Table of contents

- Preface
- Traditional financial agreements
- Who is Satoshi Nakamoto
- Fundamentals of Bitcoin blockchain

Preface

The path to Bitcoin is long and interesting

- Hundreds of notable cryptographic payment systems
  - Both e-cash and credit card based technologies
  - Academic proposals + actually deployed & tested proposals

ACC	CyberCents	iKP	MPTP	Proton
Agora	CyberCoin	IMB-MP	Net900	Redi-Charge
AIMP	CyberGold	InterCoin	NetBill	S/PAY
Allopass	DigiGold	Ipin	NetCard	Sandia Lab E-Cash
b-money	Digital Silk Road	Javien	NetCash	Secure Courier
BankNet	e-Comm	Karma	NetCheque	Semopo
Bitbit	E-Gold	LotteryTickets	NetFare	SET
Bitgold	Ecash	Lucre	No3rd	SET2Go
Bitpass	eCharge	MagicMoney	One Click Charge	SubScrip
C-SET	eCoin	Mandate	PayMe	Trivnet
CAFÉ	Edd	MicroMint	PayNet	TUB
CheckFree	eVend	Micromoney	PayPal	Twitpay
ClickandBuy	First Virtual	MilliCent	PaySafeCard	VeriFone
ClickShare	FSTC Electronic Check	Mini-Pay	PayTrust	VisaCash
CommerceNet	Geldkarte	Minitix	PayWord	Wallie
CommercePOINT	Globe Left	MobileMoney	Peppercoin	Way2Pay
CommerceSTAGE	Hashcash	Mojo	PhoneTicks	WorldPay
Cybank	HINDE	Mollie	Playspan	X-Pay
CyberCash	iBill	Mondex	Polling	

Preface

The path to Bitcoin is long and interesting

- How many survived?
  - There's a lot to learn from this history
    - Bitcoin system cleverly solves the past issues

ACC	CyberCents	iKP	MPTP	Proton
Agora	CyberCoin	IMB-MP	Net900	Redi-Charge
AIMP	CyberGold	InterCoin	NetBill	S/PAY
Allopass	DigiGold	Ipin	NetCard	Sandia Lab E-Cash
b-money	Digital Silk Road	Javien	NetCash	Secure Courier
BankNet	e-Comm	Karma	NetCheque	Semopo
Bitbit	E-Gold	LotteryTickets	NetFare	SET
Bitgold	Ecash	Lucre	No3rd	SET2Go
Bitpass	eCharge	MagicMoney	One Click Charge	SubScrip
C-SET	eCoin	Mandate	PayMe	Trivnet
CAFÉ	Edd	MicroMint	PayNet	TUB
CheckFree	eVend	Micromoney	PayPal	Twitpay
ClickandBuy	First Virtual	MilliCent	PaySafeCard	VeriFone
ClickShare	FSTC Electronic Check	Mini-Pay	PayTrust	VisaCash
CommerceNet	Geldkarte	Minitix	PayWord	Wallie
CommercePOINT	Globe Left	MobileMoney	Peppercoin	Way2Pay
CommerceSTAGE	Hashcash	Mojo	PhoneTicks	WorldPay
Cybank	HINDE	Mollie	Playspan	X-Pay
CyberCash	iBill	Mondex	Polling	

Traditional financial agreements

- Barter system

- Drawback
  - Coordination of people
    - Needs and alignment of needs
      - Same place and same time
- Two systems emerged: credit and cash

Traditional financial agreements

- Credit-based system
  - Favor/debt
    - Settle in future
      - Another trade with another person (in barter)
  - No “bootstrapping” but
    - Debt is risk, precious trades (in barter)

- Cash-based system
  - Random order and precious trades
  - Instant settlement
    - Initial “bootstrapping”

- Credit and cash are fundamental ideas, combine them

The trouble with credit cards online

- Credit card are dominant payment methods
  - A banks, credit card companies/processors, and misc intermediaries
  - Give seller your card details (security) and identity (privacy)

- Paypal-like systems
  - Act as intermediary (fee?)
    - Requires both buyer and seller to have Paypal accounts

- Today, we are getting comfortable with giving credit card details when shopping online
  - Despite knowing that companies collect data about our online shopping and browsing activity

From credit to (crypto) cash

- Cash (no debt, no default)
  - Anonymity (your credit card is who's name?)
  - Offline payments

- Bitcoin doesn't quite offer these properties, but comes closer to it
  - Anonymous to the same level as cash is.
    - No real identity to pay in Bitcoin
      - If you are not careful, your transactions on public ledger may be linked to identity (we'll see it later)
  - Doesn't work in a fully offline way either
    - "Green addresses" allow us to do offline payments in certain situations/assumptions (we'll see it later)

From credit to (crypto) cash

- David Chaum 1983 applied cryptography to cash
  - Giving out paper that say "*The bearer of this note may redeem it for a dollar by presenting it to me*" with my **signature** attached
    - People trust me
    - Consider my signature unforgeable
    - Much like banknote
  - Did the same thing electronically with digital signatures
    - "**Double spending**" problem
      - Virtual piece of paper can be copied, pass to different people
      - Can we solve double spending in this world?

From credit to (crypto) cash

- Can we solve double spending in this world?
  - Put unique serial numbers into each note I give
  - If you receiving a note from someone
    - Check my signature & calls me to verify spent-status
    - If I say no, you accept
      - I'll record serial number as spent in my ledger
    - No double spending, recipient calls me
    - Periodically bring notes for settlement (cash/fresh notes)
- Server for signing and recordkeeping
  - Tracking

From credit to (crypto) cash

- David Chaum further solved it
  - When I issue a new note to you, you pick the serial number
    - Long, random, unique
  - You write it (hiding from me), and I'll sign it
    - "Blind signature"
- This was the first serious digital cash proposal
  - Still requires a server run by a central authority
  - Server down, payments halt

From credit to (crypto) cash

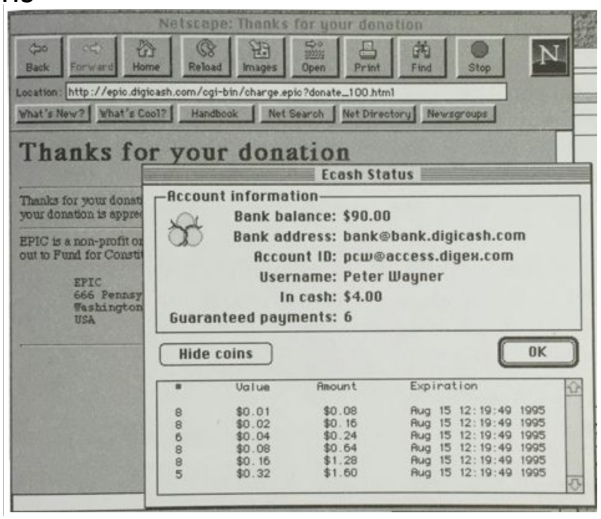
- Chaum-Fiat-Naor proposed **offline** electronic cash in 1988.
  - How can we possibly stop double spending if recipients are offline?
  - Stop worrying about preventing and focus on detecting it
    - When the merchant re-connects to the bank server
  - Credit card on an airplane, there is no network in the skies
    - Transaction are processed when airline re-connects to the network
    - If your card is denied, you'll owe the airline (or your bank)
- Traditional finance is based on detecting an error or loss, followed by recovery or punishing the perpetrator
  - Personal cheques

From credit to (crypto) cash

- Chaum-Fiat-Naor idea for detected double spending
  - Based on intricate cryptographic
  - Every digital coin issued to you encodes your identity
  - Only you and no one else can decode it
  - To spend your coin
    - The recipient will ask you to prove that you indeed own that coin
    - They'll keep a records
  - The decoding isn't enough to determine your identity
  - When you double spend a coin
    - Both recipients eventually go to the bank
    - Bank can put the two pieces of info together to decode your identity completely with very high probability
  - Can someone frame you?
    - Decoding set at spending time

From credit to (crypto) cash

- Chaum's DigiCash in 1989
  - Earliest company that tried to solve problems of online payments
  - Digicash realized Ecash
    - Clients are anonymous, but merchants aren't
    - There's no way to split your coins



From credit to (crypto) cash

- Chaum patented Digicash technology (+blind-signature)
  - It stopped others from developing similar protocol for ecash systems
  - Persuading banks and the merchants
  - Centered on the user-to-merchant transaction
  - DigiCash lost and the credit card companies won

- Other companies (e.g., Mondex) had ecash systems based on tamper-resistant hardware to prevent double spending
  - Wallet + smart card
  - Wallet was slow
  - Smart cards are like cash (bearer's)

From credit to (crypto) cash

- Cryptographers on cypherpunks mailing list wanted an alternative
  - Ben Laurie (Lucre - alternative blind-signature scheme), Ian Goldberg (coin splitting), MagicMoney (ecash alternative)
    - But, couldn't achieve as good as Bitcoin system
  - Cyperpunks was the predecessor to the mailing list where Satoshi Nakamoto announced Bitcoin to the world
  - This is no coincidence

Satoshi Nakamoto announced Bitcoin on this very mailing list, emphasizing that the creation of Bitcoin was no coincidence, but rather a culmination of years of work by these early cryptographers and privacy advocates.

Minting money out of thin air

- What makes a digital cash of $100 actually worth $100?
  - To obtain digital cash worth $100, you pay $100 to the bank/company that issues you the digital cash
- But, what is their backing?
  - e-Gold, Digigold based on gold reserves/piles
  - Ultimately, dollar or a commodity
    - Dependency
- How about digital money being its own currency, issued and valued independently of any other currency?

Minting money out of thin air

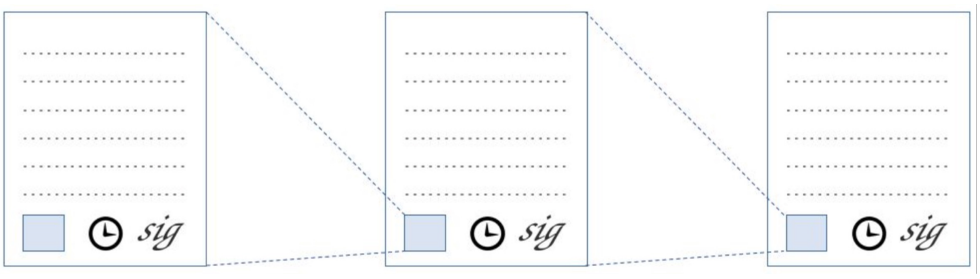
- For a digital currency to acquire real value, you need something that's scarce by design
  - Scarcity makes gold to be used as a backing for money
- Scarcity in digital world is computation power
  - Solve a computationally difficult/time-consuming "puzzle" to mint an asset
- The basic idea that *"solutions to computational puzzles could be digital objects that have some value*" is pretty old.

Minting money out of thin air

- Dwork and Naor in 1992 proposed solution to email spam
  - To send an email, your computer should solve a puzzle that would take a few seconds to solve
  - The recipient's email program ignores an incoming email if the right solution is not attached
  - For the average user, it wouldn't be a barrier
  - But if you're a spammer?
- Similarly, Adam Back in 1997 proposed Hashcash
  - Bitcoin uses essentially the same computational puzzle as Hashcash
    - Ofcourse, more complicated than Hashcash
      - Bitcoin "mining" (we'll see it later)

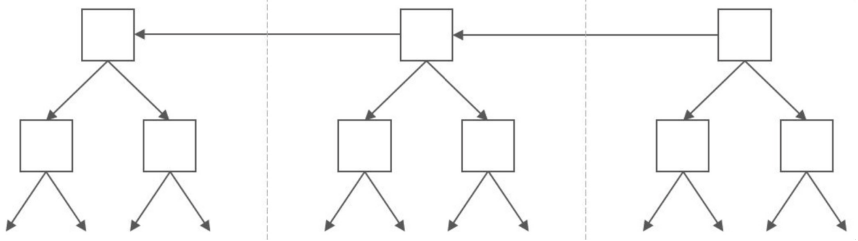
Recording everything in a ledger

- Another key component of Bitcoin is blockchain
  - A ledger in which all Bitcoin transactions are securely recorded
- The idea of blockchain is again quite old
  - Paper by Haber and Stornetta in 1991
    - Secure timestamping of digital documents
    - order of creation of documents
    - Once timestamped, a document's timestamp can't be changed
    - Signs document + current time + a pointer to previous doc



Recording everything in a ledger

- Efficiency improvement
  - No individual linking, collect into blocks and link blocks in a chain
  - In blocks, documents are linked together (tree structure not linearly)
    - Decreases verification efforts
    - Dotted vertical lines indicate time intervals



- This data structure forms the skeleton of Bitcoin's blockchain
  - Timekeeping is not of much importance in Bitcoin
  - Point is to record the relative ordering of transactions
    - In a tamper-resistant way

Recording everything in a ledger

- Bitcoin

Computational puzzles	+	Secure timestamping
to regulate the creation of new currency units		to record a ledger of transactions & prevent double spending

- Previous less sophisticated proposals to combine these two
  - b-money by Wei Dai in 1998
    - Anyone can create money using a hashcash-like system
    - There's a peer-to-peer network, sort of like in Bitcoin
    - Each node maintains a ledger, but it's not a global ledger
  - Bitgold by Nick Szabo (1998/2005 see the blogpost controversy)

Hashcash is a **proof-of-work** system created by Adam Back in 1997 to prevent spam and denial-of-service attacks by requiring the sender to perform a computationally expensive task. This system later became the basis for **Bitcoin mining**, where miners solve complex puzzles to validate transactions and secure the blockchain.

Recording everything in a ledger

- Bitcoin has several important differences from b-money & Bitgold

b-money and Bitgold	Bitcoin
1. Computational puzzles are used directly to mint currency	1. Puzzle solutions themselves don't constitute money, they are used to secure blockchain, and only indirectly lead to minting money
1. Rely on timestamping services	1. Doesn't require trusted timestamping as it aims to preserve the relative order of blocks/transactions
1. There isn't a clear way to resolve any disagreement about the ledger among the servers/nodes	1. It's the majority that decide the correct state of ledger

Who is Satoshi Nakamoto

- Satoshi Nakamoto - a pseudonym for the creator of Bitcoin
  - Says started coding Bitcoin around May 2007
  - Registered *bitcoin.org* in August 2008
  - Sent emails to people who might be interested in protocol
  - Oct 2008, publicly released a white paper describing protocol
  - Soon after, released the initial code for Bitcoin
  - Stuck around for ~2 years
    - Posted lots of messages on forums, emailed lots of people, and responded to people's concerns, submitted patches to the code
  - By December 2010, others taken over the maintenance of project
    - "He" stopped communicating
      - Satoshi is a male name

Who is Satoshi Nakamoto

- The real identity of Satoshi Nakamoto is unknown
  - Some say, the pseudonym was for the group
    - In the two years of communication
      - Multiple people sharing user accounts and passwords
      - Responding in a similar style and a similar voice
      - Made sure they didn't contradict each other
        - It much likely that at least this portion of Satoshi's activity was done by a single individual
        - It's also clear from his writings that this individual understood entire code base + all design aspects of Bitcoin
          - Reasonable to assume that same individual wrote original code base + white paper

Who is Satoshi Nakamoto

Dorian Nakamoto

Japanese-American

+ The name and his training as an engineer

- He denied it, and had not been working as an engineer for years.

Craig Wright

Australian scientist

+ Timestamps of Nakamoto's blog coincide with his blog

- The blogs and PGP keys "proving" he was founder were inconsistent

Hal Finney

Computer scientist

+ Nakamoto did correspond a lot with him

- No compelling one

Nick Szabo

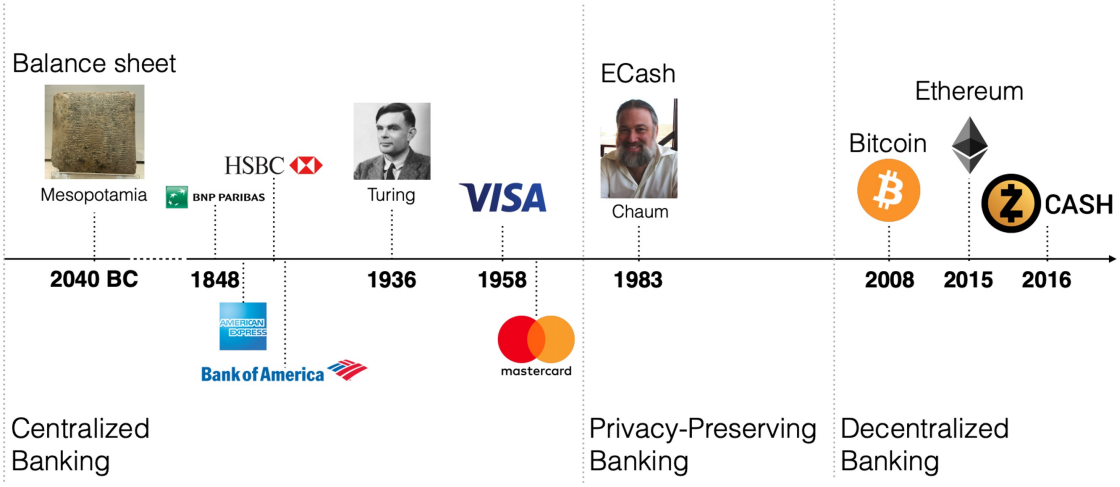
Computer scientist

+ He invented BItgold, a precursor to Bitcoin

- No compelling one

Who is Satoshi Nakamoto

- But, why hide the identity?
  - Anonymous coder "style" from the cypherpunk community
  - Legal worries
  - Personal security due to assets he owns
    - Satoshi has a lot of Bitcoin from his early mining



Fundamentals of Bitcoin blockchain

- Basic concepts to understand Bitcoin system
  - Cryptographic hash function
  - Hash pointers and data structures
  - Digital signature
  - Public keys as identities

Cryptographic hash functions

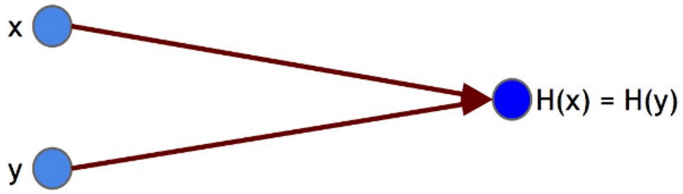
- A general hash function is a mathematical function with following properties:
  - Its input can be any string of any size
  - It produces a fixed size output
  - It is efficiently computable
    - For a given input string, you can compute the output in a reasonable amount of time

- Cryptographic hash functions (cryptographically secure) has following additional properties:
  - Collision-resistance
  - Hiding
  - Puzzle-friendliness

Hiding: Given a hash value h(x) ,it should be difficult to determine the original input x. This property ensures even if the hash is known.
Puzzle-Friendliness: For a given output hash, finding an input that hashes to this value should require a substantial amount of computational effort.This property is crucial for **proof-of-work** systems (e.g., Bitcoin mining), where solving such puzzles is intentionally made difficult.

Cryptographic hash functions

- Property 1: Collision-resistance
  - *A hash function H is said to be collision resistant if it is infeasible to find two values, x and y, such that x≠y, yet H(x)=H(y)*

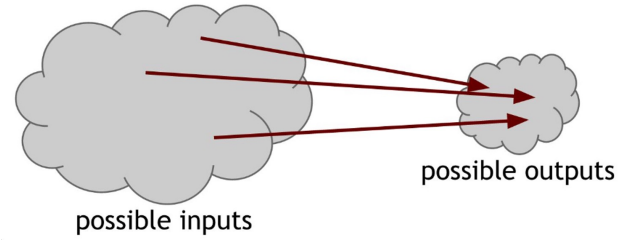


Hash collision

- Notice that we said it is infeasible to find
  - We know for a fact that collisions do exist

Cryptographic hash functions

- Property 1: Collision-resistance
  - The input space is infinite, while the output space is finite
  - To find a collision for a hash function with a 256-bit output size
    - pick $2^{256} + 1$ distinct values
    - Compute the hashes of each of them
      - Collision is guaranteed



  - Birthday paradox: We can find a collision by only examining roughly *the square root of the number of possible outputs results*

Cryptographic hash functions

- Property 1: Collision-resistance
  - Finding a collitions for a hash function with a 256-bit output
    - Compute the hash function $2^{256} + 1$ times in the worst case
    - ~ 2^{128} times on average
      - An astronomically large number
      - With 10,000 hashes/sec, you would take >10^{27} years to calculate 2^{128} hashes!
  - Thus, we have seen general but impractical algorithms to find a collision for any large-enough hash function
    - Old MD5 hash function, collisions were found after years of work

Cryptographic hash functions

- Property 1: Collision-resistance
  - Application: Message digest
    - File sharing
    - Hash serves as a fixed length digest (or unambiguous summary)
    - The file might have been GB long, the hash is of fixed length
    - Greatly reduces storage requirement

Cryptographic hash functions

- Property 2: Hiding
  - *A hash function H is hiding iff when a secret value r is chosen from a probability distribution that has high min-entropy, then given H(r || x) it is infeasible to find x*
  - The "real" input is hidden despite the output being visible
  - || denotes concatenation
  - Min-entropy is a measure of how predictable an outcome is
    - High min-entropy indicates that distribution (i.e., random variable) is very spread out
    - In other words, *r* is highly-unlikely-and-randomly-chose

Cryptographic hash functions

- Property 2: Hiding
  - We are trying to find *x,* knowing the value *H(r || x)* (and *H()* of course)
  - But we don't know *r*!
    - E.g., set for *x* could be {0,1}, but we can't decide between 0 or 1
      - Because adding a secret *r* to *x* mixes all the possible hashes
    - E.g., *Hashed_pwd = H(salt || pwd)*
      - If we know *"salt || pwd"*, we can't decide between *salt* and *pwd*
  - Trying all possible *r* and *x* for *H* will not work
    - Long enough *r* and *x*

Cryptographic hash functions

- Property 2: Hiding
  - Application: Commitment
    - Given a *msg*, generate a random value (*nonce*)
    - Create *com = commit(msg, nonce)*
    - Publish your commitment *com*
    - To reveal the committed value (*msg*)
      - Publish *msg* and *nonce*
      - Anybody can verify now

Cryptographic hash functions

- Property 2: Hiding
  - Binding property: *It is infeasible to find two pairs (msg, nonce) and (msg', nonce') such that msg ≠ msg' and H(nonce || msg) = H(nonce' || msg')*
    - Implied by the collision-resistant property
    - Reverse implications?
      - No!
      - It's possible that you can find collisions, but not in the form *H(nonce || msg) = H(nonce' || msg')*
        - E.g., *com = H(nonce || msg) = H(nonce' || msg)*
          - Scheme is still binding, but the underlying hash function is not collision-resistant

Cryptographic hash functions

- Property 3: Puzzle friendliness
  - *A hash function H is said to be puzzle-friendly if for every possible n-bit output value y, if k is chosen from a distribution with high min-entropy, then it is infeasible to find x such that H(k || x) = y in time significantly less than 2^n*
    - Given the output of a hash function (*y*) and part of the input (*k*), it is difficult to find an input (*x*)
    - The problem here is to try all the possible *x* till you find one that gives the correct hash *y*
      - So, you will end up finding one but it will take time

Cryptographic hash functions

- Property 3: Puzzle friendliness
  - Application: Search puzzle
    - A mathematical problem
    - Requires searching a very large space in order to find the solution
    - A search puzzle has no shortcuts
      - There's no way to find a valid solution other than searching that large space
  - Bitcoin mining is a sort of computational puzzle

Secure Hash Algorithm

- Bitcoin primarily uses SHA-256
- We need a hash functions work on inputs of arbitrary length
  - If we have a hash function that works on fixed-length inputs
  - Merkle-Damgard transform can convert it into a hash function that works on arbitrary-length inputs
    - And, if the underlying compression function is collision resistant, then the overall hash function is collision resistant as well

Secure Hash Algorithm

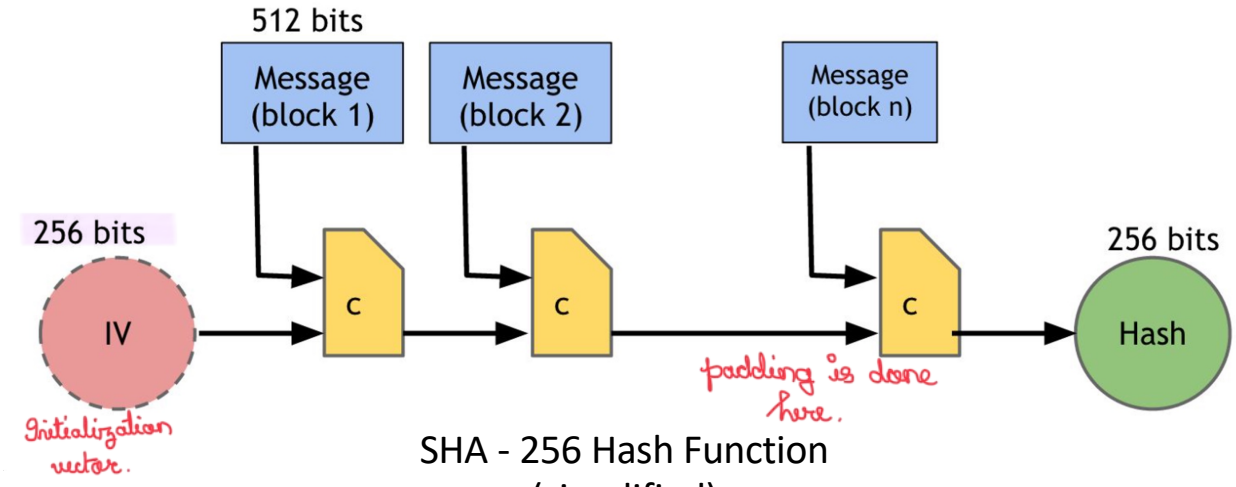
- If compression function takes inputs of length *m* and produces an output of a smaller length *n*
- The arbitrary size input to the hash function is divided into blocks of length *m-n*
- Pass each block + the output of previous block into the compression function
  - Input length will be *(m-n)+n=m;* accepted input

This is the reason why the block is divided in the length of m-n size. So that when we add the hash output from the previous block it sums up to m, which is the size of the block.

- No previous block output for the first block, use an Initialization Vector (IV)
  - Padding at last block (multiple of 512 bits for SHA-256)

Secure Hash Algorithm

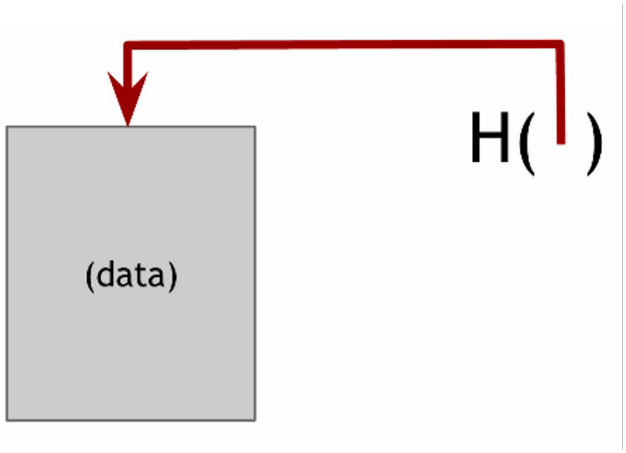
- SHA-256's compression function takes 768-bit input and produces 256-bit outputs, block size is 512 bits



SHA - 256 Hash Function (simplified)

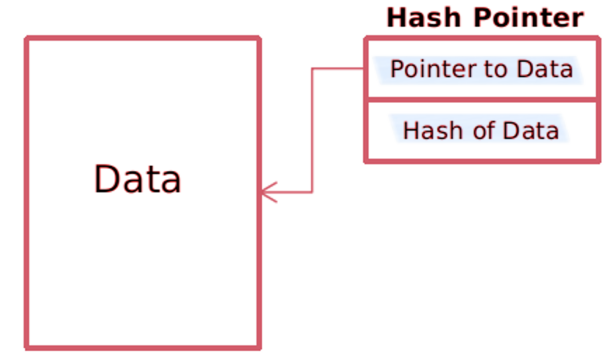
Hash pointers and data structures

- Hash pointer is
  - a pointer to where some info is stored with
  - a cryptographic hash of the info



Hash pointers and data structures

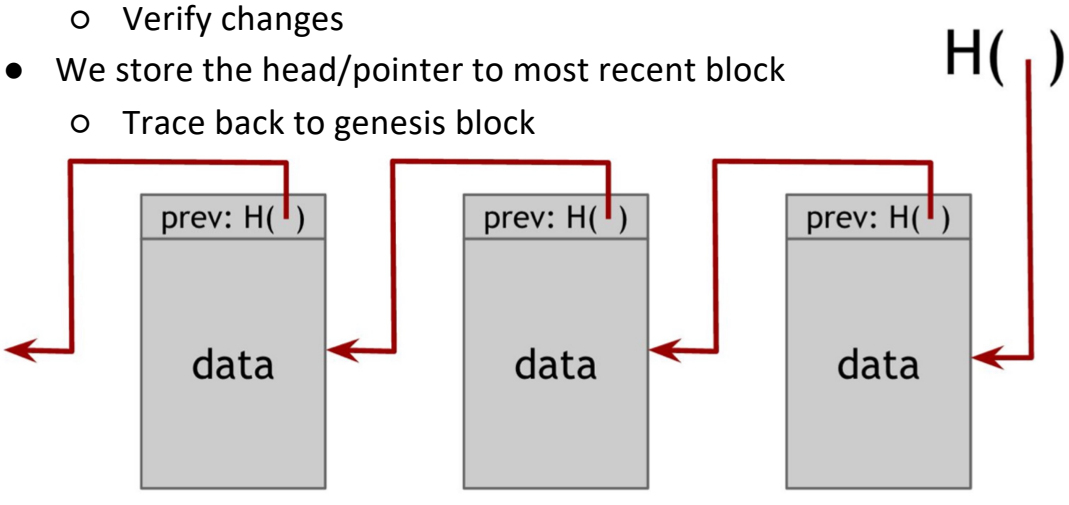
- Think of it as C Structure that contains two objects: pointer + hash



- With a hash pointer
  - Get the info back
  - Verify that it hasn't changed

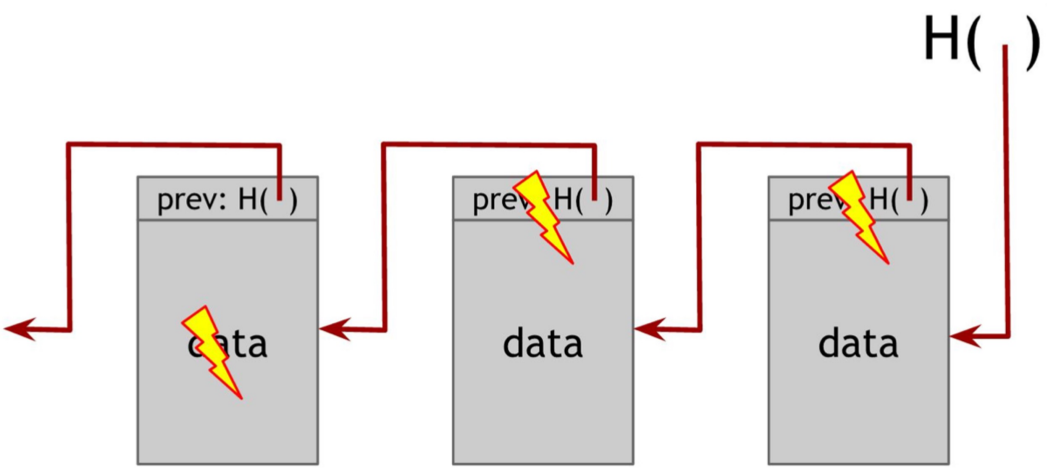
Hash pointers and data structures

- Build data structure with hash pointers
  - Block chain: a linked list built with hash pointers instead of pointers
    - Each block
      - Points to previous block
      - Contains a digest of that previous block
        - Verify changes
      - We store the head/pointer to most recent block
        - Trace back to genesis block



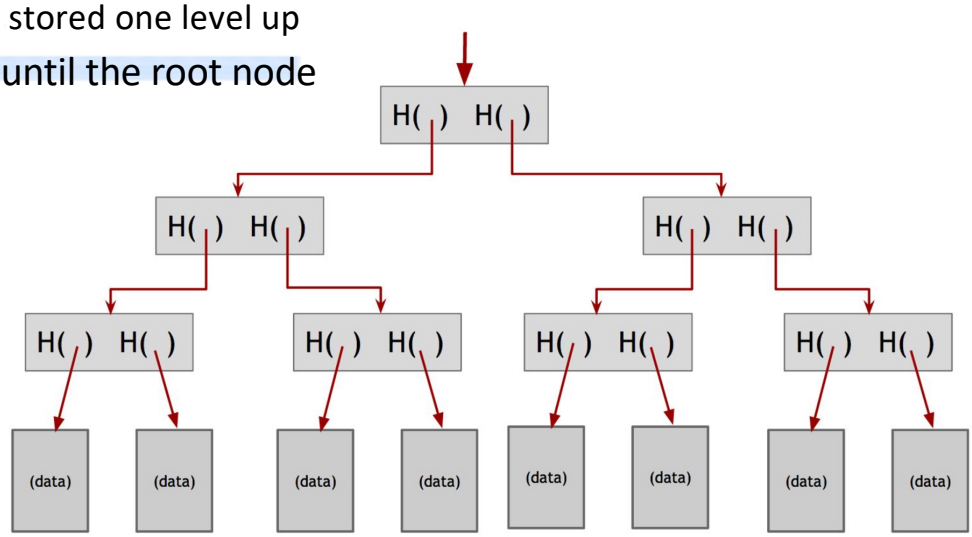
Hash pointers and data structures

- Use case: tamper-evident log



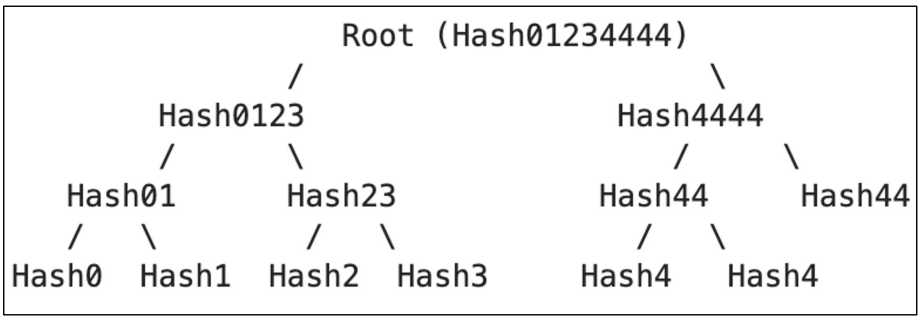
Merkle trees

- Merkle tree is a binary tree with hash pointers
  - Data blocks are grouped in pairs
    - Hash of each block in a parent node
  - Parent nodes are again grouped in pairs
    - Hashes stored one level up
  - Continues until the root node



Merkle trees

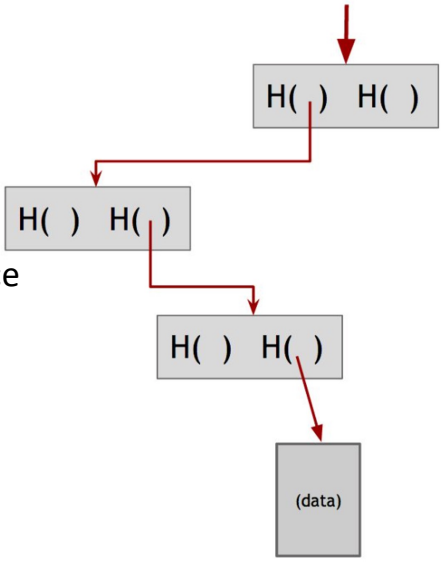
- Merkle tree is a binary tree with hash pointers
  - If odd number of nodes on any level
    - Last node is duplicated and hashed with itself



  - In Bitcoin
    - Transaction (Tx) copies aren't stored
      - Such duplication is done only at calculation time
    - Internal hashes from merkle tree aren't stored (only merkle root)

Merkle trees

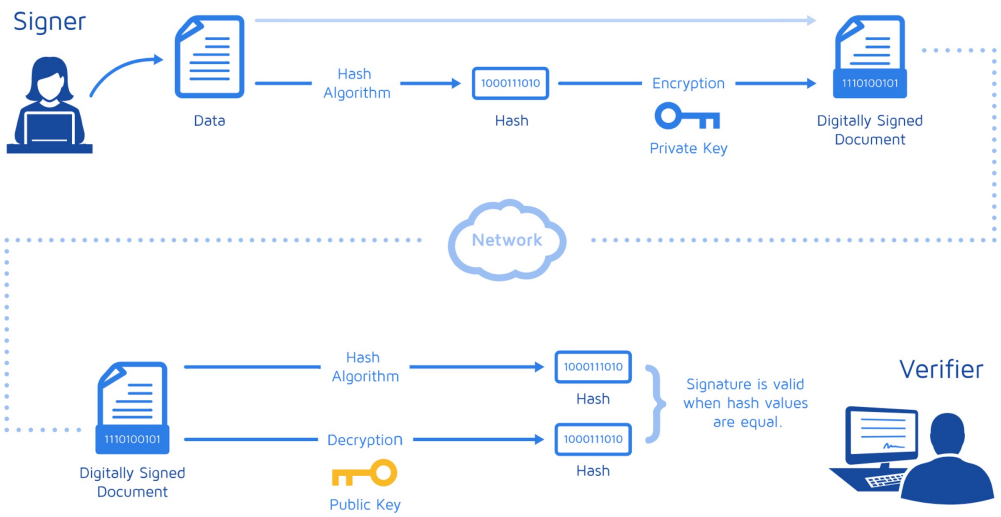
- Tamper-proofing
- Proof-of-membership/inclusion
  - Prove that a data block is included in the tree
    - Only show the blocks on the path from that data block to the root
- Sorted merkle trees
  - Sort blocks at the bottom in some order
  - Proof of non-membership
    - Verify non-membership in a log time-space



Digital Signature

- Integrity
- Authenticity
- Nonrepudiation
  - Confidentiality?
    - NO!

There is no confidentiality as the original document along with the encrypted hash is now the digitally signed document.
The same hash as the sender is applied by the receiver on the received document, and the encrypted hash is decrypted. Then both the hashes are compared to check if they are equal.



Digital Signature

- Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA) digital signature scheme
  - ECDSA over the standard elliptic curve "secp256k1"
    - Provides 128 bits of security
      - difficult to break 2^{128} symmetric-key cryptographic operations

In Bitcoin, key generation using Elliptic Curve Cryptography (ECC) works as follows:

- **Private Key: A randomly generated 256-bit number.**
- **Public Key: Generated by multiplying the private key with a predefined point G**

G on the elliptic curve: Public Key=Private Key×G

Public Key=Private Key×G

- **The public key is used to derive the Bitcoin address, while the private key remains secret and is used to sign transactions.**

Public keys as identities

- If a message with signature verifies under a public key *pk*
  - As if *pk* is saying the message
  - To use identity *pk,* you should know secret key *sk*
- Make a new identity as and when you please
  - Create a new fresh key pair, *sk and pk*
- Public keys are large
  - In Bitcoin, *pk* uncompressed: 512 bits | *pk* compressed: 257 bits
  - Use the hash of *pk*
  - To verify that a message comes from an identity, check
    - *pk* indeed hashes to the identity
    - Message verifies under *pk*