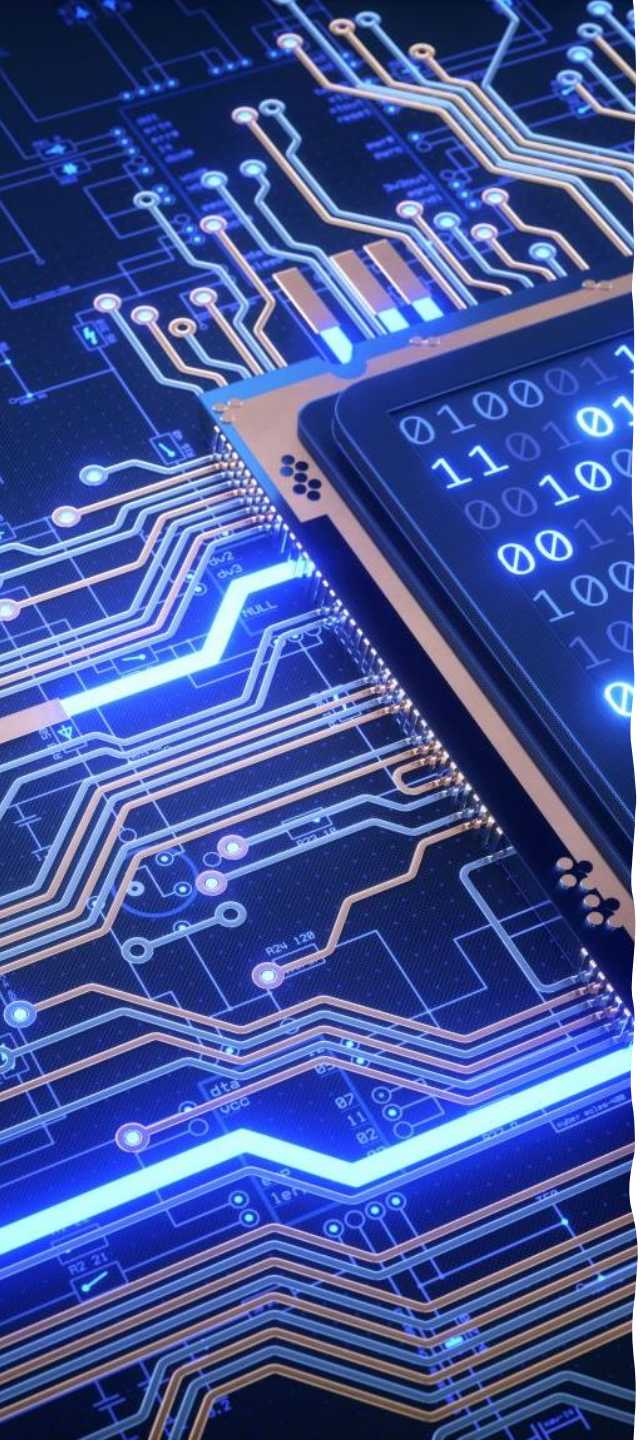


IoT and Blockchain

PoW Background

Phillip G. Bradford

University of Connecticut, Stamford CT. **USA**

phillip.bradford@uconn.edu

Outline

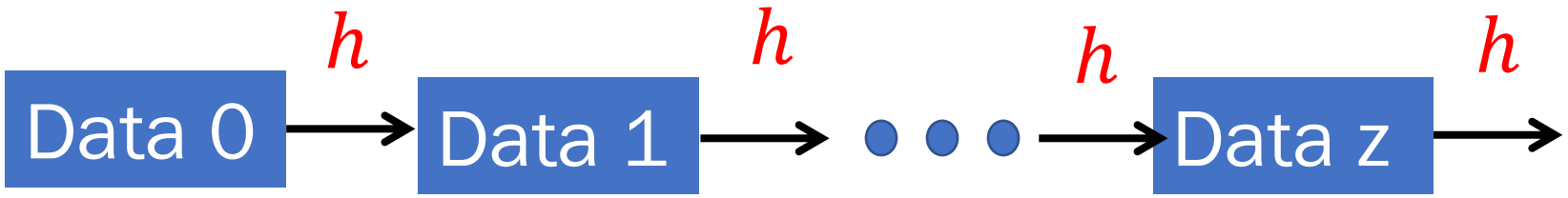
Hash chains

Blocks

Mining

Consensus

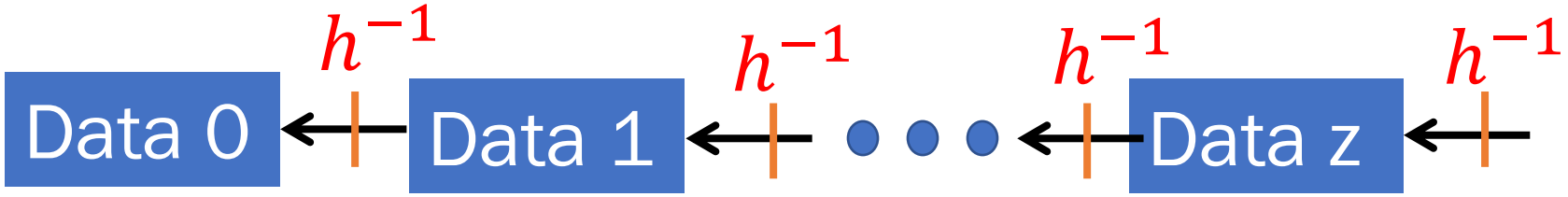
Hash chain



Easy to validate going forward computing h
pseudo unique – collision intractability

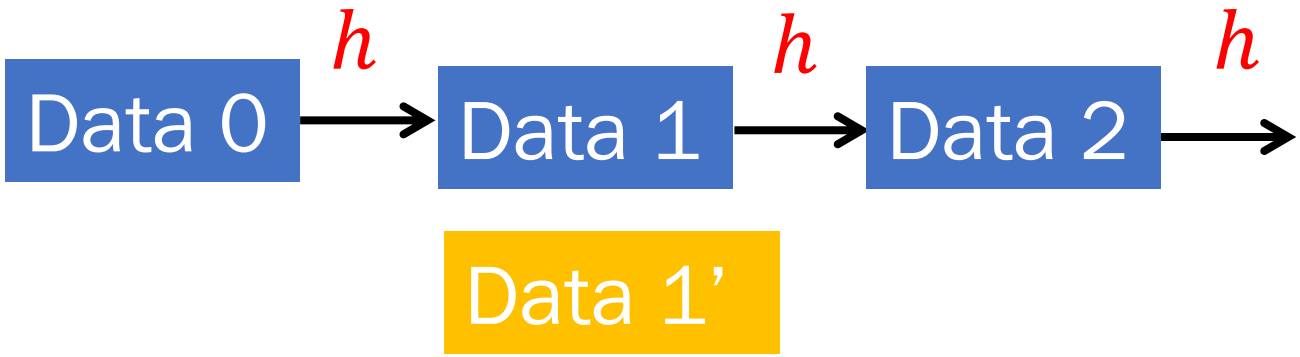
Intractable to go-backwards h^{-1}

Hash chain: intractable backwards



Intractable to any step go-backwards h^{-1}

Hash chain - security



Challenge! Collision intractability

Common assumption Data 0 is valid

Ingredient of blocks: consensus

Valid Chain

Previous hash

Data

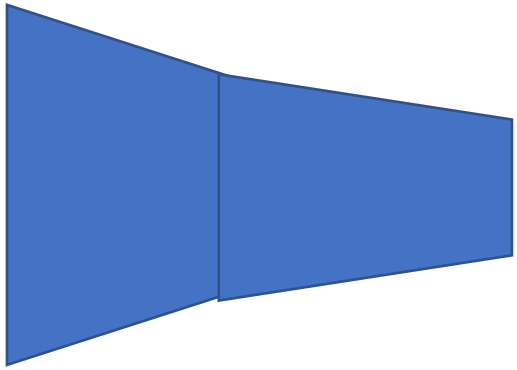
Building a winner by consensus

How?

Our simple Blocks

class Block:

previous hash
data
nonce

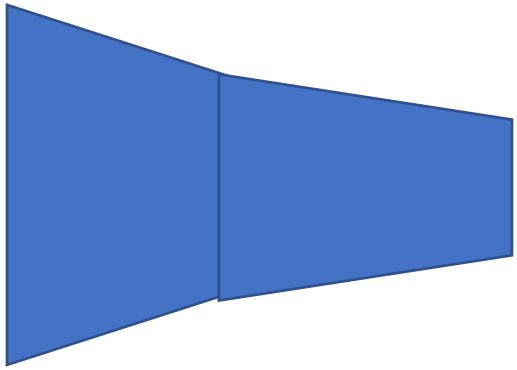


hash

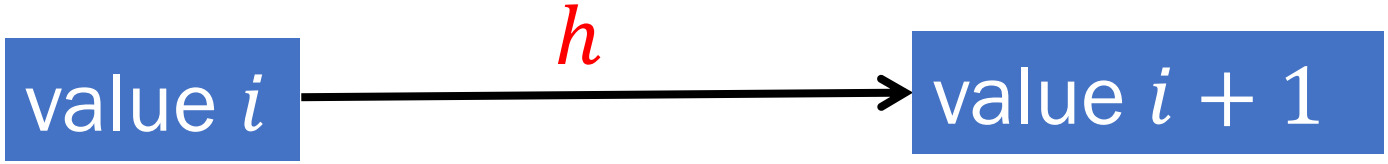
Our simple Blocks

class Block:

previous hash
data i
nonce i

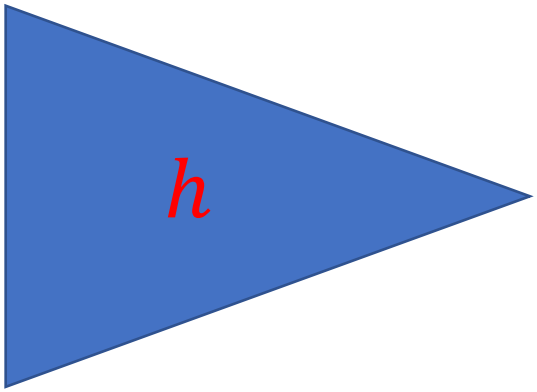


hash

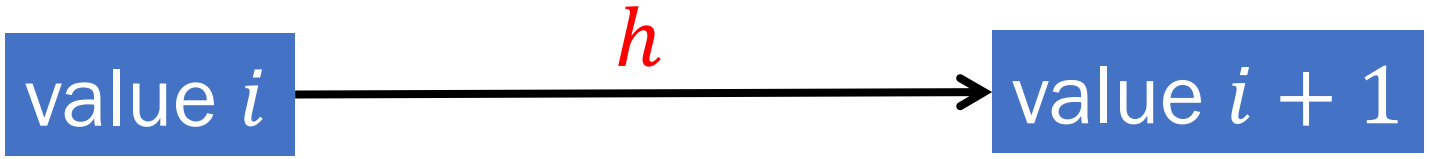


Our simple Blocks

previous hash
data i
nonce



hash
data $(i + 1)$
nonce $(i + 1)$



Basic blocks

RPI_1> git clone \
https://github.com/wonder-phil/2022Blocks

Host> git clone \
https://github.com/wonder-phil/2022Blocks

Python compHash

```
def compHash(self):
    hashFunction = hashlib.new('sha256')
    myStr = str(self.prevHash)+str(self.data)+str(self.time)
        +str(self.nonce)
    myBytes = myStr.encode()
    hashFunction.update(myBytes)
    self.bHash = hashFunction.hexdigest()
    return self.bHash
```

Simple block hashes: base 16

Hash for block 1 :
e28731c23c541c3b682e80a332ee30bda1d5ac8b8acb5f5f206cef25db3529fb

Hash for block 2 :
75154608b0943abefc3494bd9f84a39e9a5df362175a9cbb9903d2e443593b25

Hash for block 3 :
b0a0643f8c7d867e9eae994b72d9acd13f172591ab07fd87b93cd23a33d10191

What is the likelihood of a leading 0 ?

Our simple Blocks

class Block:

difficulty

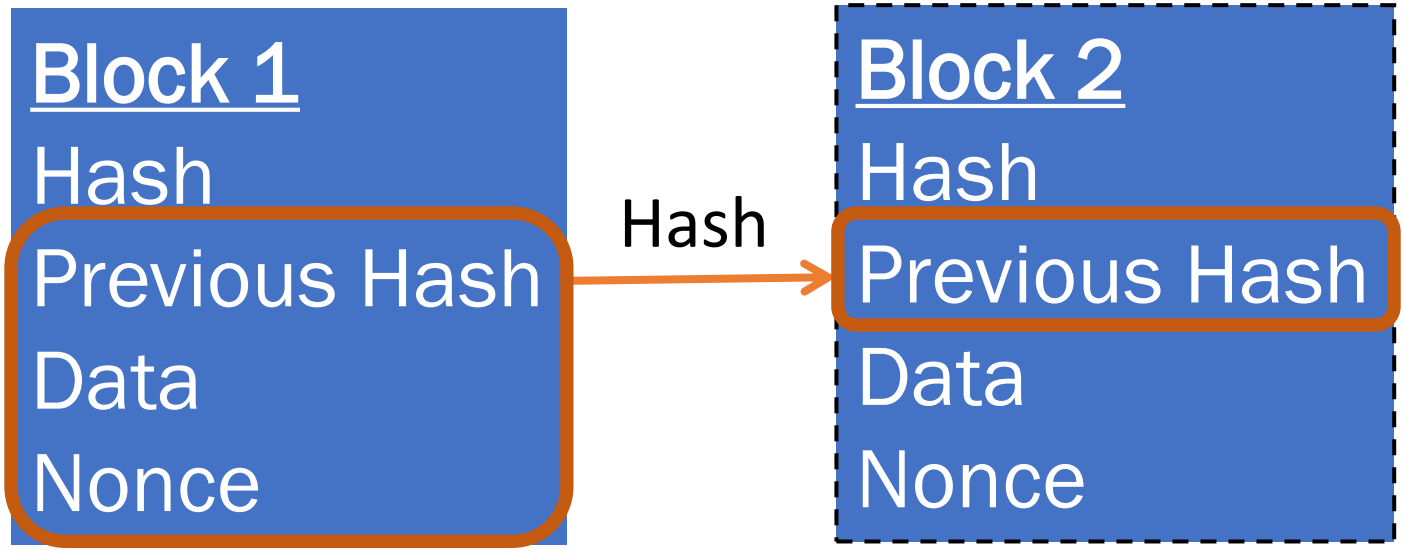
previous hash

data

nonce

hash

Generating Blocks



Simple block hashes: base 16

Hash for block 1 :

028731c23c541c3b682e80a332ee30bda1d5ac8b8acb5f5f206cef25db3529fb

What input can we change to get a leading 0 ?

Nonce

First to get difficulty leading zeros

Who finds the next block?

Key idea: leading zeros

P[x leading zeros] = $\frac{1}{16^x}$

First generating a block with the given inputs

While varying the nonce

Enough leading zeros

Example

```
b=Block("empty","Genesis block")
b.nonce
b.bHash
b.nonce = 436
b.compHash()

for i in range(16):
        b.nonce = b.nonce + 1
        b.compHash()
```

Block mining

Given a fixed difficulty

Picking up data from a common data block

All miners grab this data

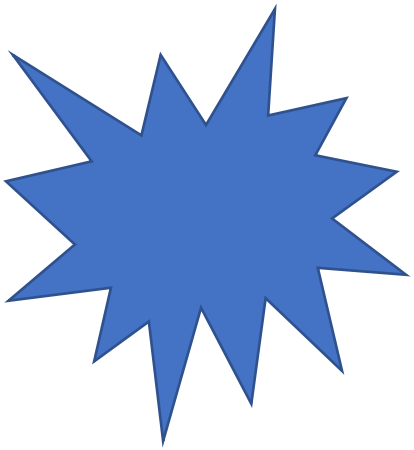
Change nonce & re-hashing to get difficulty leading zeros

Python Miner

```
def mineBlock(self,diff):
    self.target = "0"*diff
    while self.bHash[0:diff] != self.target:
        self.nonce = self.nonce + 1
        self.compHash()
    print("Block mined: ", self.bHash)
    return self
```

Consensus

Who is the first to get difficulty leading 0s?



51% validation of new block

Acceptance of new block