UNSW SYDNEY

UNSW THE UNIVERSITY OF NEW SOUTH WALES

CRYPTO CURRENCIES

Never Stand Still Faculty of Engineering School of Electrical Engineering and Telecommunications

Prof. Aruna Seneviratne, Room EE 312
E-mail – a.seneviratne@unsw.edu.au

Based on : Intro to Crypto and Crypto Currencies by Ed Felton etal.

Hash Functions Again

- Hash Function Properties
  - Takes any string as input
  - Generates a fixed size-output
  - Efficiently computable
  - Has the following security features
    1. *Almost* Collision free
    2. Hiding
    3. Puzzle friendly

1. Collision Free

- Assume, given a space of *N* possible hash values, already picked a single value.
- *N−1* remaining values that are unique from the first
  - The probability of randomly generating two integers that are unique from each other I = *(N-1)/N*
- *N−2* remaining values (out of a possible *N*) that are unique from the first two
  - Probability of randomly generating three integers that are all unique = *(N-1)/N x (N-2)/N*
    - Multiply the probabilities because each random number generation is an independent event

- In general,
  - The probability of randomly generating *k* integers that are all unique is:

$$(N-1)/N\times(N-2)/N\times\cdots\times(N-\{k-\})N\times(N-\{k-1\})/N$$

- Can be approximated by:

$$e^{-k(k-1)/2N}$$

1. Collision Free cont.

无碰撞 就是输入一个x 的到h(x),不会有一个y它的h(y)==h(x),unless x = y

- If you have 2^{130} randomly chosen inputs
  - 99.8% chance that two of the inputs will collide
- This is the case regardless of what ***H( )*** is
- Takes a long time to compute

- **No** *H( )* has been proven to be collision free

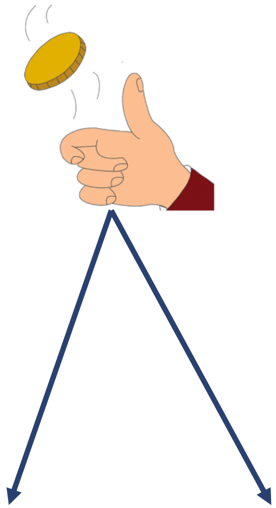
- Despite – safe to assume
  - If *H(x)* = *H(y)* then *x* = *y*
  - Therefore you can use *hash* to recognise large files *(fingerprint)*

2. Hiding

- Given *H(x)*, it is infeasible to find *x*

- If *r* is chosen from a probability distribution that has *high minimum entropy**
  - *H (r |x)* it is infeasible to find x

- *High minimum entropy* means that the distribution is very spread out
- Probability of a particular value being chosen is negligibly small



H(“heads”) H(“tails”)

Example

- Commit to a value, reveal later
  - Putting vale (message) in a box (commitment) and locking it with a key

- Possible commitment API
  - (*com*,*key*) := ***commit*** (*msg*)
    - *com* is the commitment and *key* is the secret key for unlocking the box

  - match (true/false) := ***verify*** (*com*, *key, msg*)
    - Given the commitment, key and message, one can verify that the messages are the same (*true of false*)

  - Locking box
    - (*com*,*key*) := ***commit*** (*msg*) and publish *com* and *msg*
  - Opening box
    - match := ***verify*** (*com*, *key, msg*)

Implementing Commitments

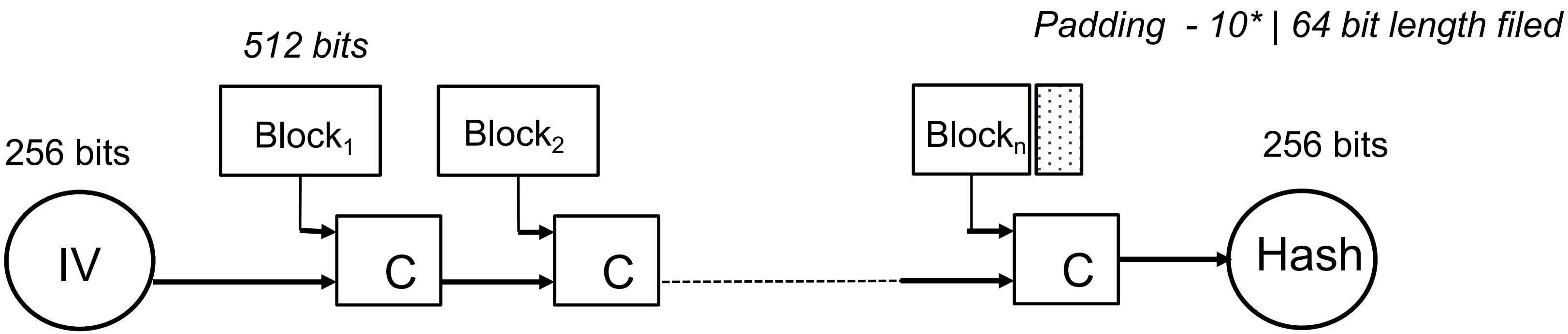
- Security Properties
  - Hiding: Given *com*, infeasible to find *msg*
  - Binding: Infeasible to find msg !=msg' such that
    - ***verify*** (*commit(msg)*, *msg'*) = = true

- Implementation
  - commit(*msg*) := (H(*key* |*msg*), H(*key*)
    - Where key is a random 256 bit value
  - Verify (*com*, *key*, *msg*) := (H(*key* | *msg*) == *com*)

3. Puzzle Friendly

- For every possible output *y* from the hash function,
  - if *k* is chosen from a distribution with *high minimum entropy*, then it is not possible to find *x* such that
    - *H(* k | x*)* = y
- Search Puzzle
  - Mathematical problem that requires the searching of a very large space in order to find the solution
  - Given a "puzzle ID" (min-entropy distribution) and a target set *Y (make the hash function fall into)*
  - *Y is a target range of set of hash results we want puzzle ID to specify the specific puzzle and a solution to the puzzle x*
- Try to find a "solution" *x* such that
  - ***H(***puzzle ID | x***)*** ∈ *Y*
- Only way -> Try random values of *x*

SHA-256

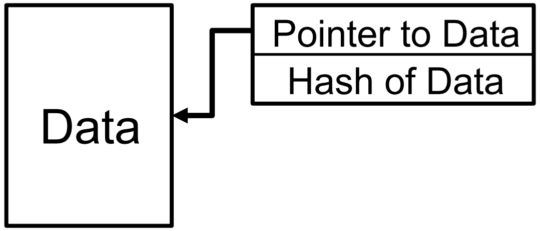


- If *C*, the *compression function* is *collision free*, then the final hash is also collision free

Hash Pointers

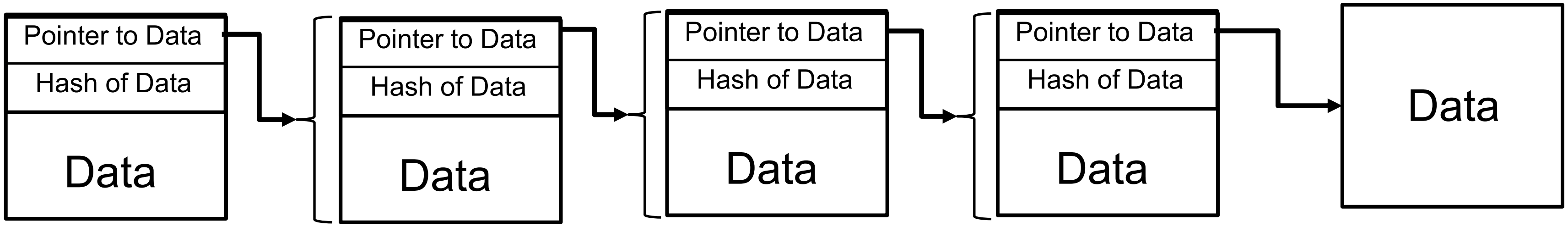
- Hash pointer
  - Pointer to where some information is stored
  - Cryptographic has of the info

- Hash stored in the hash pointer is the
  - Hash of the ***whole data*** of the previous block
  - And the ***hash pointer*** to the block before that one



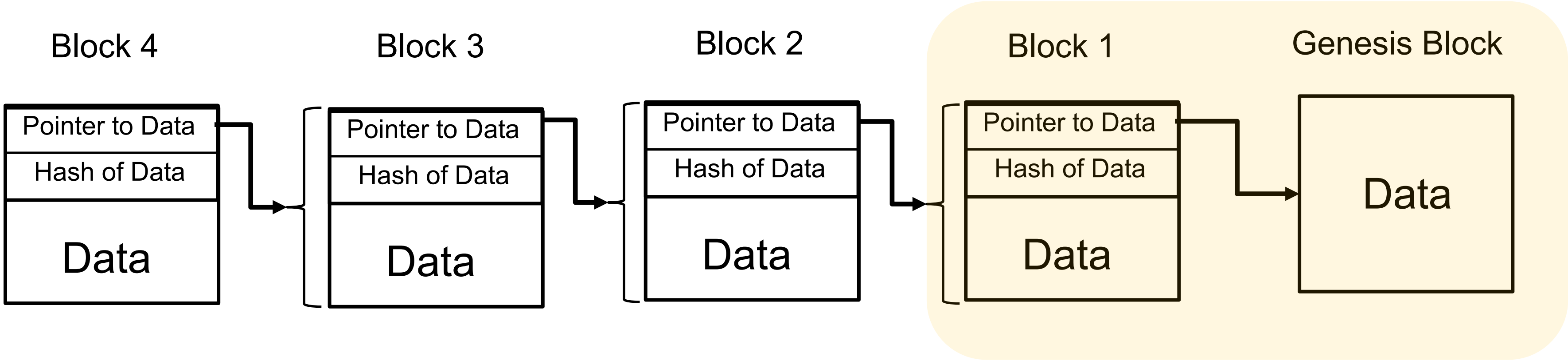
Hash Pointers cont.

- Can be used to create any data structure as long as there are no cycles
- Simple linked list

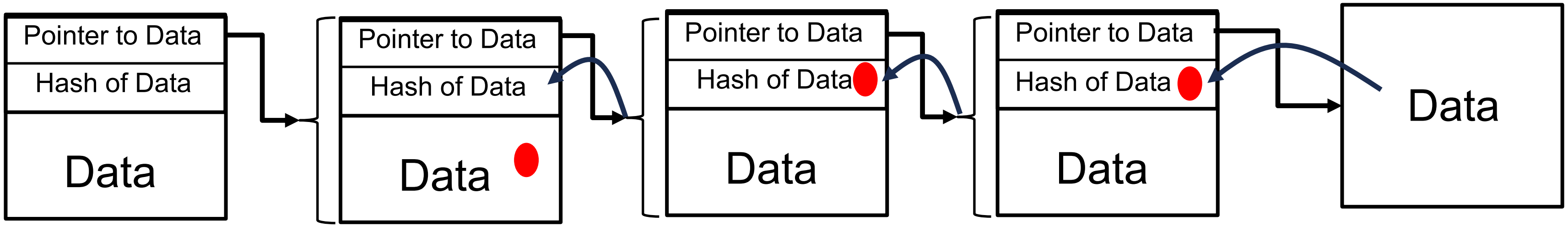


- If we have a hash pointer
  - Ask to get the information back
  - Verify that it has not changed

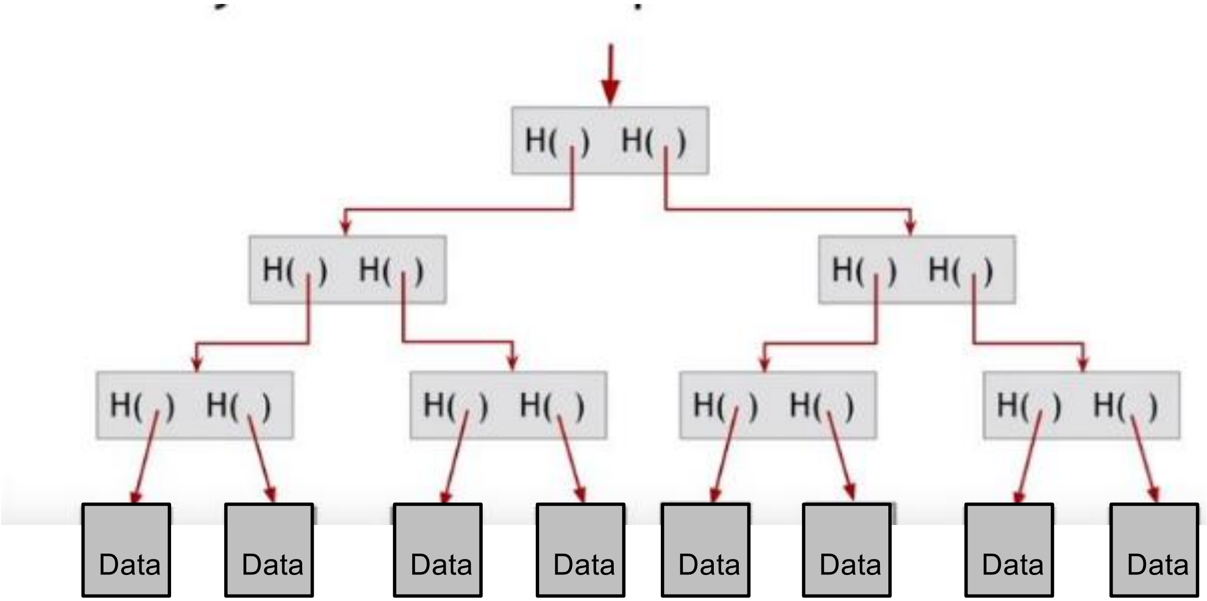
Blockchain



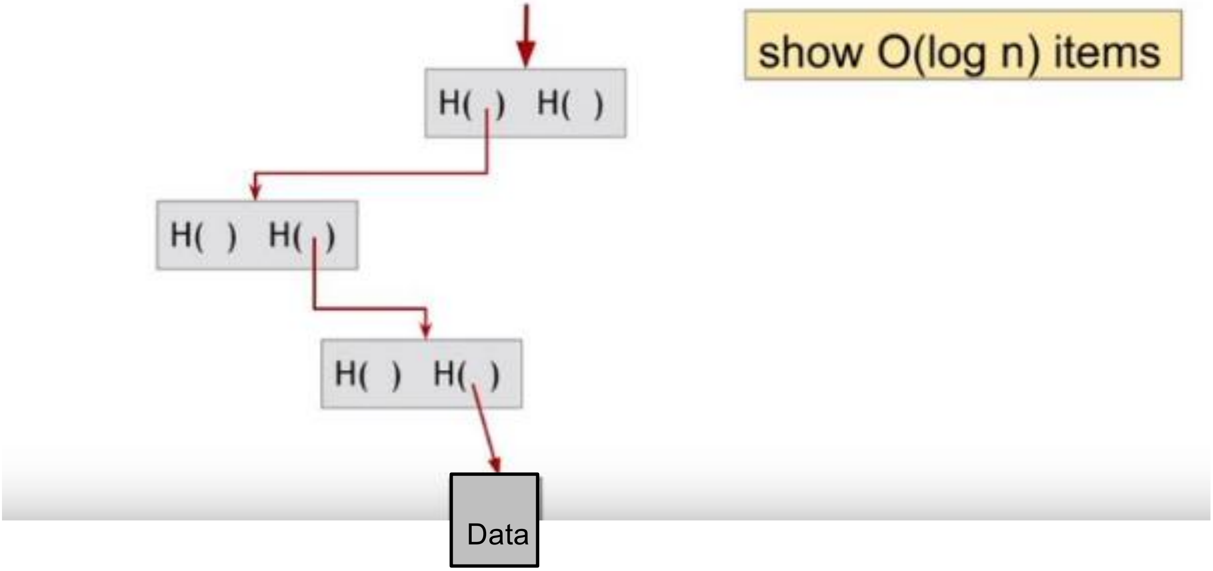
- Tamper-evident block



Binary Tree – Merkle Tree



Membership of a Merkle Tree



show O(log n) items

Advantages

- Can hold many items but need remember the root hash
- Can verify membership in O(log n) time/space
- Sorted Merkle trees
- Can verify non membership in O(log n)
  - Show items before, after missing one

- More generally
  - Hash pointers can be used in any pointer-based data structure that has no cycles

Digital Signatures again.

- Only you can sign, but anyone can verify
  - Uses public private key pair
- Signature is tied to particular document
  - Cannot cut and paste
  - Public Key == an identity

- Decentralised ID
  - Anybody can make a new identity at any time
  - No central point of coordination
  - *Address* in crypto currencies

A Simple Cryptocurrency

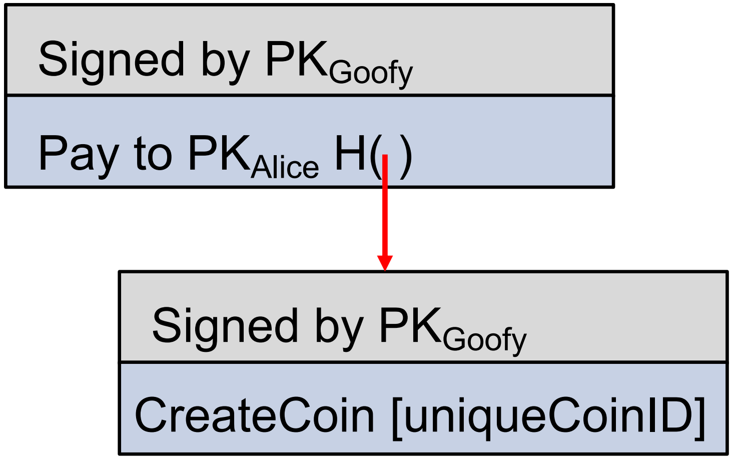
Goofy coins

Goofy can create new coins

Signed by PK_{Goofy}

CreateCoin [uniqueCoinID]

Whoever owns a coin can spend it



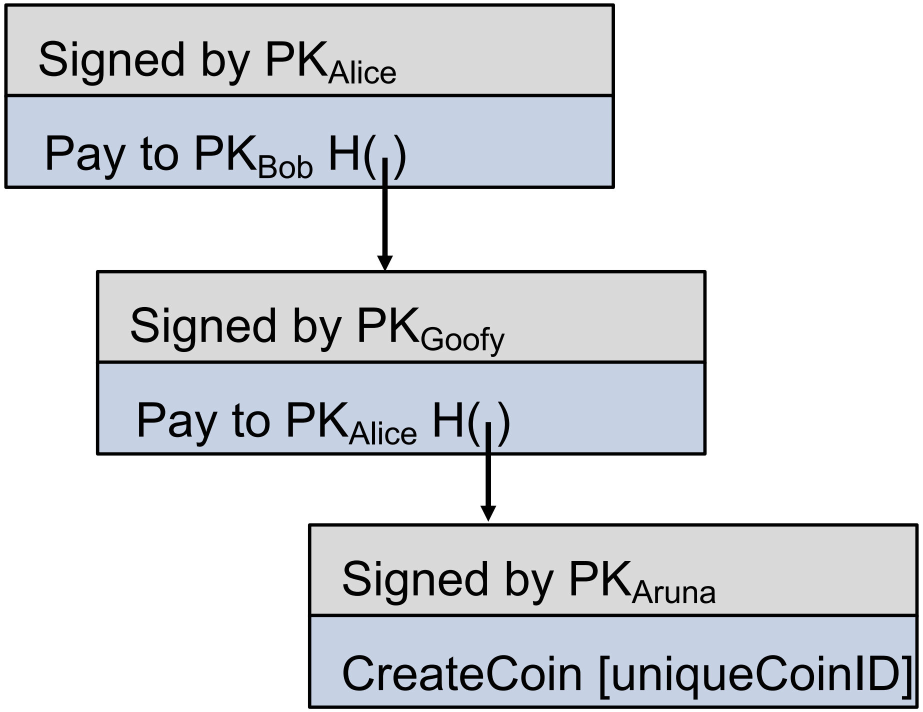
Signed by PK_{Goofy}

Pay to PK_{Alice} H( )

Signed by PK_{Goofy}

CreateCoin [uniqueCoinID]

A Simple Cryptocurrency cont.



Signed by PK_{Alice}

Pay to PK_{Bob} H( )

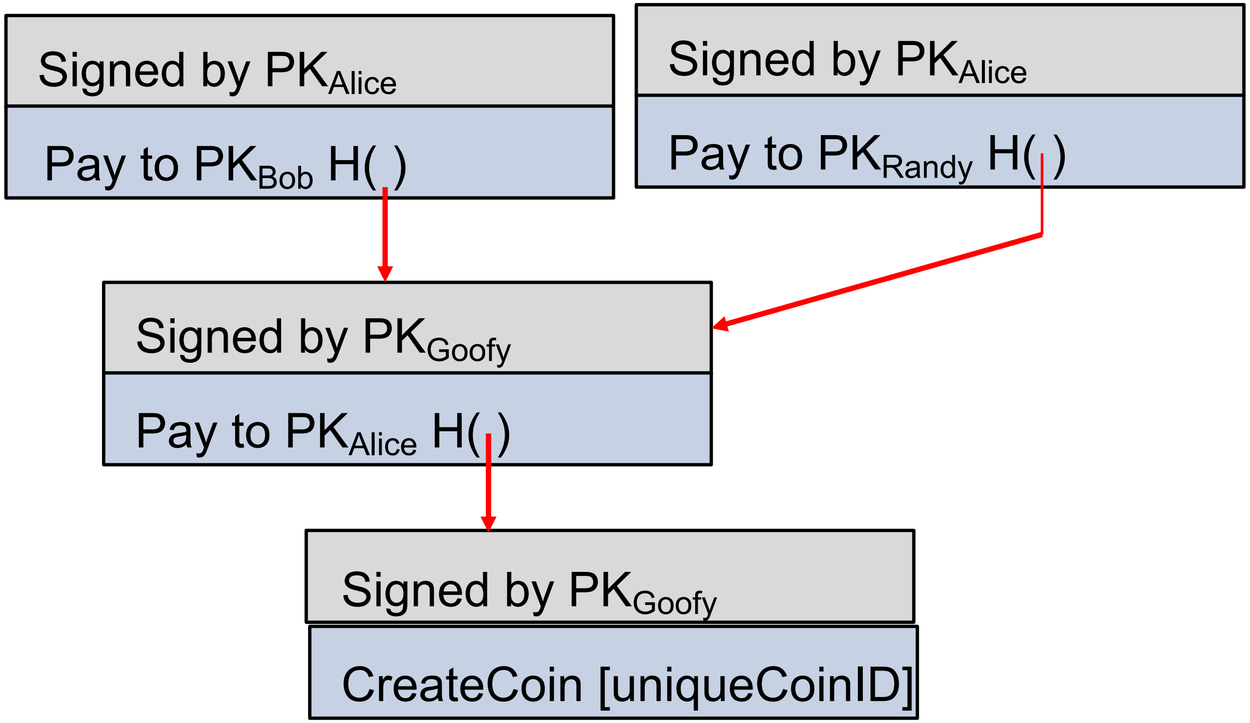
Signed by PK_{Goofy}

Pay to PK_{Alice} H( )

Signed by PK_{Aruna}

CreateCoin [uniqueCoinID]

Simple Cryptocurrency – Double Spending



Signed by PK_{Alice}

Pay to PK_{Bob} H( )

Signed by PK_{Alice}

Pay to PK_{Randy} H( )

Signed by PK_{Goofy}

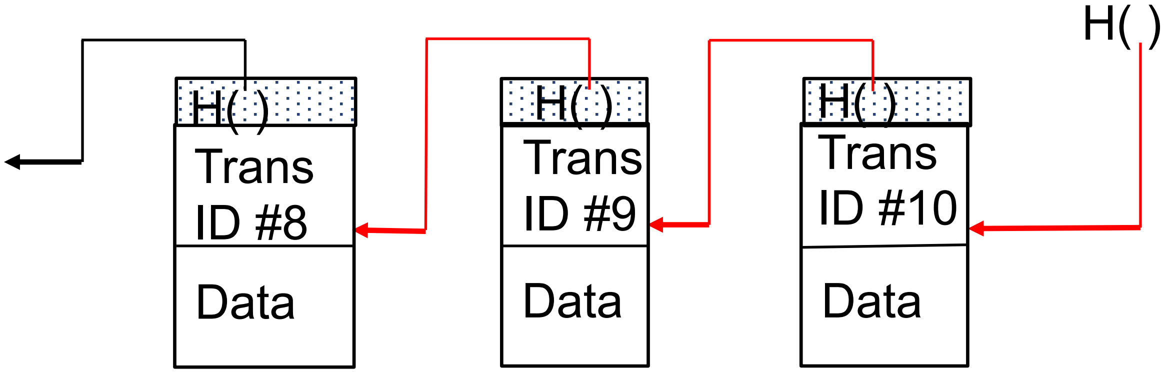
Pay to PK_{Alice} H( )

Signed by PK_{Goofy}

CreateCoin [uniqueCoinID]

Overcoming Double Spend

- Creator published a history of all transactions
  - Blockchain, signed by the creator (goofy)



- Can optimise by putting multiple transaction into the same block
- What does the history do?
  - Detect double spending

Overcoming Double Spend cont.

- *CreateCoints* transaction creates new coins

transID #10		type:CreateCoins	
Coins created			
num	value	recipient	
0	3.3	0x….	Coin10(0)
0	1.4	0x….	Coin10(1)
0	7.1	0x….	Coin10(2)

Paycoins

- *PayCoins* transaction consumes (and destroys) some coins and creates new coins of the same total vale

transID #10 type:PayCoins

Consumed coinIDs:
68(1), 42(0), 72(3)

Coins created

num	value	recipient
0	3.3	0x….
0	1.4	0x….
0	7.1	0x….

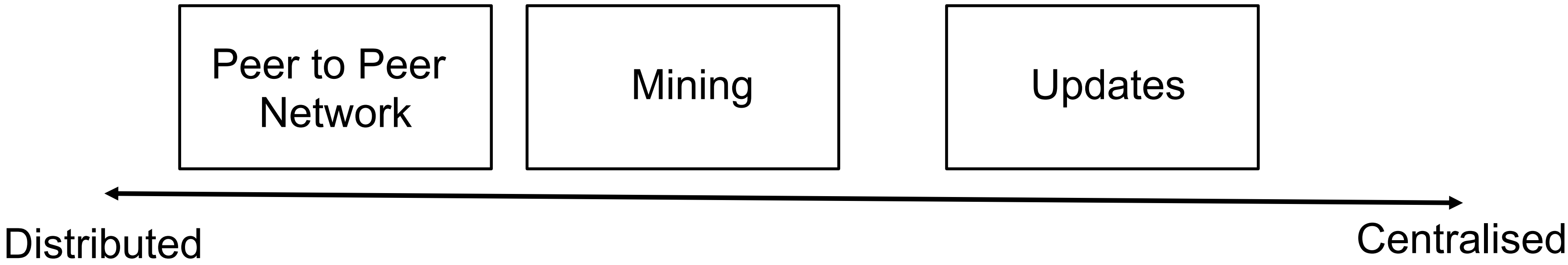
signatures

- Valid if:
  - *Consumed coins are valid*
  - *Not already consumed*
  - *Total value = Total value in, and*
  - *Signed by owners of all consumed coins*

What is the problem?

- Centralisation
  - What if Goofy is dishonest?
- Solution - decentralisation

Distributed System



Peer to Peer Network

Mining

Updates

Distributed

Centralised

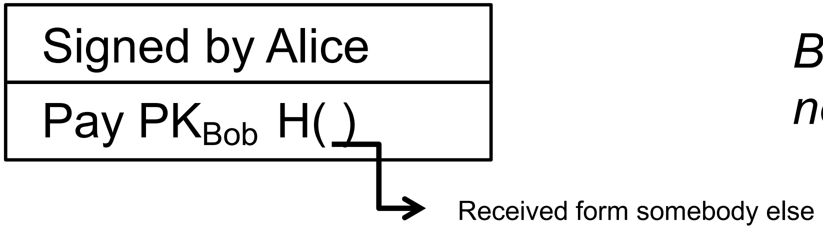
- Distributed Consensus
    - Fundamental problem is maintaining consistency
- Numerous applications
    - DNS, Public Key Directory, ……

❑*Adapted from Arvind Narayanan*

Consensus

- Definition
  - Protocol terminates all legitimate nodes decide on the same value
  - The value has to have been proposed by a some legitimate node

- P2P Bitcoin
  - Alice broadcasts the transaction to all nodes on P2P network



Signed by Alice

Pay PK_{Bob} H( )

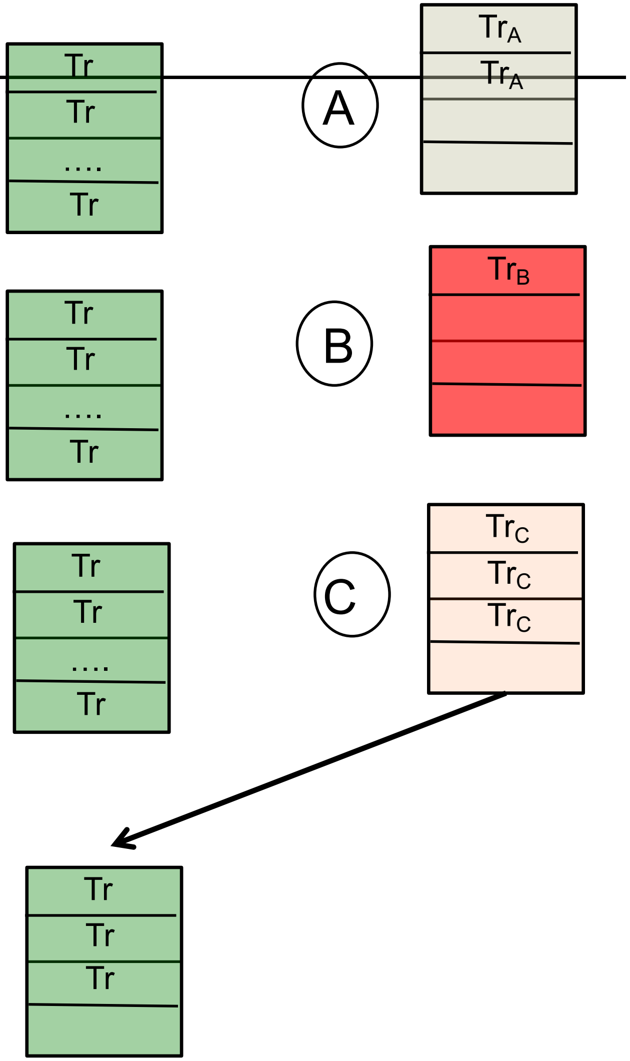
Received form somebody else

Bob's machine does not need to be on the network

- The which transactions were broadcast and order in these transactions took place

Consensus – Bitcoin

- All nodes have
  - A sequence of blocks of transactions that they have received consensus on
  - A set of outstanding transactions they head about
- Select any valid block
- Really hard technical problem



Adapted from Arvind Narayanan

Why

- Nodes may crash or be malicious
- Network is imperfect
  - Not all pairs of nodes are connected
  - Faults in the network
  - Latency
- Many impossibility results
  - Byzantine generals problem
  - Fischer-Lynch-Paterson – consensus impossible with a single faulty node
- Some well known protocols
  - Paxos: Never produces inconsistent results, but can get stuck (rarely)

Adapted from Arvind Narayanan

Some Observations

- Models say more about the model than the problem
- Models were developed to study systems like distributed data bases
- Bitcoin is a practical solution
- Bitcoin
  - Introduces the notion of incentives
  - Embraces randomness
    - No specific end-point
    - Consensus happens over long time scales (~1 hour): as time goes on the probability increases

Adapted from Arvind Narayanan

Consensus without Identities

- Identity is hard in P2P systems
  - No central entities
- Pseudonmity is a goal
- Implicit Consensus
  - Assume that it is possible to pick a random node
  - In each round pick a node at random
  - The selected ndoe proposes the next block in the chain
  - Others nodes accepts the and extends the blockchain, if all transactions are valid(unspent, valid signature) or
  - Reject this block extends the blockchain from an earlier block

Adapted from Arvind Narayanan

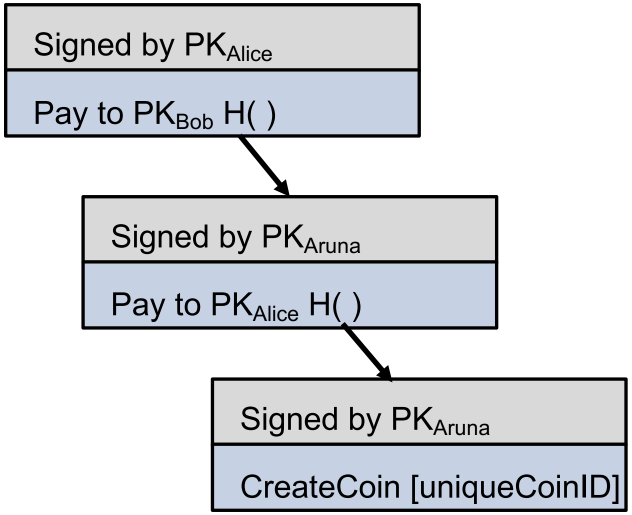
Consensus Algorithm

1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a new block
3. In each round a random node gets to broadcast its block
4. Other nodes accepts the block
5. Nodes express their acceptance of the block by including its hash in the next block they create
6. Extend the longest valid branch

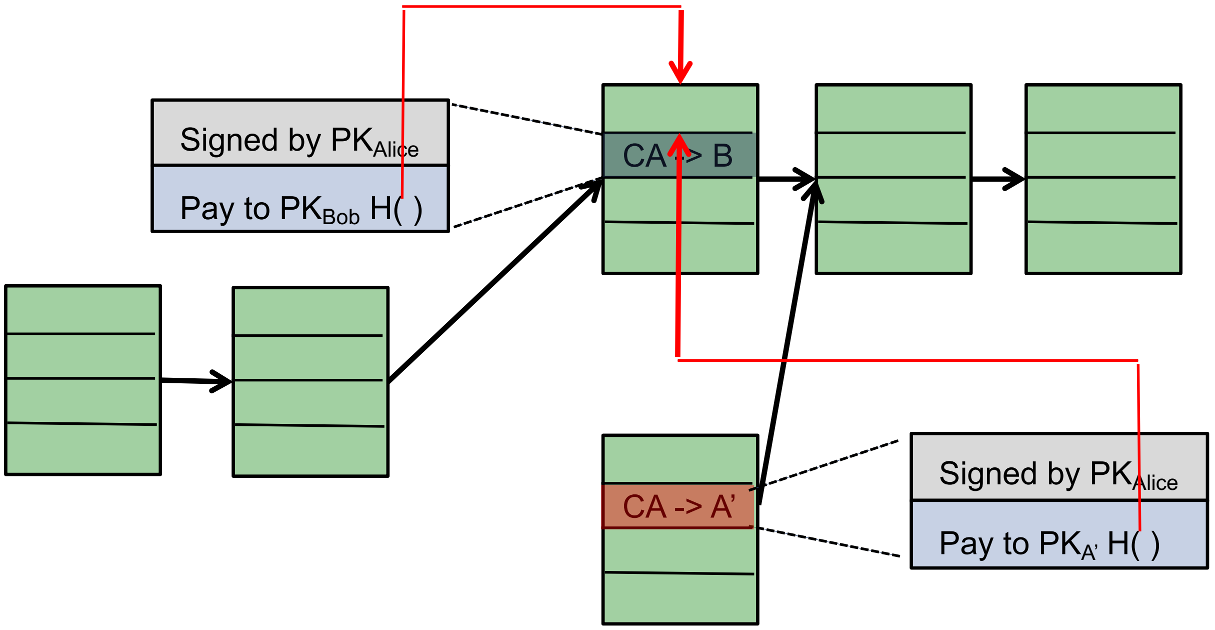
Adapted from Arvind Narayanan

Validation

- Stealing somebody else's bit coins?
  - Cannot because cannot forge signature
- Denial of service by not including any of the transactions from a give user
  - Only an a delay
- Double spending

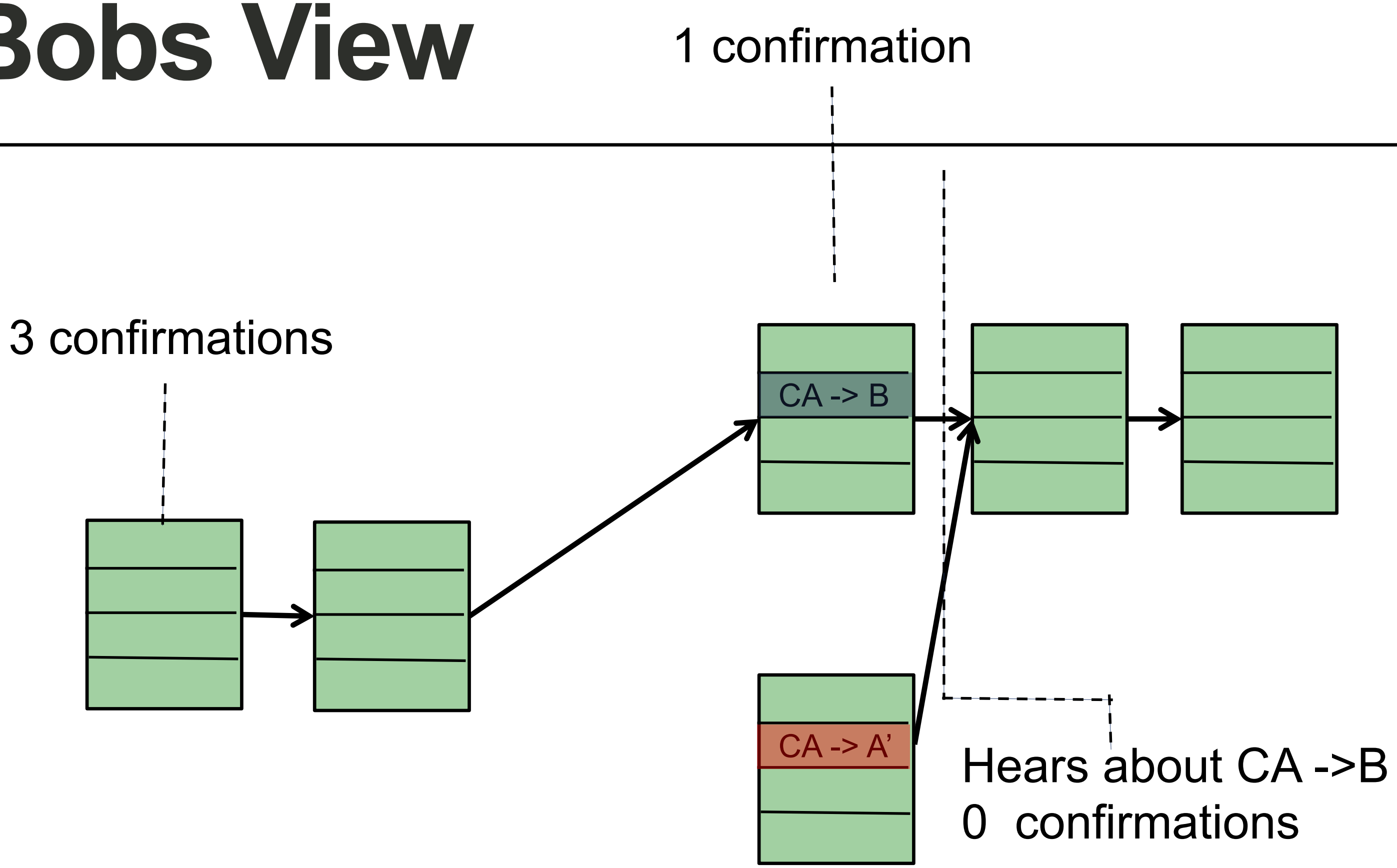


Double Spend (1)



Adapted from Arvind Narayanan

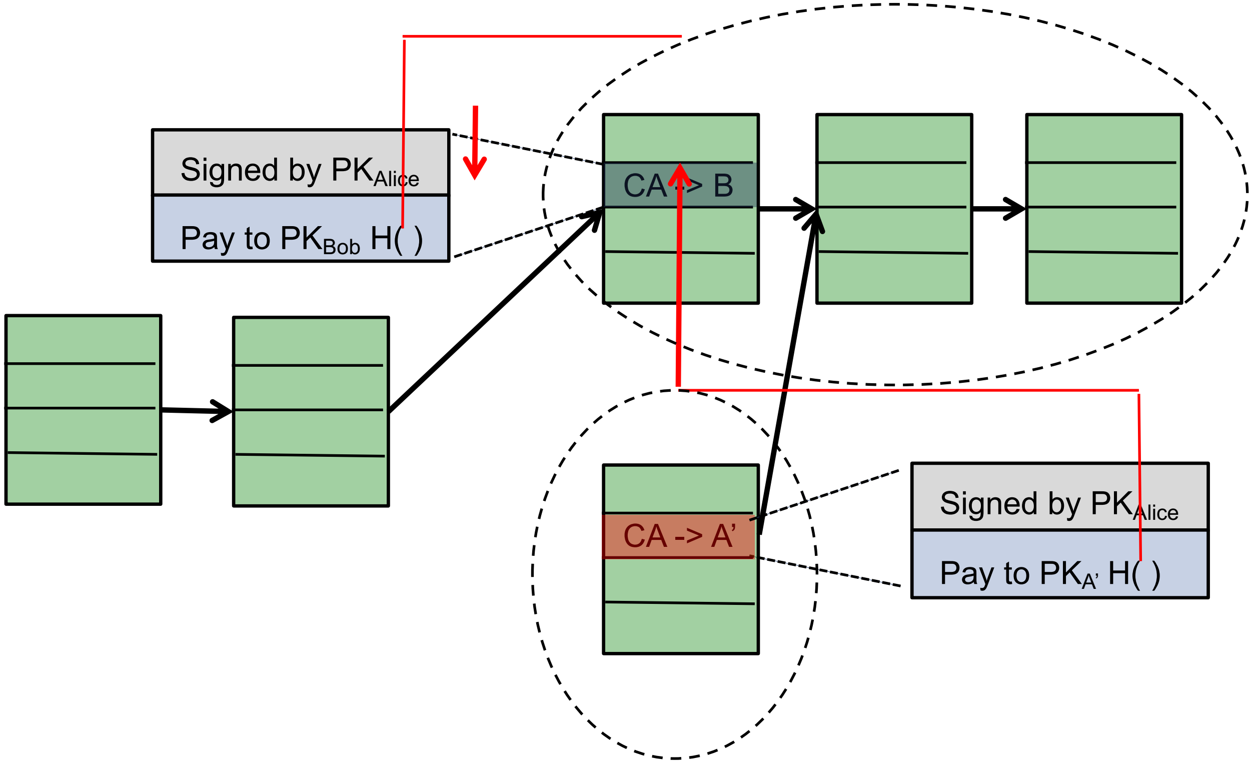
Bobs View



- Double spend probability decreases exponentially with the number of confirmations
- Common heuristic: 6

Incentives not to act maliciously

Reward the node that created these blocks



Punish the node that created this block

Adapted from Arvind Narayanan

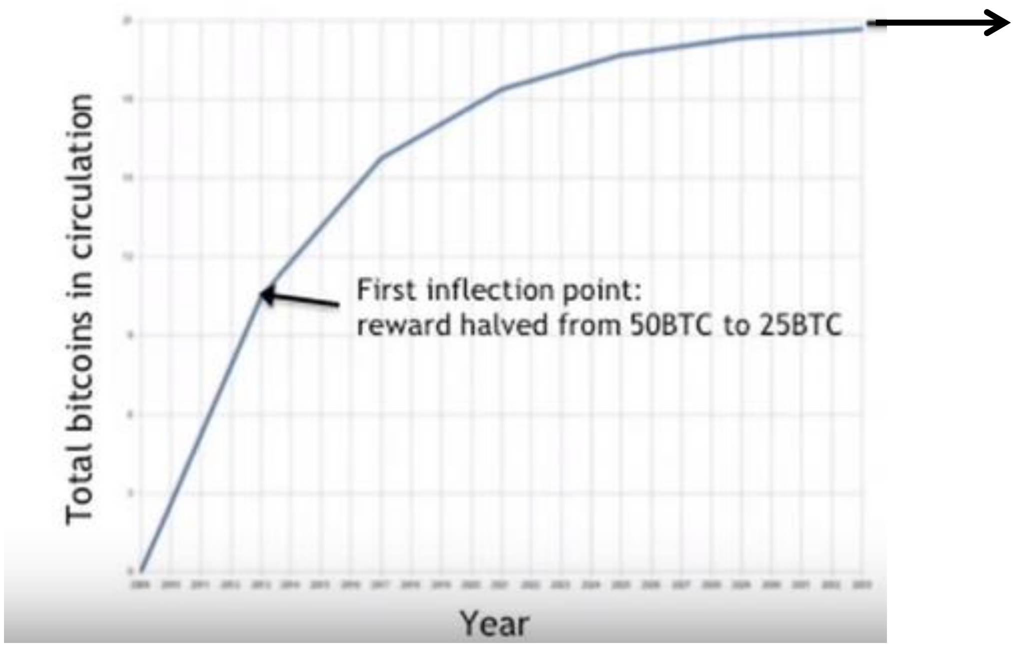
Incentive 1: Block Reward

- Creator of a block gets to:
  - Include special con-creation transaction in the block
  - Choose the recipient address of this transaction
- Block creator gets to to “collect” the reward only if the block ends up on the long-term concensus branch

- Value is fixed: currently 25BTC, halves every 4 years

Adapted from Arvind Narayanan

Finite Supply



First inflection point:
reward halved from 50BTC to 25BTC

Total supply:21 million

- Block reward is how new bit coins are created
- Runs out in 2040

Adapted from Arvind Narayanan

Transaction Fee

- Creator of a transaction can choose to make output value less than the input value
- Like a tip – voluntary

- Problems
  - How to pick a random node
  - How to avoid a free for all
  - How to prevent Sybil attacks

Proof of work

- Selecting a random node
  - Select nodes in proportion to a resource that no one can monopolise
    - In proportion to the computing power: proof of work
    - In proportion to owership: proof of stake
- Let nodes compete for the right to create a block
- Make it difficult to create new identities

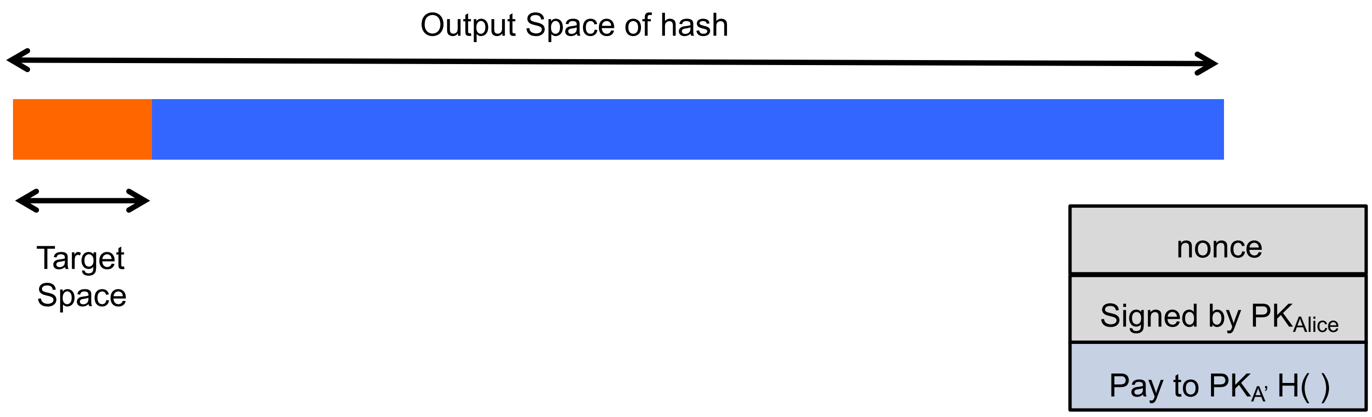
Adapted from Arvind Narayanan

Puzzle Friendly

- For every possible output *y*, if *k* is chosen forma distribution with high minimum entropy, then it is not possible to find *x* such that
- ***H(***k|x***)*** = y
- Search Puzzle
  - Given a "puzzle ID" and target set *Y*, find a solution *x* such that
  - ***H(***puzzle ID|x***)*** ∈ Y
- Only way -> Try random values of *x*

Has Puzzles

- To create a block, find a nonce such that
- ***H***(*nonce* |*prev_has*|*tx*|…..|*tx*) is very small



If the hash function is secure, the only way to find such a nonce keep trying until you get lucky

Some finer Details

- Nodes automatically re-calculate the target every two weeks
    - Goal: average time between blocks ~10 mins.

      Attacks are infeasible if majority of miners weighted by has power follow the protocol

Adapted from Arvind Narayanan

Summary

- Identities
  - No real-world ID any user can create an ID
- Transactions
  - Messages that are broadcast to the P2P network giving instructions as to what to do with coins
  - Coins are chain of transactions
- Peer to Peer Network
  - Transfers the transactions to all the nodes in the network – best effort. Security comes form the blockchain and consensus protocol
- Blockchain and Consensus
  - Transaction to be in a blockchain needs a number of confirmations (6 is the heuristic). Could have a orphan of blocks.
- Hash puzzles and mining
  - Randomly finding nodes

Adapted from Arvind Narayanan