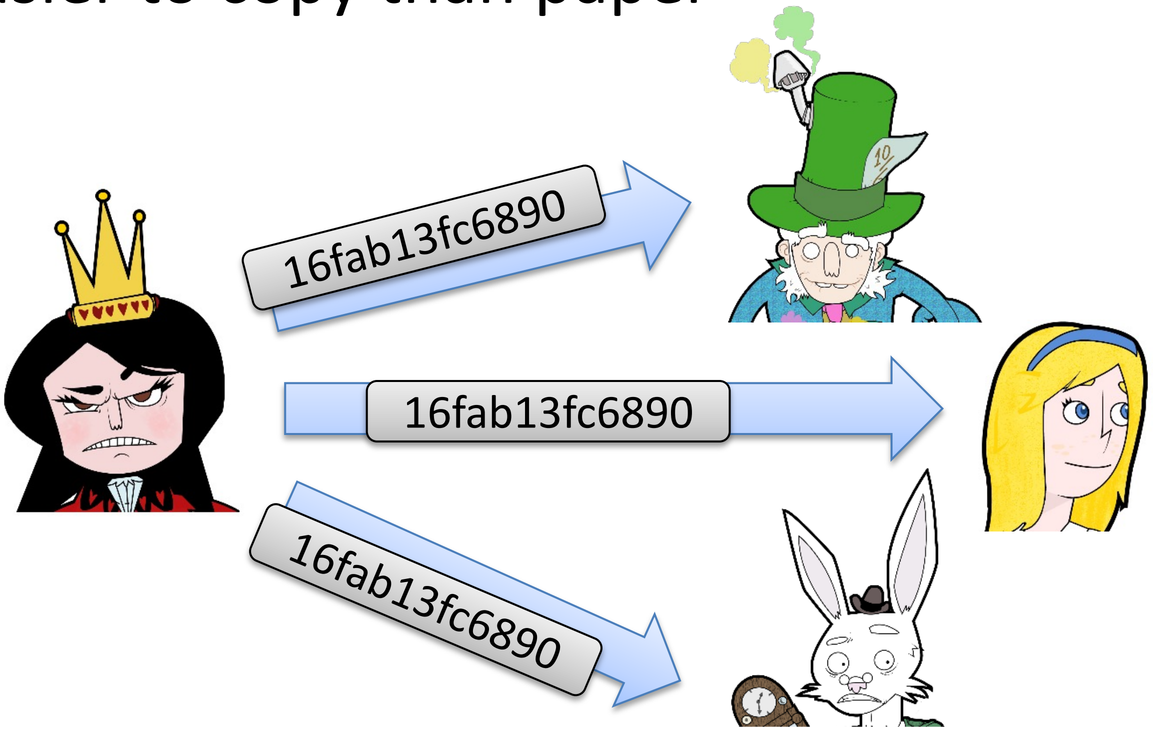
Downsides of Decentralization

- There are **no regulators**
  - MtGox (handling 70% of all Bitcoin transactions) shut down on Feb 2014, reporting 850.000 BTC (**450 million USD**) stolen
- Transactions **cannot be reversed**
  - But see a later lecture for alternatives
- Software bugs immediately exploited as hackers can **make money**
  - Ransomware
  - Virus stealing bitcoins

Design Principles

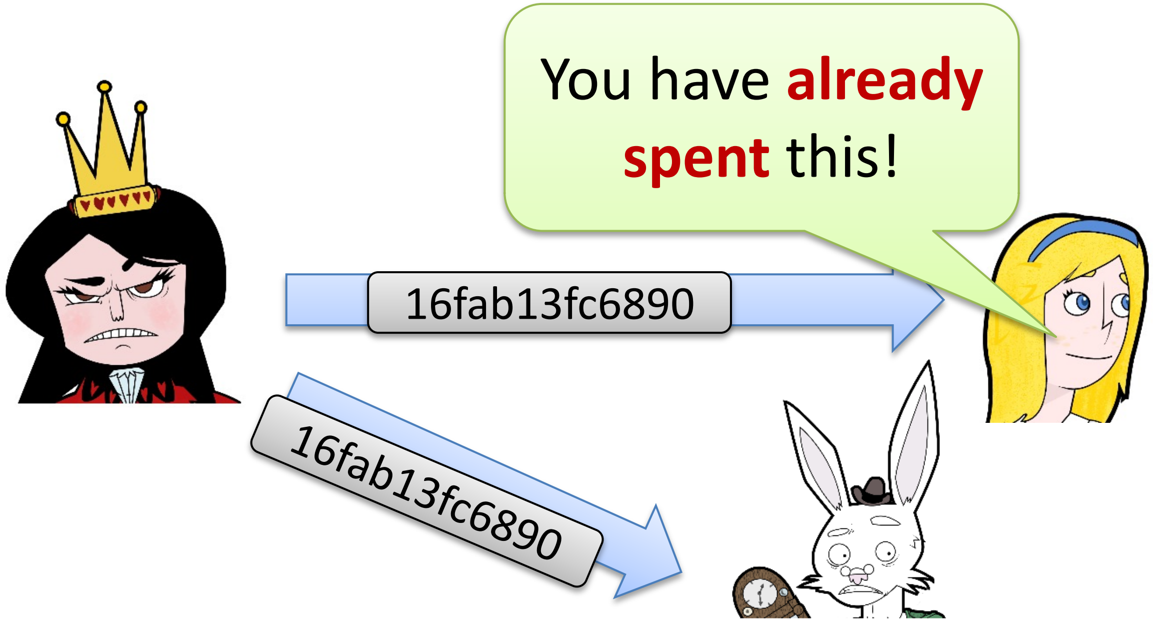
Doublespending

- Main problem with the digital money is that it is **much easier to copy** than real money
  - Bits are easier to copy than paper



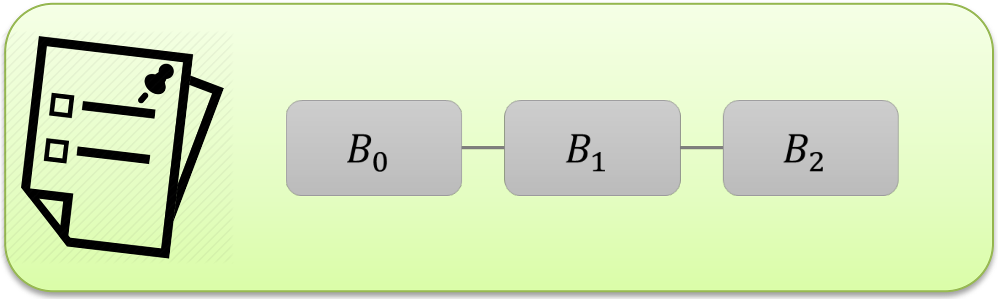
Bitcoin's Idea (Simplified)

- The users **emulate a public bulletin-board** containing a list of transactions
  - A transaction if of the form: "User P_1 transfers a coin #16fab13fc6890 to user P_2"

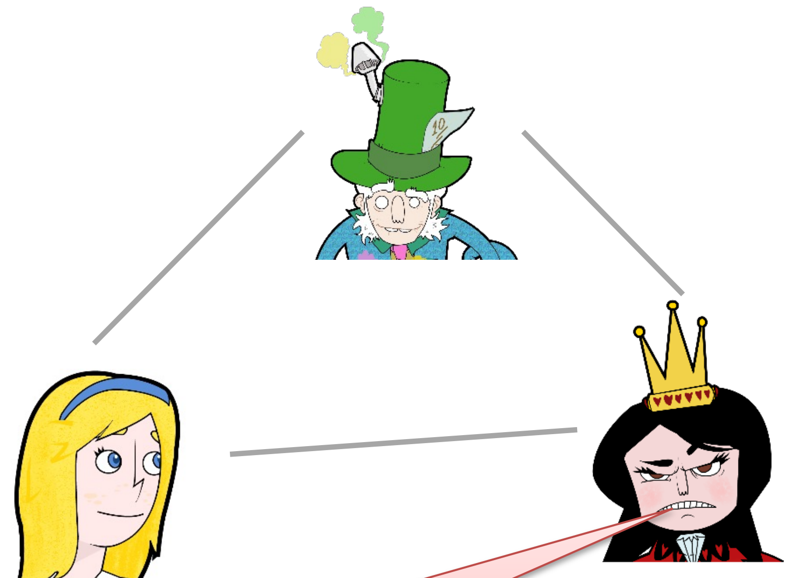


Trusted Bulletin-Board Emulation

Ideal World



Real World



Main difficulty:

Some **parties can cheat**!

An Idea

- Assume **honest majority** and implement the bulletin-board by **voting**
  – Every transaction is **broadcast**

Is this the **correct** bulletin-board?

Transaction id	Value
ddbs21239864k…	0.084 BTC
edd98763hn3nr…	1.2 BTC
mkk8765g4g2j3…	0.036 BTC

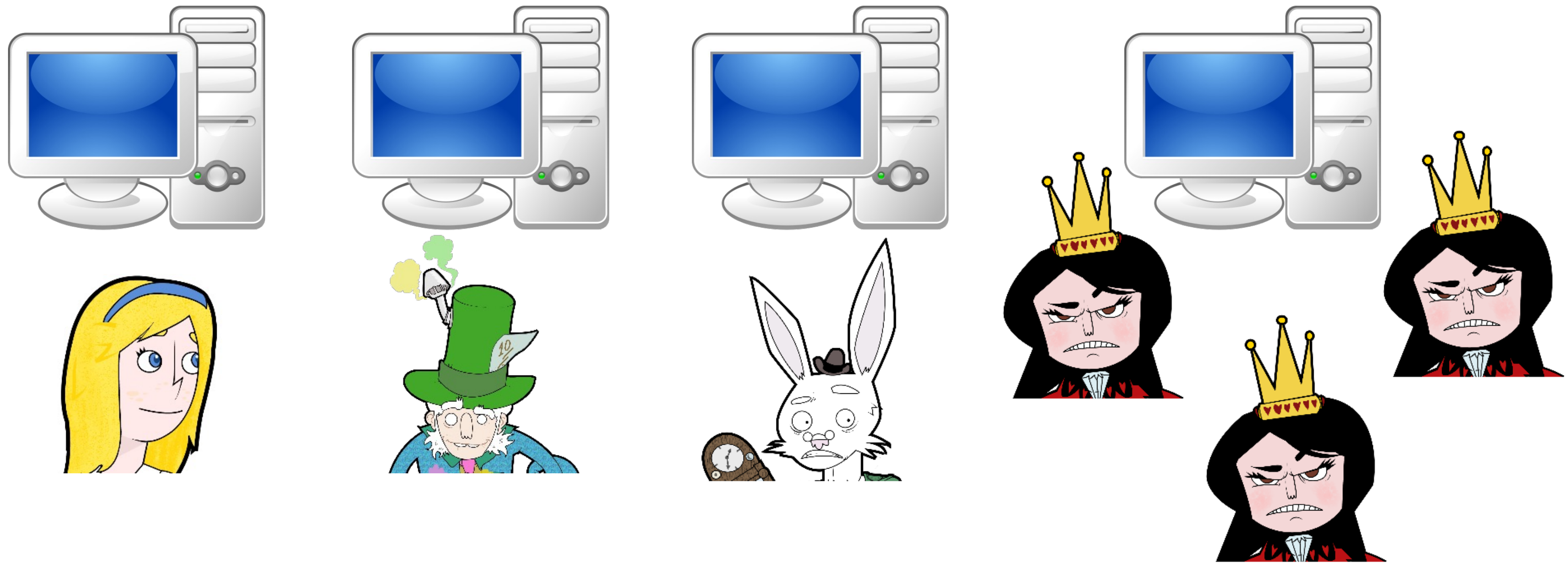
In cryptocurrencies this is called the **consensus protocol**

How to Implement Consensus?

- A very well-studied problem in distributed computing
  - Agreement **requires** honest majority
- Problem: **Sybil attack**
  - How to define majority in a context where **everybody** can join the network?

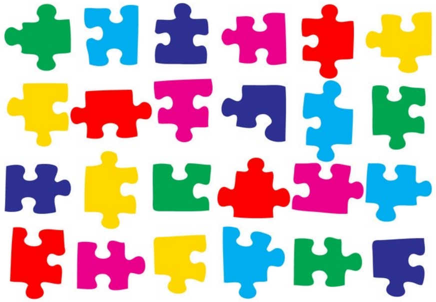
Bitcoin's solution

- Majority = Majority of **computing power**
- Now creating multiple identities does not help

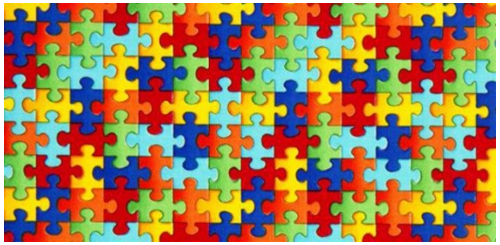


How is this verified?

- Use Proofs of Work (PoW) – Dwork & Naor ‘92
- Basic idea: User solve **moderately hard** puzzle



Hard to find solution

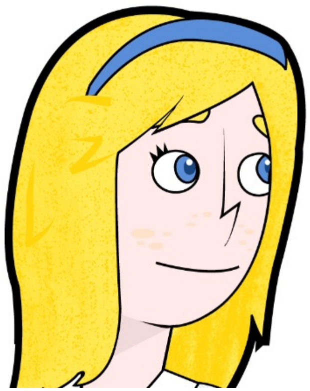


Easy to verify

- Digital puzzle: Use cryptographic hashing
  - Hash function $\mathbf{H}$ with running time $\mathrm{TIME}(\mathbf{H})$
  - Solve: **Find** input s.t. output starts with n zeroes
  - Verify: **Compute** hash

Simple PoW

Hash function $\mathbf{H}$ with running time $\mathrm{TIME}(\mathbf{H})$



Random x ←

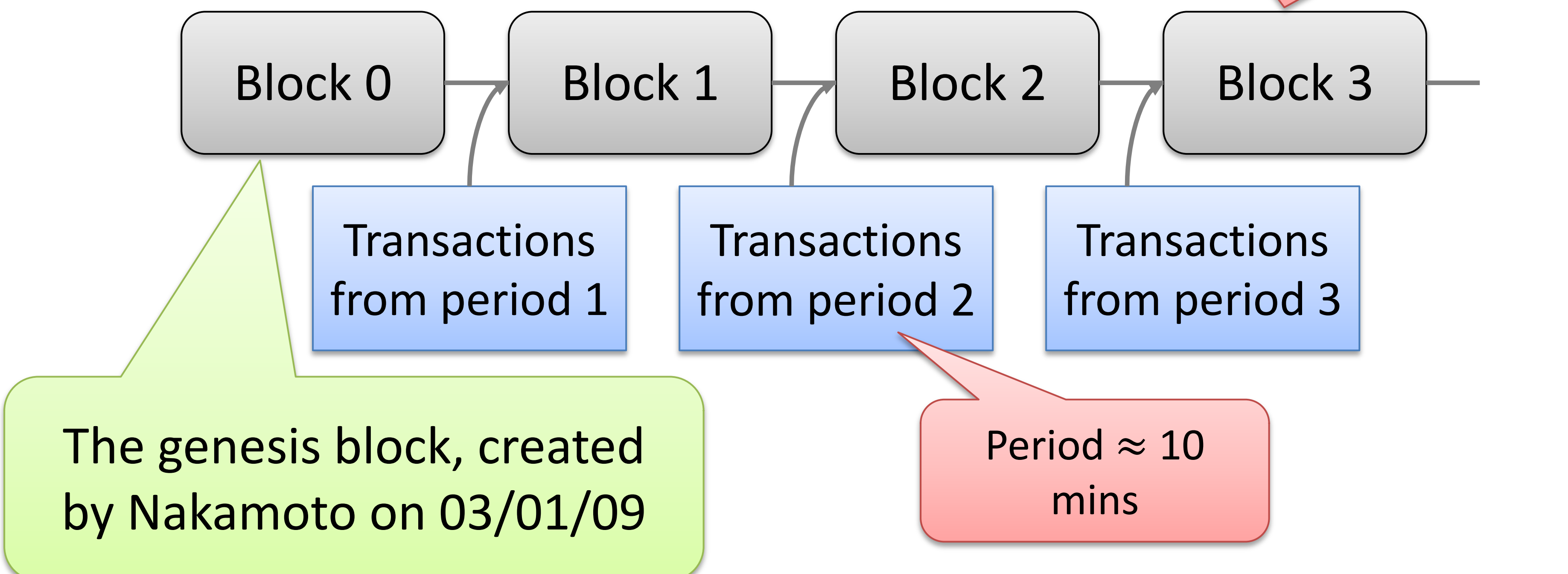
Answer s →

Find s s.t. $\mathbf{H}(s||x)$ starts with n zeroes (time $2^n \cdot \mathrm{TIME}(\mathbf{H})$)

Check that $\mathbf{H}(s||x)$ starts with n zeroes (time $\mathrm{TIME}(\mathbf{H})$)

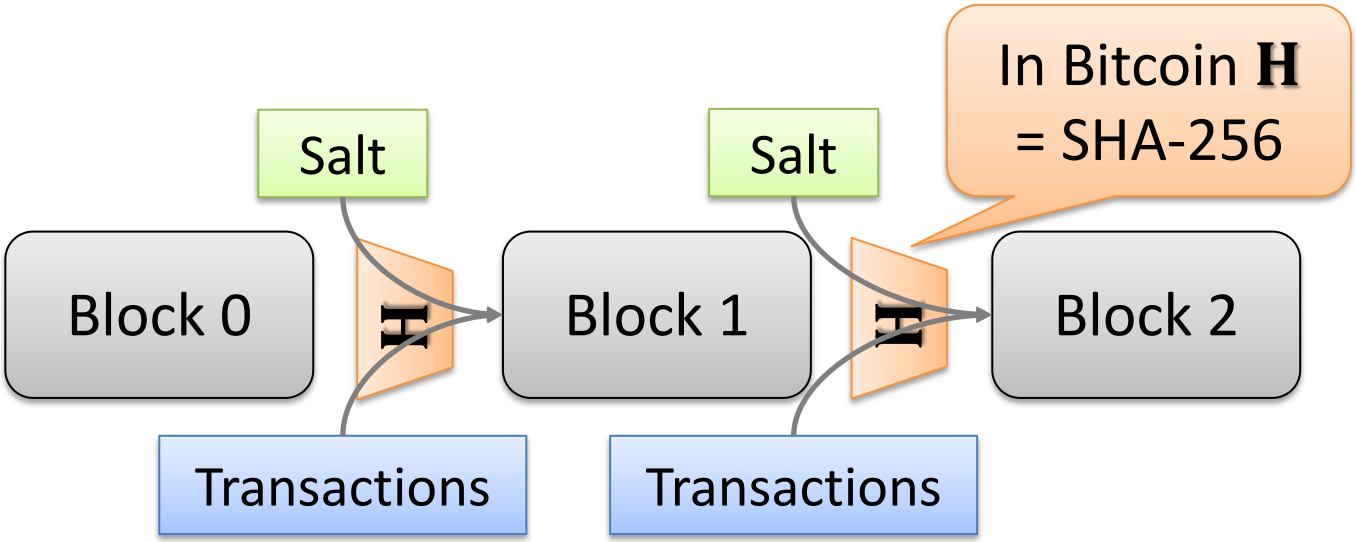
Setup for the Bulletin-Board

- Users maintaining the bulletin-board are called **miners**
- Miners maintain a chain of blocks:



Extending the Blockchain

- The chain is **extended** by using the PoW

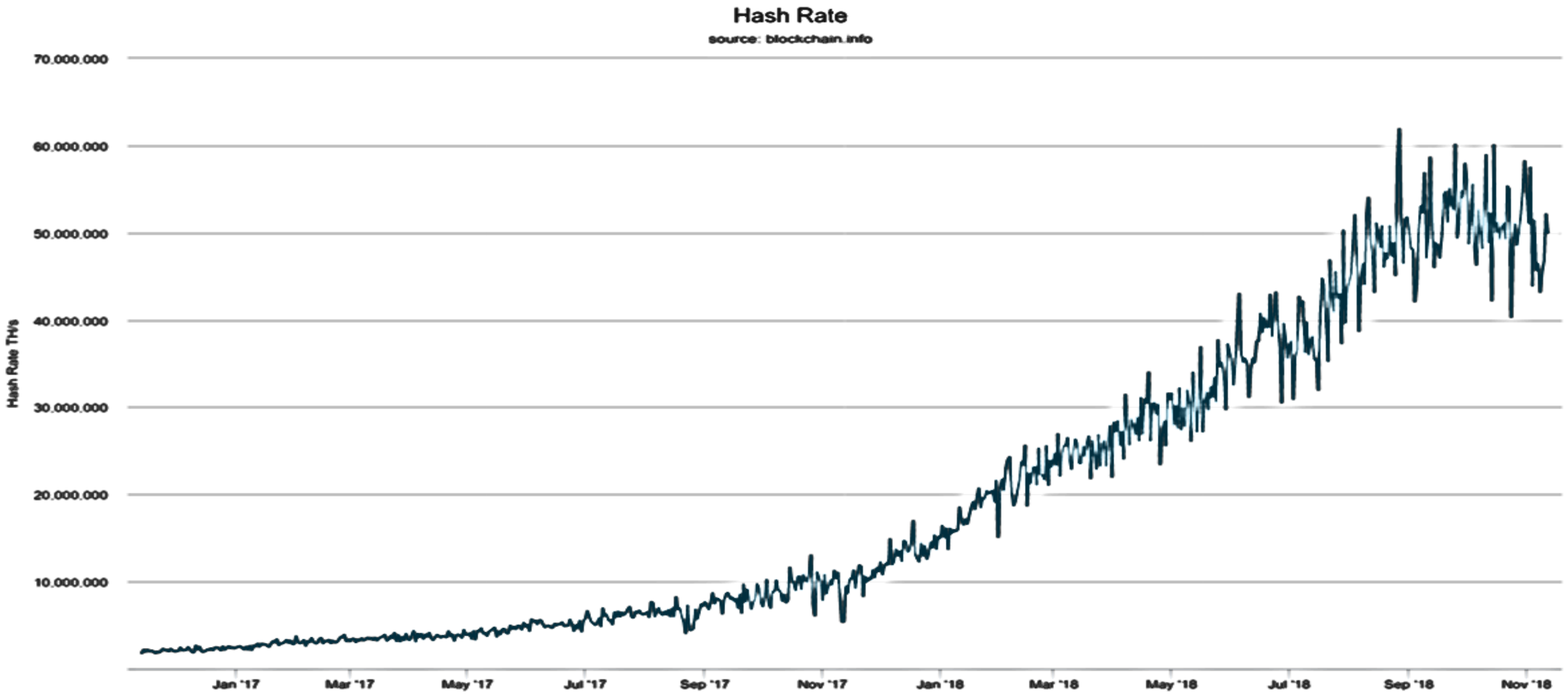


- PoW challenge: $\mathbf{H}(\text{Salt}||\mathbf{H}(\text{Block}_i)||\text{TX})$ starts with n zeroes (**hardness parameter**)

Adjusting the Hardness Parameter

- The computing power of the miners **changes**
- Miners should generate a new block every **10 minutes** (on average)
- Thus the hardness parameter is periodically **adjusted to the mining power**
  - It happens once every **2016 blocks**
  - **Automatic** process, in a way that depends on the time it took to generate the 2016 blocks
  - Possible because each block contains a **timestamp**

Hash Rate



- January 2017: 2,550,000 TH/s
- January 2018: 15,000,000 TH/s
- September 2018: 50,000,000 TH/s

October 2019: 114 EH/s

How it Looks in Real Life

Height	Timestamp	Transactions	Miner	Size
550168	6 minutes ago	2796	DPOOL	1,1 MB
550167	11 minutes ago	2348	BTC.com	1,5 MB
550166	27 minutes ago	2227	…	…
550165	44 minutes ago	…	…	
550164	49 minutes ago	…	…	

Bitcoin Block Reward Halving Countdown

Days	Hours	Minutes	Seconds
942	02	19	01

Reward-Drop ETA date: **13 Jun 2020 16:33:51**

The Bitcoin block mining reward halves every 210,000 blocks, the coin reward will decrease from 12.5 to 6.25 coins.

Total Bitcoins in circulation:	16,679,225
Total Bitcoins to ever be produced:	21,000,000
Percentage of total Bitcoins mined:	79.42%
Total Bitcoins left to mine:	4,320,775
Total Bitcoins left to mine until next blockhalf:	1,695,775
Bitcoin price (USD):	$6,360.50
Market capitalization (USD):	$106,088,210,612.50
Bitcoins generated per day:	1,800
Bitcoin inflation rate per annum:	4.02%
Bitcoin inflation rate per annum at next block halving event:	1.80%
Bitcoin inflation per day (USD):	$11,448,900
Bitcoin inflation until next blockhalf event based on current price (USD):	$10,785,976,888
Total blocks:	494,338
Blocks until mining reward is halved:	135,662

How to Post on the Board

- Broadcast over the internet your transaction to the miners
- **Hope** they will add it to the next block
  – Miners are **incentivized** to do so
- Miners **never** add **invalid transactions** (e.g., doublespending)
  – A chain with an invalid transaction is itself not valid, so no rational miner would do it
- When a miner finds an extension he **broadcasts** it to all the users

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Where Do These Bitcoins Come From?

- A miner that solves the PoW gets a **reward**
  - 50 BTC for the first 210000 blocks ($\approx$ 4 years)
  - 25 BTC for the next 210000 blocks
  - 12.5 BTC for the next 210000 blocks
  - … and so on
- Note that:

$$210000(50 + 25 + 12.5 + \cdots) = 21000000$$

More in Details…

- Each block contains a transaction that transfers the reward to the miner
  - A so-called **coinbase transaction**
- Advantages:
  - It provides an **incentive** to be a miner
  - It makes miners interested in broadcasting the new block as soon as possible

An Important Feature

- Assuming everybody follows the protocol, the following **invariant** is maintained:

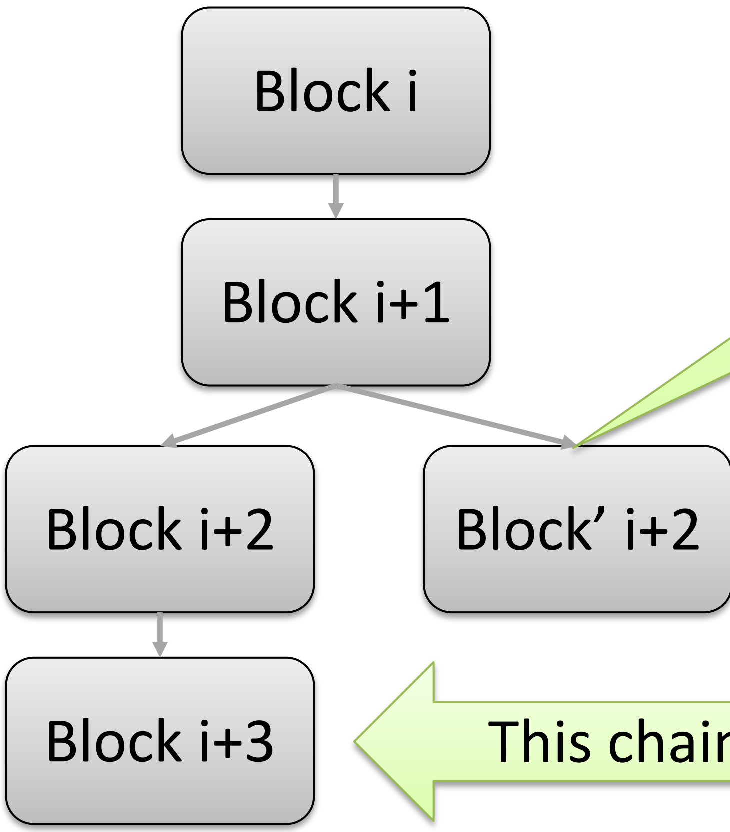
> Every miner P_i whose computing power is a α_i-**fraction** of the total computing power mines a α_i-**fraction** of the blocks

- Fract. of computing power ≈ fract. of revenue
- This is because P_i's chances of solving the PoW are **proportional** to the number of times P_i can evaluate the hash function

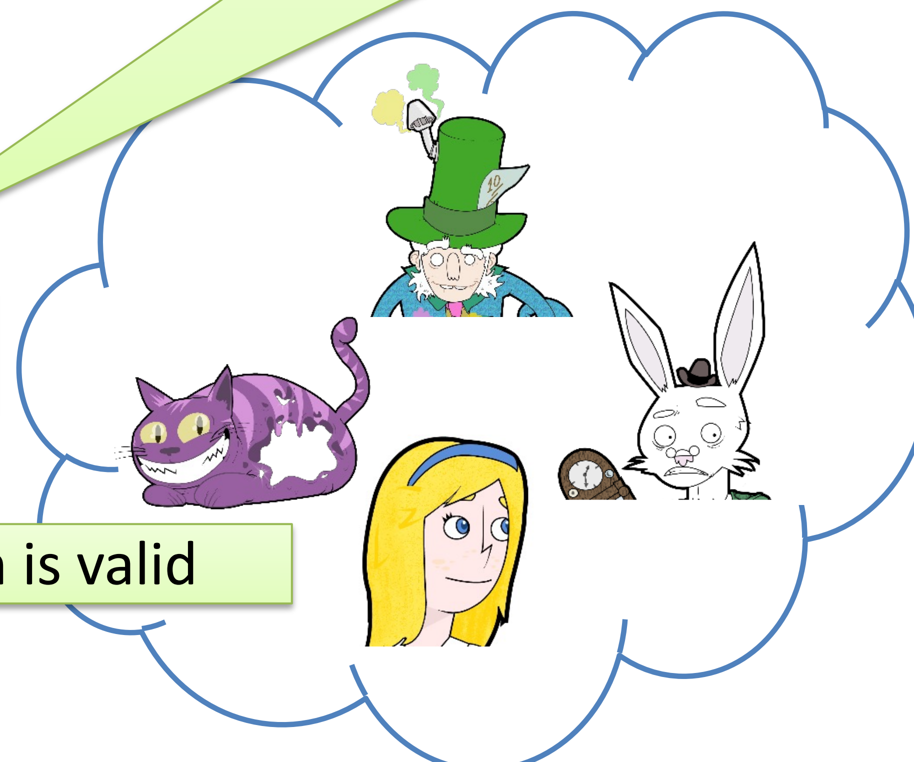
Forks

- The **longest chain** counts!

Makes **no sense** to work on a shorter chain, as everybody else is working on extending the **longest** one



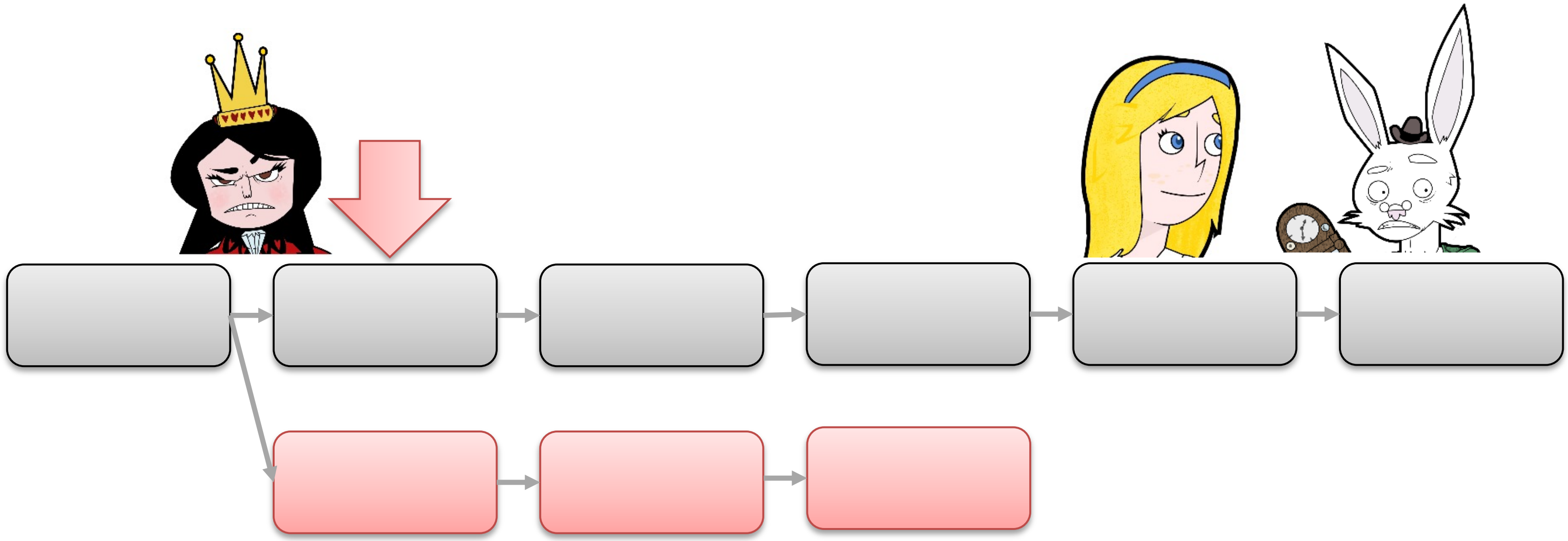
This chain is valid



Consequences

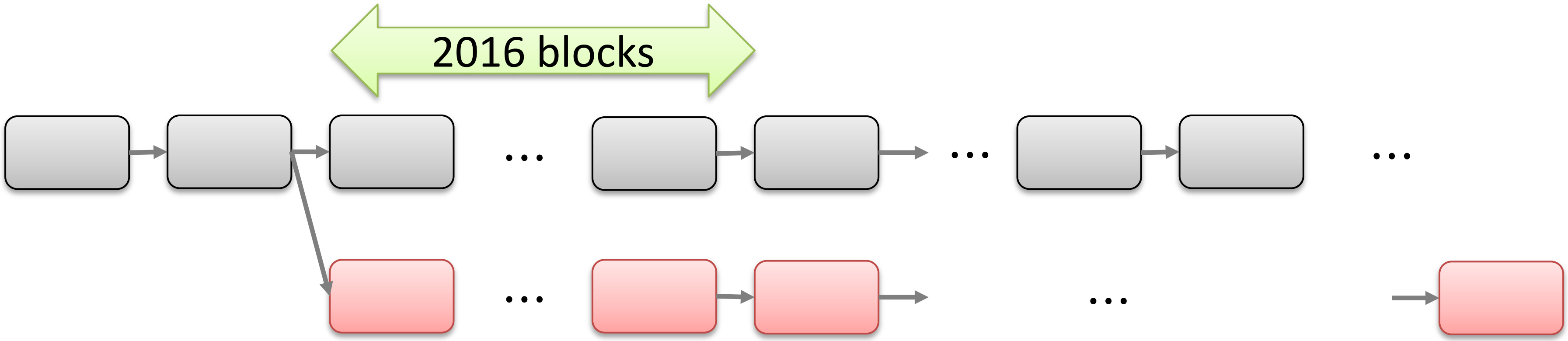
- The system should quickly **self-stabilize**
- If there is a fork, then one branch will die
  - What if your transaction ends up in a **dead branch**?
  - **Recommendation:** To make sure it doesn't happen wait **6 blocks** ($\approx$ 1 hour)

Can Transactions be Reversed?



- Requires a **fork in the past**
  - Unlikely with **minority** computing power
  - Honest miners always ahead of the adversary

Attack based on Hardness Parameter



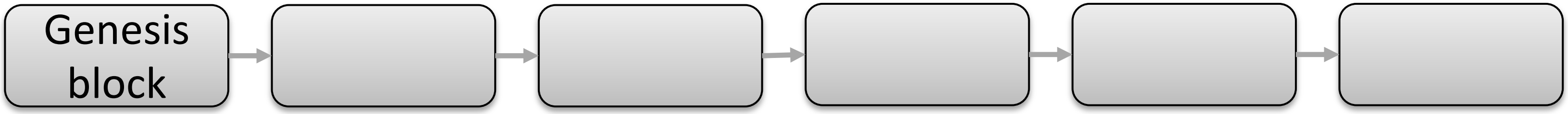
1) **Secretly** compute another chain with **fake timestamps** (indicating that it took a long time to produce it)

2) The difficulty **drops** drammatically, so can quickly produce a chain **longer** than the valid one and publish it

The Strongest Chain

- For this reason, in Bitcoin it is **not the longest** chain that matters, but rather **the strongest**
- Strength of each block is 2^n
- Strength of the chain is the sum of the strength of all blocks
  - This clearly prevents the previous attack

Freshness of the Genesis Block



I did not know the genesis block before Bitcoin was launched (Jan 3, 2009)

Here is a **heuristic** proof: "The genesis block contains a hash of a title from a front page of the London Times on Jan 3, 2009."

Why Does it Matter?

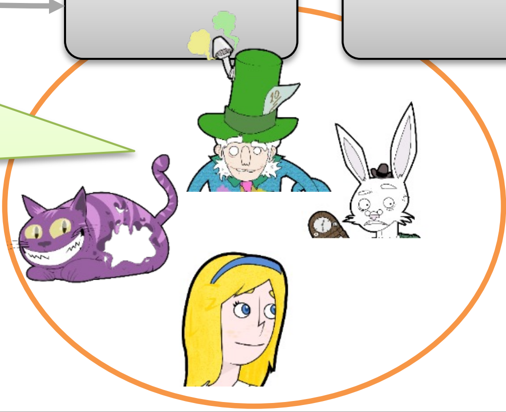
- Otherwise Satoshi could «pre-mine»

3) On Jan 3, 2010 publish **secret chain**

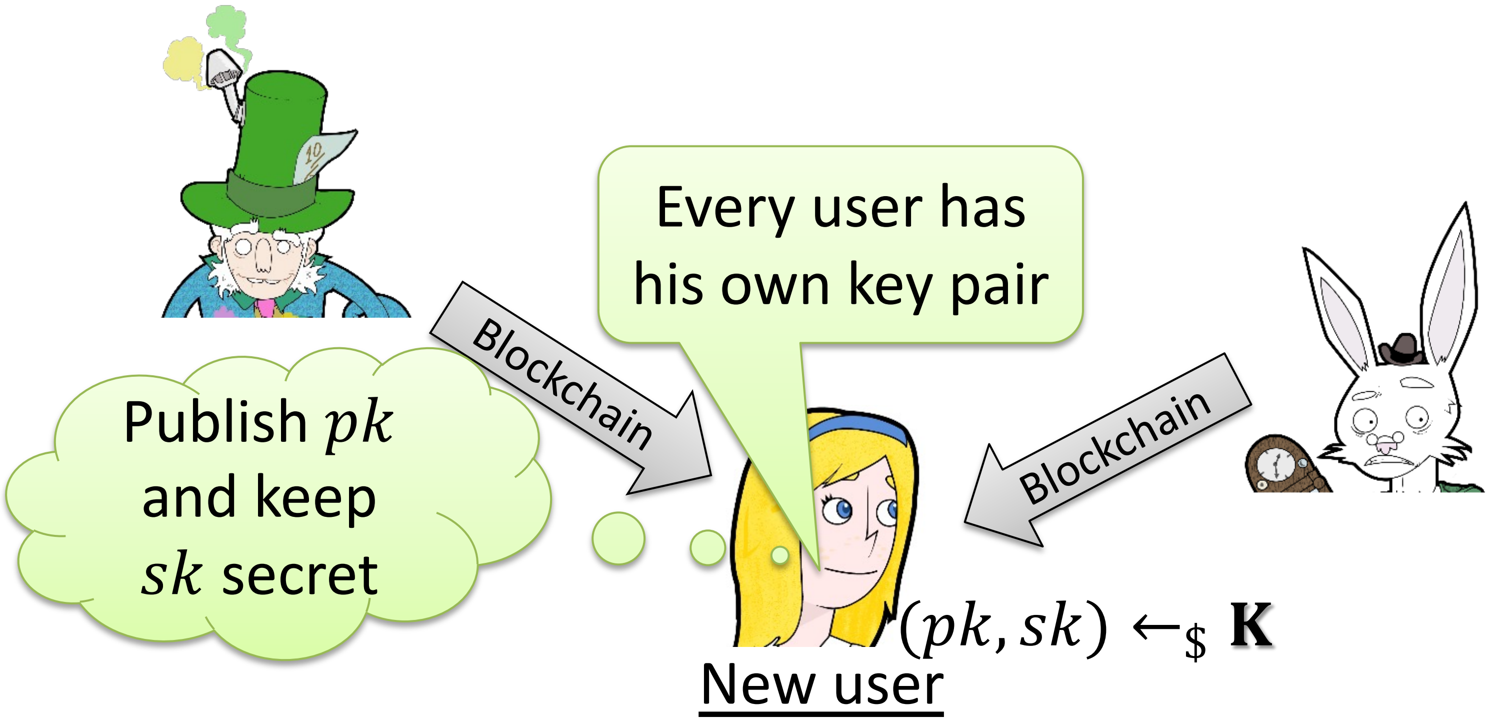
1) **Secretely** start mining in 1980 and produce a very **strong** chain

Genesis block

2) Honest miners start working on Jan 3, 2009; since they have **less time** after 1 year their chain is still **weaker**



Joining the Network



- How to identify a user? Use a **digital signature** scheme $(\mathbf{K}, \mathbf{S}, \mathbf{V})$
  - Bitcoin uses **ECDSA**

Digital Signature Standard (DSS)

- Approved by US government in 1994
  – Designed by NIST & NSA
  – Originally using SHA-1, but now SHA-2 is recommended
  – DSS is the standard and DSA is the algorithm
- A variant of **ElGamal PKE**
  – Security based on the **hardness of DL**
  – Creates a **320-bit signature** (vs 1024 bits with RSA)
  – Most of the computation is mod a **160-bit prime**

DSA Key Generation

- Shared global **public values** (p, q, α)
  - Prime p of size 1024 bits
  - Prime q of size 160 bits (factor of $p - 1$)
- Value $\alpha \in \mathbb{Z}_p^*$ of order q
  - Pick $g \in \mathbb{Z}_p^*$ and compute $\alpha = g^{(p-1)/q} \bmod p$
  - Repeat if $\alpha = 1$
- Each user generates (a, β)
  - **Private key** $a \leftarrow_{\$} \mathbb{Z}_q$
  - **Public key** $\beta = \alpha^a \bmod p$

DSA Signing

- Let $x \in \{0,1\}^*$ be the message to be signed
  - Pick random $k \leftarrow_{\$} \mathbb{Z}_q$
  - Let $r = \left(\alpha^k \bmod p\right) \bmod q$
  - Let $s = (\mathbf{SHA2}(x) + a \cdot r)k^{-1} \bmod q$
  - Repeat if $r = 0$ or $s = 0$
- Signature is $y = (r, s)$
  - Value k should be **destroyed** and **never reused**

Signature Verification

- Give message x and signature $y = (r, s)$
  - Compute $u = s^{-1} \cdot \mathbf{SHA2}(x) \bmod q$
  - Compute $t = s^{-1} \cdot r \bmod q$
  - Let $v = (\alpha^u \beta^t \bmod p) \bmod q$
- Accept iff $v = r$
- **Correctness:**

$$v = (\alpha^{u+at} \bmod p) \bmod q$$
$$= \left(\alpha^{s^{-1}(\mathbf{SHA2}(x)+ar)} \bmod p\right) \bmod q$$
$$= \left(\alpha^{s^{-1}ks} \bmod p\right) \bmod q = r \bmod q$$

Remarks on DSA

- Important to check $r, s \neq 0$
  - If $r = 0$, then $s = \mathbf{SHA2}(x) \cdot k^{-1} \bmod q$ is **independent** of the secret key a
  - If $s = 0$, then $s^{-1} \bmod q$ cannot be computed
  - Both events very **unlikely** (probability $\approx 2^{-160}$)
- Operations on both sides are performed mod q, only one operation is performed mod p

Elliptic Curve DSA (ECDSA)

- Variant of DSA using **elliptic curve groups**
- Signature is 320 bits
- All operations are mod a 160-bit prime (or slightly more)
  - Minimum size 163 or 192 bits
- Security depends on **hardness of solving DL** in an elliptic curve group

Validating the Blockchain

- What is needed in order to decide which blockchain is valid?
- One needs to know:
  - The **initial rules** of the game
  - The **genesis block**
- Given many candidates pick the one that:
  - Verifies correctly
  - Is the **longest** (i.e., the **strongest)**
- Verification can take **several hours** (blockchain size ≈ 185GB as of September 2018)

Checkpoints

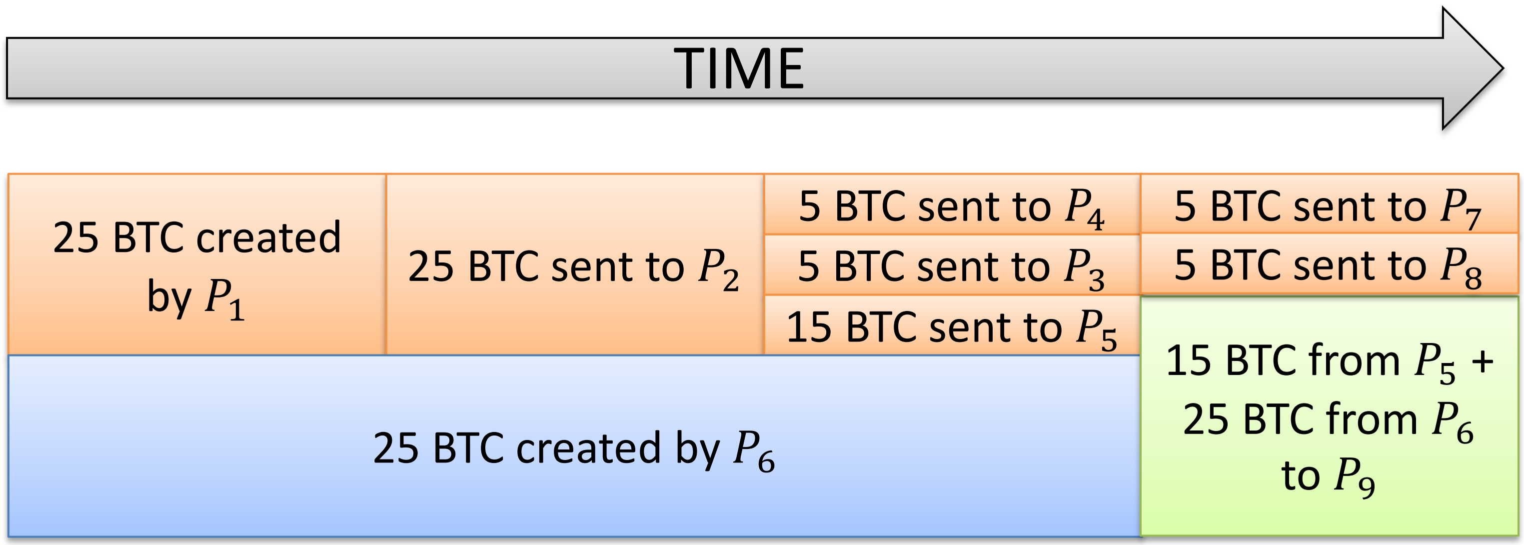
- Old block hash **hardcoded** into Bitcoin software
- In theory: Not needed
- Goes against the decentralized spirit of Bitcoin
- But **useful** in practice:
  - Prevent some **DoS attacks** (flooding nodes with unusable chains)
  - Prevent attacks involving **isolating nodes** and providing them fake chains
  - **Optimization** for initial blockchain download

Protocol Updates

- The Bitcoin protocol can be **updated**
- Proposals can be submitted to the Bitcoin foundation in the form of **Bitcoin Improvement Proposals** (BIPs)
- Only the miners can vote
  - Votes included in the minted blocks
  - Currently, need **75% approval** which roughly corresponds to 75% of computing power

Bitcoin's Money Mechanics

- Bitcoin is **transaction** based
- Technically there is **no notion of coin**



- Users P_7 and P_8 hold 5 BTC, whereas user P_9 holds 40 BTC

Syntax of Transactions (Simplified)

P_1 $T_1 =$ User P_1 creates 25 BTC

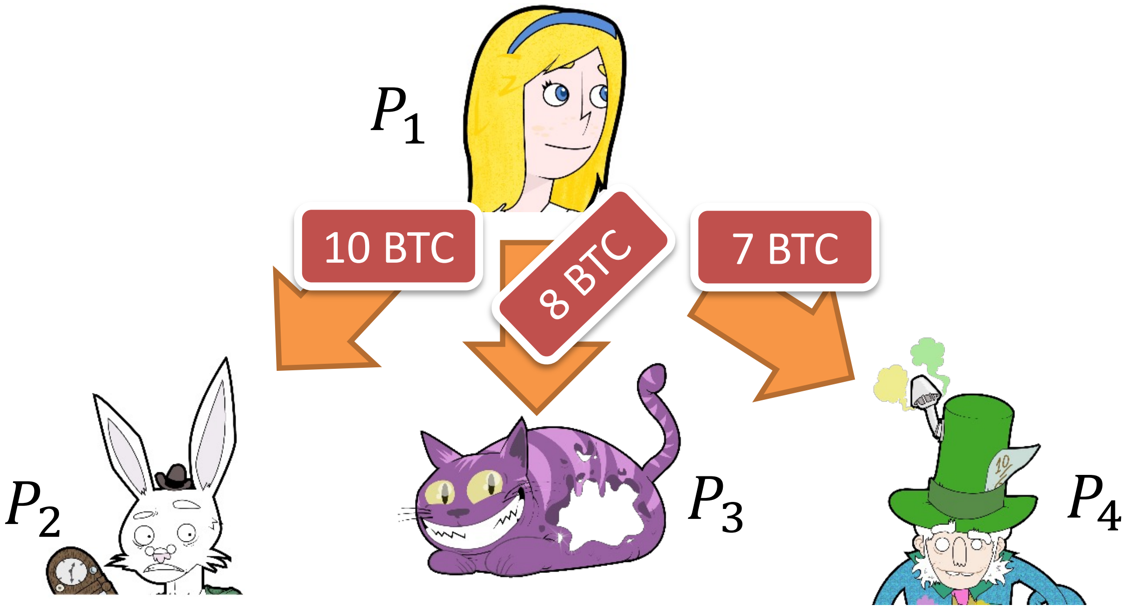
During the mining process

P_2 $T_2 =$ User P_1 sends 25 BTC from T_1 to P_2 | Signature of P_1 on T_2

We say T_3 redeems T_2

P_3 $T_3 =$ User P_2 sends 25 BTC from T_2 to P_3 | Signature of P_2 on T_3

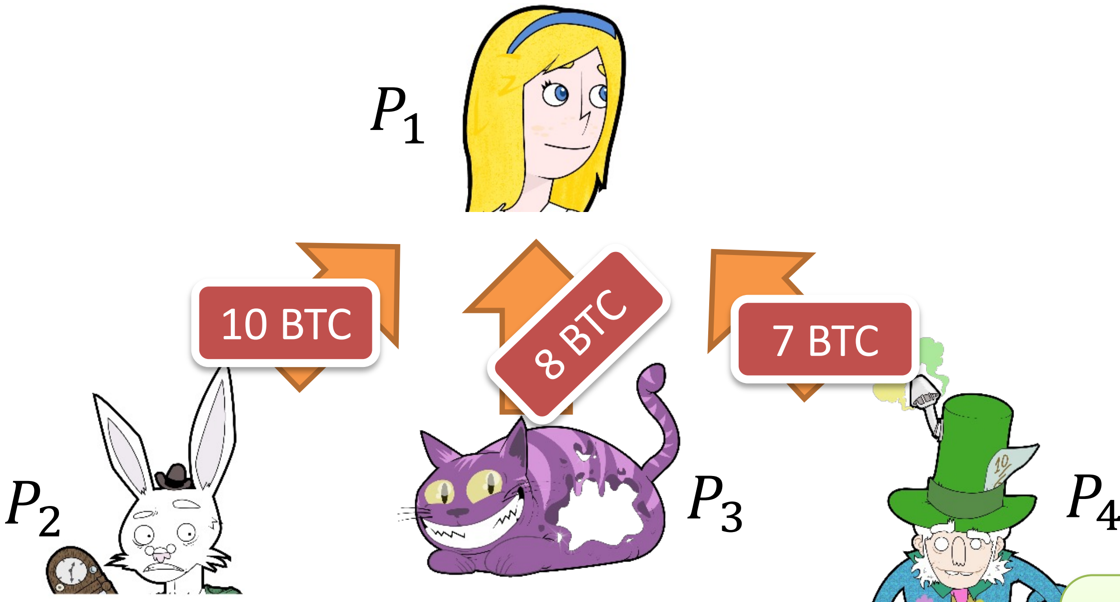
Multiple Output Transactions



$T_2 =$

User P_1 sends 10 BTC from T_1 to P_2 User P_1 sends 8 BTC from T_1 to P_3 User P_1 sends 7 BTC from T_1 to P_4	Signature of P_1 on T_2

Multiple Input Transactions



All signatures need to be **valid**

$T_4 =$

User P_2 sends 10 BTC from T_3 to P_1	Signature of P_2 on T_4
User P_3 sends 8 BTC from T_3 to P_1	Signature of P_3 on T_4
User P_4 sends 7 BTC from T_3 to P_1	Signature of P_4 on T_4

Time Locks

Transaction specifies time t **after which** it is considered valid

$T_2 =$

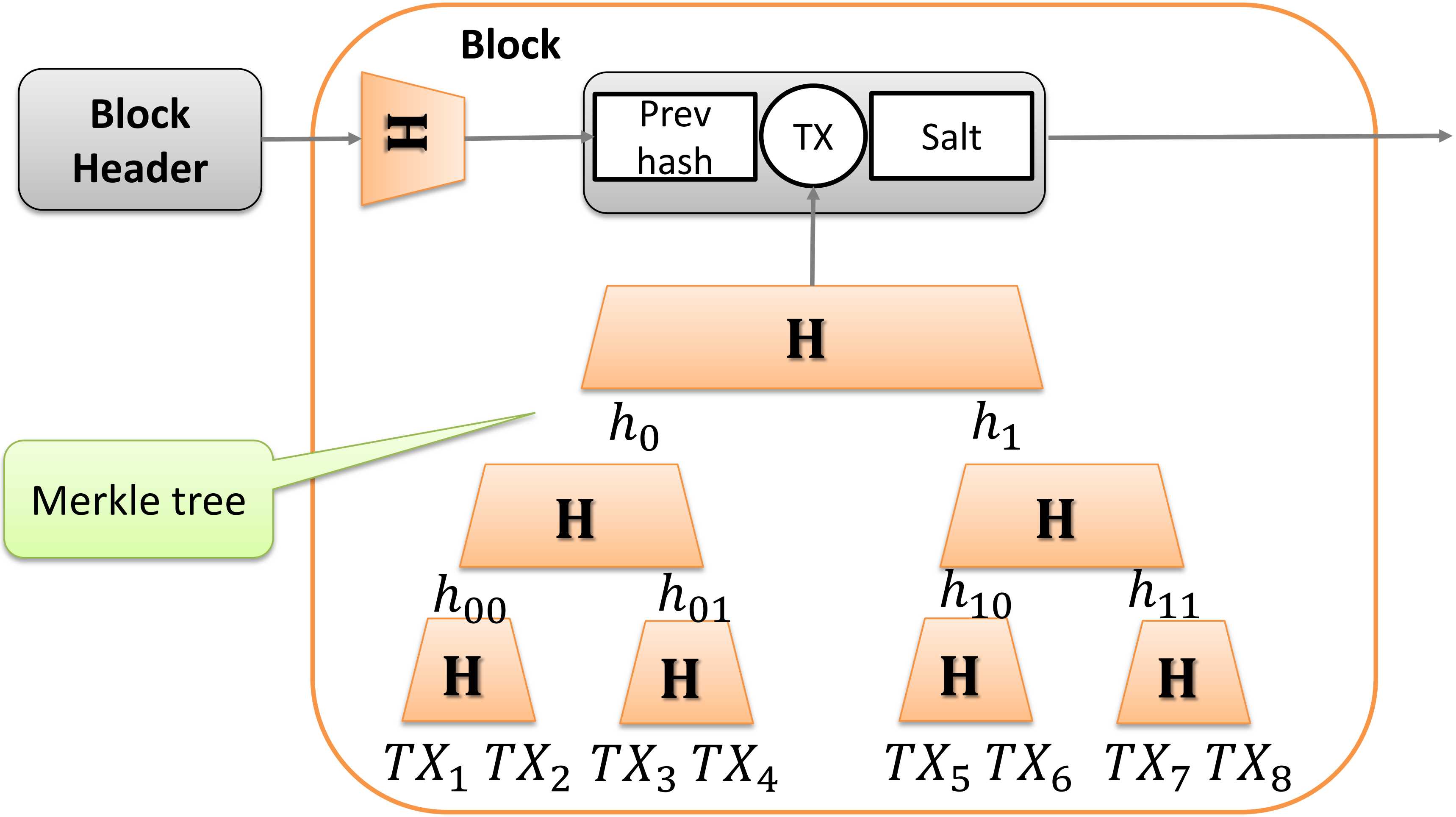
User P_1 sends 25 BTC from T_1 to P_2 if time t has passed	Signature of P_1 on T_2

Measured in blocks or real time

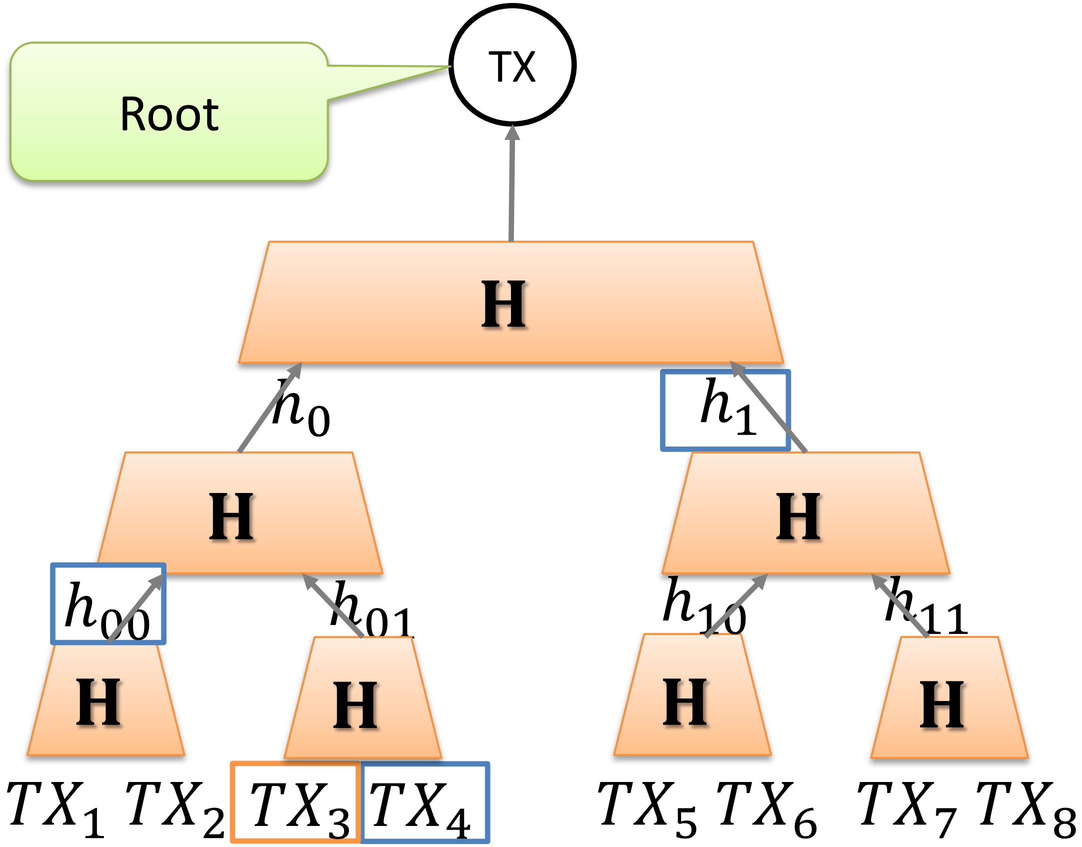
Generalizations

- All these features can be combined
- The total value of in-coming transactions can be larger than the total value of outgoing transactions
  - The difference is called **the fee**
  - Goes to the miner
- The conditions for redeeming a transaction can be more general (the so-called **smart contracts**)

Block Structure in More Details



How to Verify Merkle Trees



Proofs are log(depth) and verification requires log(depth) time

Why Merkle Trees?

- Merkle root always of **same small size**
  - Easily transmittable for pooled mining
  - Simplifies writing hashing algorithms in hardware
- **Light** clients
  - No need to process the entire block
- Pruning of old spend transactions
  - Old transactions are **not needed** in order to verify the validity of the blockchain

Mining Pools and Attacks

Solo Mining

- Variance of income **too high for solo miners**
- Here is a rough estimate:

Total hash rate as of Nov. 2018

$$\frac{40{,}000{,}000\ \text{THash/s}}{14\ \text{THash/s}} \approx 2857142$$

$$\approx 54.4 \cdot (365 \cdot 24 \cdot 6)$$

ASICS Antminer S9 – 14 THash/s (3,000 USD)

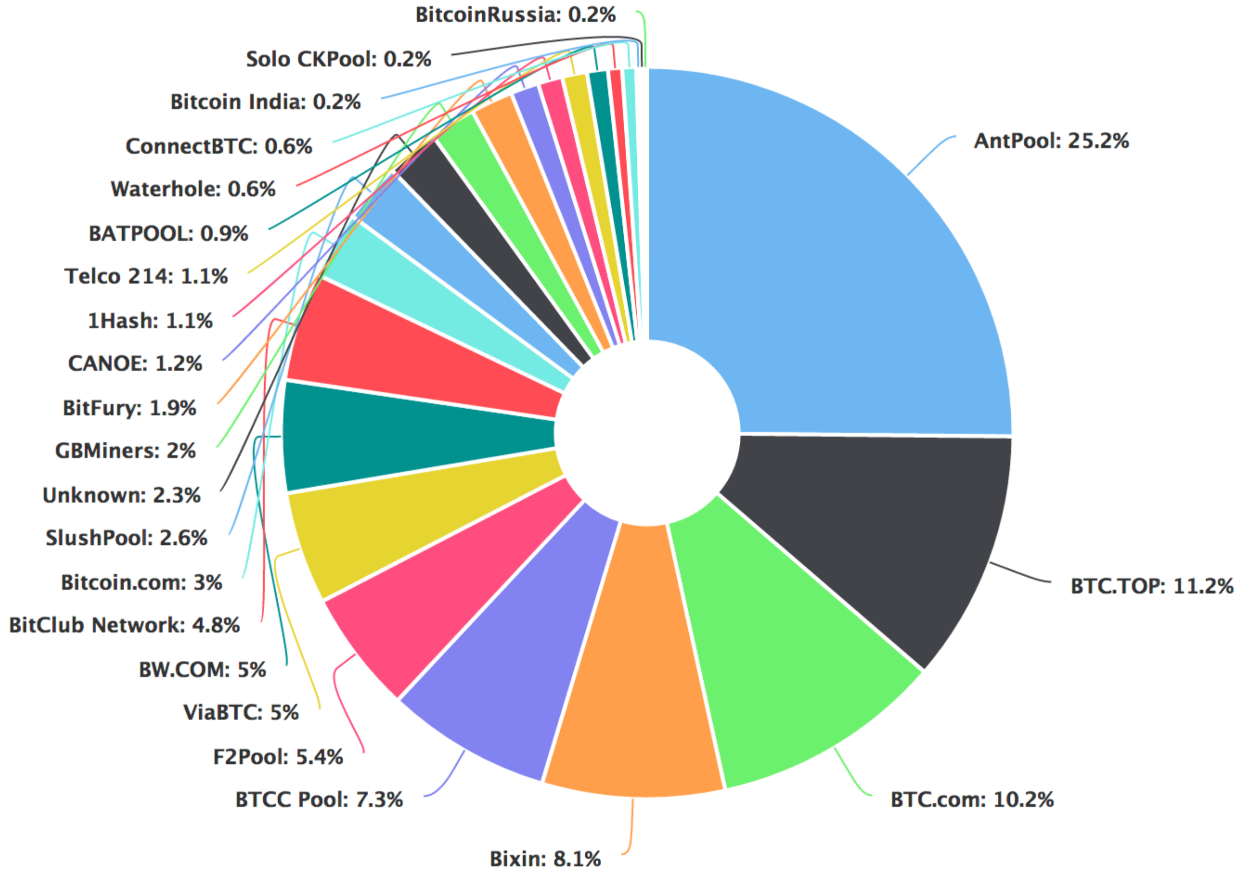
- Waiting time for mining a block ≈ 50 years

Mining Pools

- Miners create cartels called **mining pools**
- Mining pools are either operated **centrally** or in a **peer-to-peer** fashion
- Some of the pools charge **fees** for their service
  - E.g., if the operator gets 25 BTC for mining, then it will share $25 - \varphi$ BTC (where φ is the fee)
- Expected revenue is lower on average, but **variance** is significantly smaller
  - But how to prevent cheating? How to reward the miners?

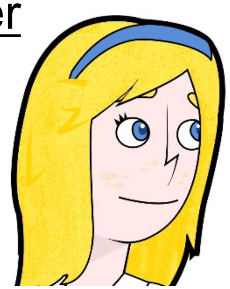
Biggest Mining Pools

As of July 13, 2017



How to Design a Mining Pool?

Miner



Includes coinbase transaction transferring money to pk

Mining Pool Operator

pk

A transaction T^i and a hash $\mathbf{H}(B_i)$

Tries to find s_i such that $\mathbf{H}(s_i, \mathbf{H}(B_i), T^i)$ starts with n zeroes

Current hardness parameter

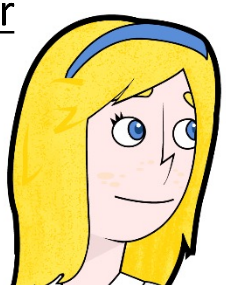
Nonce s_i

Once nonce is found by one of the pool members, each of them is rewarded **proportionally** to his work

But how to verify **how much work** a miner did?

Proportional Method

Miner



Mining Pool Operator

pk

A transaction T^i and a hash $\mathbf{H}(B_i)$

Tries to find s_i such that $\mathbf{H}(s_i, \mathbf{H}(B_i), T^i)$ starts with n zeroes

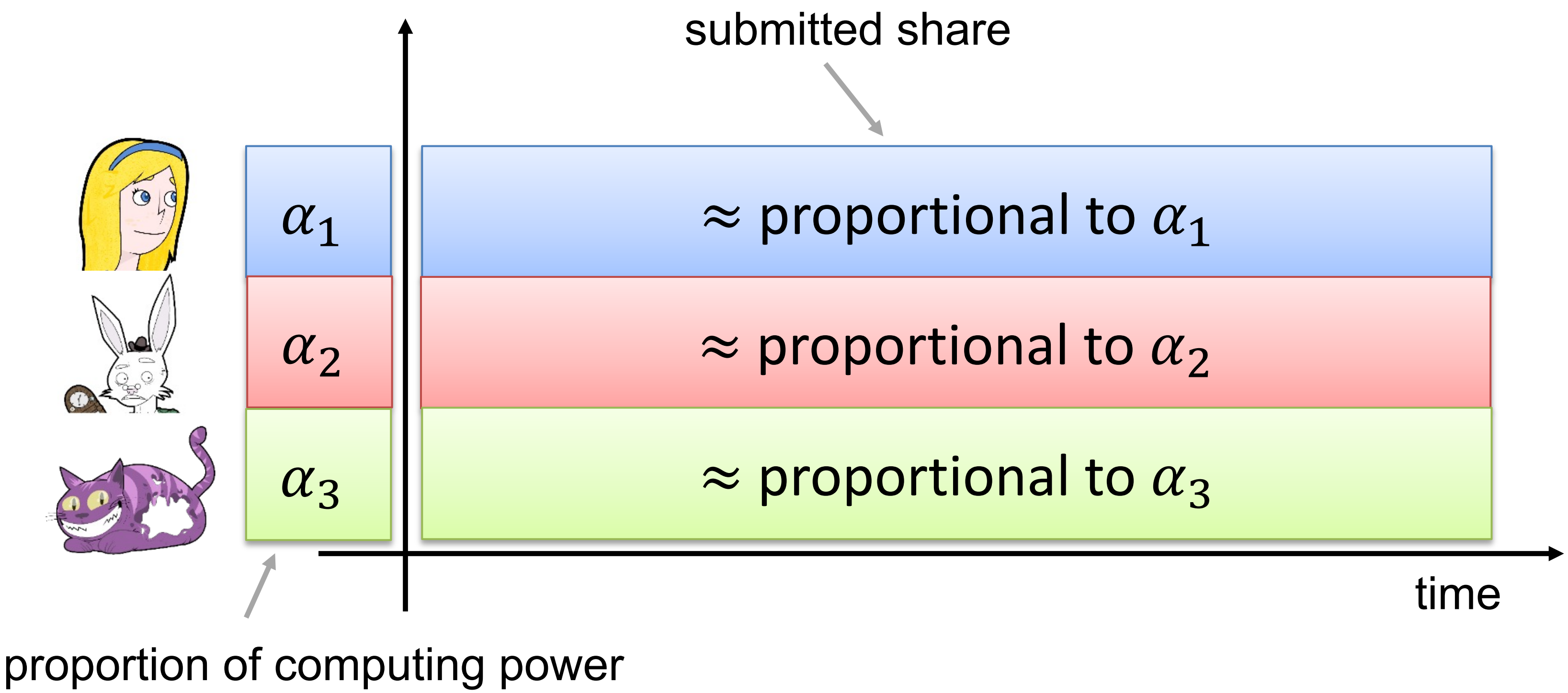
Current hardness parameter

Nonce s_i

But also submit **partial solutions**, i.e. values $s_i{}'$ such that $\mathbf{H}(s_i, \mathbf{H}(B_i), T^i)$ starts with $n' \ll n$ zeroes

Amount of work measured in # of **partial solutions**

Probability of Success



- Probability of **pool winning** is $\alpha_1 + \alpha_2 + \alpha_3$
- Reward for Alice: BTC $25 \cdot \frac{\alpha_1}{\alpha_1+\alpha_2+\alpha_3}$

Expected reward BTC $25 \cdot \alpha_1$

Pool Hopping

- What if miners **change pool**?
  - Expected revenue is α_i (from new pool)
  - Plus the revenue form old pool (small extra)
- It is **profitable** to escape from pools with lots of share holders
  - Because such pools have too many "mouths to feed"

Slush's Method

- Solution: Use a **scoring function** that assigns to each share a score σ
- Then assign rewards **proportionally to the score** σ
- Slush's scoring function: $\sigma = e^{T/c}$
  - T: time since beginning of this round
  - c: some constant
- Intuitively this gives advantage to miners who joined late

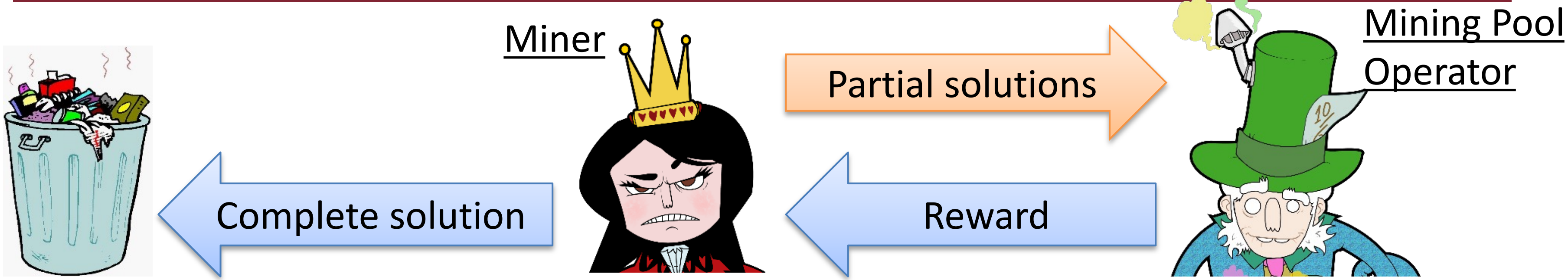
Other Methods

- **Pay-per-share:** Operator pays for each partial solution, no matter if he mined the block
  - Risky for operator (leading to higher fees)
- **Score-based:** Geometric method, double geometric method…
- See also:
  - M. Rosenfeld. "Analysis of Bitcoin pooled mining reward systems." 2011

Security of Mining Pools

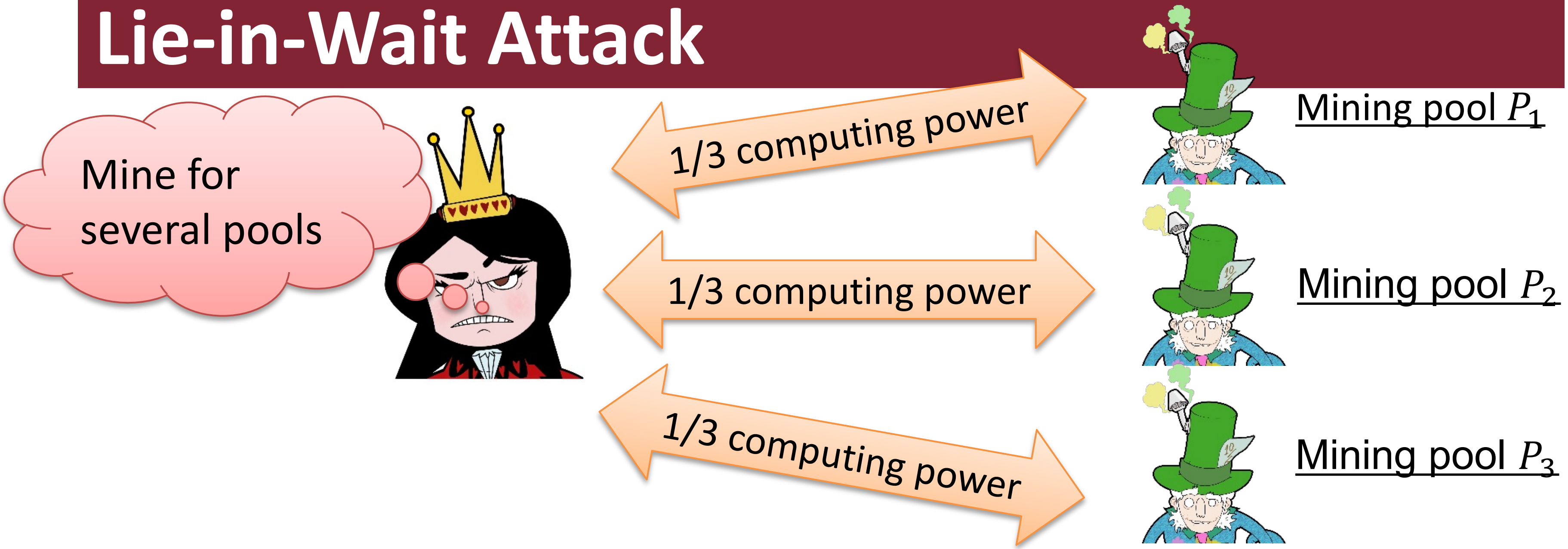
- Typically assume the operator is **honest**
  - Because he has reputation
- Miners are instead untrusted
- We will describe two attacks:
  - Sabotage attack
  - Lie-in-wait attack
- Both attacks are based on **withholding blocks**

Sabotage Attack



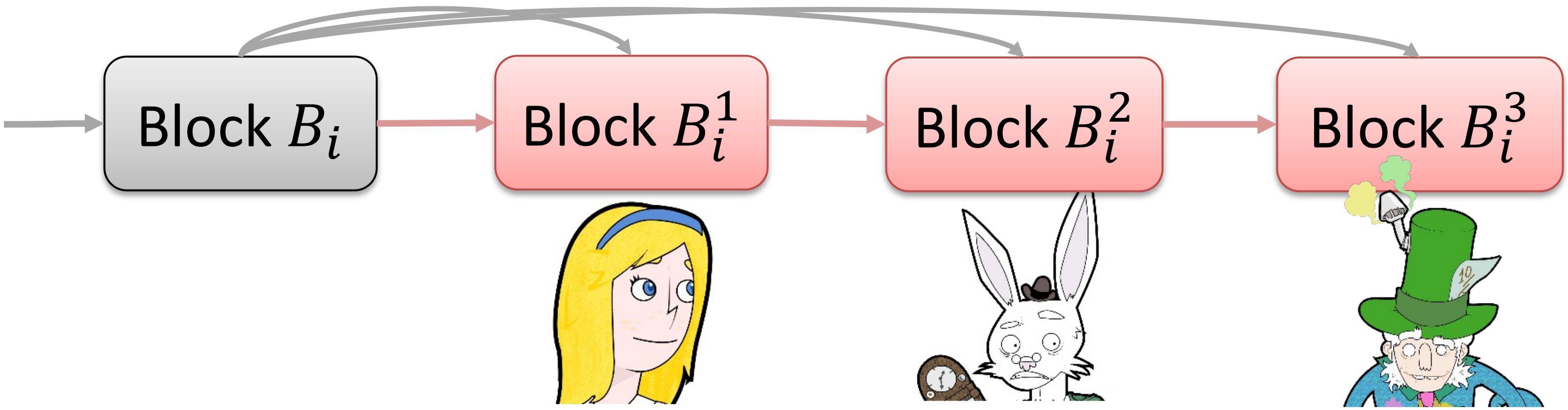
- Based on submitting only **partial solutions**
  - Pool **loses money**
  - Dishonest miner does not earn anything (actually it loses a little bit)
- Ultimate goal: Make the pool go **bankrupt**
  - E.g., because it is a competing pool
  - Mining pool Eligus lost 300 BTC back in 2014

Lie-in-Wait Attack



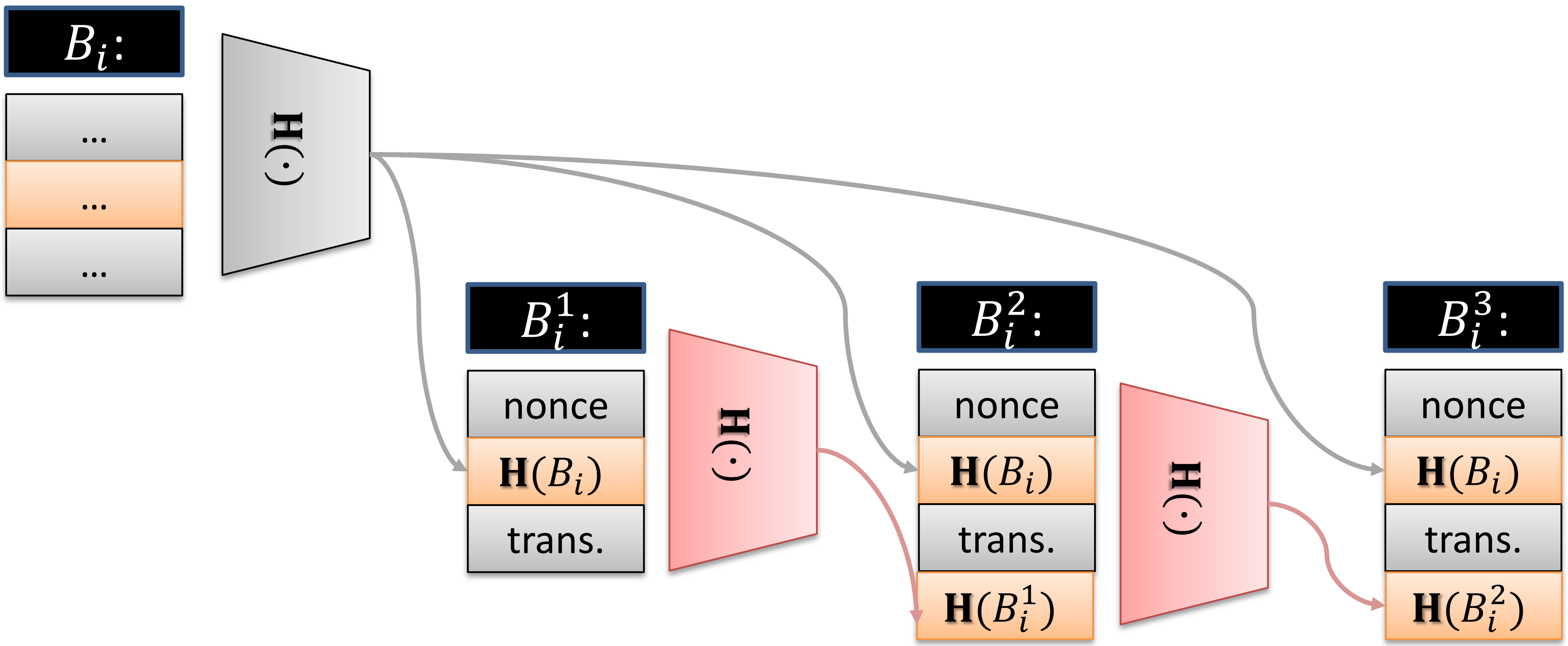
- Once solution is found (say for P_2)
  - **Wait submitting** it and mine for P_2 only
  - Send it to P_2 after some time
- Intuition is that this is profitable because P_2 is a very **likely winner**

Peer-to-Peer Mining



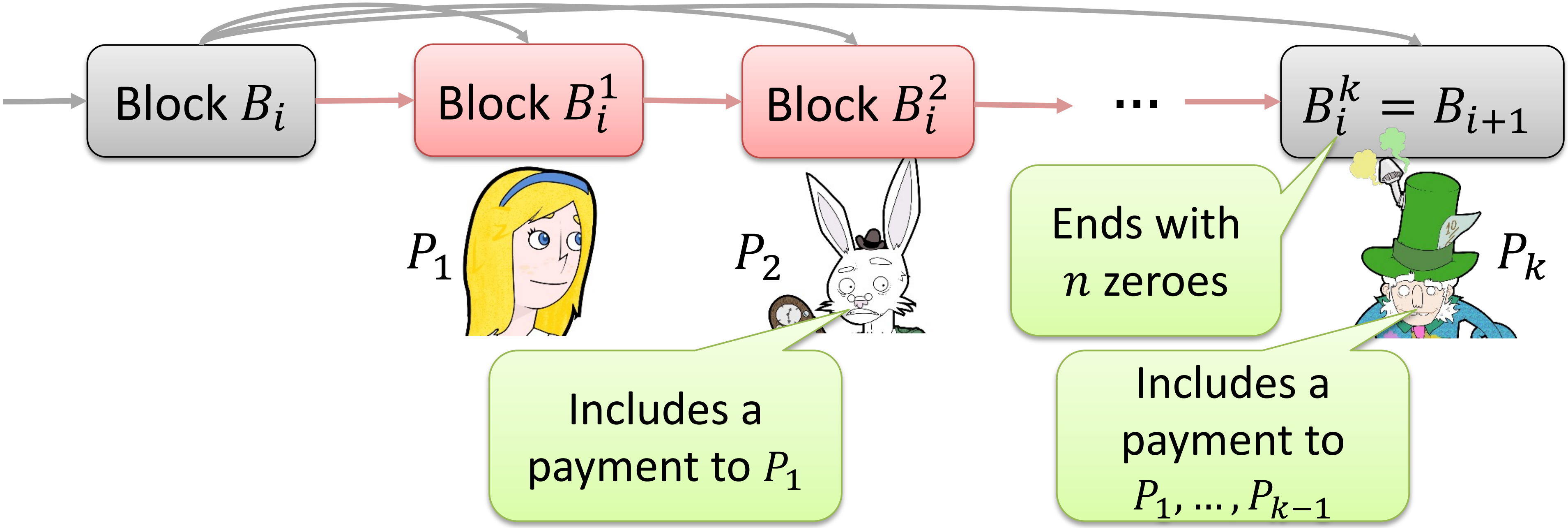
- Main idea: Create a blockchain with hardness parameter $n' \ll n$ **on top** of the last block
  - Every B_i^j is a valid extension of B_i (hardness n')
  - Requires to use **other fields** in the block
- Parameter n' chosen in such a way that new blocks **appear often** (say every 30 sec)

How to Do it



- The blocks contain **extra space** that can be used to store the hash values $\mathbf{H}(B_i^j)$

Reward



- Block B_i^k enters the **main** blockchain as B_{i+1}
- Reward can be computed using some formula
- Each miner is **incentivized to behave nicely**

Possible Attack Goals

- **Double spending**
- Get **more money** than you should
- Short selling
  - Bet that the price of BTC will drop and then destroy the system (i.e., make the price of BTC go to zero)
- Someone (government?) interested in **shutting down** Bitcoin

The 51% Attack

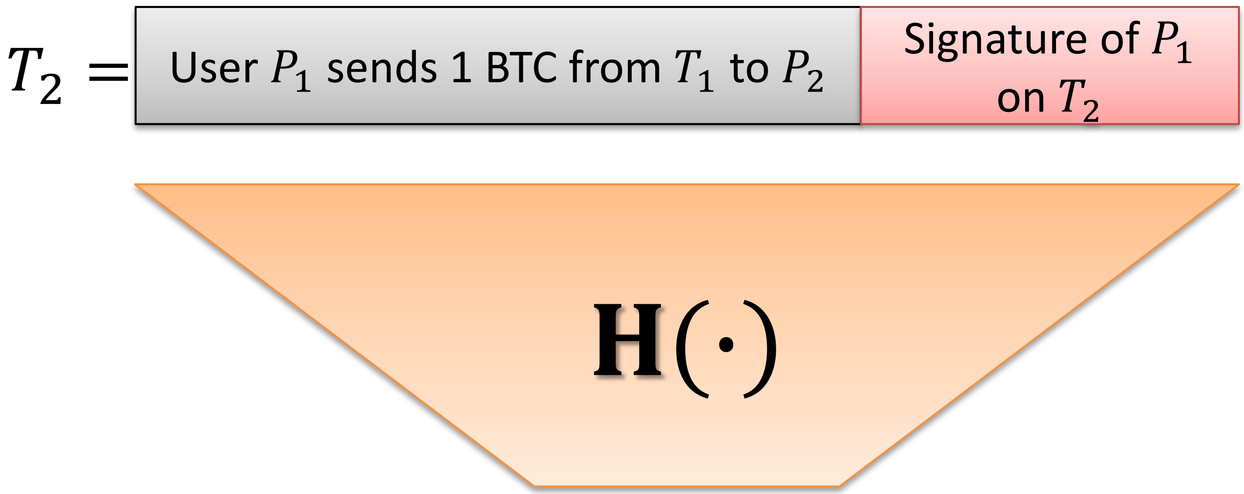
- An adversary **controlling majority** of computational power **cannot**
  - Steal money from earlier transactions (requires forging a signature)
  - Generate money without effort (still needs to solve PoW)
- However such an adversary **can**
  - Fork the chain and doublespend
  - Reject all other miners' blocks
  - Exclude certain transactions

Programming Errors

- Block 74638 (Aug 2010) contained a transaction with 2 outputs summing to over 184 billion BTC
  - **Integer overflow** in Bitcoin software
  - Solved by software update + **manual fork**
- Fork at block 225430 caused by an error in the **software update**
  - Solved by reverting to older version
- Moral: **Nothing** can be fully decentralized
  - Sometimes human intervention is needed

Transaction Malleability

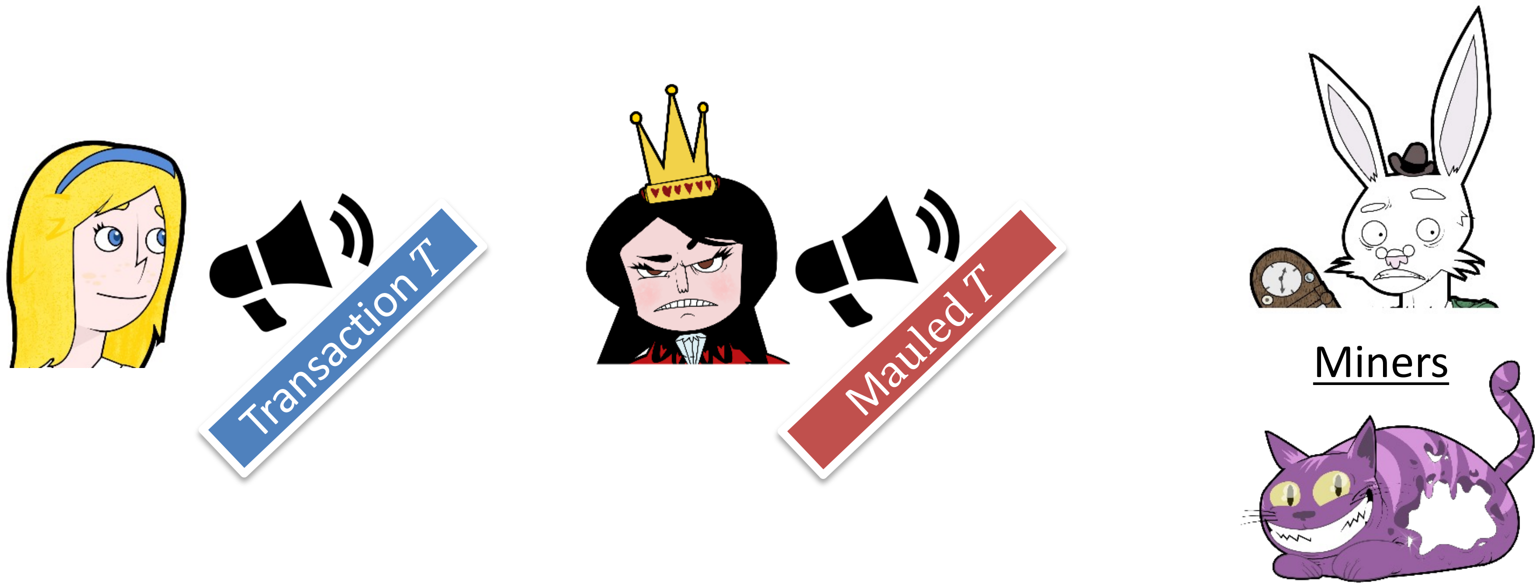
- Transactions are **identified** by their hashes



$$\text{TxId} = \mathbf{H}(T_2)$$

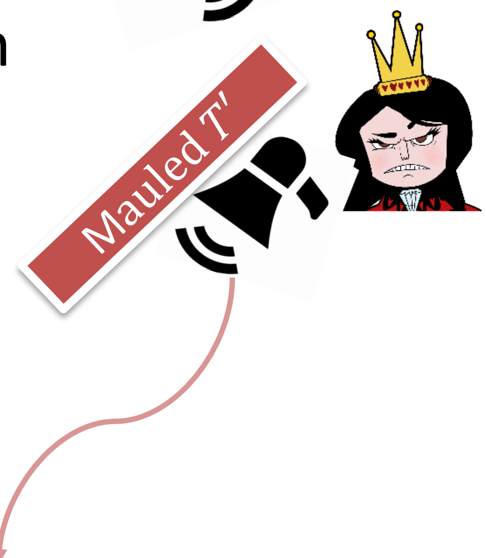
- One can change TxId by **mauling** a signature
  - In ECDSA if $\sigma = (r, s)$ is a valid signature of message m, so is $\sigma' = (r, -s)$

How to Exploit Malleability

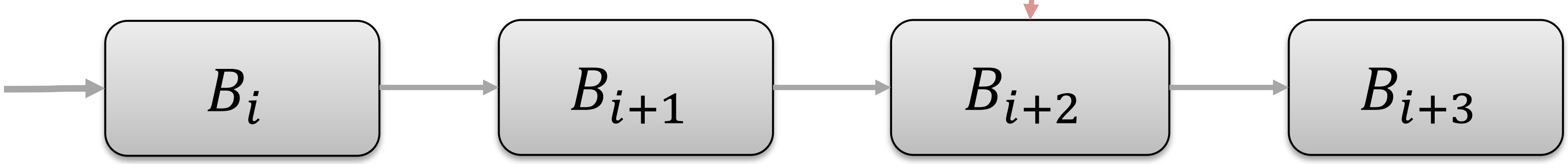


- As a result **TxId changes**!
- Often not a problem as **semantically** nothing changed
- Problematic for Bitcoin contracts

Claimed Attack on MtGox

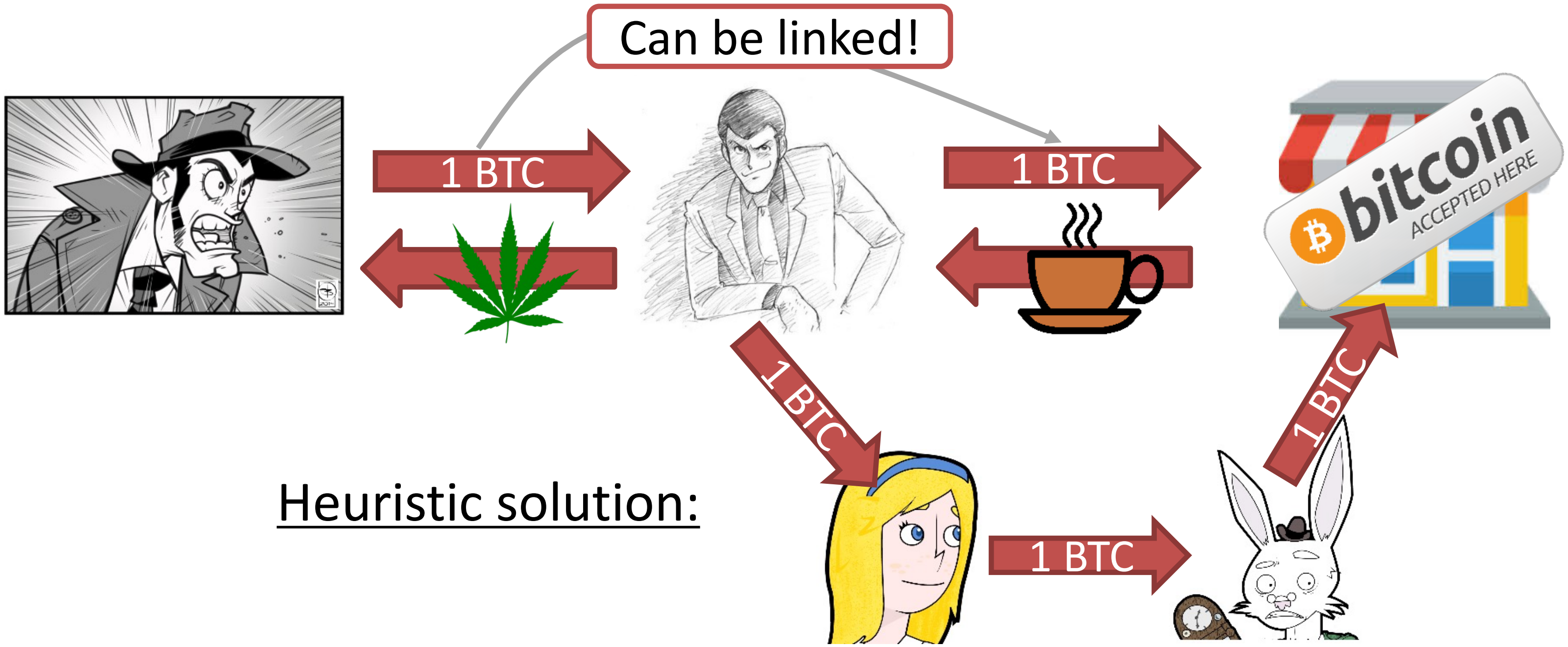


- MtGox **cannot see transaction** with TxId $\mathbf{H}(T)$ in the blockchain
  - As if transaction did not happen
  - Doublespending possible
- Decker-Wattenhofer 14: This is probably **not true**



Lack of Anonimity

- Bitcoin only guarantees **pseudonymity**



Heuristic solution:

- Can sometimes be **de-anonymized**
  - Meiklejohn et al.: A Fistful of Bitcoin, 2013

Hardware Mining

- Evolution of mining habits
  - CPU -> GPU -> FPGA -> ASIC
- Several drawbacks
  - Makes the whole process **non-democratic**
  - Might be exploited by very powerful adversary
  - Excludes some applications (e.g., mining as micropayment)
- Advantages
  - Security against botnets and makes miners interested in long-term stability of the system

How long term? Hash rate can go up by 100x in a year

Risks Associated with Pool Mining

- June 2014: The Ghash.io pool got > 50% of the total hash power
  - What we were promised: A distributed currency independent of the central banks
  - What we got (June 2014): Currency controlled by **single company**
- Miners lost control of which blocks they mine
  - Not possible to choose Bitcoin transactions
  - Common believe: 99% of the miners only care about highest possible **block reward**

How to Break Bitcoin?

- Start a number of mining pools with a **negative fee**
- Wait to get $> 50\%$ computational power
- Will the miners join?
  - Well, yes if they only care about **block reward**
- Is Bitcoin secure?
  - Need to assume that majority behaves honestly **even if it has incentives not to do so**
  - Maybe the only reason why it is still unbroken is that nobody was really interested in doing it

Majority is not Enough (Selfish Mining)

- I. Eyal, E. G. Sirer. "Bitcoin Mining is Vulnerable." Commun. ACM 61(7), 2018
- Basic idea: When a new block is found **keep it** for yourself
- Goal: Make the honest miners **waste their effort** to mine blocks that will never make it to the blockchain
- The proportion of minted blocks is **higher**, yielding **a revenue greater** than the fair share

Bitcoin is not Incentive Compatible

- Recall with the honest strategy every miner with α-fraction of computing power gets α-fraction of the revenue
- But if there is a strategy that is more beneficial than the honest strategy, miners have an **incentive to misbehave**
  - The larger α the more beneficial the dishonest strategy is
  - Hence miners have incentive to join a large pool that uses this strategy

Simplifying Assumption

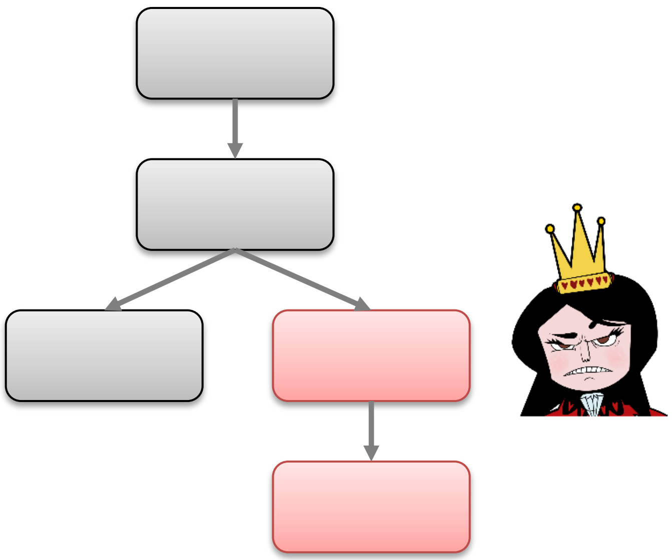
- What happens if there is a fork?

> Bitcoin specification: "From two chains of equal length mine on the one that was received first."

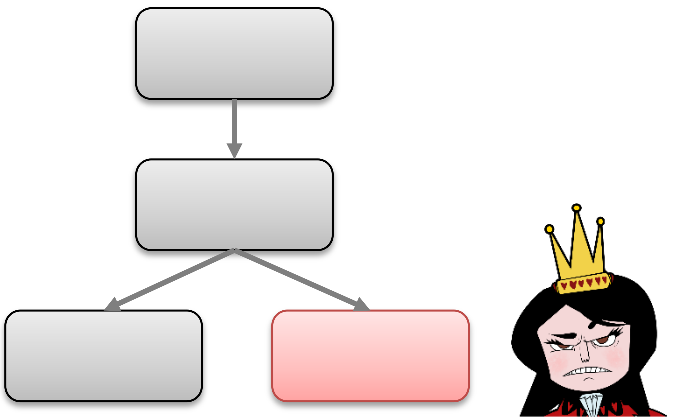
- Assume that the adversary is **always first**
  - E.g., he puts a lots of fake nodes acting as sensors
  - We will **remove** this assumption later

Selfish Mining (Basic Idea)

- Adversary finds new block and **keeps it**
- Two things can happen:



Publish the chain when the public one equalizes



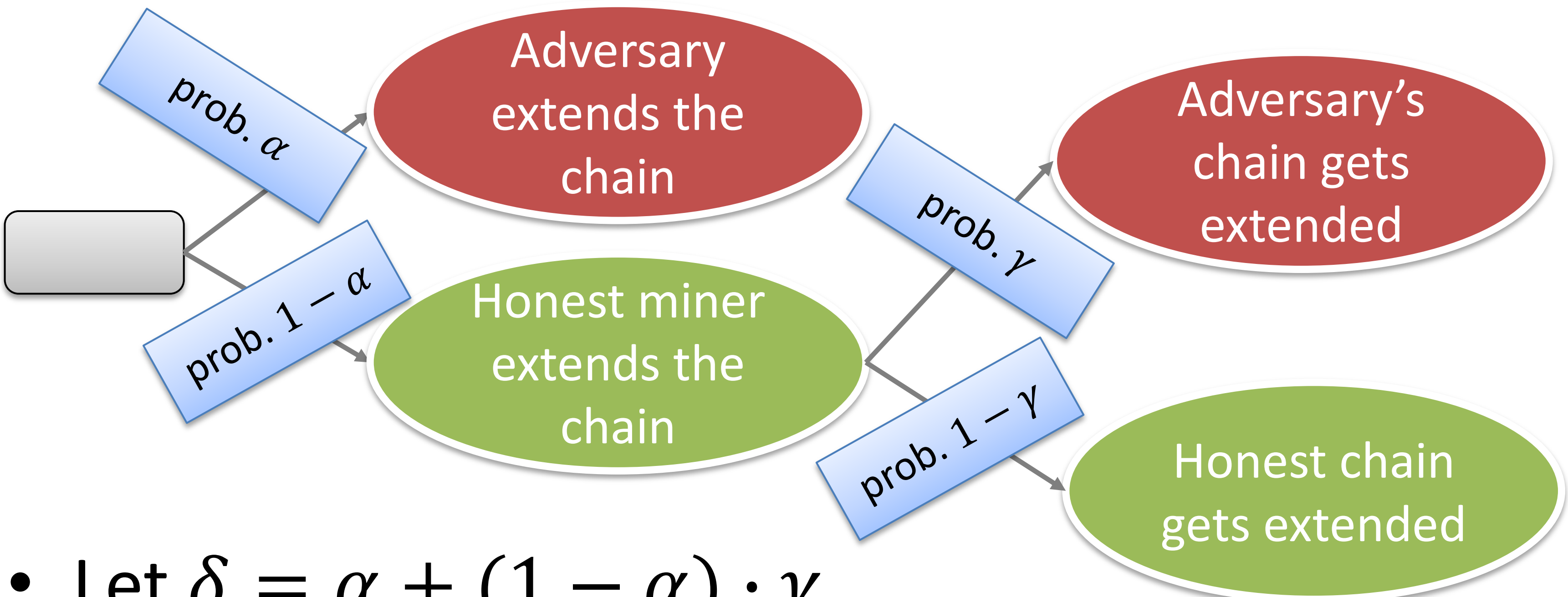
In this case the adversary publishes his own block and **loses nothing**

Towards the Full Attack

- The assumption that the adversary is always first might look **unrealistic**
- Eyal and Sirer show a modification of the attack that works **without this assumption**
- Let γ be the probability that a honest miner will choose to mine on the adversary's chain
- Assume the adversary controls an α- fraction of the computing power
  – The other miners hold $(1 - \alpha)$-fract. for $\alpha < 1/2$

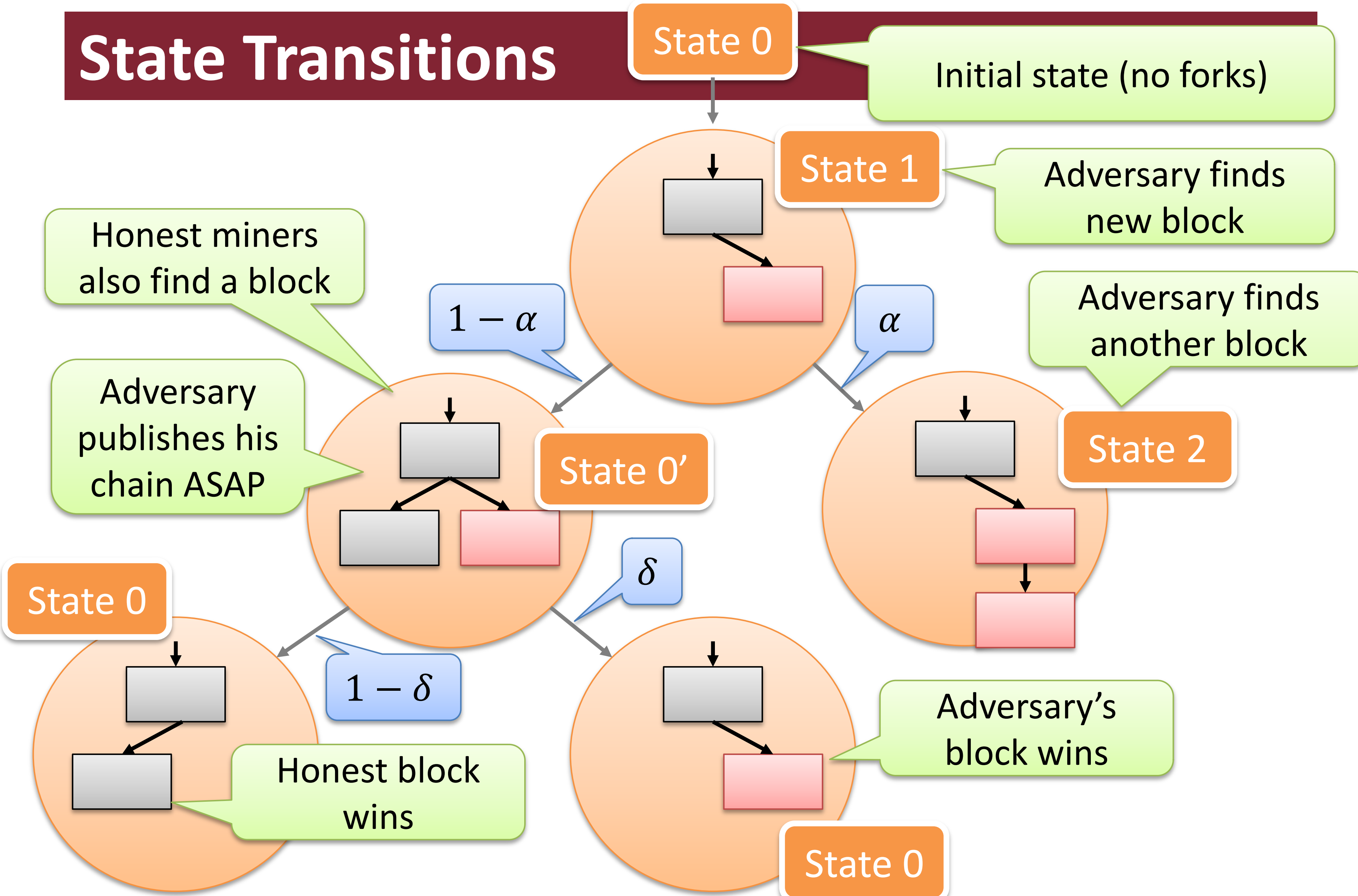
An Observation

- What is the probability that the **adversary's chain** is selected?

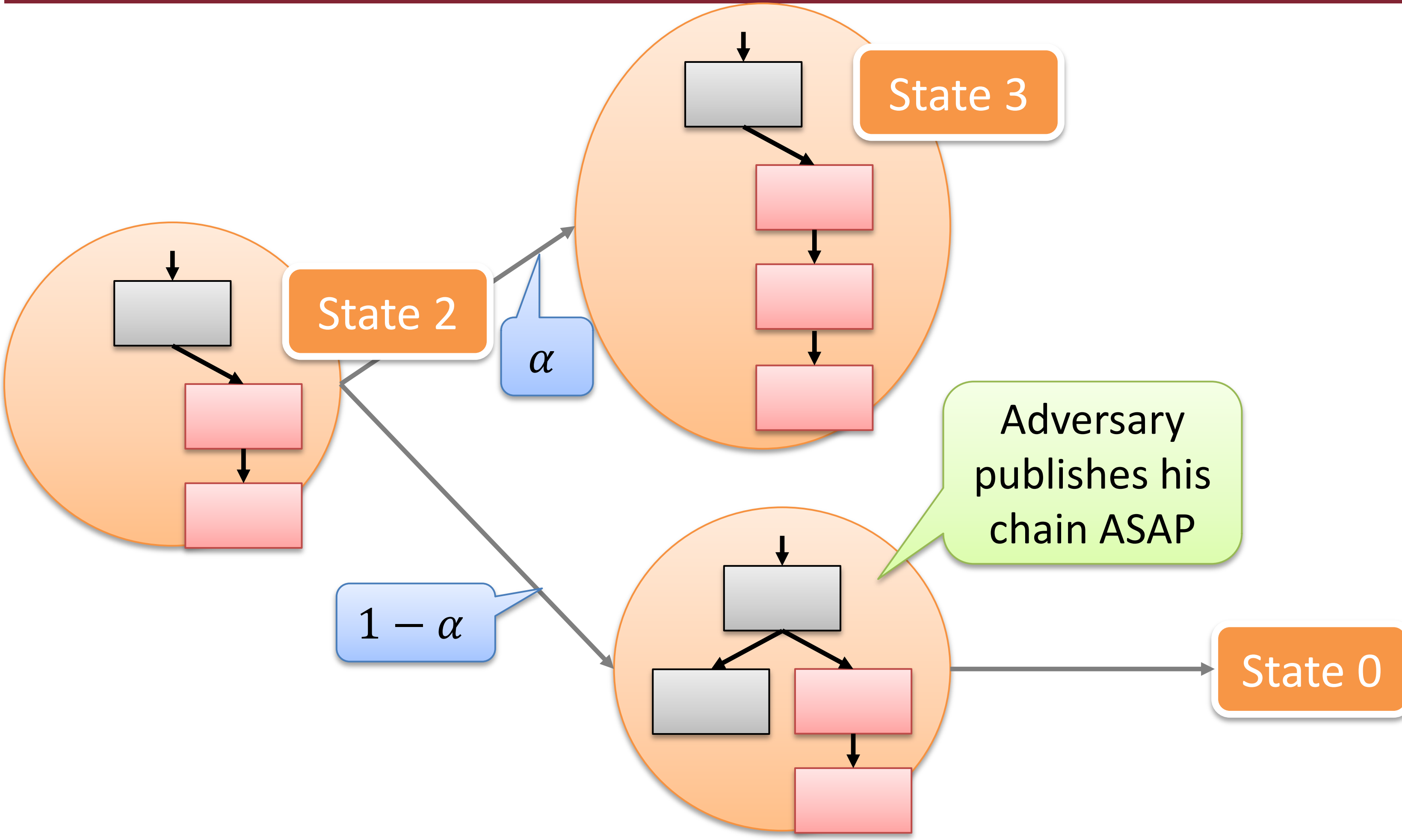


- Let $\delta = \alpha + (1-\alpha) \cdot \gamma$

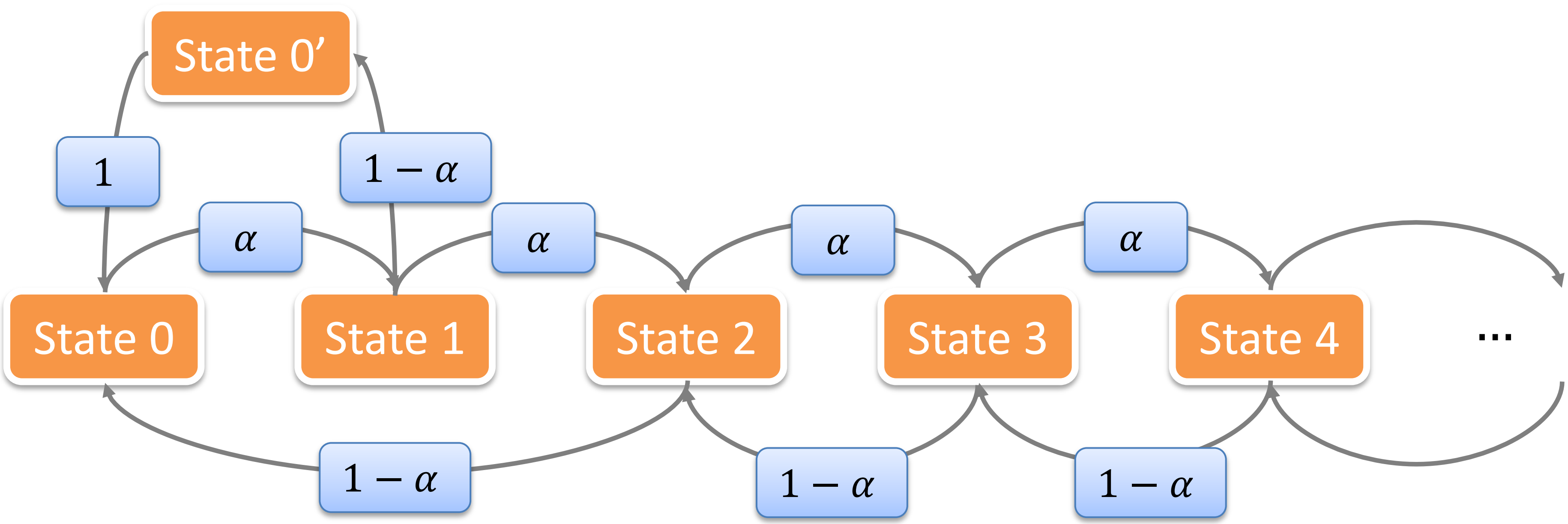
State Transitions



Continuing from State 2

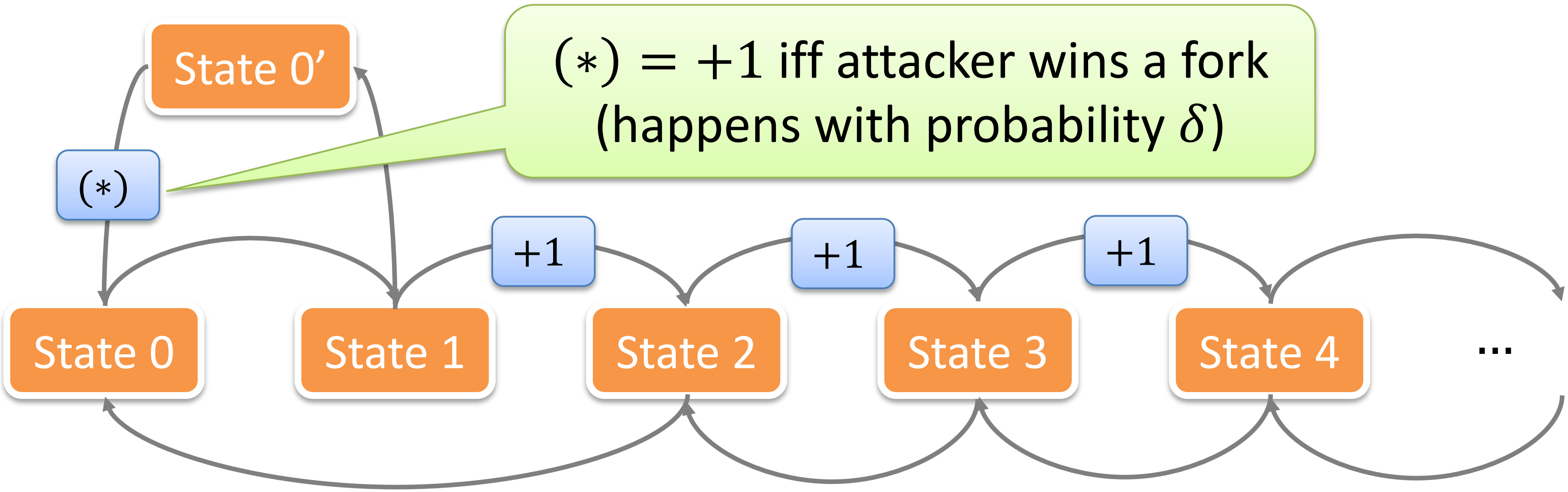


Resulting State Machine



Calculating the Revenue

- Apply theory of Markov chains
  - Stationary distribution: $p_0, p_{0'}, p_1, p_2, \ldots$

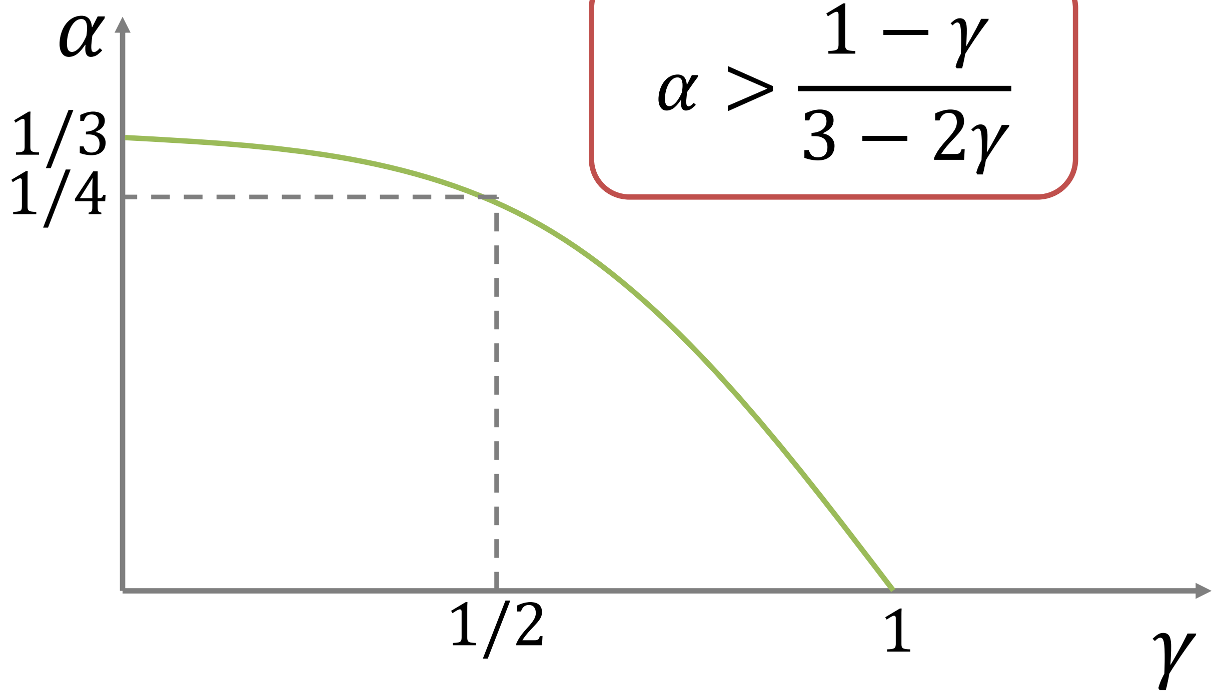


Expected Revenue: $\delta \cdot p_{0'} + \alpha \cdot p_1 + \alpha \cdot p_2 + \cdots$

The Final Result

- Eyal and Sirer show that the expected revenue **exceeds that of the honest strategy** as long as

$$\alpha > \frac{1-\gamma}{3-2\gamma}$$



How to Fix it?

- One simple idea is to choose $\gamma = 1/2$
  - This means choosing which fork to mine **uniformly at random**
- The threshold for α moves to ¼
  - Need to assume that **¾-fraction** of computing power is honest
  - Smaller than the believed **½-fraction** but better than current reality

Summary of Other Attacks

- Whale transactions
  - Make transactions with **huge fees**
  - Incentivizes miners to mine on **old blocks**
  - Accidentally happened in the past
- Flood attack
  - Send big amount of **small transactions**
  - Countermeasure: Increase transactions fees

What Does Bitcoin Actually Achieve?

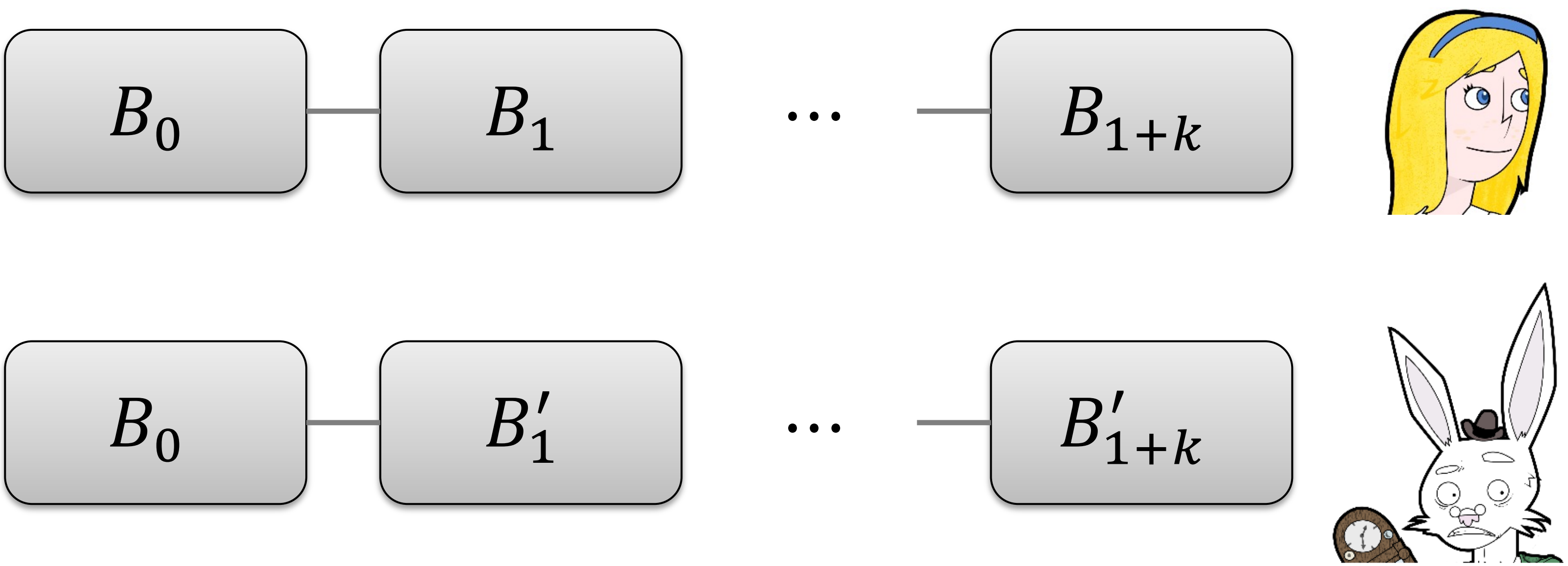
- What are the **exact security properties** achieved by Bitcoin? And under what **assumptions**?
  - J. A. Garay, A. Kiayias, N. Leonardos. "The Bitcoin backbone protocol: Analysis and applications." EUROCRYPT 2015
  - R. Pass, L. Seeman, A. shelat: "Analysis of the Blockchain protocol in asynchronous networks." EUROCRYPT 2017
  - And many more, …

Eventual Consensus

- The following properties holds with **overwhelming probability**
  - **Safety:** If two or more honest parties report a transaction as stable ($> k$ blocks deep), then it will always be in the same position
  - **Liveness:** Every transaction is eventually committed by all honest nodes
- The above two properties can also be derived from the following alternative properties

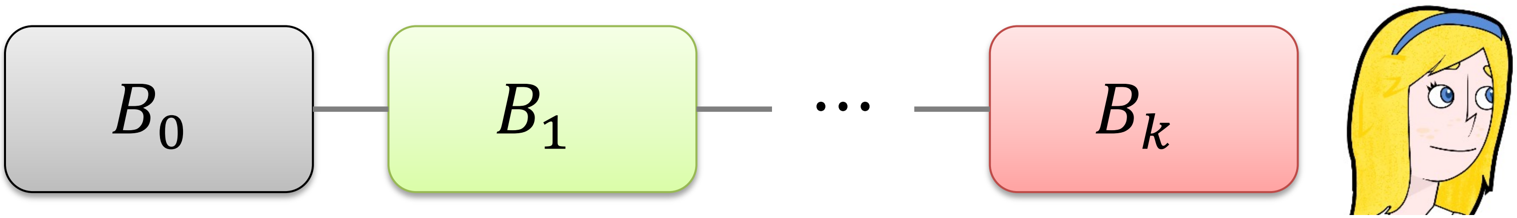
Common Prefix

- For any chains C_1, C_2 possessed by honest parties, **pruning** k blocks of one of the two chains yields a **prefix** of the other chain



Chain Quality and Chain Growth

- **Chain Quality:** For any chain C adopted by honest parties, at least one of the last k blocks was honestly generated



- **Chain Growth:** For any chain C adopted by honest parties, then the number of blocks appearing in any portion of C spanning s prior slots is at least τs

Nakamoto's Consensus

- Recall
  - **Longest chain wins.** Each node mines on the longest chain
  - **Disseminate blocks.** Upon adopting a new longest chain, via mining or by receiving from others, a node broadcasts the newly acquired block(s)
  - **Commit.** A node commits a block if it is buried at least k blocks deep in the longest chain adopted by that node

The Main Result

- We will show:

Theorem. Let $g = e^{-\alpha\Delta}$. Nakamoto's consensus satisfies **safety** and **liveness** as long as

$$\alpha \cdot g^2 > \beta$$

- Here, α and β are the honest and malicious **mining rates** and Δ is the **network delay**
- The value g^2 is the **loss** due to network delays

The Model

- Sinchrony: **known** message delay bound Δ
  - For P2P networks, take **diameter** into account
  - The attacker **controls the delay** within $(0, \Delta)$
- Simple **memoryless** mining
  - Poisson processes
  - α, β are the **collective** honest and malicious **mining rates**
  - α, β **do not change** (perfect difficulty adjustment)
- The assumption on β is **very strong**, but will allow for a **simple proof**

Poisson Processes

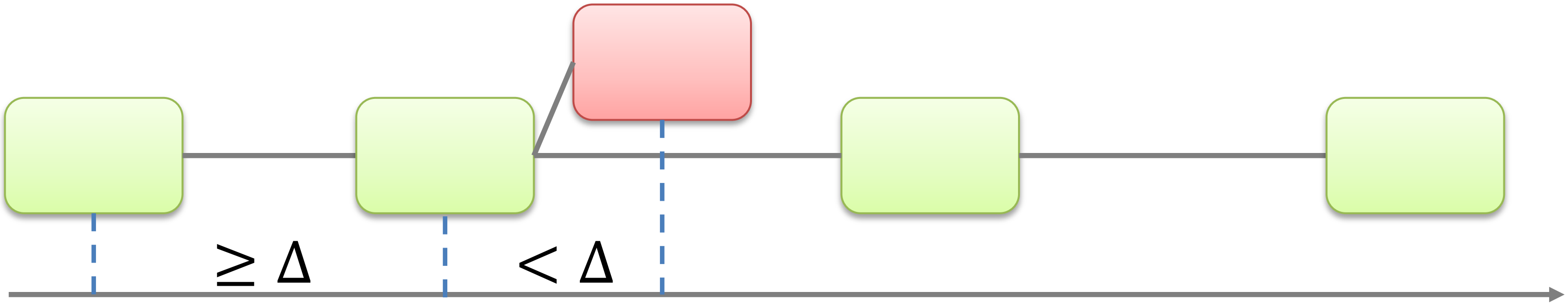
- Models arrivals of **memoryless stream**
  - Main parameter is **the rate** λ
- In a **time window** of size t, the probability of having k events is $e^{-\lambda t}\,(\lambda t)^k/k!$
  - The time till the next block **does not depend** on how much time elapsed since the previous block
- The gap time T between two **consecutive** blocks follows an i.i.d. exponential distribution
  - $\Pr[T > \Delta] \leq e^{-\lambda\Delta}$

Proof Intuition

- To prove safety, we would like to show that
  - Honest blocks **contribute** to safety
  - Malicious blocks **undermine** it
  - So, safety holds as long as $\alpha > \beta$ (**honest majority**)
- But we need to consider **network delays**
  - Not all honest blocks extend one another on the same chain (due to **forks**)
  - In the proof we show most of them do

Tailgaiters and Non-tailgaters

- Assume for simplicity there is **no adversary**
- Two honest blocks do **not** extend each other if they are mined **too close** (i.e. $< \Delta$)
  - Suppose an honest block B is mined at time t
  - If no other honest block is mined between $t - \Delta$ and t, then B is a **non-tailgater**
  - Otherwise, B is a **tailgater**



Properties of Non-tailgaters

- Non-tailgaiters do **not** have the **same height**
  - Because the two blocks are Δ apart
  - So the later block will be at a height **higher** than the earlier block (the longest chain rule)
- Moreover, we can compute the **fraction** of honest tailgaters and non-tailgaters
  - By Poisson, $\Pr[T > \Delta] \leq e^{-\alpha\Delta} = g$
  - Each honest block is a tailgater w.p. $1 - g$ and a non-tailgater w.p. g

Concluding Liveness

- On **expectation**, the number of non-tailgaters grows at a rate of $g\alpha$
- Because non-tailgaters have **different heights**, the longest chain also grows at a rate of $g\alpha$
- Since $g\alpha > g^2\alpha \geq \beta$, we get liveness
  - The actual proof is a bit **more complex**, as one needs to show that the **actual outcome** is unlikely to deviate much from the expected outcome

Loners

- Suppose an honest block B is mined at time t
- We say B is a **loner** if no other honest block is mined between time $t - \Delta$ and $t + \Delta$
- A loner is the **only** honest block at its height
  - Simply because a loner and any other honest block **do not tailgate one another**
- A loner requires **two** back-to-back non-tailgaters
  - The probability of being a loner is g^2

Concluding Safety (1/2)

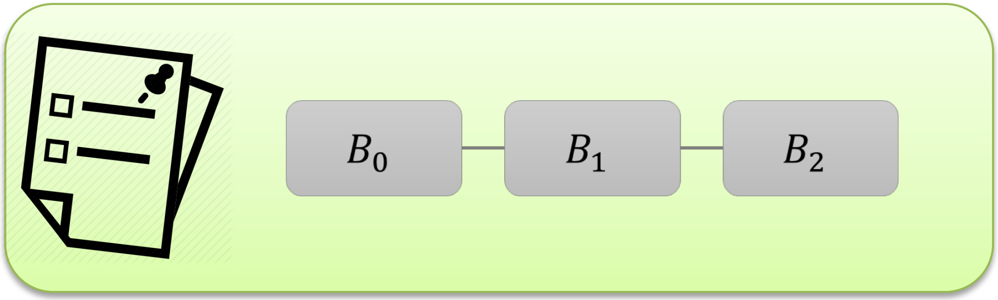
- Violating safety requires two chains that **diverge** by more than k blocks
  - Both adopted by **honest** nodes
- Consider the time **window** in which these two diverging chains are mined
  - As loners do not share heights with honest blocks, we can **pair** each loner with a **malicious** block
  - Thus, there has to be **more malicious blocks than loners**

Concluding Safety (2/2)

- Thus, to **violate safety**
  - At some point, the adversary **mines more blocks** than honest nodes **mine loners**
  - If honest nodes can mine loners **faster** than the adversary can mine blocks, then **safety holds**
- Since the **expected** loners rate is $g^2\alpha$, the theorem follows
  - The actual proof is a bit **more complex**, as one needs to show that the **actual outcome** is unlikely to deviate much from the expected outcome

Bitcoin as a Robust Transaction Ledger

- C. Badertscher, U. Maurer, D. Tschudi, V. Zikas. "Bitcoin as a transaction ledger: A composable treatment." CRYPTO 2017

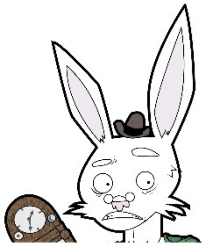


Payment Channels

Specifying Conditions in Bitcoin

- **Strange** transactions:

A boolean function

P_2 

$T_2 =$ | T_1 | 1 BTC | Condition C_2 to spend T_2 | Witness W_2 |

$[T_3]$

P_3 

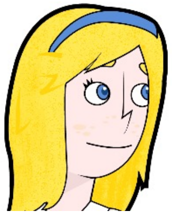
$T_3 =$ | T_2 | 1 BTC | Condition C_3 to spend T_3 | Witness W_3 |

- T_3 **redeems** T_2 if C_2 outputs true upon $([T_3], W_3)$
- Standard transactions:

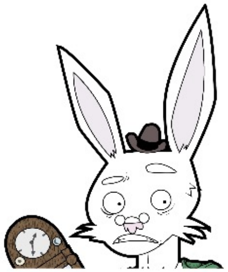
$$C_2([T_3], W_3) = \mathbf{V}(pk_2, [T_3], W_3)$$

Example

A previous transaction that can be spent by Alice



$T_2 =$	T_1	1 BTC	Condition $C([T], p, q, \sigma) = 1$ iff $p, q > 1$ and $p \cdot q = 2501$ and σ is Bob's signature on $[T]$	Alice's signature



$T_3 =$	T_2	1 BTC	Can be spent using Bob's signature	$p = 41$ and $q = 61$ and Bob's signature on $[T_3]$

How to do this?

- The conditions are specified using Bitcoin's scripting language
  - **Not Turing complete** (as we want transactions to be verified quickly)
  - Hard to post strange transactions (miners **might not accept them**)

OP_DUP OP_HASH160
02192cdf64739gt5es9sdfq13apeoir984de4r4o
OP_EQUALVERIFY OP_CHECKSIG

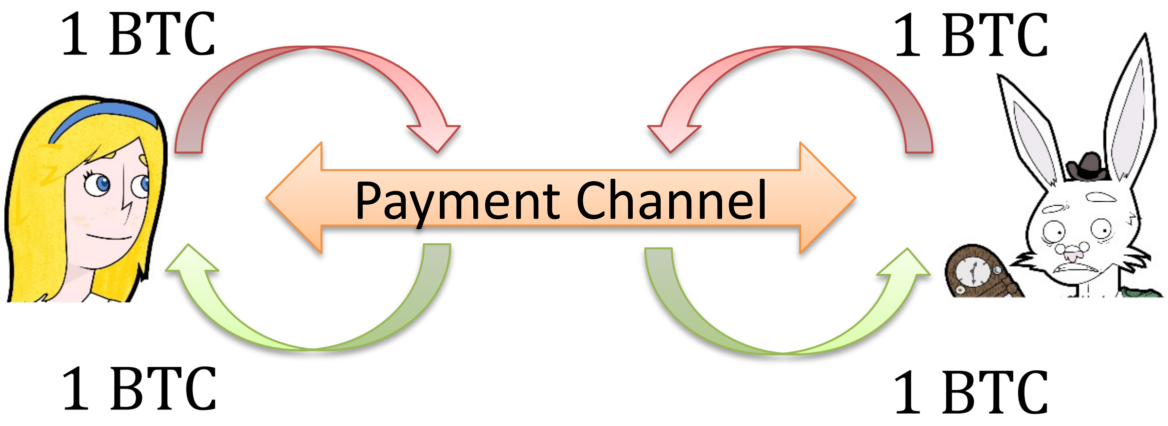
Bitcoin Contracts

- The strange transactions can be used to create so-called **Bitcoin contracts**
- Examples
  - Payment channels
  - Pay money to whoever knows some password
  - Assurance contracts
  - Put a deposit to prove you are not a spammer
  - Pay money only if some event happens
  - Decentralized organizations (avoid lawyers)

Micropayments

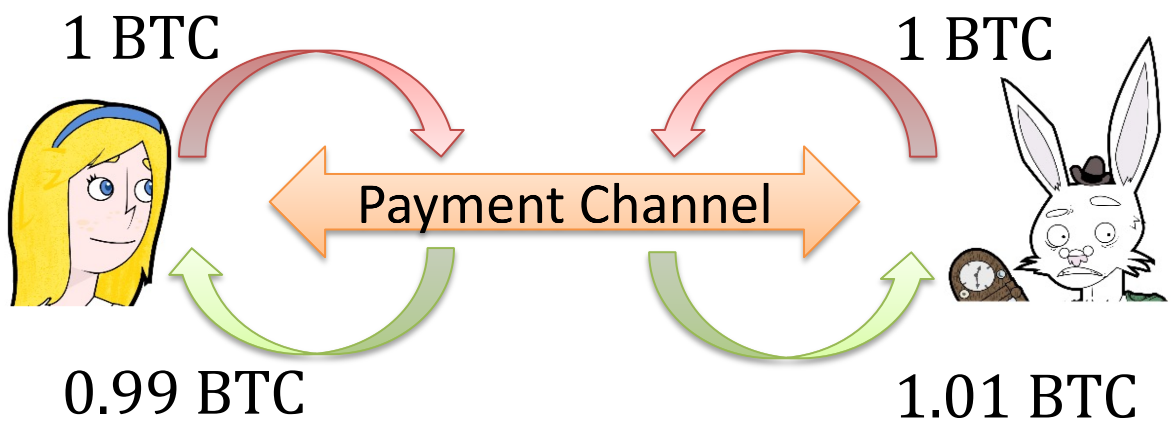
- Hard to make **micropayments** in Bitcoin
  - I.e., payments worth a fraction of a cent
  - E.g., for wifi connection or for downloading data
- Reasons:
  - Non-negligible transactions fees
  - Long transaction confirmation time
- **Inherent** limitation (7 trans/sec)
- Can be solved via so-called **payment channels**
  - E.g., the Lightning Network

Payment Channels (1/4)



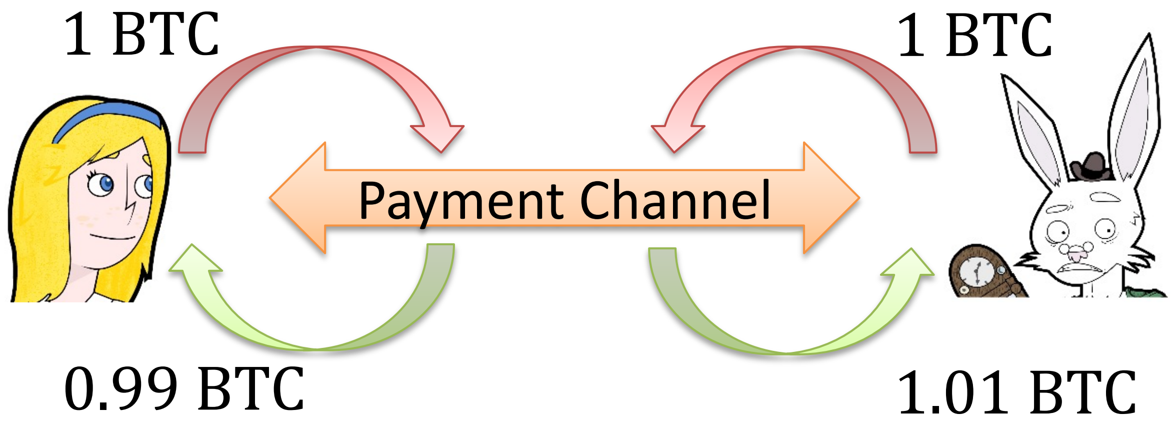
- **Opening** the channel: Agree to establish the channel and **charge it** with, say, 1 BTC
  - This requires operations on the blockchain
  - Agree on how much each party gets out at end (virtual agreement, not on the blockchain)

Payment Channels (2/4)



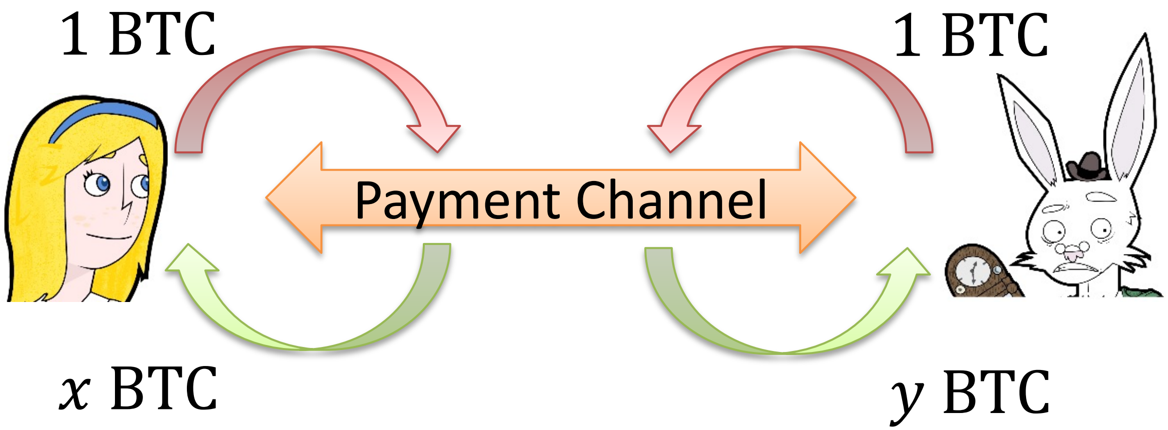
- Suppose Alice wants to pay 0.01 BTC to Bob
  - Say for using his website
- **Adjust** the state of the channel accordingly
  - Without informing the blockchain

Payment Channels (3/4)



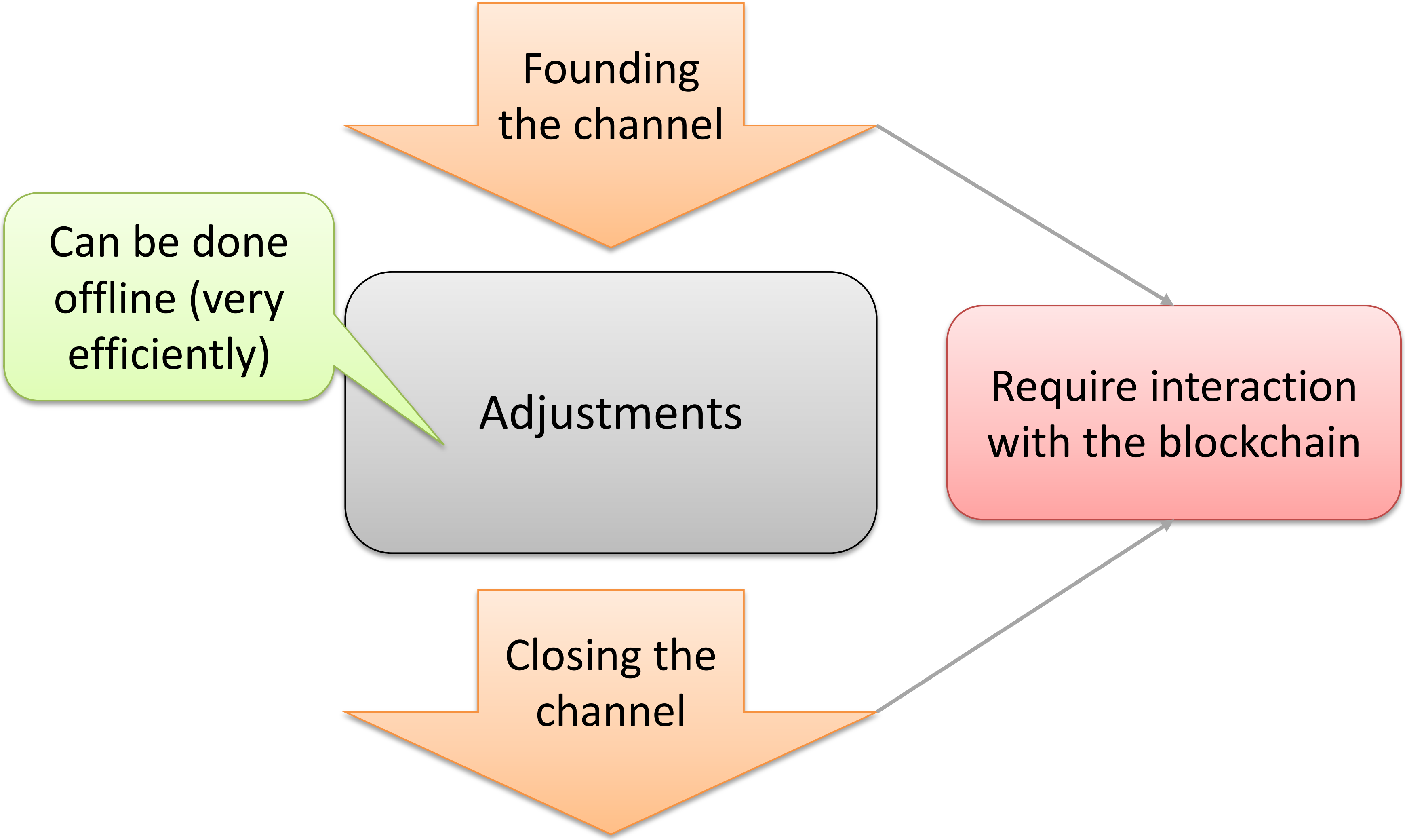
- In general **for any state** of the channel (x, y) such that $x + y = 2$
  - If Alice wants to pay $x' \leq x$ to Bob, the new state becomes $(x - x', y + x')$
  - Neglecting transactions fees

Payment Channels (4/4)

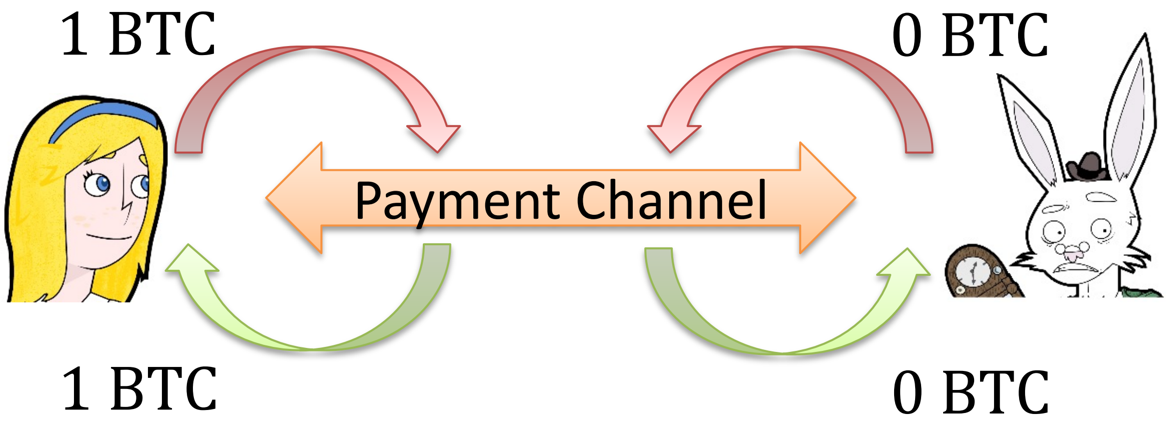


- **Closing** the channel: At the end Alice and Bob can close the channel and get real money back
  - Either because the micropayments are over
  - Or because they enter in some form of **disagreement**

General Picture



Unidirectional Channels



- Let's start with the case where only Alice can pay to Bob
  - This is called a **unidirectional** payment channel

Tool: Multi-Signature Transactions

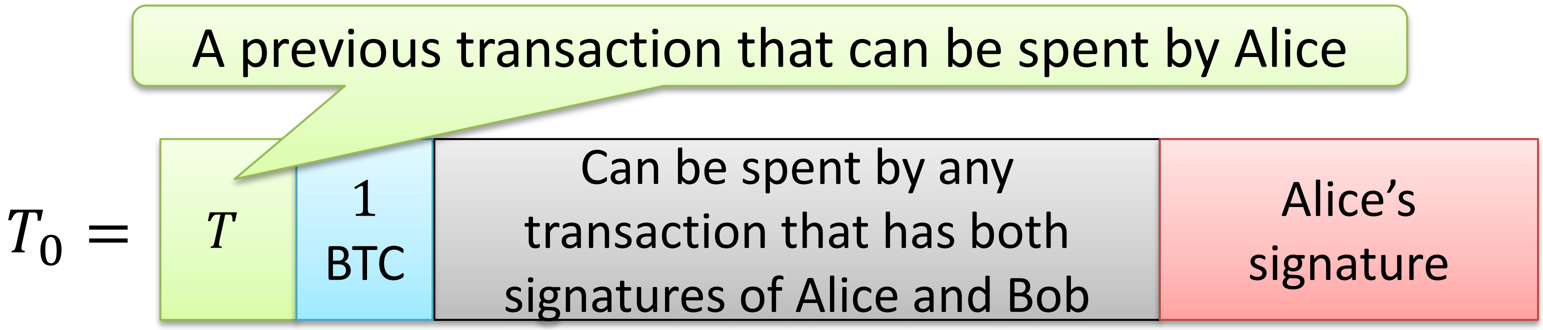
- Special transactions that can be claimed only by providing signatures from k users (out of a set of n users)
  - This is a k-out-of-n **multisignature** transaction

A previous transaction that can be spent by Charlie

$T_2 =$	T_1	1 BTC	Can be spent by any transactions that has both signatures of Alice and Bob	Charlie's signature
$T_3 =$	T_2	1 BTC	Can be spent by Dave	Alice sig. Bob sig.

Founding a Channel (1/3)

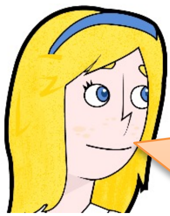
- Alice creates a **founding transaction** as follows:



- Can Alice post T_0 on the blockchain?
  - It is a bit **risky**
  - If Bob does not coperate her money could be **locked forever**!

Founding a Channel (2/3)

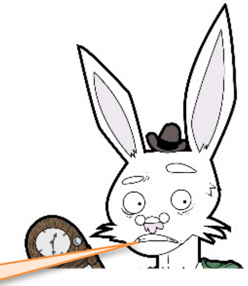
- Solution: Ask Bob to sign a **refund transaction** T' with a timelock



Please sign this:

$T' =$

T_0	1 BTC	Can be spent by Alice after 30 days have passed



Here is my signature: Bob's signature

  - Good news: This can be done **without knowing** T_0
  - Bad news: There are problems with transactions malleability (let's ignore them here)

Founding a Channel (3/3)

- Alice sure **she gets her money** back in 30 days
  - By adding her own signature on T'
  - And posting T' on the blockchain

$T' =$

T_0	1 BTC	Can be spent by Alice after 30 days have passed	Alice's sig. Bob's sig.

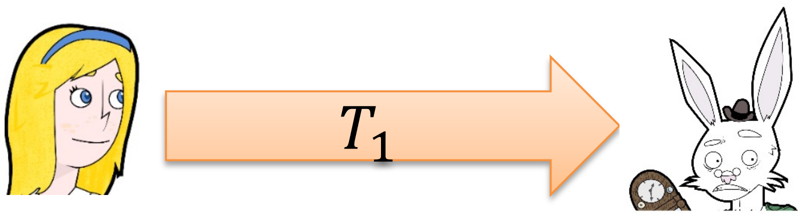
- So, she can now **safely post** T_0 to found the payment channel

Making Micropayments (1/2)

- In order to send 0.01 BTC to Bob, Alice sends him the following transaction T_1:

$T_1 =$

Alice sends 0.99 BTC from T_0 to Alice Alice sends 0.01 BTC from T_0 to Bob if 29 days have passed	Alice's sig.



- How can Bob get real money from T_1?
  - Can just sign T_1 and post it on the blockchain
  - But has to do so **before day 30**, otherwise Alice **can steal all the money**

Making Micropayments (2/2)

- In general in order to send y BTC to Bob, if the last transaction sent by Alice was

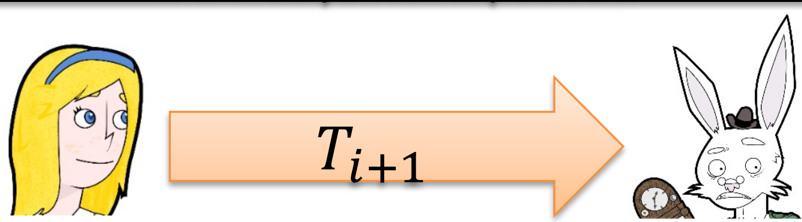
$T_i =$

Alice sends x BTC from T_0 to Alice Alice sends $1 - x$ BTC from T_0 to Bob if 29 days have passed	Alice's sig.

- She can adjust it as follows:

$T_{i+1} =$

Alice sends $x - y$ BTC from T_0 to Alice Alice sends $1 - (x - y)$ BTC from T_0 to Bob if 29 days have passed	Alice's sig.



Closing the Channel

- To close the channel, Bob simply adds his signature to the last transaction T_i and posts it (by day 29)

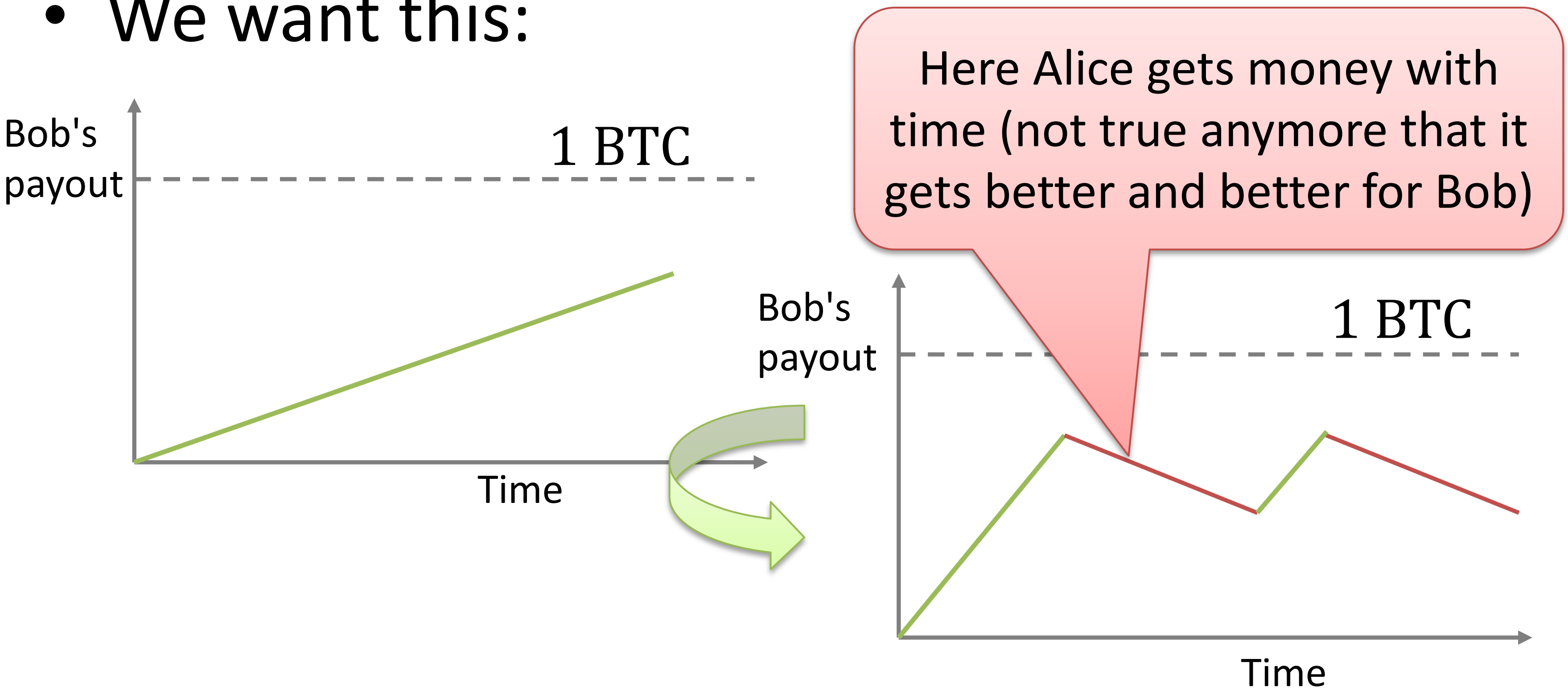
$T_{i+1} =$

Alice sends $x - y$ BTC from T_0 to Alice Alice sends $1 - (x - y)$ BTC from T_0 to Bob if 29 days have passed	Alice's sig. Bob's sig.

- Why the last?
  - As the T_i's only get **better and better** for him
- To close the channel, Alice has to **wait (or ask Bob)**

Bi-Directional Channels (1/3)

- What if Bob wants to **pay something back** to Alice?
- We want this:



Bi-Directional Channels (2/3)

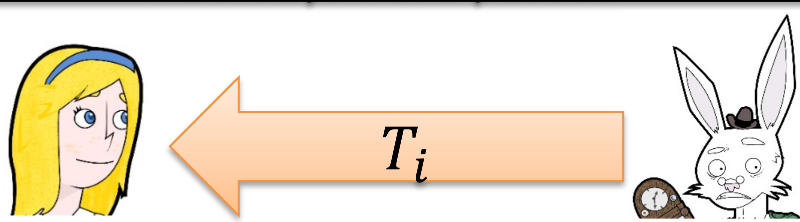
- Let's focus on a **single inversion**
- Assume the state of the channel is:

$$(1 - y \text{ BTC to Alice}, y \text{ BTC to Bob})$$

- Now Bob sends signed transactions to Alice
  - E.g. to transfer y' BTC:

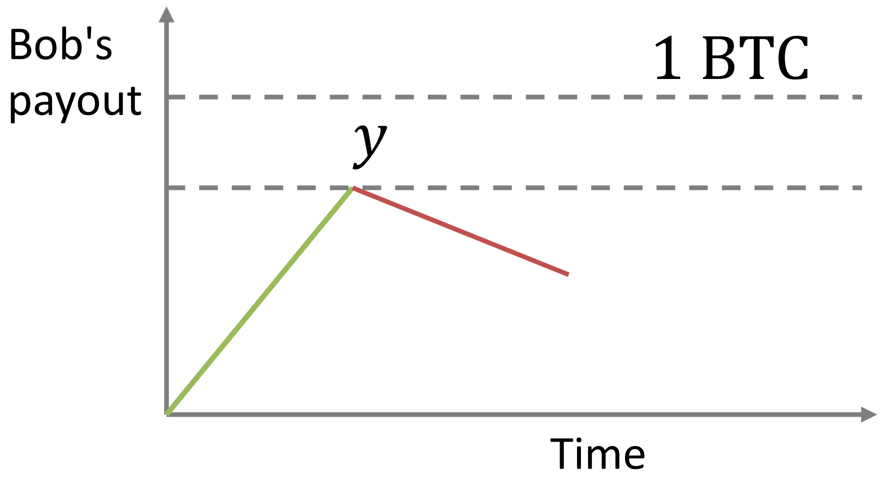
$T_i =$

Alice sends $1 - (y - y')$ BTC from T_0 to Alice Alice sends $(y - y')$ BTC from T_0 to Bob if 28 days have passed	Bob's sig.



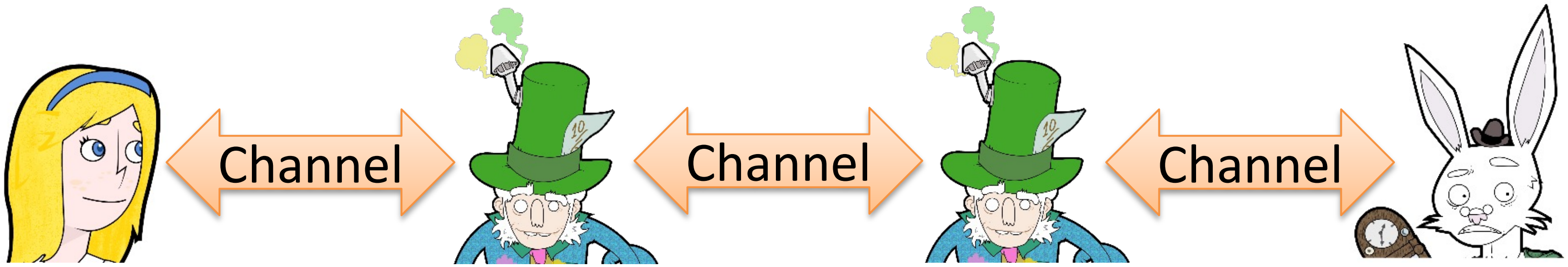
Bi-Directional Channels (3/3)

- Why the timelock is now **28 days**?
  - Remember: Bob is now **losing money**
  - At day 29 he could post the transaction that gives him y BTC
  - We need to allow Alice to **react earlier**



Payment Networks

- Previous solution requires **a different channel** per pair of parties
- Can we do better?
  - Yes, let's make the parties **route the payments**
  - Possibly at a fee



Payment Networks (2/3)

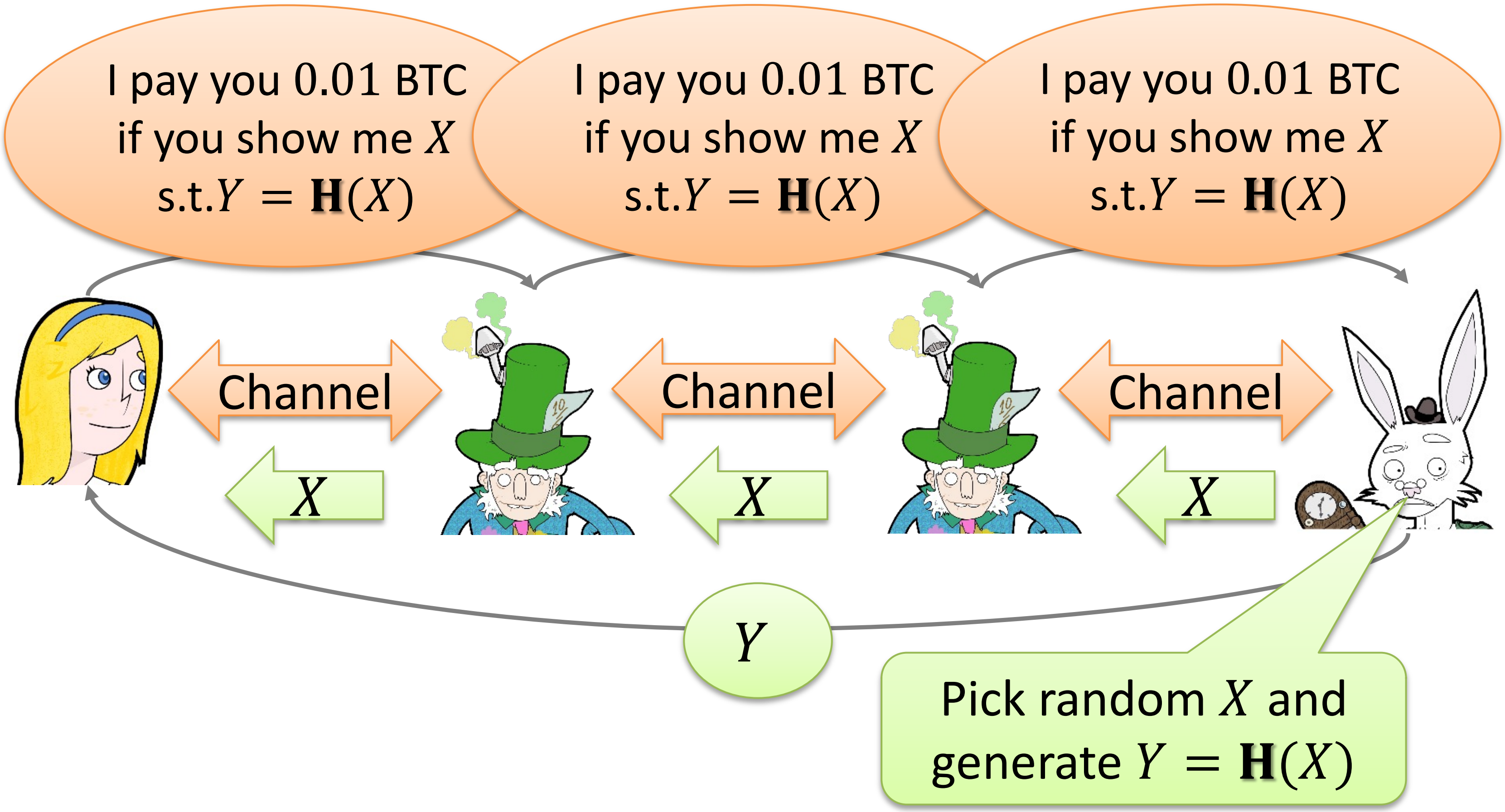
- What if the intermediaries are **untrusted**?
- Solution based on **hash-locked transactions**
  - Let $\mathbf{H}$ be a hash function and $Y = \mathbf{H}(X)$
  - Can be redeemed only by publishing X

$T_2 =$

T_1	1 BTC	Can be spent using Bob's signature and X s.t. $Y = \mathbf{H}(X)$	Alice's signature

Payment Networks (3/3)

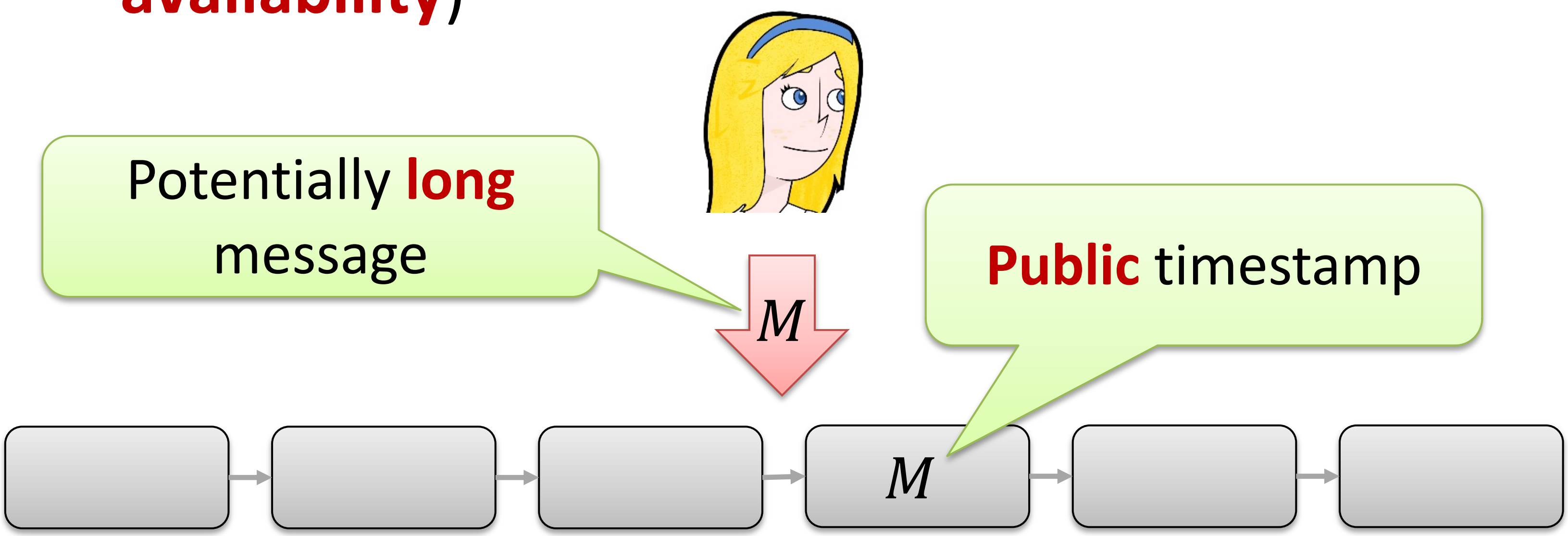
- Sketch of the solution:



Plasma

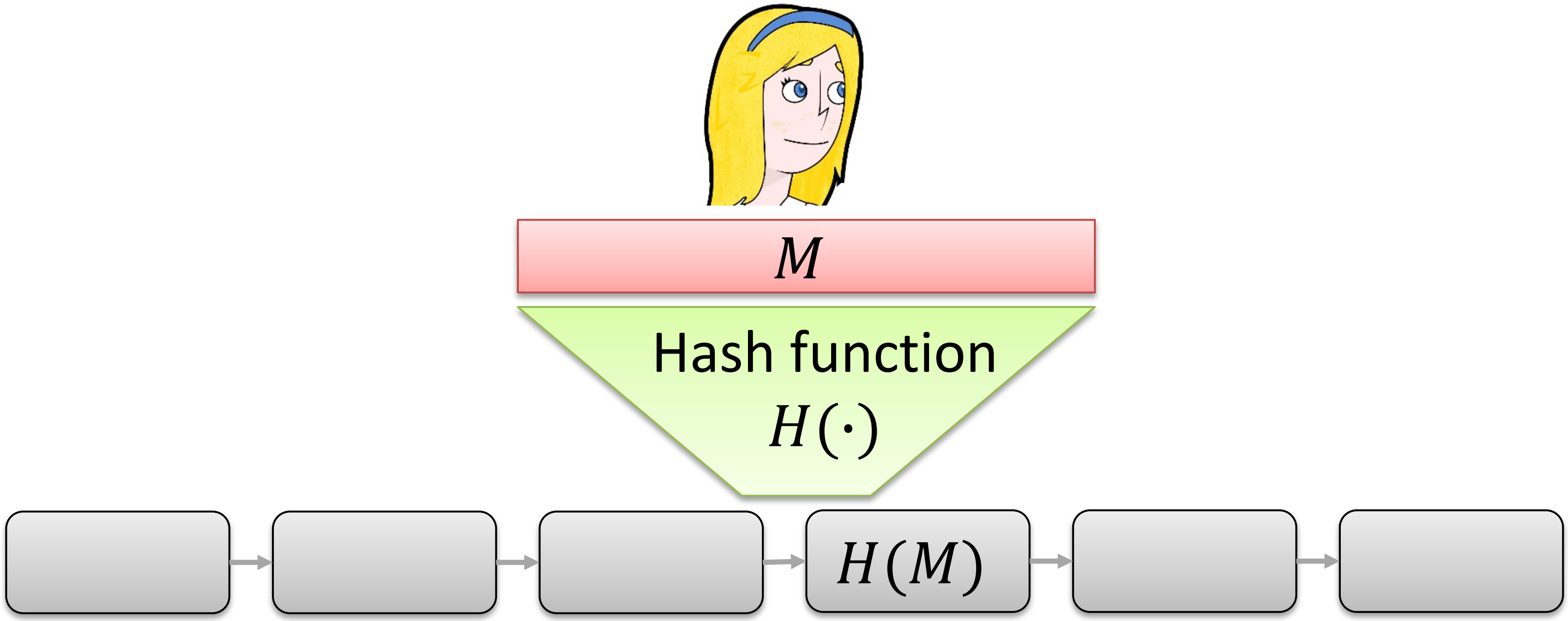
Blockchain as Public Timestamping

- Each user can **prove** that:
  - He **knew** some message M at some point
  - The message M was **publicly available** (**data availability**)

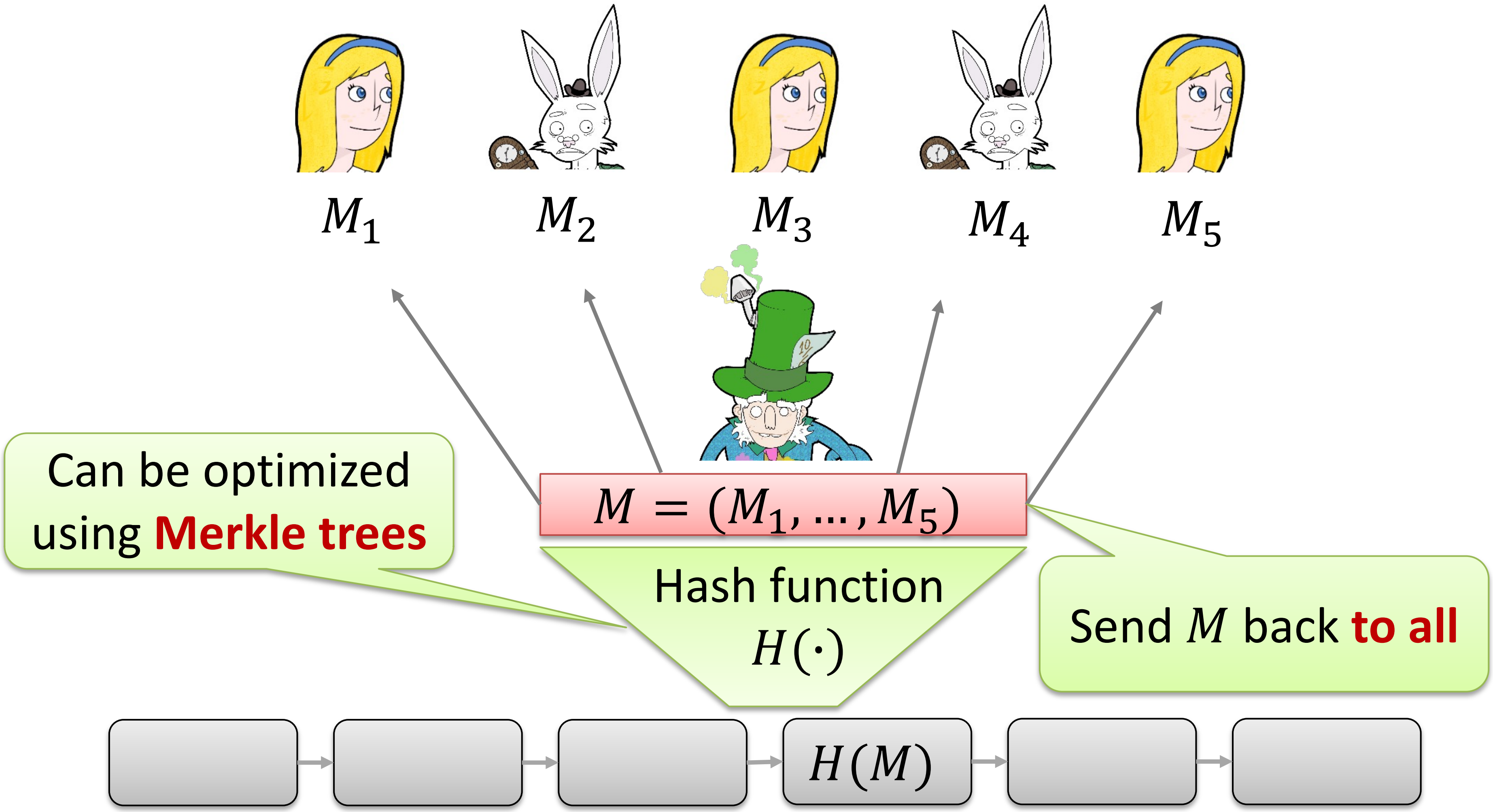


A Relaxation

- Assume we **give up** on **data availability**
  - Namely, we only care about proving knowledge
- Then, we can **improve efficiency**

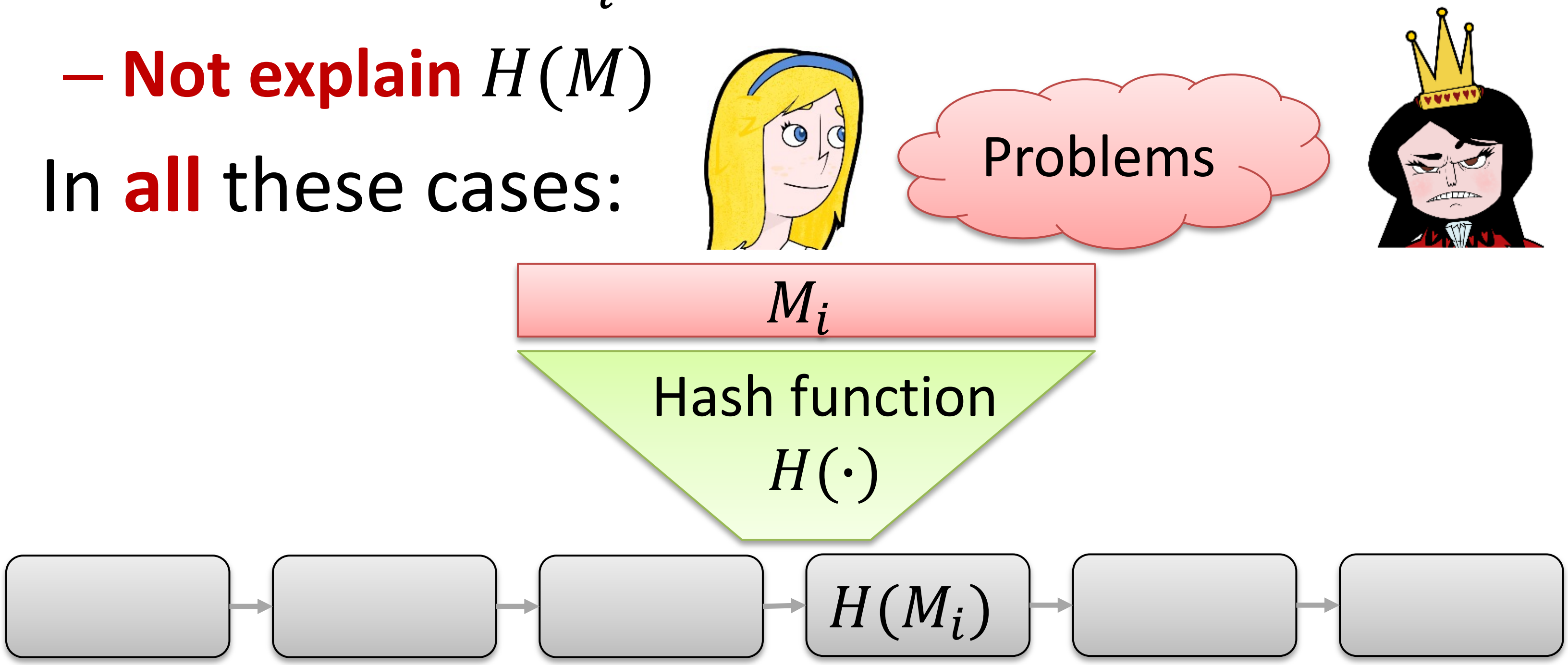


Even Better…


M_1
M_2
M_3
M_4
M_5
Can be optimized using **Merkle trees**
$M = (M_1, \ldots, M_5)$
Hash function $H(\cdot)$
Send M back **to all**
$H(M)$

What Can Go Wrong?

- A **malicious** operator can:
  - **Not publish** $H(M)$
  - **Exclude** some M_i
  - **Not explain** $H(M)$
- In **all** these cases:



In Summary

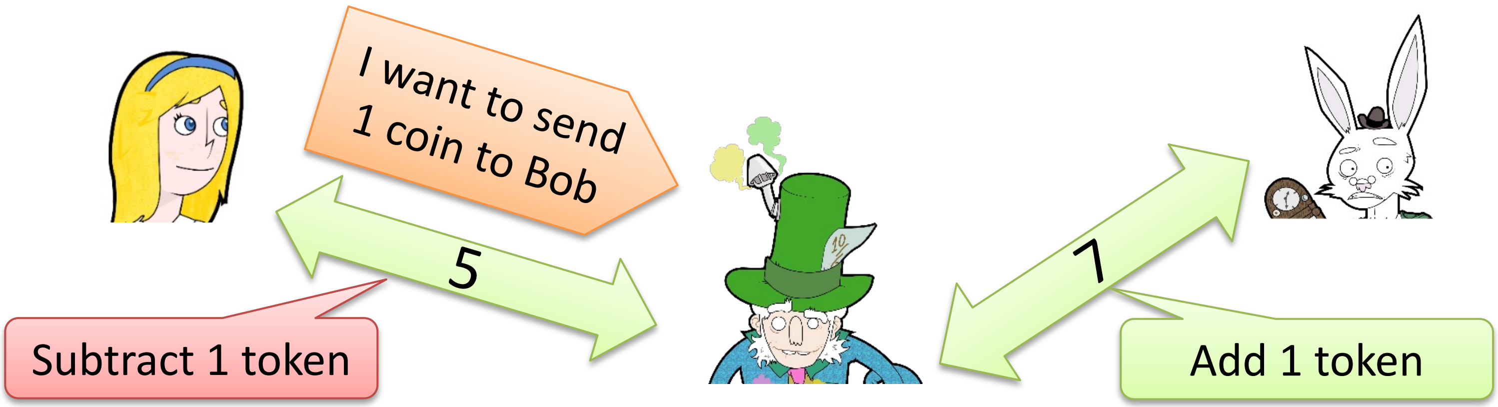
- If the operator is **honest**, this solution **saves** a lot of blockchain space
  - Call this the **optimistic scenario**
- If the operator is **malicious**, **nothing** really bad happens
  - Timestamping just takes **more time**
  - Call this the **pessimistic scenario**
- **Weak** vs **strong** timestamping
  - Can the latter can be obtained using the former?

Plasma

- Main idea: Apply the **hash-and-timestamp** idea to the **ledgers**
  - J. Poon, V. Buterin: Scalable Autonomous Smart Contracts, 2017
- A single operator maintains **its own ledger** L
  - This is called the "Plasma ledger"
  - The ledger L is published **off-chain** (say, on the operator's website)
  - **Periodically**, the operator publishes $H(L)$ **on-chain**, in a smart contract that he deployed

Plasma Ledger

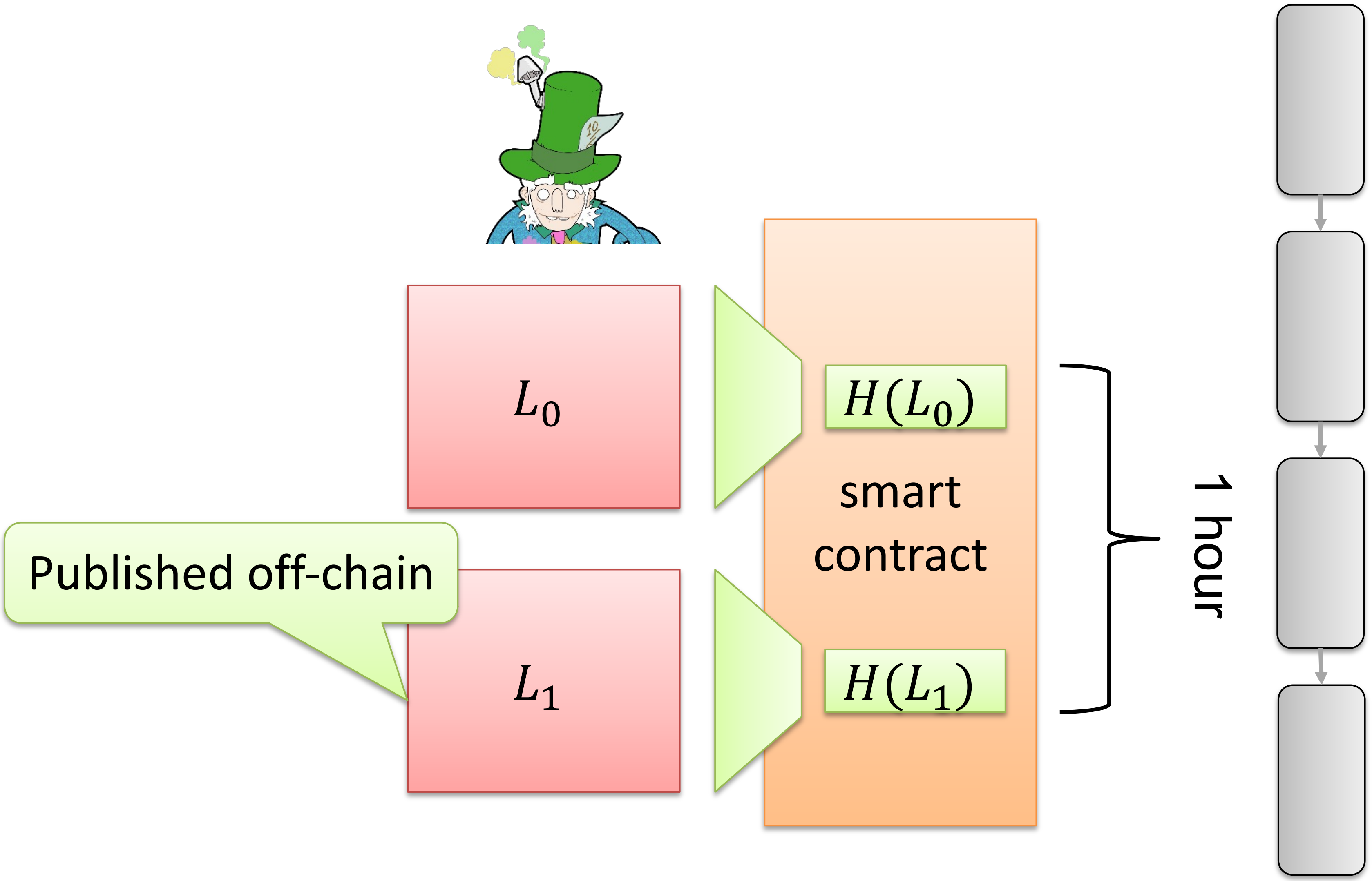
- Users own **tokens**
  - Can be **exchanged** with coins on the main chain
  - Can exit the Plasma ledger **at any time**



User	Tokens
	5
	7

User	Tokens
	4
	8

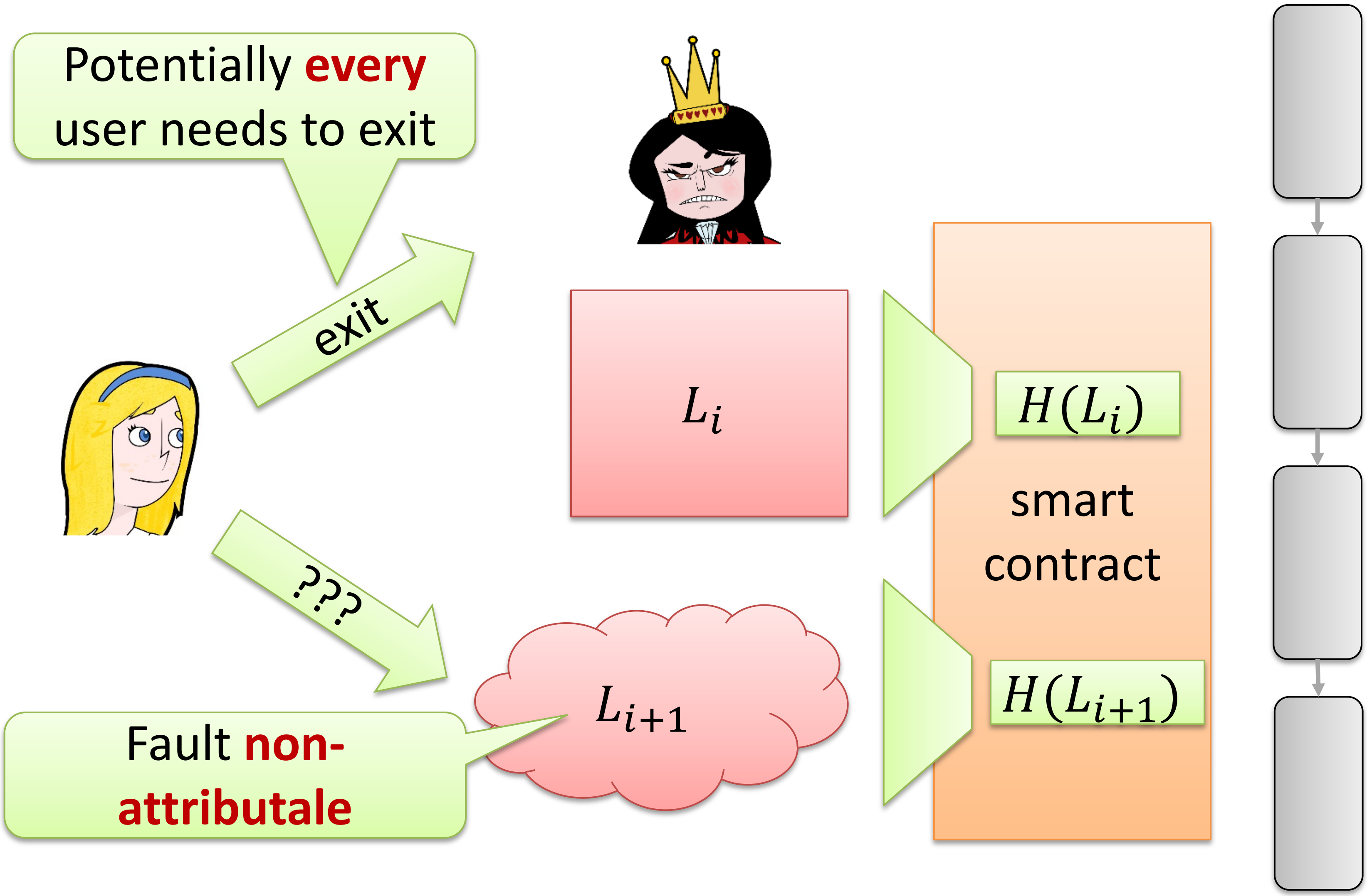
Periodic Commitments



Main Features

- As long as the operator is **honest**, Plasma provides **huge savings** on transactions fees
- A **dishonest** operator **cannot steal** money
- In case of problems, disputes can be resolved **on-chain** via the smart contract
  - Each user must **monitor** the operator’s webpage and the main page
- Main challenge: How to deal with **data unavailability**

What If There Is Data Unavailability?



Attributable Faults

- **Uniquely** attributable
  - The smart contract **knows what** went wrong (e.g., a user signs contradictory messages)
  - Malicious parties can be, e.g. **financially penalized**
- **Non-uniquely** attributable
  - The smart contract knows something went wrong but **can't determine** whose fault it was
  - Who does pay the fee? Natural idea would be 50/50, but rich players may not care (**griefing**)

Kinds of Plasma

- Plasma Cash
  - Tokens as **individual entities** (cannot be merged)
  - Ledger $L: \{1, \ldots, n\} \rightarrow \{\bot, U_1, \ldots, U_m\}$, where $\bot$ means that a token was withdrawn
  - Exit with k coins takes $\Omega(k)$ communication
- Fungible Plasma (Plasma MVP)
  - Tokens can be **merged**
  - Ledger $L: \{U_1, \ldots, U_m\} \rightarrow \mathbb{R}$
  - Exit with k coins takes $O(\log k)$ communication

Comparison

	Exit Size	NUA Faults
Plasma Cash	Large	NO
Fungible Plasma	Short	YES

- **Can't get** the best of both worlds
  - S. Dziembowski et al. Lower Bounds for Off-Chain Protocols: Exploring the Limits of Plasma. ITCS'21