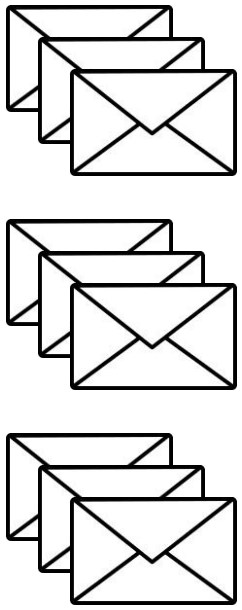
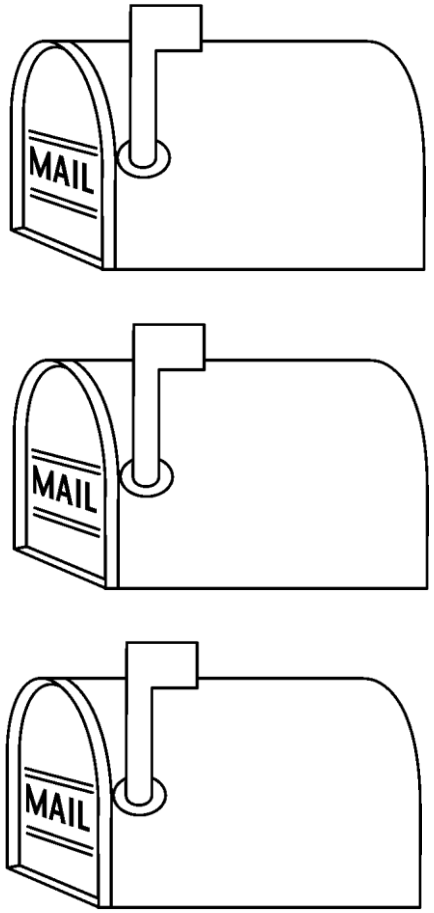
Blockchain Seminar
Proof of work

By Idan Gerichter

Junk email problem

Nigerian Prince

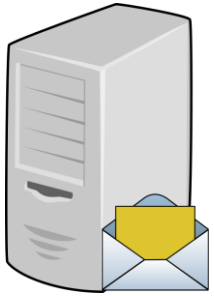




Restrain spammers

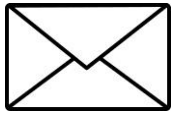
- Spammers can send thousands (or more) emails per second
- We want to limit the amount of emails a spammer can send

Possible solution #1

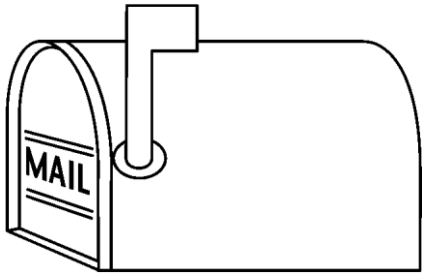


More than *X* secs passed from “Prince” last email

Prince

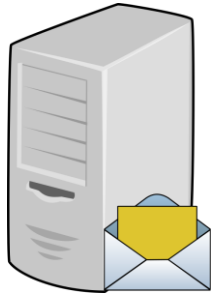


Timestamp

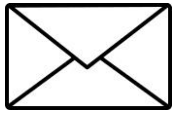


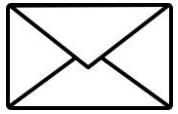
Bob

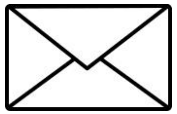
Problems with solution #1

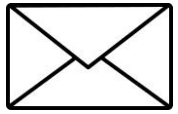


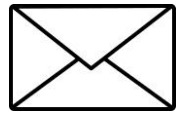
More than *X* secs passed from "Prince 2" last email

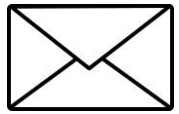

Timestamp


Timestamp


Timestamp


Timestamp


Timestamp


Timestamp


Prince 1


Prince 2


Prince 3


Prince 4


Prince 5

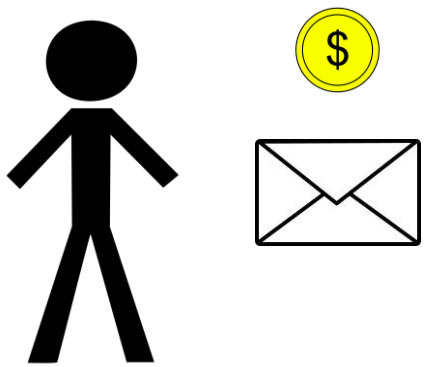

Prince 6

Problems with solution #1

- A spammer can create many accounts
- The same apply for ip addresses
- We require a trusted third party

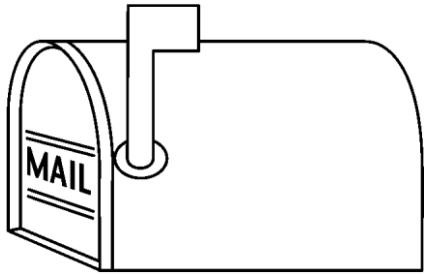
Possible solution #2

- Let's take snail mail as example
- Every email will cost a nominal fee - virtual stamp



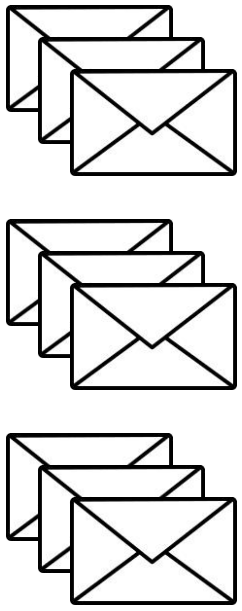
Alice

Negligible for regular user

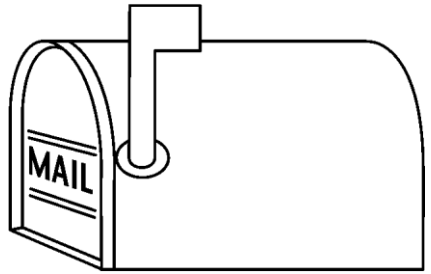


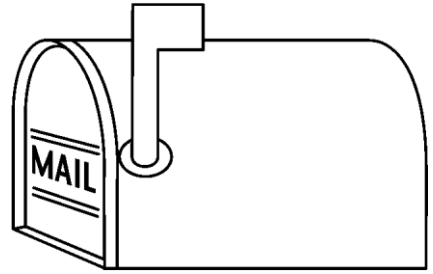
Bob

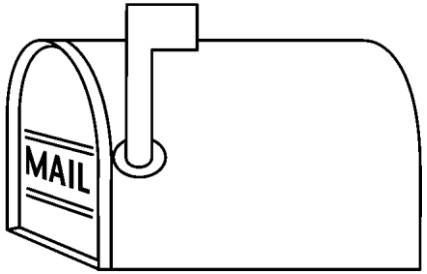
Solution #2 - spammer



A lot of money
for a spammer







Problems with solution #2

- Usage fees can be deterrent for most users
- We don’t want a system where sending notes between friends will cost similarly to a postage stamp.

Computational approach

- Usage fees by computational power
- The sender will be required to invest computing power - preventing him from spamming
- First proposed in *"Pricing via Processing or Combatting Junk mail" [Dwork & Naor 92]*

Dwork

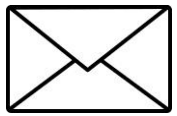
Naor

The missing link - Proof of Work

- We need a way of efficiently verifying a computational effort has been made by the sender.

Challenge-Response PoW scheme

4. Compute



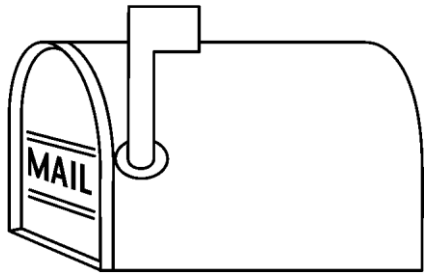
1. Ask for service

3. Send challenge: Factorize 9,487

5. Response: 9487 = 53 * 179

7. Grant access

2. Choose challenge



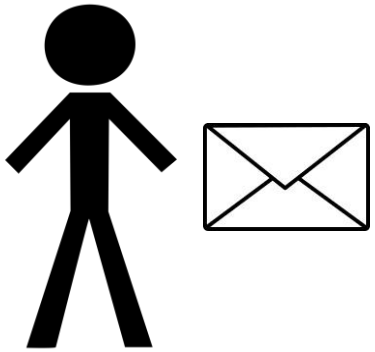
6. Verify

Alice

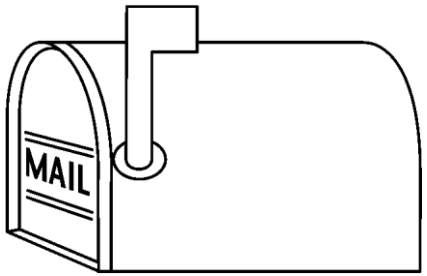
Bob

Theoretical model definitions

- Resource – The object we want to limit access to (e.g. my mail box)
- User – Desires access to the resource



User



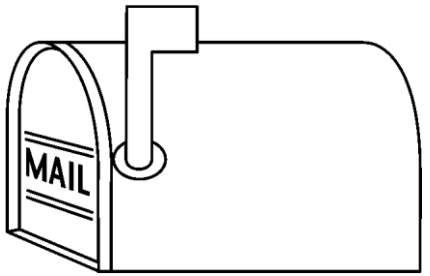
Resource

Theoretical model definitions

- Resource manager – Regulates access to the resource
- Pricing function – Moderately difficult to compute but not infeasible. Easy to verify.

User Pricing function proof

Resource manager



Resource

Pricing function definition

- Let f be a *pricing function* if:
  - f is moderately easy to compute
  - f is not amenable to amortizations:
    - The computational cost of computing $f(m_1), \ldots, f(m_k)$ is comparable to the sum of cost of computing $f(m_i)$ where $1 \leq i \leq k$
  - f is not amenable to preprocessing
  - Given x, y it is easy to determine if $y = f(x)$

Introduction to Cryptographic hash functions

- Preimage resistance
  - Given y, hard to find x such that $H(x) = y$
- 2nd preimage resistance:
  - Given x hard to find $x' \neq x$ such that $H(x') = H(x)$
- Collision resistance:
  - Hard to find $x \neq x'$ such that $H(x) = H(x')$

Random oracle model

"Ideal cryptographic hash function"

Pricing function classes of difficulty

- Focus on the relative difficulty of computational tasks rather than asymptotic growth
- A good pricing function will have a difficulty parameter

Easy problems

Efficient solution exists
Ex: Verify preimage

Moderate Problems

Somewhere in the gap

Hard Problems

Infeasible in reasonable time
Ex: Hash preimage

Pricing function example - Hashcash

- $H: \{0,1\}^* \rightarrow \{0,1\}^n$ is a cryptographic hash function
- If random oracle, for a string s:

  Every bit in $H(s)$ is 0 or 1 with probability $\frac{1}{2}$
- The probability the first k bits are 0 is $\frac{1}{2^k}$

Hashcash continue

- Find s such that $H(s)$ has k leading zeros. s is the proof
- In a brute force attack the complexity of such PoW is $O(2^k)$ in expectation
- We assume H is partial preimage resistance

Hashcash – Preprocessing

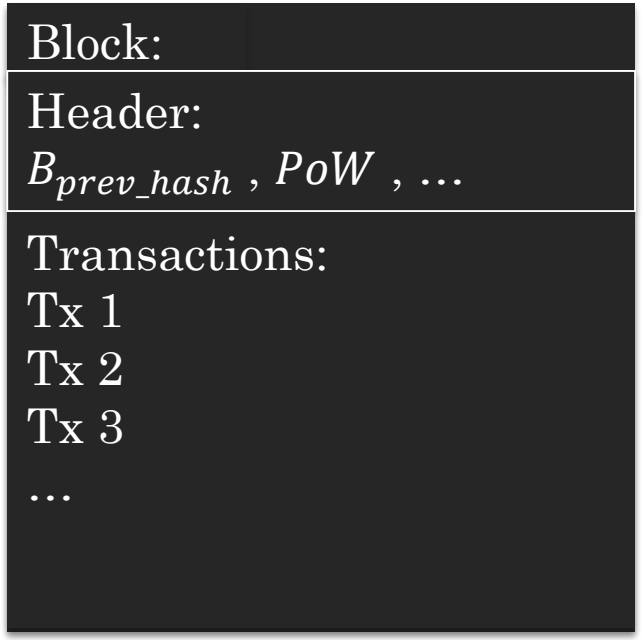
- Pricing function should be “not amenable to preprocessing”
- t is a timestamp
- Task: find s such that $H(t \mid s)$ has k leading zeros
- A better function might be $H(t \mid s) < target$
  Used in bitcoin

Connection to blockchain

- What miners do in blockchain is solving PoW puzzles!
- But why?
- PoW secures the integrity and order of blocks

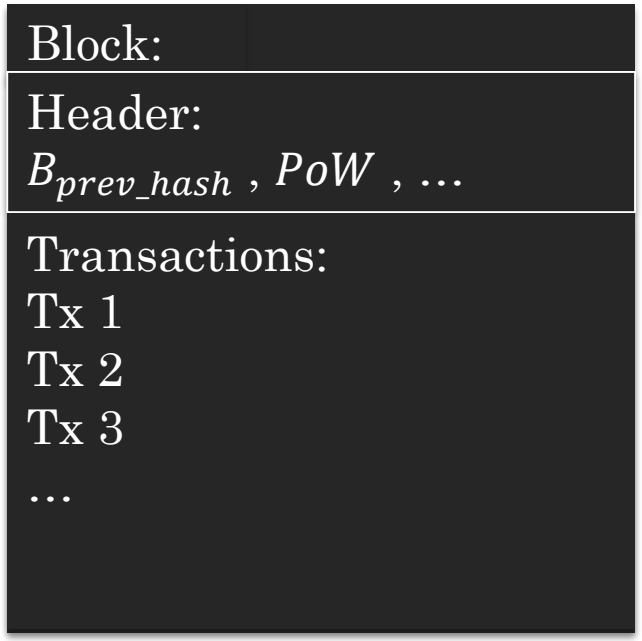
Blockchain pricing function

- The pricing function should be a function of the current block and the previous block.
- Blockchain block internals:

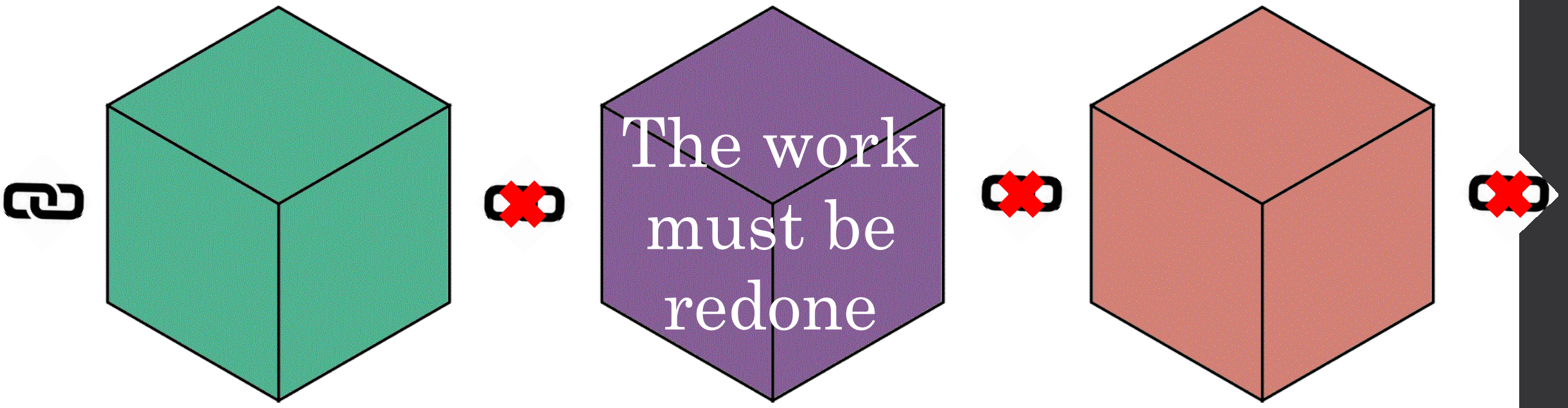


Pricing function task

- Find $PoW \in \{0,1\}^*$ such that $H(block)$ has k leading zeros.
- Note: actual PoW uses only the merkle root.



Proof of Work in blockchain



Prev hash: QJ65
PoW: JI56

Hash: F168

Prev hash: F168
PoW: 6DLK

Hash: F058

Prev hash: I2OP
PoW: KRX3

Hash: I2OP

LCR and Proof of Work

- Tampering with a block c annot be done without **redoing the work**
- Not only of the current block but **all the blocks chained after** it
- Not only the work must be redone for those block but also **faster** than the pace new blocks are generated!

Questions?

Thank you for listening!