Blockchain, Bitcoin

Technology Basics

SBM, NMIMS

Aug-25

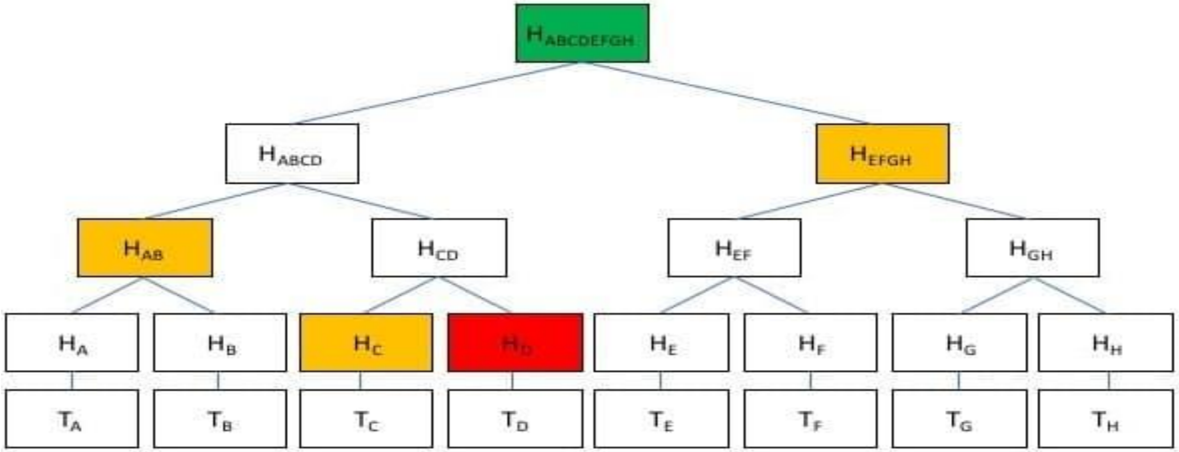
Cryptography Basics

- Hash function:
  - Hash function H
  - Takes a message x
    - of arbitrary but finite size
  - Outputs a fixed size hash h (also called digest)
- Cryptographic hash function
  - Hash function with four additional properties

- Easy to compute:
  - computationally easy to calculate the hash of any given finite message
  - $h = H(x)$; where h is of fixed length
- Pre-image resistance:
  - Infeasible to generate a message that has a given hash value
  - Given a hash h it is infeasible to find any message x such that $h = H(x)$
  - Also called one-way functions
- Second pre-image resistance:
  - Given a message, it is infeasible to find another message which produces identical output, i.e., a collision, when given as input to the hash function
  - Given a message m it is infeasible to find another message m' such that $m \neq m'$ and $H(m) = H(m')$
- Collision resistance:
  - Infeasible to find any two different messages which produce identical outputs, i.e., a collision, when given as input to the hash function
  - It is infeasible to find any two different messages $m \neq m'$ for which $H(m) = H(m')$

Cryptography Basics

- Merkle Tree: hash tree, or authentication tree
  - Binary trees in which the leaf nodes are labelled with the values that need to be authenticated
    - Each non-leaf node is labelled with the hash of the labels or values of its child nodes
  - Advantages:
    - Tamper Proof
    - Uses fewer resources
    - Easy to verify if a specific transaction has been added to the block



Cryptography Basics

- Asymmetric cryptography: Public-key encryption
  - Defined as a triple of efficient algorithms $\mathcal{E}$ (G, E, D), where
    - G: key generation algorithm; takes no input
      - Outputs a key pair (*pk*, *sk*)
        - *pk* : public key: can be shared publicly
        - *sk* : secret key: should be kept private
      - $(pk, sk) \leftarrow G()$
    - E: encryption algorithm: two inputs
      - Public-key *pk*
      - Message $m \in \mathcal{M}$
      - Single output:
        - Cipher text $c \in \mathcal{C}$ encrypted under the public-key *pk*
          - Associated with the public/secret key pair (*pk*, *sk*)
            - **Of the intended recipient**
      - $c \leftarrow E(pk, m)$

Cryptography Basics

- Asymmetric cryptography
  - Public-key encryption
    - D: decryption algorithm: two inputs
      - Secret-key *sk*
      - Cipher text $c \in \mathcal{C}$
      - Outputs:
        - Message $m \in \mathcal{M}$ (encrypted under the public-key *pk* associated with *sk*)
        - or <<*error*>> if keys are incorrect
      - $m \leftarrow D(sk, c)$

    ⇒ if the respective operations are reversible
      - $\forall (pk, sk)$ of G
        $\forall\ m \in M$
        - $D(sk, E(pk, m)) = m$

Cryptography Basics

- Digital Signatures
  - Defined as a triple of efficient algorithms $\mathscr{S}$ *(G, S, V)* where
    - *G*: key generation algorithm; takes no input
      - Outputs a key pair (*pk*, *sk*)
        - *pk*: public key - can be shared publicly
        - *sk*: secret key - should be kept private
      - $(pk, sk) \leftarrow G()$
    - *S*: signing algorithm - takes two inputs
      - Secret key *sk*
      - Message $m \in \mathcal{M}$
      - Single output: signature $\sigma \in \Sigma$
        - Can be communicated publicly together with the message
      - S is invoked as:

        $S : \sigma \leftarrow E(sk, m)$

Cryptography Basics

- Digital Signatures
  - V: algorithm: three inputs
    - Public-key pk
    - Message $m \in \mathcal{M}$
    - Signature $\sigma \in \Sigma$
    - Single output: either `accept` or `reject`
      - Depending on the validity of the signature σ on message m
    - $\{accept, reject\} \leftarrow V(pk, m, \sigma)$

$\Rightarrow$ a signature generated by S is accepted by V iff (pk, sk) is a valid public/secret key pair

- So $\forall$ (pk, sk) of G it holds that:
  - $\forall\, m \in \mathcal{M} : V(pk, m, S(sk, m))$ = accept

Introduction

- Digital file or ledger that contains transactions
  - Changed when money is exchanged
  - No backing (Gold/Govt.) of the transactions involved
  - Mutual trust in its value drives the system
- Every participant can maintain their own copy of the ledger
  - Not reqd. for general users; only for those who maintain the system
  - Everyone can see everyone else's balances
  - Real system only uses account numbers and not names
- Ledgers kept in sync
  - Sender informs everyone else
  - Broadcast a message with sender, beneficiary a/c no, amount
  - Everyone across the entire world (nodes) then updates their ledger, and pass along the information to other nodes
  - Math based security: enabling a system that lets a group of computers maintain a ledger
  - Ledger concept is similar to what is done in banks
- Value of bitcoin
  - Does not represent anything valuable in the physical world
  - Only has value because we believe it is valuable, just like any other fiat currency

Account Security

- Ledger concept is similar to that in banks
  - Complication: No single entity, maintained by a group of strangers: issue of trust
  - Everyone knows about everyone else's financial transactions
- Design of the system is meant to protect every aspect of the system
  - Usage of Mathematical functions
- Mechanism to prevent unauthorised usage of account numbers
  - Nodes have to be sure that the request is authentic: only the rightful owner has sent the message
  - Needs a signature to prove that the sender is the real owner of an account
  - Concept of signature is based on maths: kind of password to unlock and spend funds
  - Not a simple static password, completely different Digital Signature is required for every transaction

Account Security

- New Bitcoin account number:
  - Associated with a private key: password equivalent
  - Mathematically linked to that account number
  - Bitcoin wallets holds these keys: allows creation of signatures
- Signature:

```
signature = f(message, private key)
verify(message, public key, signature)
```

  - Intermediary that proves that the sender has the password without requiring the sender to reveal it
  - Proves the authenticity of a message: uses a mathematical algorithm that prevents copying or forgery in the digital realm
  - Derived out of (a) private key (b) transaction text
  - Can be verified through the public key
    - Public keys are actually the "send to" addresses in Bitcoin
    - When we send money to someone, we are really sending it to their public key
  - Other nodes in the network use the signature to verify that it corresponds with the public key
  - Specific to a transaction: cannot be copied/reused
  - No proof as to when the transaction originated

Creating a Transaction

- To send an amount of BTC to a recipient, sender must:
  - Reference other transactions where he received the said amount or more
  - Referenced transactions are called "inputs"
  - Other nodes verifying this transaction will check the inputs
    - Make sure the sender was in fact the recipient
    - Inputs add up to the said amount or more Bitcoins
    - Inputs can affect privacy
  - Extra amount needs to be sent back to the sender
- Referenced input linkages ensure that
  - Ownership of Bitcoins is passed along in a kind of chain
  - Validity of each transaction is dependent on previous transactions
- On first install, each node validates ALL transactions from the very first one

Transactions

- Once a transaction has been used, it is considered spent, and cannot be used again
- When verifying a transaction, nodes also make sure the inputs haven’t already been spent
  - Nodes check every other transaction ever made
- Figuring out your own balance requires iterating through every transaction ever made and adding up all the unspent inputs
- Any "user-error" mistakes can result in the permanent loss of Bitcoins
  - Loss of private key => any funds associated with the corresponding public key will be lost forever

Ordering Transactions

- Ordering of transactions is important
  - Determines the order of payment
- Problem: Difficult to determine transaction order in Bitcoin
  - No single intermediary (bank) – multiple individuals involved
  - Network delays might cause transactions to arrive in different orders at different places
    - Effectively allowing fraudulent sender to spend money twice
- Solution: All participants decide on transaction order
  - New transactions go into a pool of pending transactions
  - Subsequent grouping of transactions into a series (chain) with locked order
  - Selection of the next transaction in the chain: mathematical lottery
    - Participants select a pending transaction of their choice
    - Begin trying to solve a special problem
    - First participant to find a solution gets to have its transaction selected as the next in the chain

Ordering Transactions (Contd.)

- Problem which has to be solved for linking transactions:
  - Cryptographic hash: combines inputs to create a number
  - Irreversible: no easy way to start with an output and then find an input
  - Approach: making guesses – use random numbers (along with other inputs) until the output meets certain criteria
    - Other inputs: transaction from the pending pool and chain
- It takes about 10 minutes on average for someone to find a solution on the network
  - Unlikely that two people will solve it at the same time
- Occasionally, more than one block will be solved at the same time, leading to several possible branches
  - Build on top of the first one that is received
  - Others may have received the blocks in a different order
    - Will be building on the first block they received
  - The tie gets broken when someone solves another block

Ordering Transactions (Contd.)

- General rule:
  - Always immediately switch to the longest branch available
  - Rare for multiple blocks to be solved at the same time
  - Rarer for this to happen multiple times in a row
- End result:
  - Block chain quickly stabilizes
  - Everyone is in agreement about the ordering of blocks a few back from the end of the chain

Ordering Transactions (Contd.)

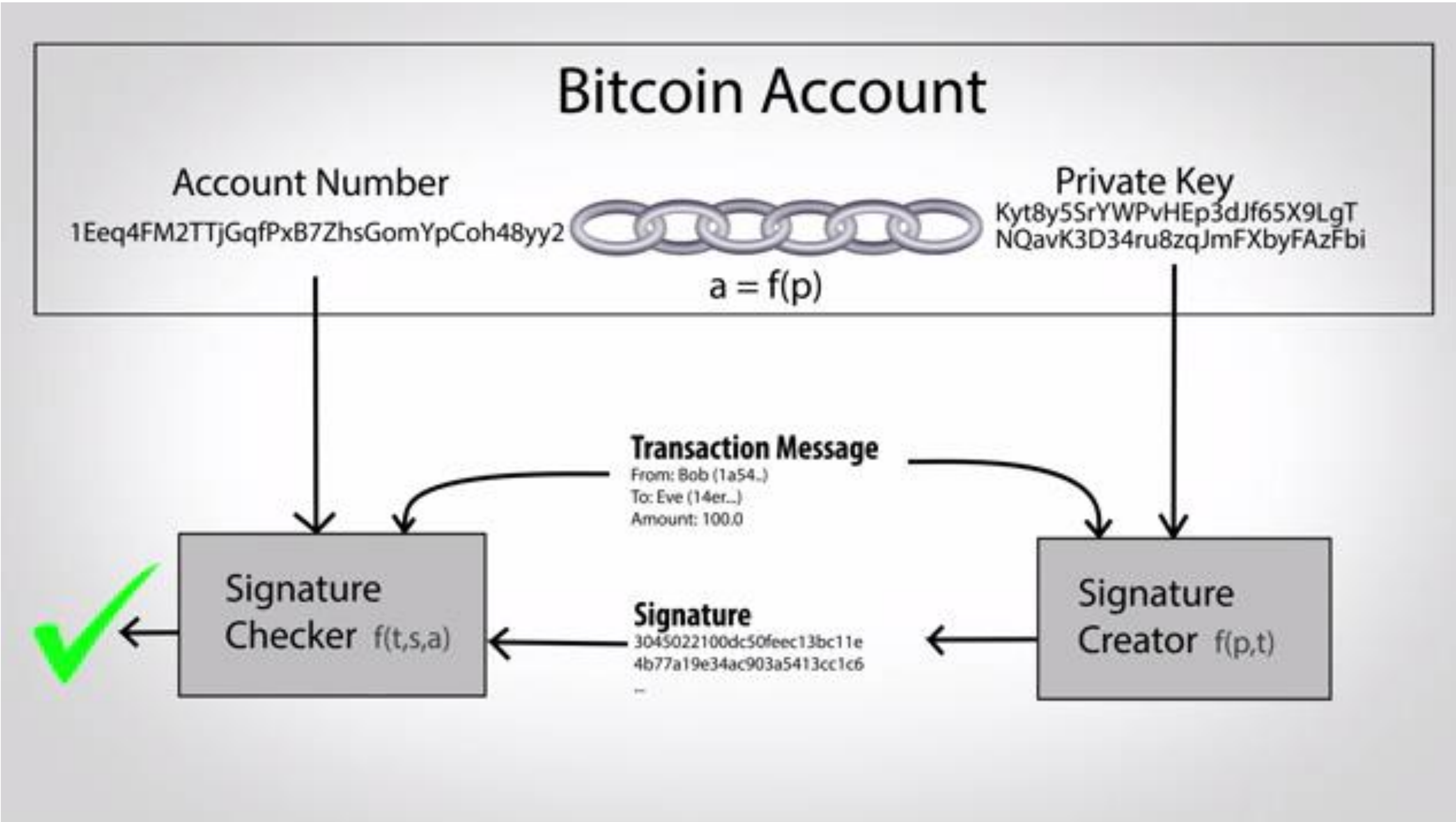
- Also helps ensure that all participants agree about past transactions
  - New participant will get different versions of the transaction chain
  - Need to choose the one that the majority is using
  - Done on the basis of a kind of voting, linked to guesses in solving the "problem" for ordering transactions
  - Each guess for solving the "problem" is effectively a vote for that chain
  - Each guess has a cost in computing power
    - Makes it unlikely that a single person or group could ever afford to outvote or out-compute the majority of users
  - Possible to look at any given answer and estimate how many guesses it took to find it: can be tallied to check number of "votes"
  - Problems are of comparable difficulty levels (recalibrated every two weeks)

Money Creation

- Linked with a win in the mathematical lottery
- New Bitcoins are created by the system with each win
- These are awarded to their account
  - In addition to getting to pick the next transaction in the chain

- I have 100 BTC => I know the private key related to a public key and its balance is 100 BTC that is unspent

Signature Creation and Verification



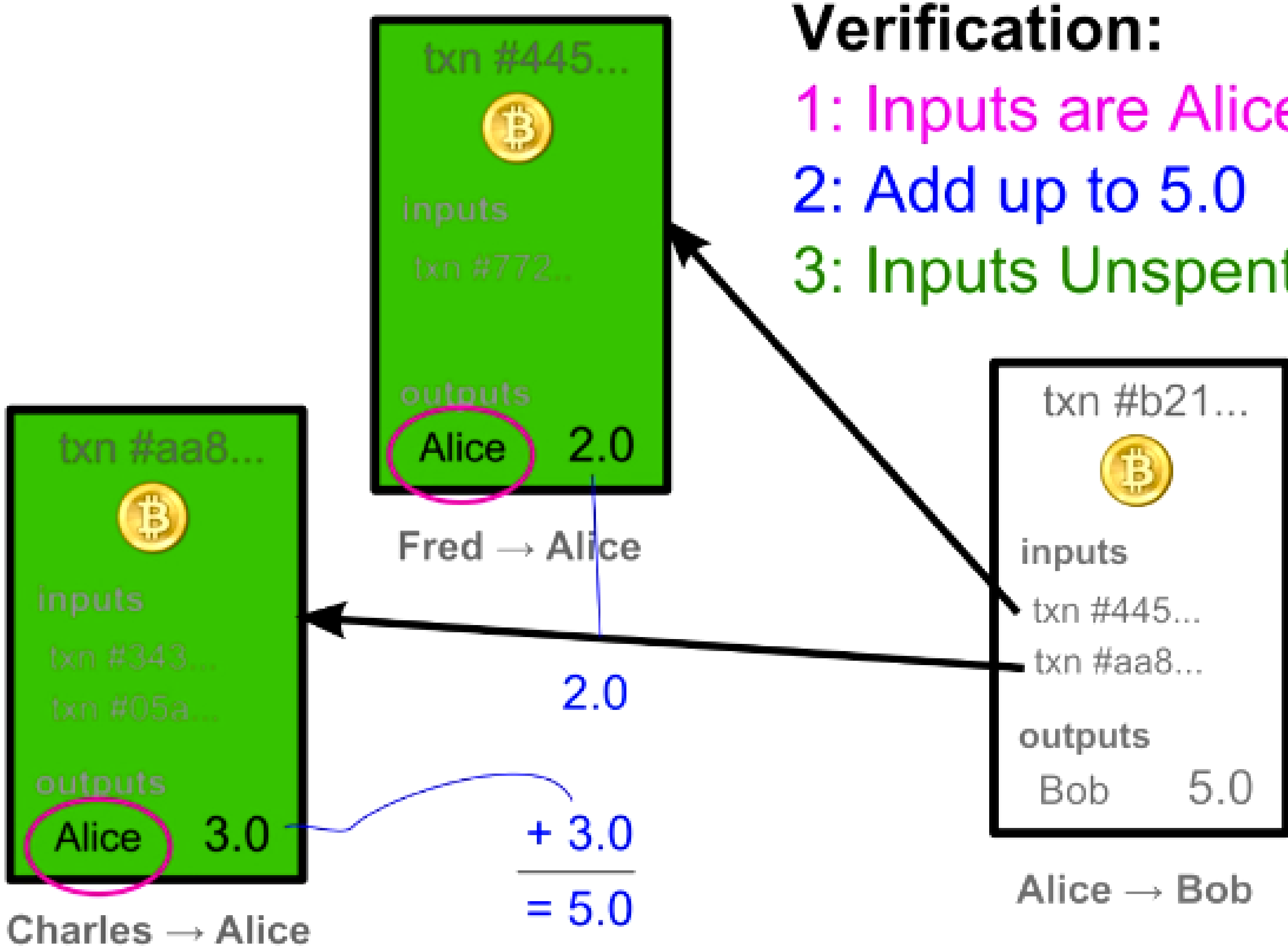
Transaction Verification

Verification:

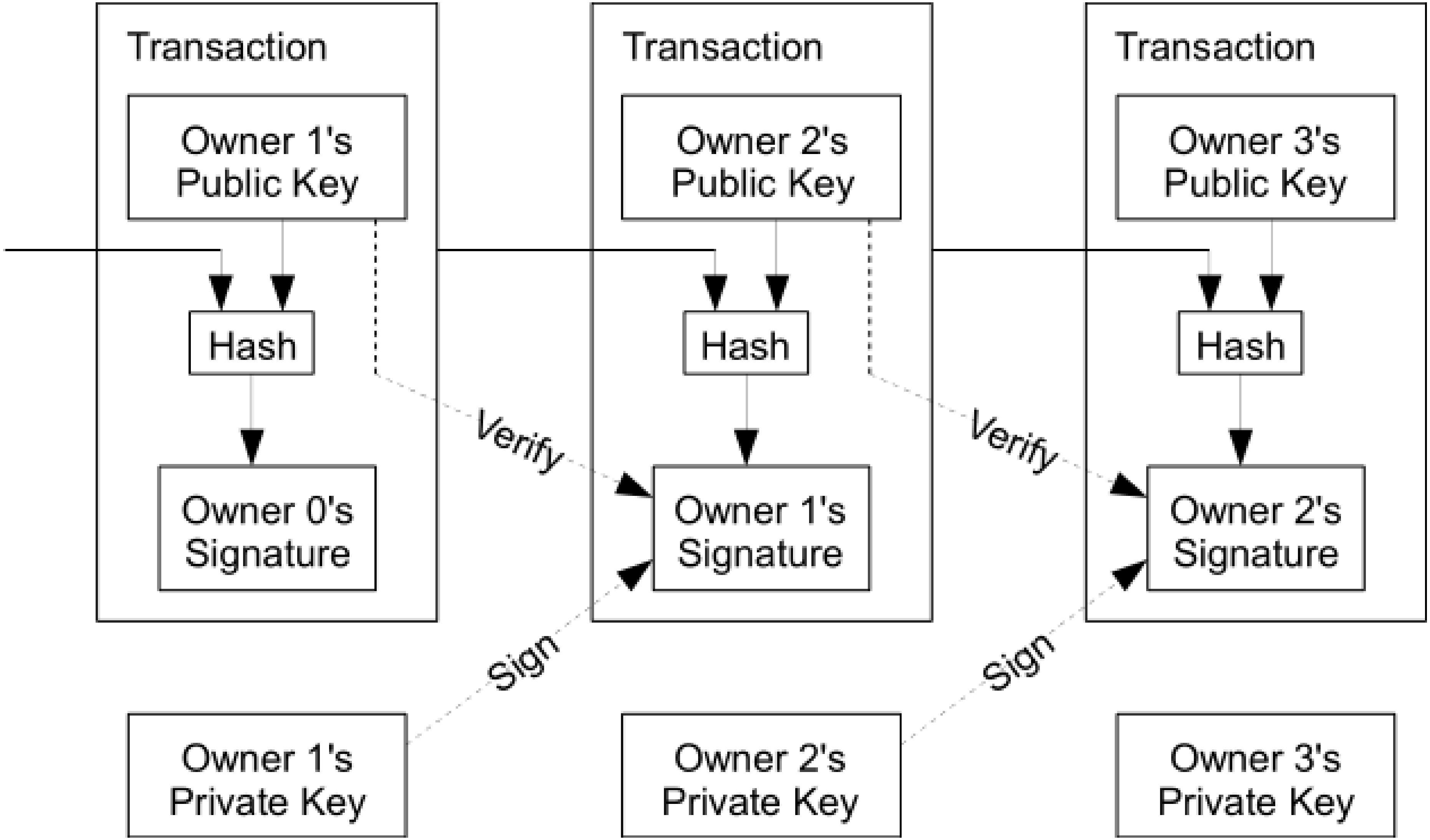
1: Inputs are Alice's

2: Add up to 5.0

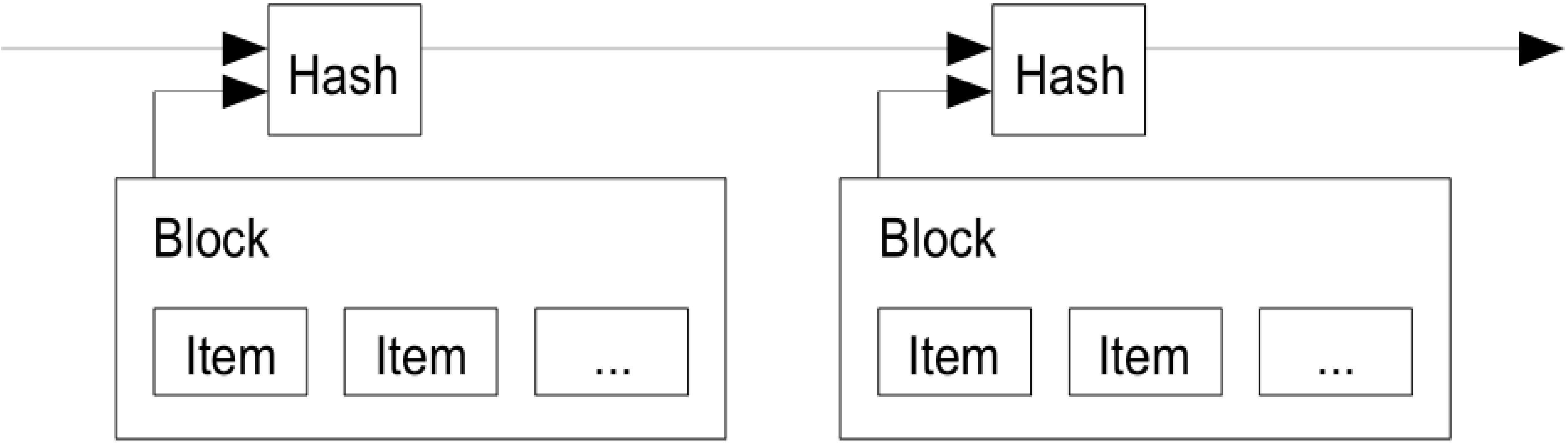
3: Inputs Unspent



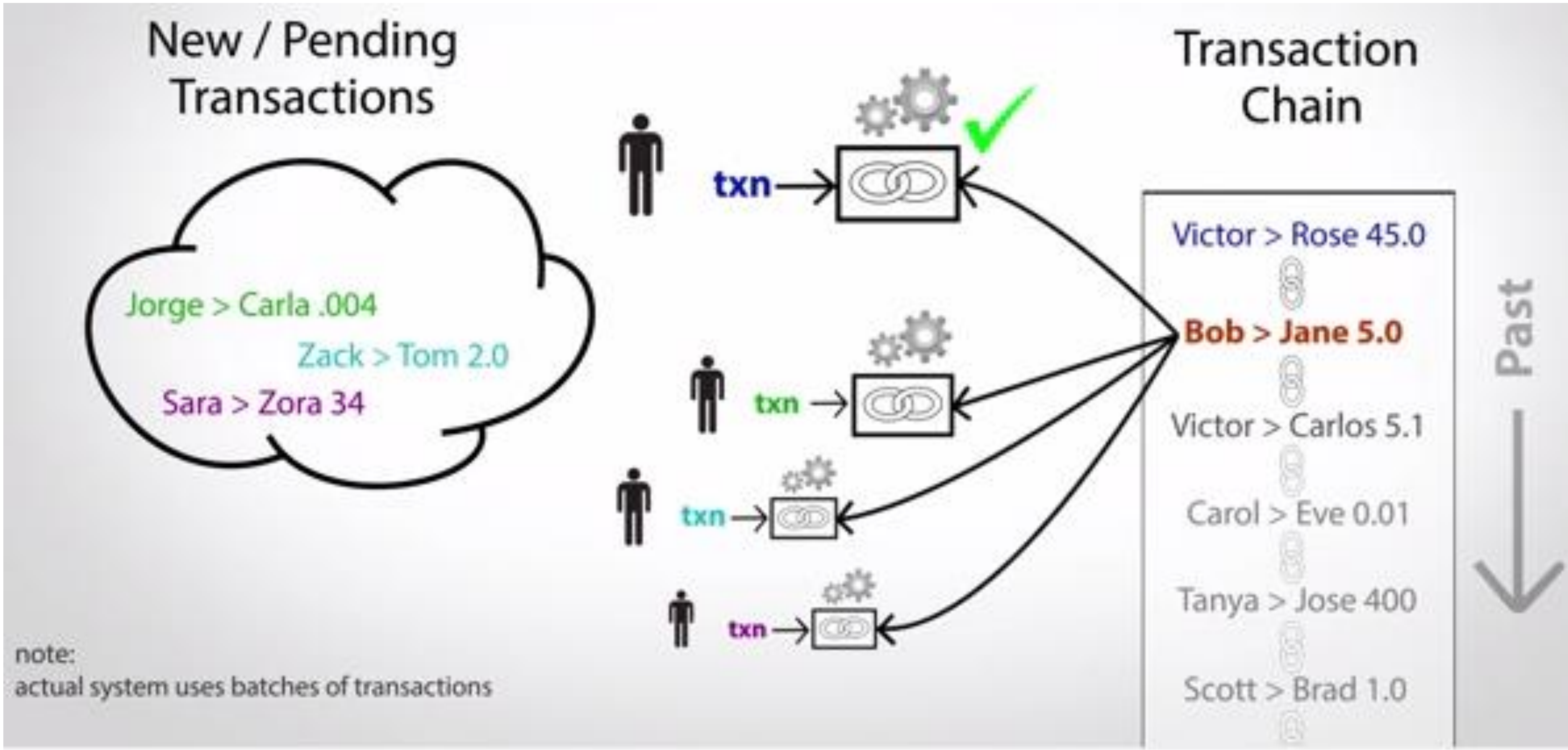
Transactions



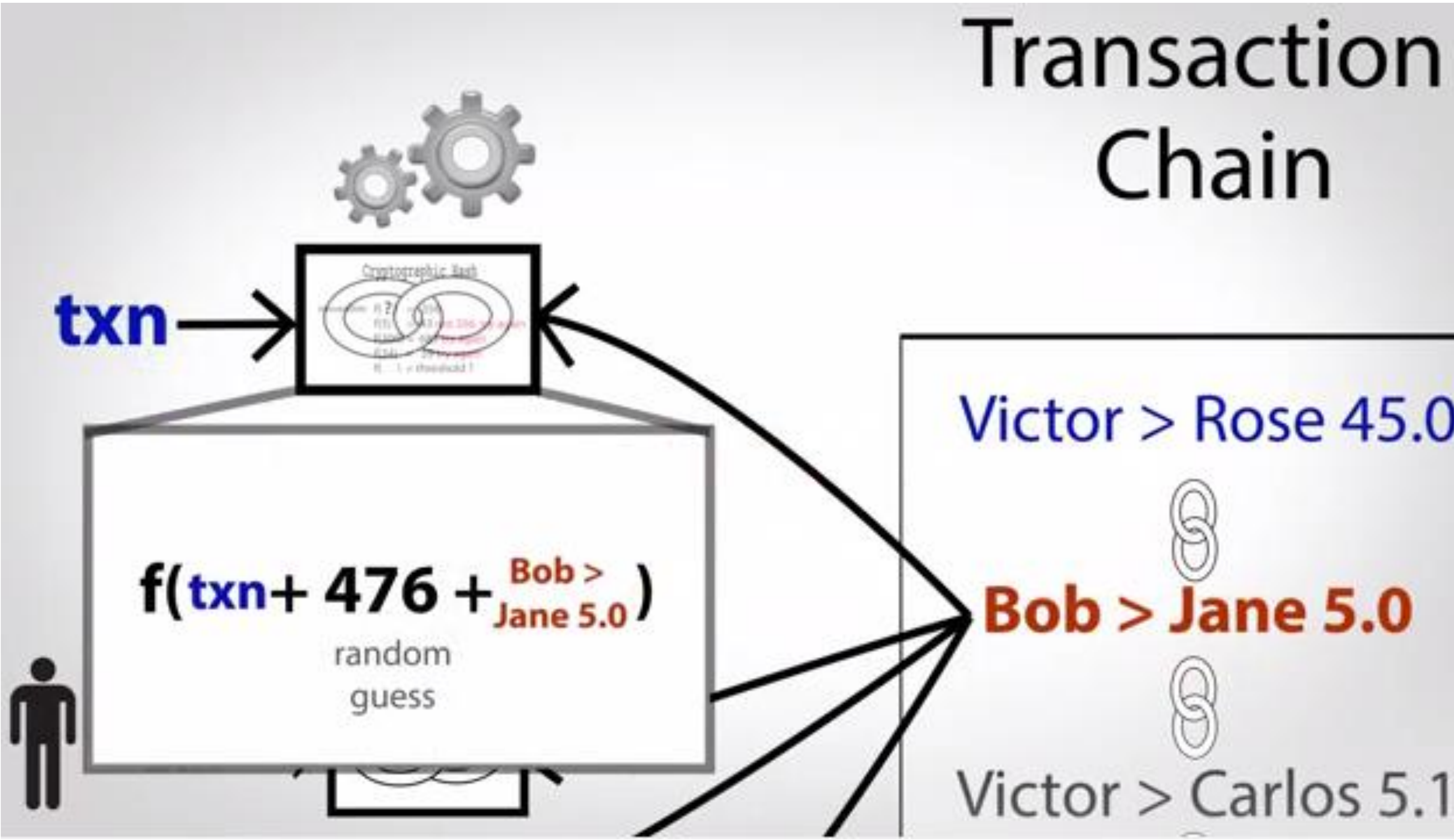
Transactions in the Block



Ordering Transactions



Transaction



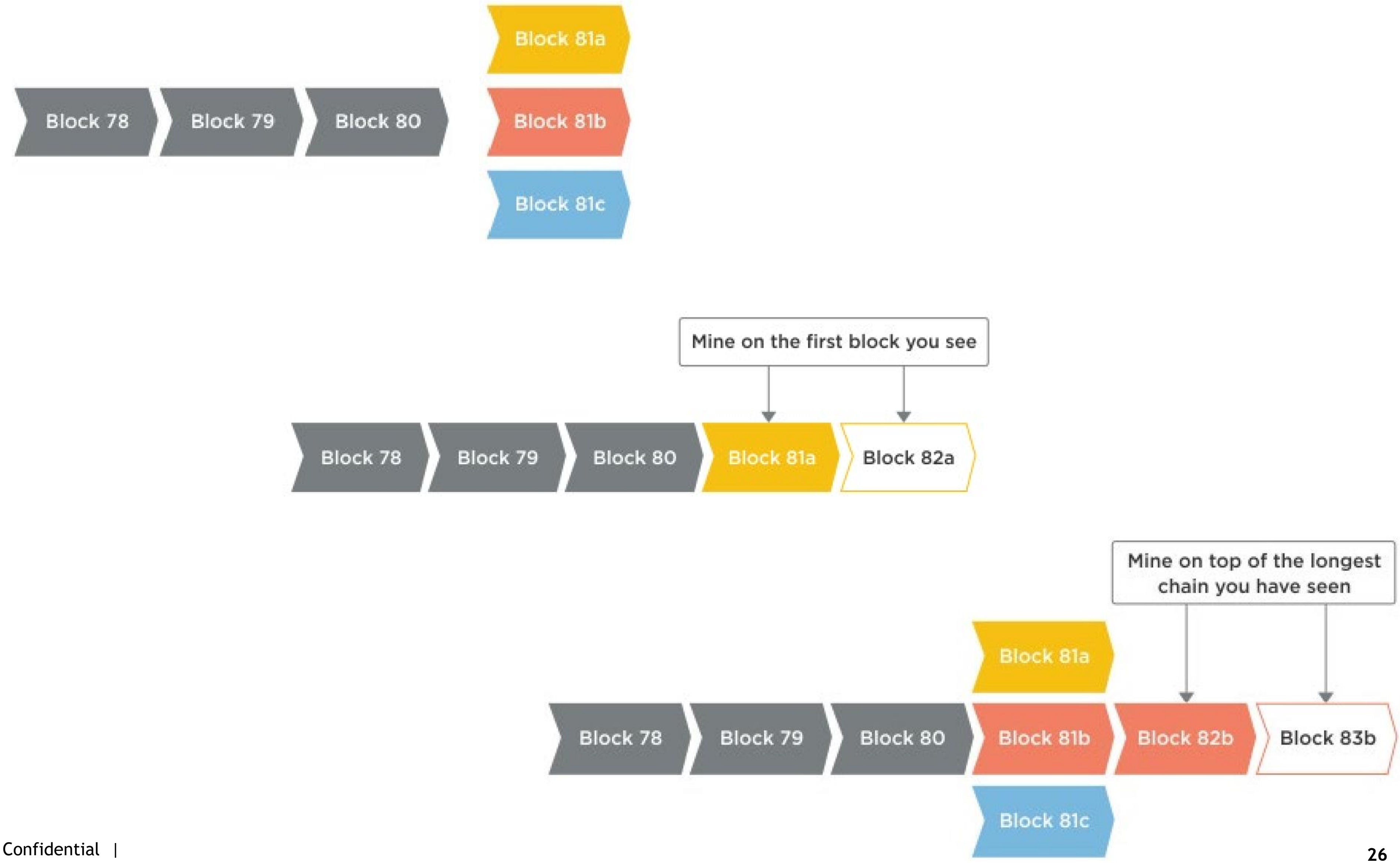
Solving a block

Cyrpto Hash Locks Blocks in Place

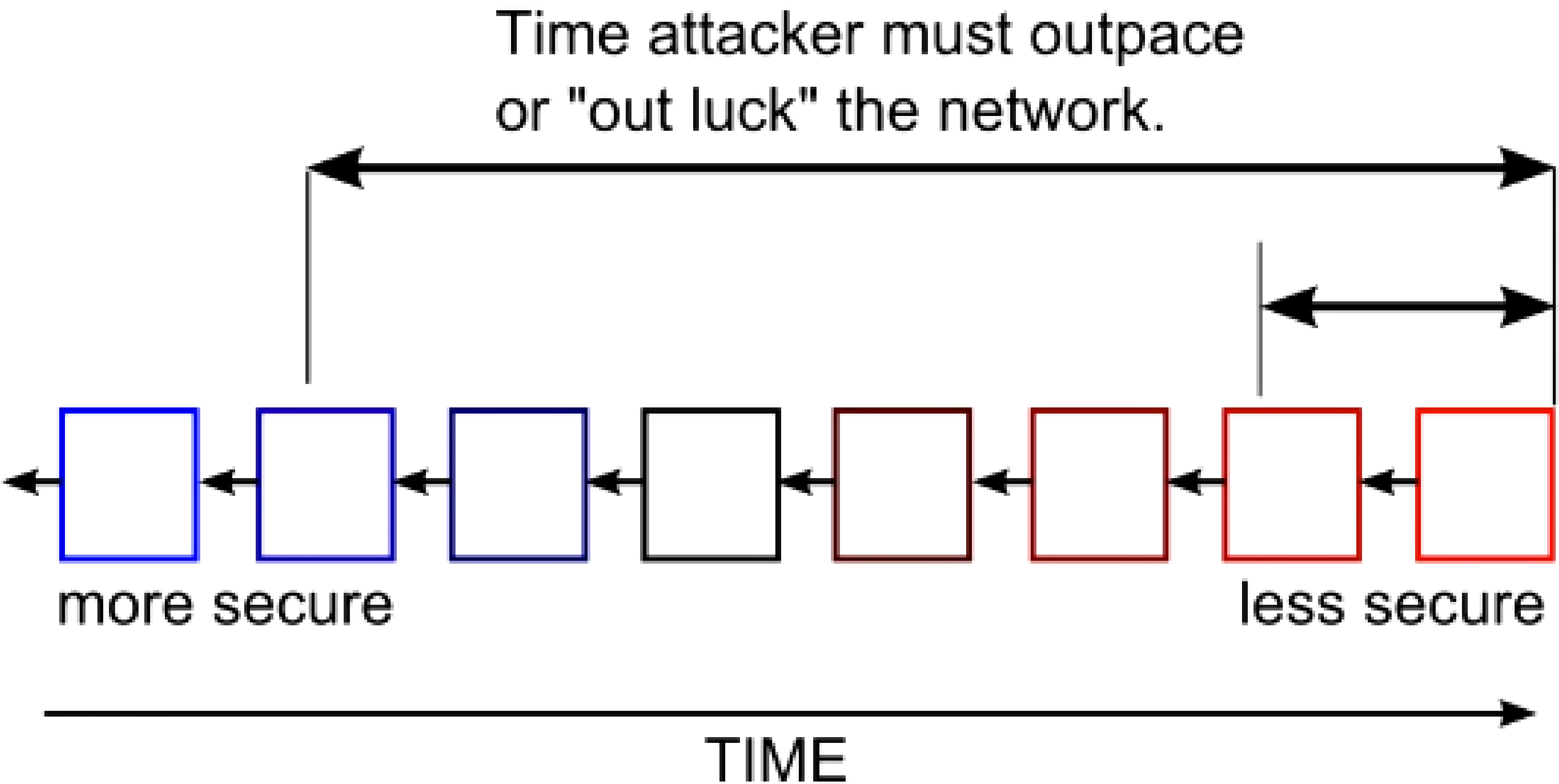
block contents

prev block ID	transactions	random guess (nonce)		hash result	?	target
f(#78A...,	tx#839, tx#a76,...,	3001	) =	438...	<	100...
f(#78A...,	tx#839, tx#a76,...,	3002	) =	988...	<	100...
f(#78A...,	tx#839, tx#a76,...,	3003	) =	587...	<	100...
f(#78A...,	tx#839, tx#a76,...,	3004	) =	087...	<	100...

Longest chain rule: Consensus



End of the chain



Thank you