CSCI 1515 Applied Cryptography

This Lecture:

- SNARGs from PCP (Continued)
- Blockchain

Proof Systems for Circuit Satisfiability

NP relation $R_{L_C} = \{(x, w) : C(x, w) = 1\}$

	NP	Σ-Protocol	(Fiat-Shamir) NIZK
	$P(x, w) \xrightarrow{w} V(x)$	$P(x, w) \rightleftarrows V(x)$	$P(x, w) \xrightarrow{\pi} V(x)$
Zero-Knowledge	NO	YES	YES
Non-Interactive	YES	NO	YES
Communication	$O(\lvert w\rvert)$	$O(\lvert C\rvert \cdot \lambda)$	$O(\lvert C\rvert \cdot \lambda)$
V's computation	$O(\lvert C\rvert)$	$O(\lvert C\rvert)$	$O(\lvert C\rvert)$

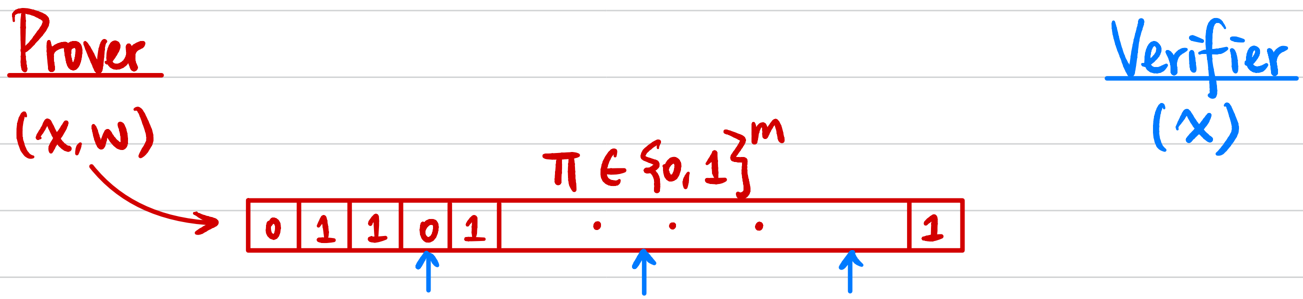
Can we have communication complexity & verifier's computational complexity sublinear in $|C|$ & $|w|$?

Succinct Non-Interactive Argument

- SNARG: Succinct Non-Interactive Argument
- SNARK: Succinct Non-Interactive Argument of Knowledge
- zk-SNARG / zk-SNARK: SNARG / SNARK + Zero-Knowledge

- Succint: $|\pi| = poly(\lambda, \log |C|)$
  Verifier runtime $poly(\lambda, |x|, \log |C|)$
- Argument: In Soundness / Proof of Knowledge: $\forall$ PPT P^*

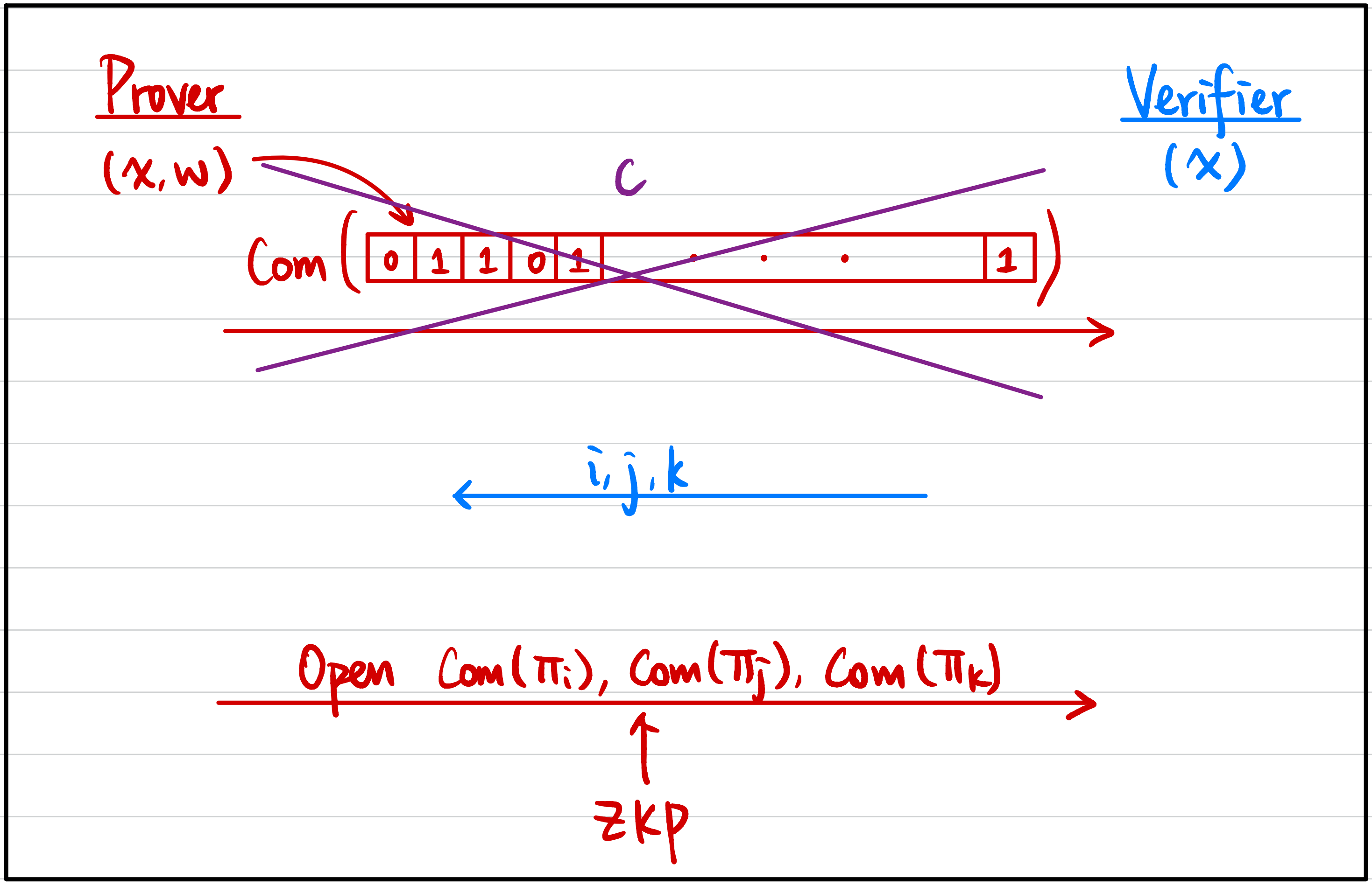
Probabilistically Checkable Proof (PCP)



PCP Theorem (Informal):

Every NP language has a PCP where the verifier reads only a constant number of bits of the proof.

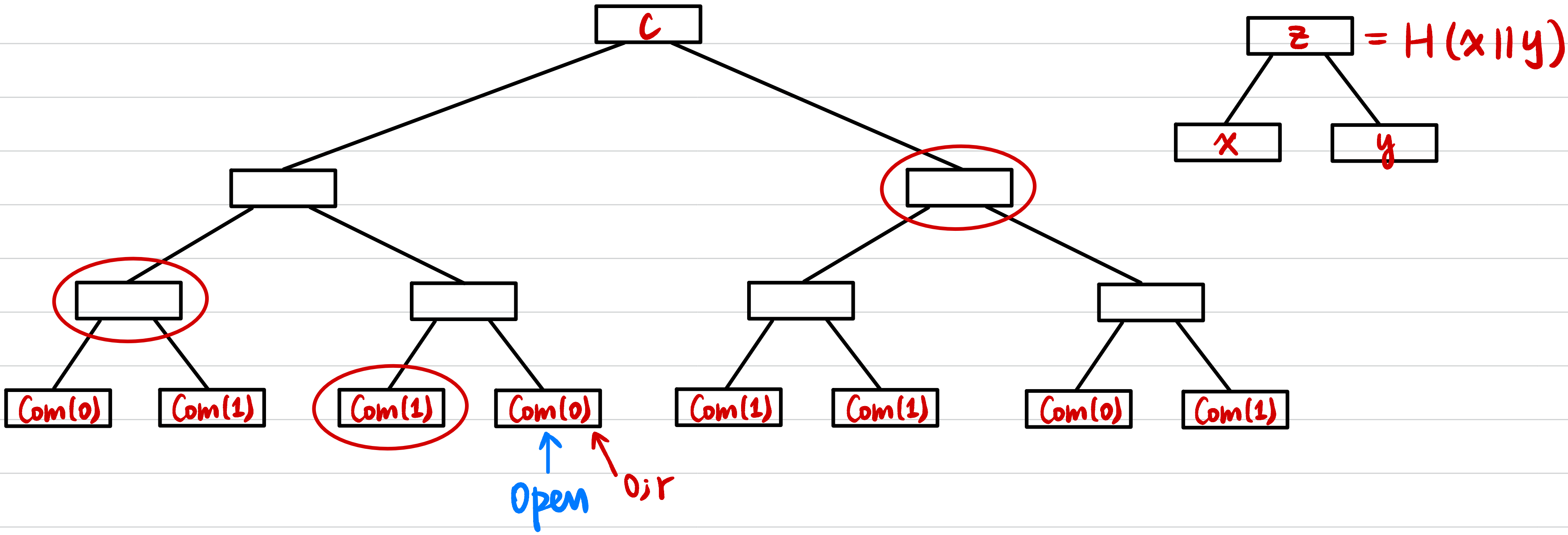
First Attempt



Is it succint?

Is it ZK?

Merkle Tree



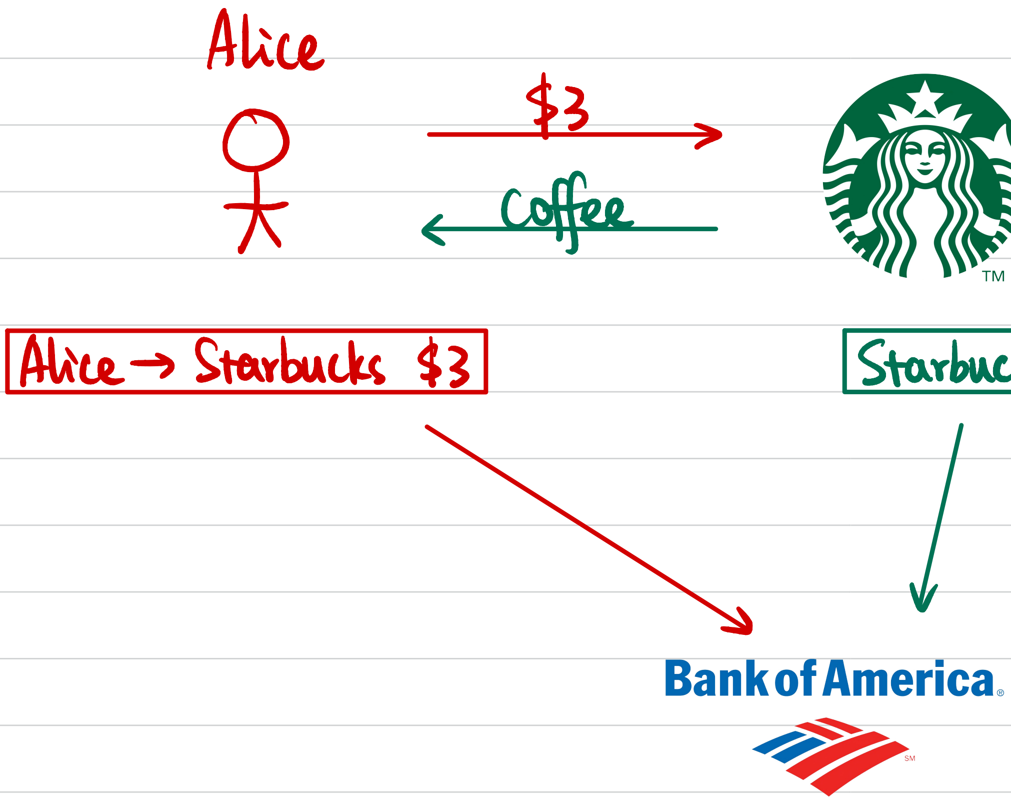
How to open commitment?

Why hiding & binding?

Hiding ← Hiding of Commitment Scheme

Binding ← Collision Resistance of Hash
Binding of Commitment Scheme

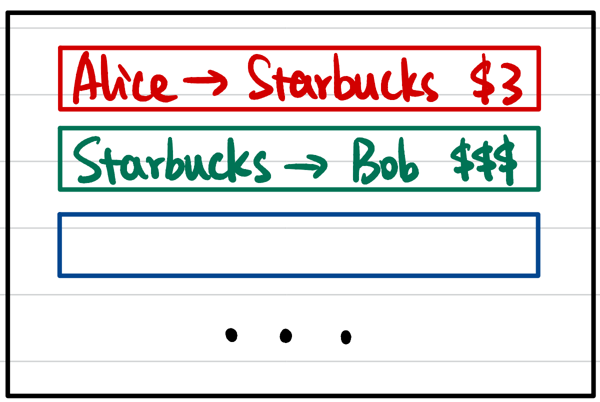
Transactions in Real Life



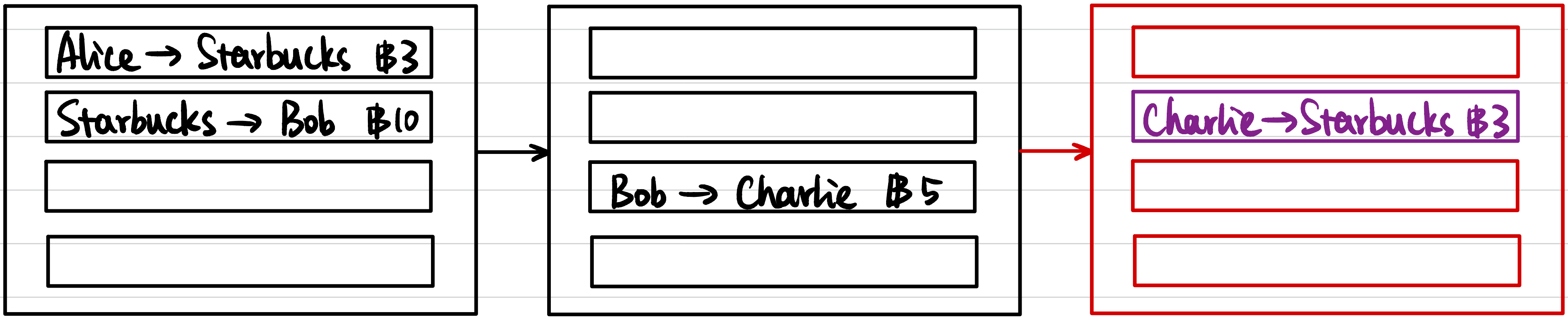
① initiated by sender

② enough balance in sender's account

A trusted party that maintains a private ledger



Blockchain



- Public ledger that everyone can view & verify
- Maintained by "miners" in a distributed way

Step 1: Charlie wants to make a transaction Charlie → Starbucks ฿3
↳ broadcasts it to the entire network

Step 2: All miners collect all transactions in the network

- Verify validity
  - ① initiated by sender ← How?
  - ② enough balance in sender's account
- Agree on next block ← How?

Step 3: Repeat

Transaction Authentication

Alice: $(vk_A, sk_A) \leftarrow KeyGen(1^\lambda)$

Bob: $(vk_B, sk_B) \leftarrow KeyGen(1^\lambda)$

Charlie: $(vk_C, sk_C) \leftarrow KeyGen(1^\lambda)$

Starbucks: $(vk_S, sk_S) \leftarrow KeyGen(1^\lambda)$

Bob → Charlie ₿5 :

$m_1 = (vk_B, vk_C, 5)$ $\sigma_1 \leftarrow Sign_{sk_B}(m_1)$

Charlie → Starbucks ₿3 :

$m_2 = (vk_C, vk_S, 3)$ $\sigma_2 \leftarrow Sign_{sk_C}(m_2)$

Consensus Protocol

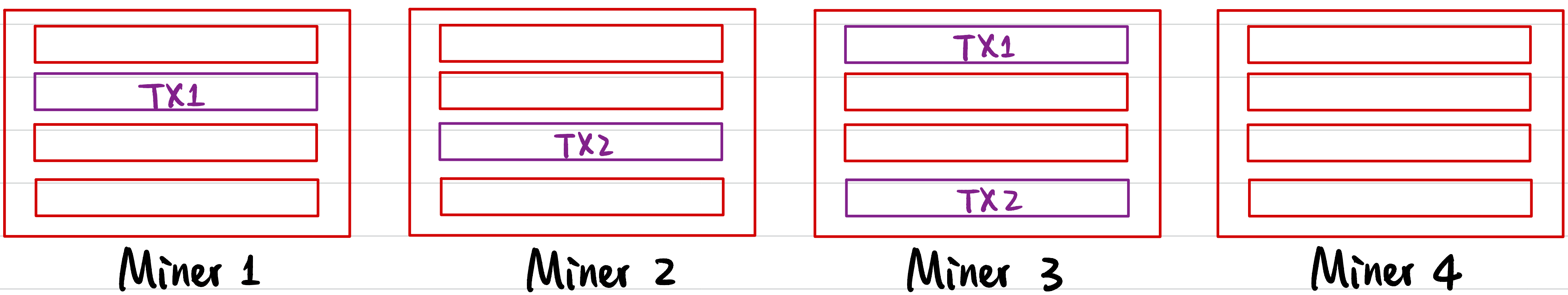
TX1 = Charlie → Starbucks ฿3 :

$m_2 = (vk_C, vk_S, 3)$ $\sigma_2 \leftarrow \text{Sign}_{sk_C}(m_2)$

TX2 = Charlie → Alice ฿4 :

$m_3 = (vk_C, vk_A, 4)$ $\sigma_3 \leftarrow \text{Sign}_{sk_C}(m_3)$

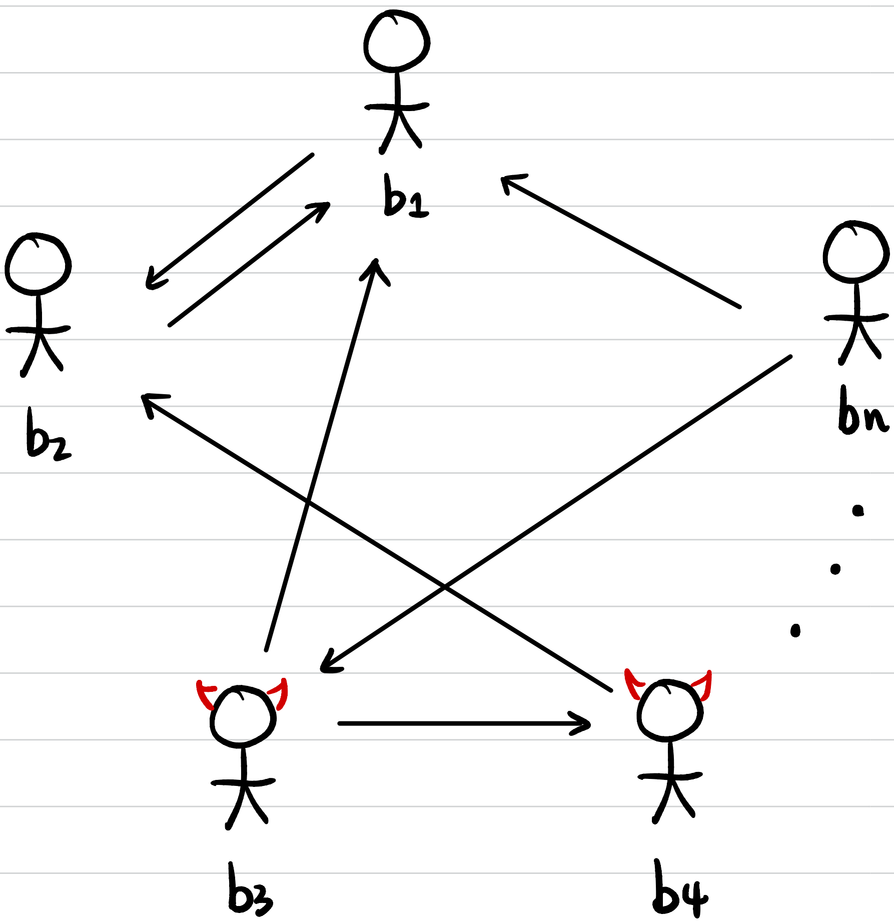
"permissionless"



WANT: ① All miners agree on the same block

② New block is valid

Byzantine Agreement



⇓

Agree on a block

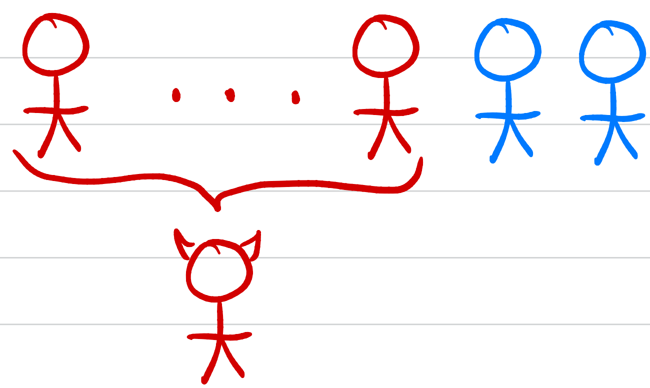
Byzantine Fault Tolerance (BFT) Protocol:

If $n \geq 3t+1$, (necessary)

then it's possible to reach consensus.

Assume $t < n/3$, then agree on a valid block.

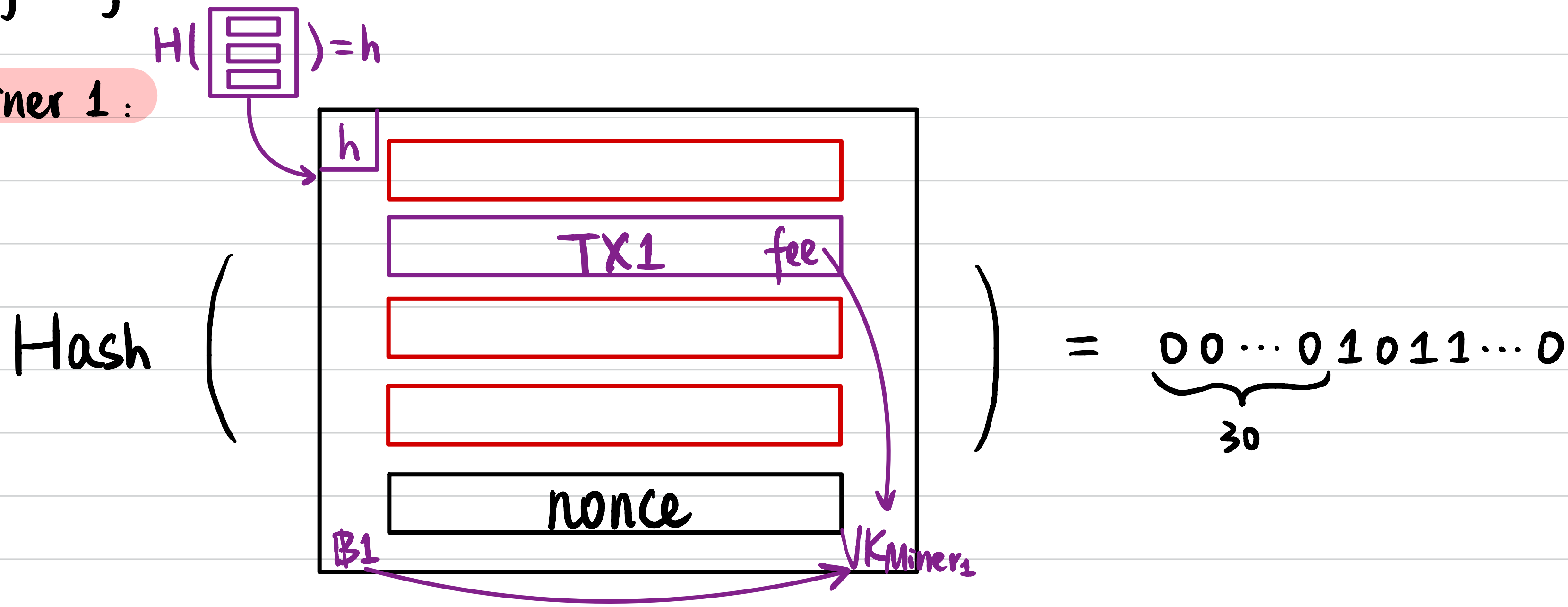
Any problem?



"Sybil Attack"

Proof of Work (PoW)

Miner 1:

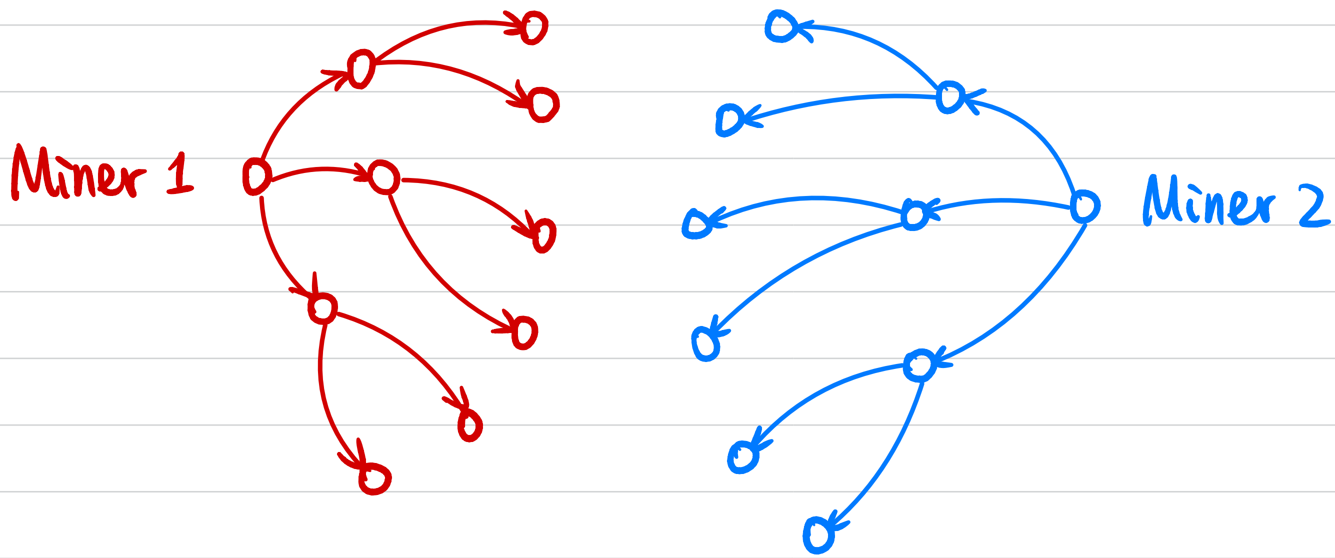


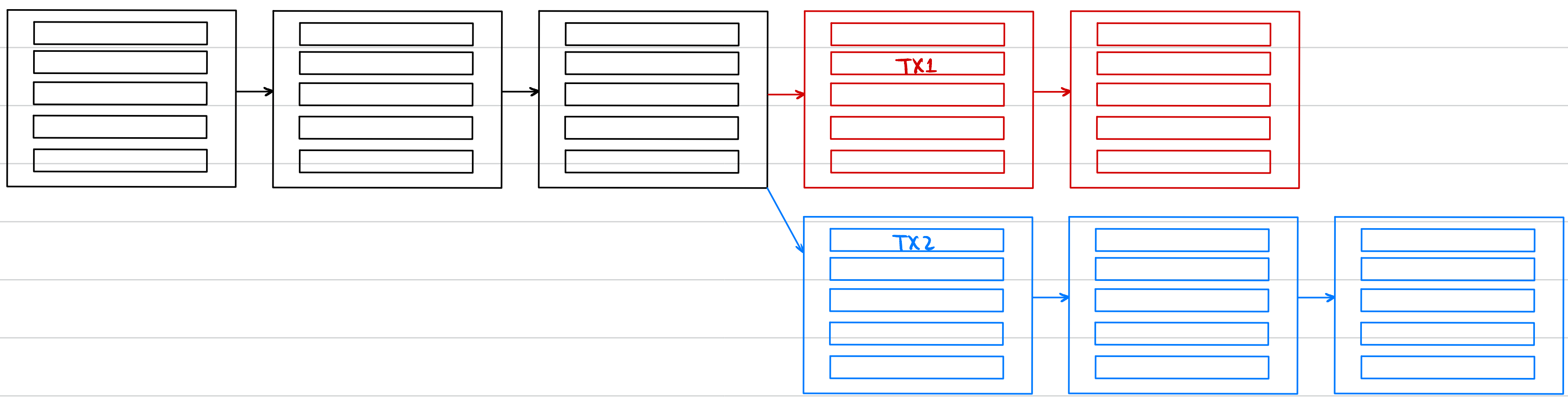
Find nonce s.t. Hash(block) has ≥ 30 leading 0's.

Consensus Protocol:

Whoever first finds a block that hashes to a value w/ ≥ 30 leading 0's, that block becomes the next block.

Proof of Work (PoW)





Longest Chain Rule: Always adopt the longest chain.

Assuming honest majority of computation power, the longest chain is always valid.

Blockchain

- Efficient verification of sufficient balance: Merkle Tree

- Settlement of a transaction:
  Included in a block which is ≥ 6 blocks deep (~ 1 hr)

- Dynamically adjust # leading 0's s.t. each block takes ~10min to mine
  Last 1 hr: > 6 blocks: increase # leading 0's
  < 6 blocks: decrease # leading 0's

- Miners' motivation:
  - transaction fee
  - new coin generated in each block goes to miner

- Extensions
  - Proof of Stake (PoS)
  - Anonymous transactions (zk-SNARGs)
  - Smart Contracts
  - Public Bulletin Board