Cryptographic Primitives Used in Blockchains

Focus is on abstraction they provide…

(Take CS 485/585 for how they work)

Portland State
Computer Science

Public-key, Private-key cryptography

But first, symmetric encryption

- Three main algorithms:

```
k = Keygen(n)
C = Encrypt(k, M)
M = Decrypt(k, C)
```

- Use the **same (secret) key** to encrypt and decrypt
  - Secret key shared between sender and receiver
  - If you can encrypt, then you can also decrypt
- Fast, easy to accelerate, good for large amounts of data
  - But, has a key distribution problem
- Examples:
  - Block ciphers: AES (Advanced Encryption Standard)
  - Stream ciphers: Salsa20/ChaCha

Asymmetric encryption (Public Key, Private Key)

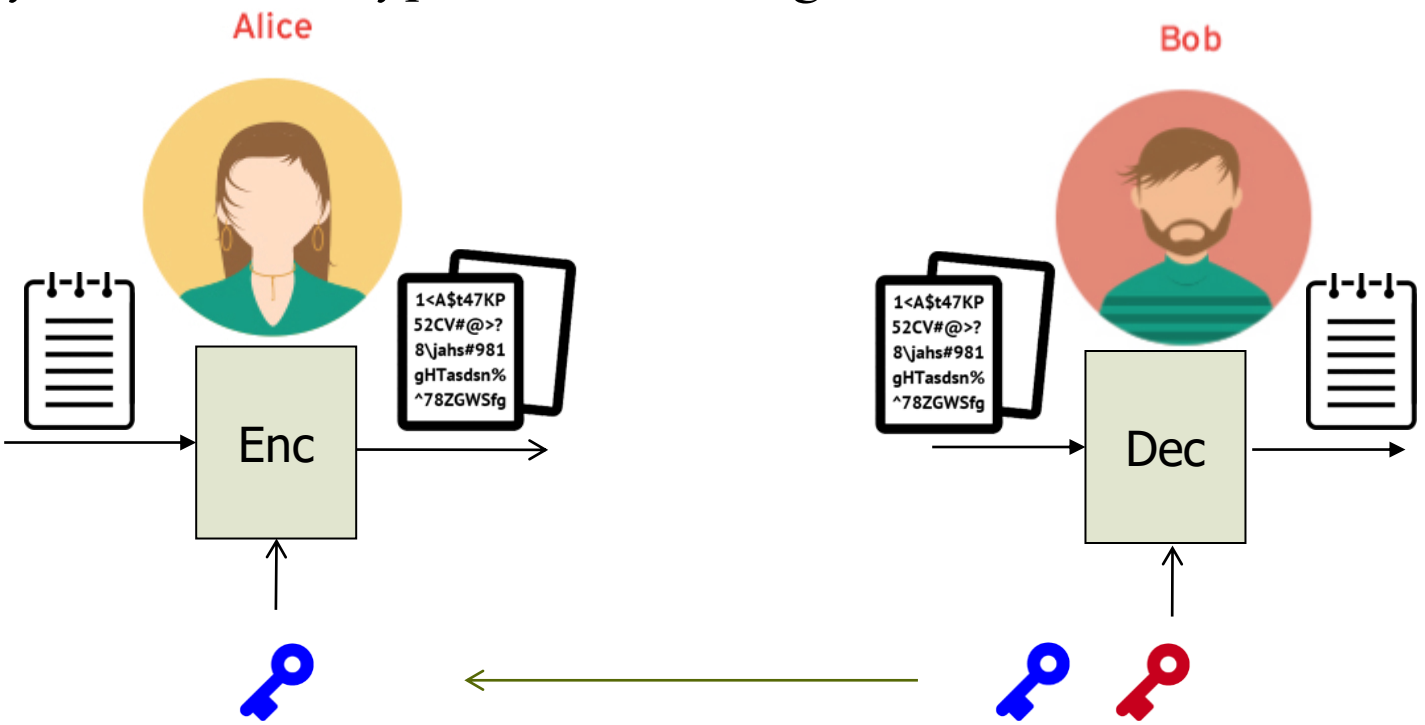
- Also has three main algorithms
  - Key generation
  - Encryption
  - Decryption
  - Plus more (later)
- Uses different keys to encrypt and decrypt ("**asymmetric**" crypto)
  - Anyone can encrypt a message with the public key
  - Only the owner of the private key can decrypt
- Slow, hard to accelerate, good for only small amounts of data
  - But, easy to distribute public keys (on a blockchain, it's simply your wallet address!)
- Examples:
  - RSA
  - ECDSA

Figure definitions

- Public key 
- Private key (kept secret) 
- Plaintext 
- Ciphertext 

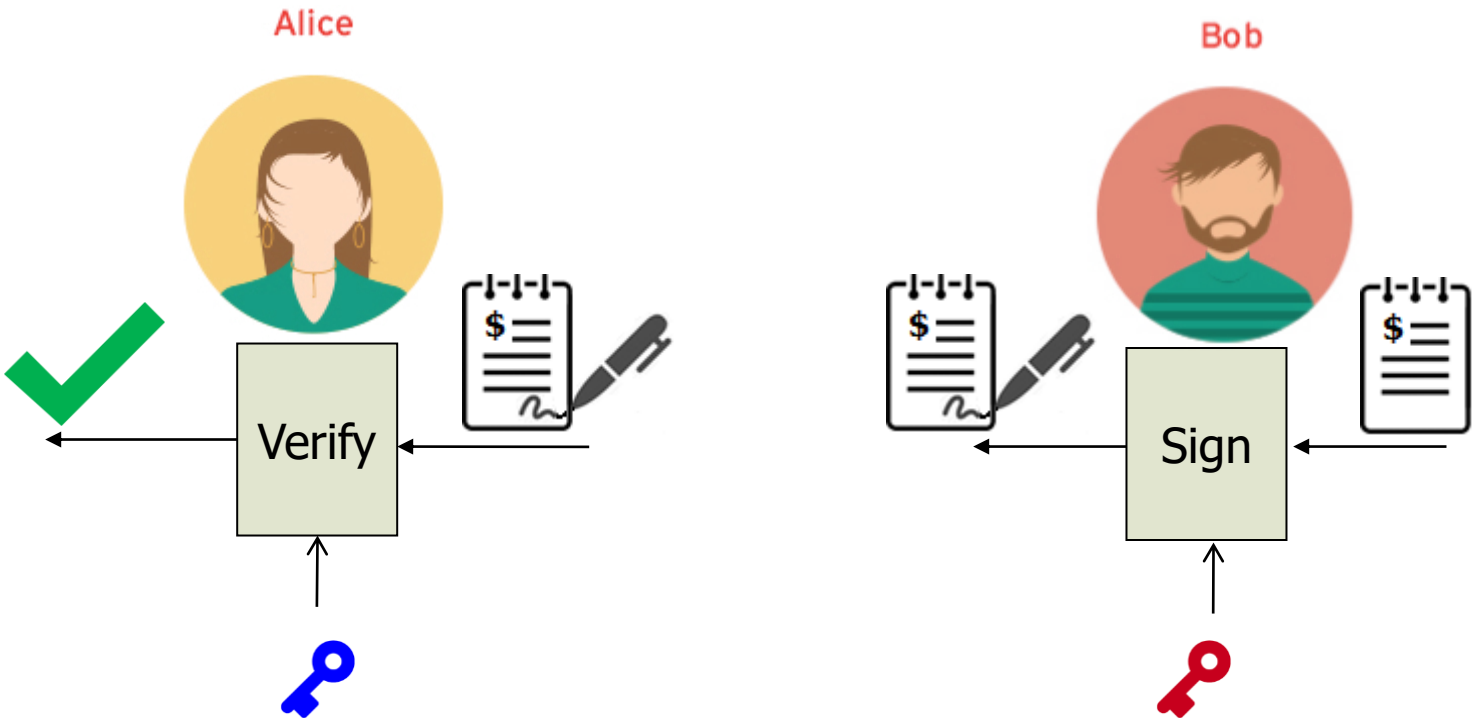
Asymmetric encryption

- Bob uses key generation algorithm to generate keys
  - Bob's public key
  - Bob's private key
- Bob publishes
- Alice encrypts her message with and sends it to Bob
- Only Bob can decrypt Alice's message with



Digital signatures

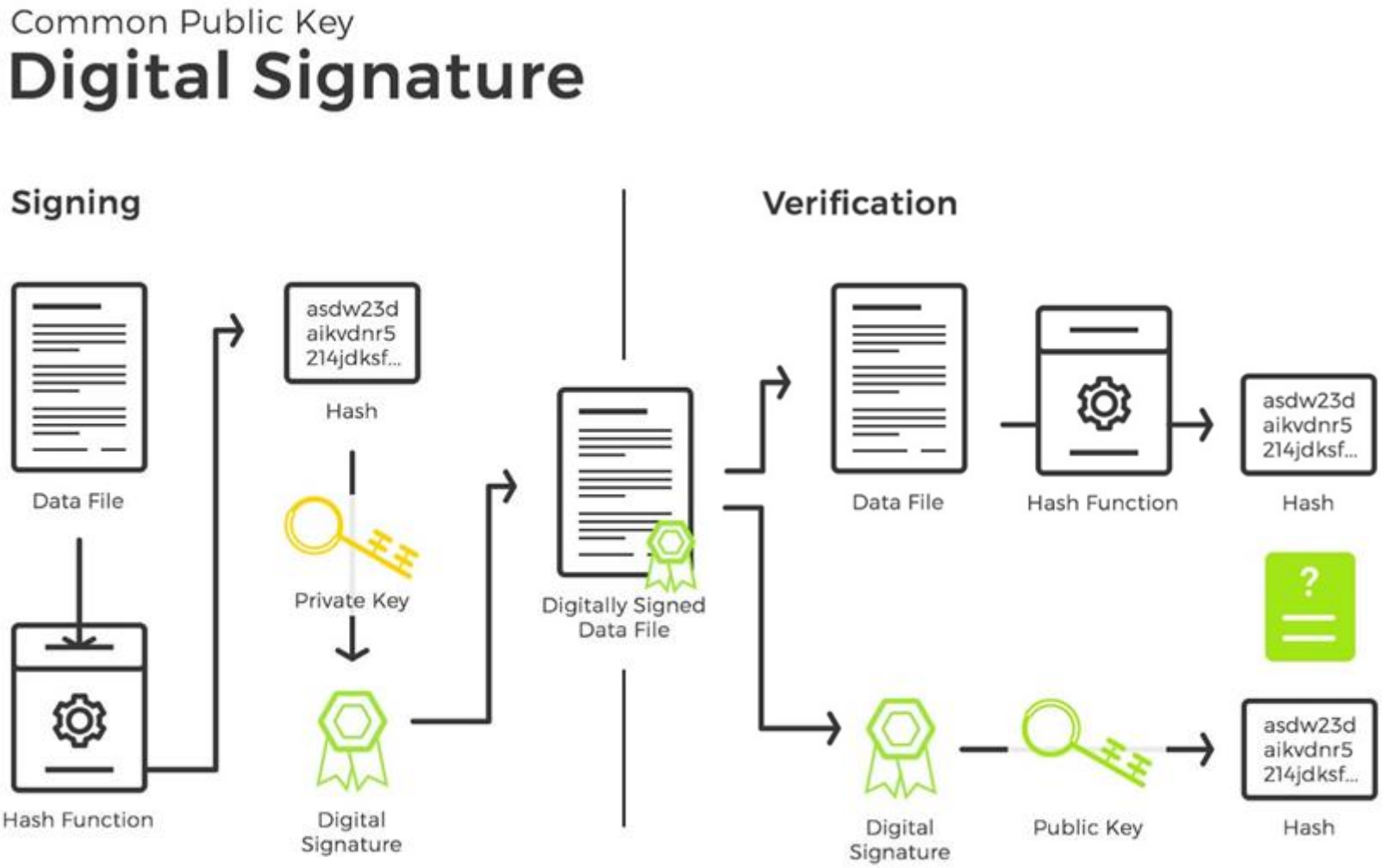
- Public-key also supports digital signing and verification algorithms
  - Used to generate signatures to authenticate data (non-repudiation)
- Bob with a message to withdraw $1 from Bank of Alice
  - Bob signs message using private key
  - Sends message with signature to Alice
  - Alice uses Bob's public key to verify only Bob could have signed message
  - Debits Bob's account $1 and sends him $1



Digital signatures in practice

- Wallet addresses ==  that nodes use to validate signatures
- Q: Where are places that digital signatures are used in practice?
  - Certificate authority store
  - Web site certificates
  - Software signing keys (apt, Windows updates)

Typically, hash of message signed due to performance issues



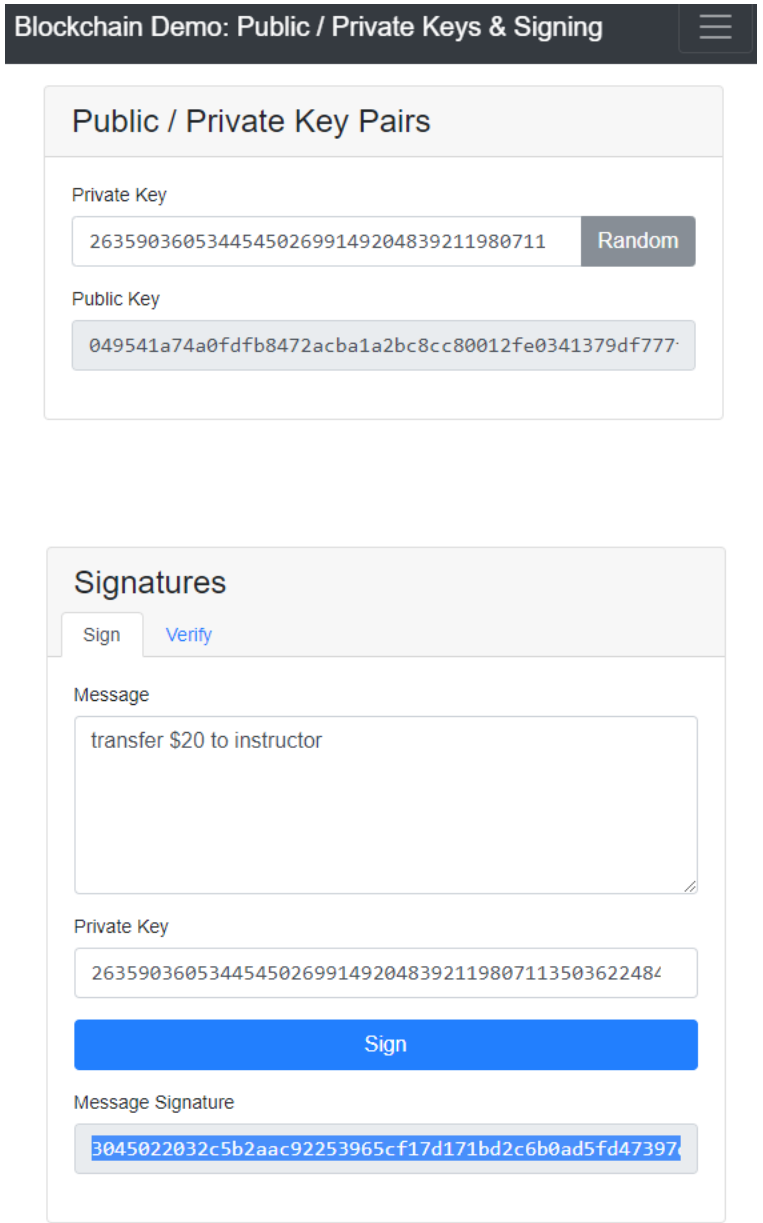
Demo (play along)

- Go to https://bc.oregonctf.org/keys
  - Set private-key, public-key pair
  - Keep tab open for subsequent demos

- Go to https://bc.oregonctf.org/signatures
  - Use private key to sign message `"transfer $20 to instructor"`

- Copy signature
- Go to "Verify" tab (`/signatures#verify`)
  - Paste signature and Verify
- Modify message to transfer $200
  - Verify again

- Every signature of every transaction in a block must be validated in this manner (see next demo)



Demo (play along)

- Visit https://bc.oregonctf.org/keys to see key pair
- Visit https://bc.oregonctf.org/transaction
  - See the public key (e.g. wallet address) used in transfer "From:" field
  - See the private key used to sign transaction
  - Use the private key to sign the transfer for the "From:" address
- Copy signature

- Go to "Verify" tab (`/transaction#verify`)
  - Paste signature and Verify
- Modify amount
  - Verify again

Private key

- *Must* be generated securely
- What happens if the generation code is faulty?
  - Guess the private key easily and grab all the ETH
  - Sneaky thieves "ethercombing" (4/2019)

    - "*The thieves seemed to have a vast, pre-generated list of keys, and were scanning them with inhuman, automated speed.*"
- Or what if generation code is maliciously written?
  - Get the private key as it's being generated!
    - Phishing sites for key generation
    - Spell-checker used on key generation step

- *Must* be kept accessible
- What happens if you lose yours?

Ah Sh*t, I lost my Ethereum Wallet

Recovery for Ethereum wallets

Eric Olszewski Follow

Aug 3, 2018 · 9 min read

  - "out of the 21 million Bitcoins that will ever exist, between 2.8–4 million (14–20% of the total supply) have already been lost."

WORLD

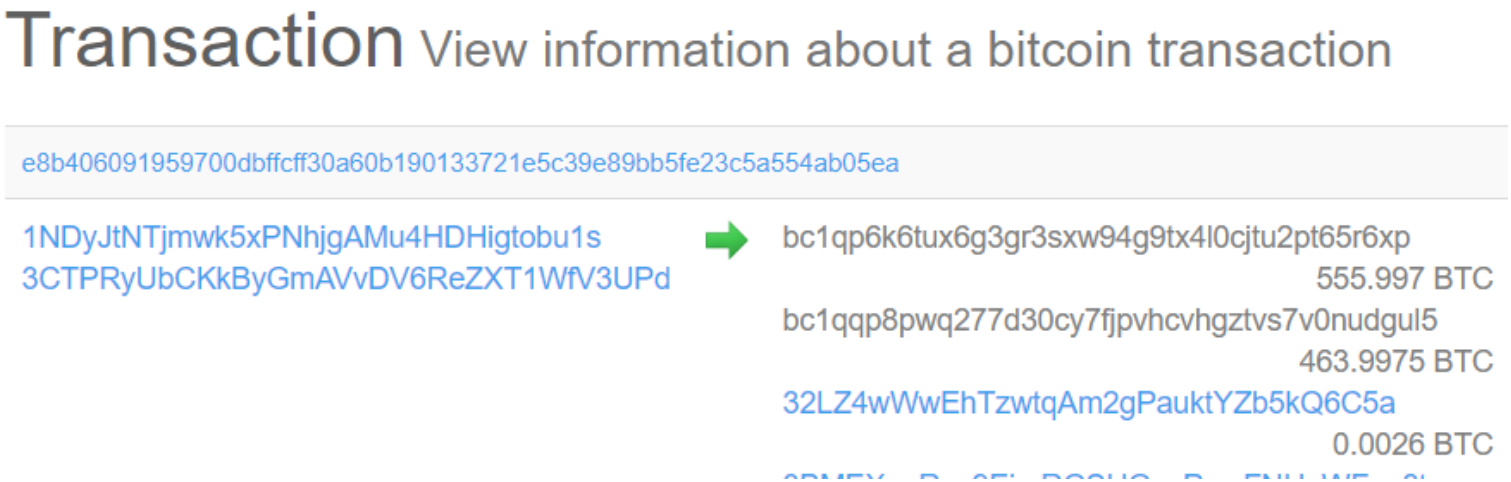
MAN ACCIDENTALLY THREW BITCOIN WORTH $108 MILLION IN THE TRASH, SAYS THERE'S 'NO POINT CRYING ABOUT IT'

BY ANTHONY CUTHBERTSON ON 11/30/17 AT 12:25 PM

  - Motivates cold-wallets stored in bank safe deposit boxes

- *Must* be kept secret
- What happens if you get yours stolen?
  - Binance $40 million loss (5/2019)

  - https://www.blockchain.com/btc/tx/e8b406091959700dbffcff30a60b190133721e5c39e89bb5fe23c5a554ab05ea



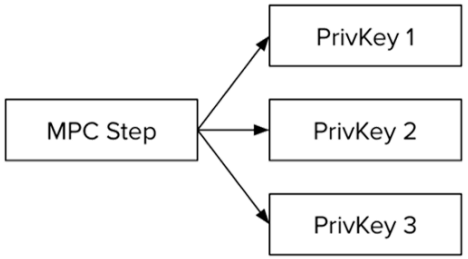
  - Do we really want the ledger to be immutable and reward this behavior?
  - What would it take to roll back? (more later)

Multisignature schemes

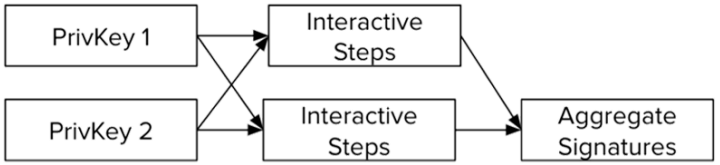
- Problem
  - Compromise of a single set of private-keys can cost you all of your $
- Multisignatures
  - Require *m-of-n* signers to authorize a transaction
  - Loss of a private-key or an adversary compromising a private-key doesn't allow for funds to be lost
  - Used to manage larger amounts of cryptocurrency balances
- Can be done with code and single signatures
  - Examples: BTC's P2SH (Pay-to-Script-Hash)
- Can be done with cryptographic schemes natively

Native multisignature schemes

- https://blockchainatberkeley.blog/alternative-signatures-schemes-14a563d9d562
- Threshold ECDSA (Keep Network, Kzen)

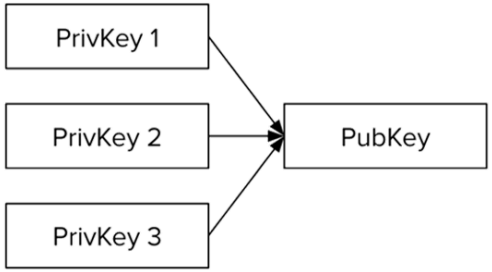


MPC Step: Several rounds of communication are necessary

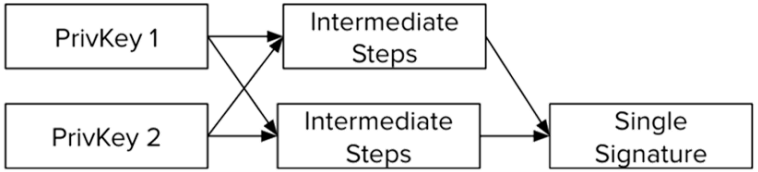


Transaction Signing: Several rounds of communication are necessary.

- Threshold Ed25519 (Kzen), Schnorr (Bitcoin)



Public Key Generation: Here we derive a single public key from the existing set of private keys



Transaction Signing: Fairly interactive signing process required. Various implementations have different amount of interaction require

Cryptographic hash functions (Immutability)

Cryptographic hash functions

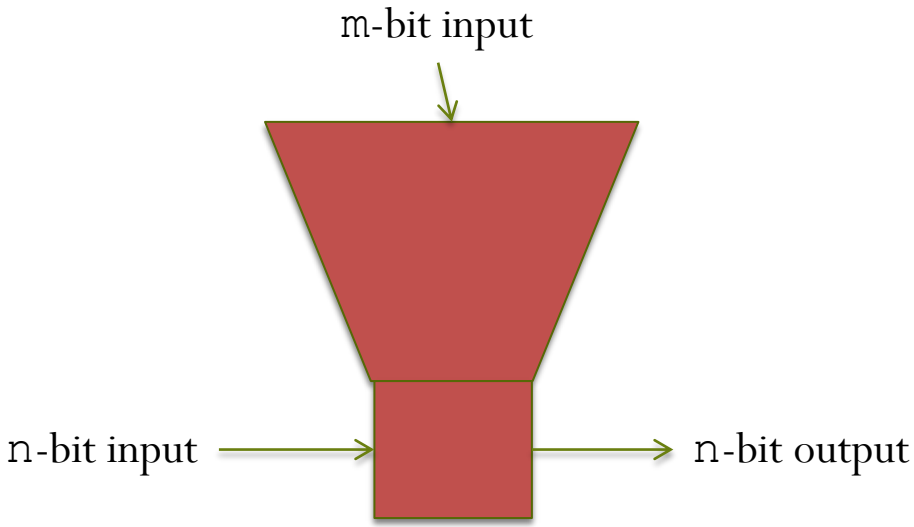
- One way functions that take arbitrary-sized input and generates a random-looking, fixed-length output

- Hash function `H`, Input `x`, hash function output `h`

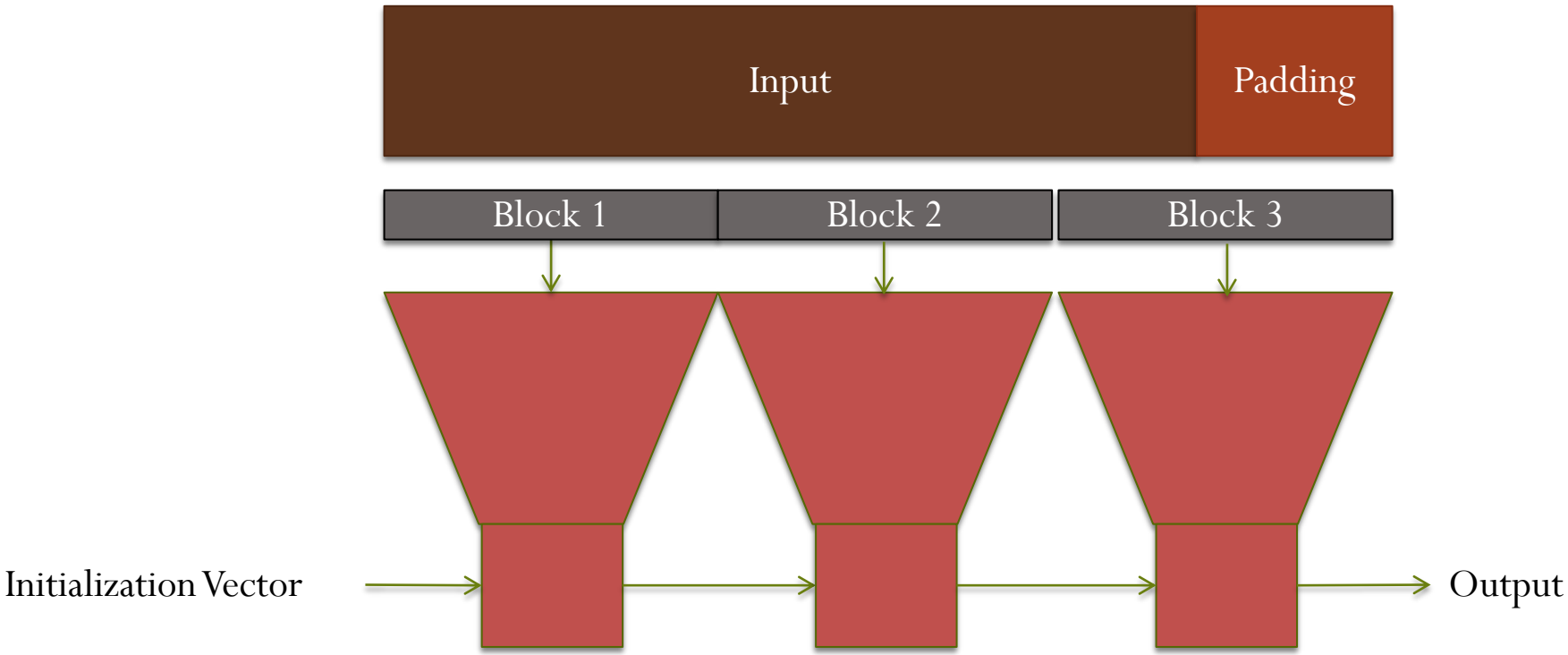
  `H(x)=h`

Merkle-Damgard Hash Construction

- Repeated use of a "compression function"
  - Maps `m` bits of input to `n` bits of output (`m` > `n`)



Merkle-Damgard Hash Construction



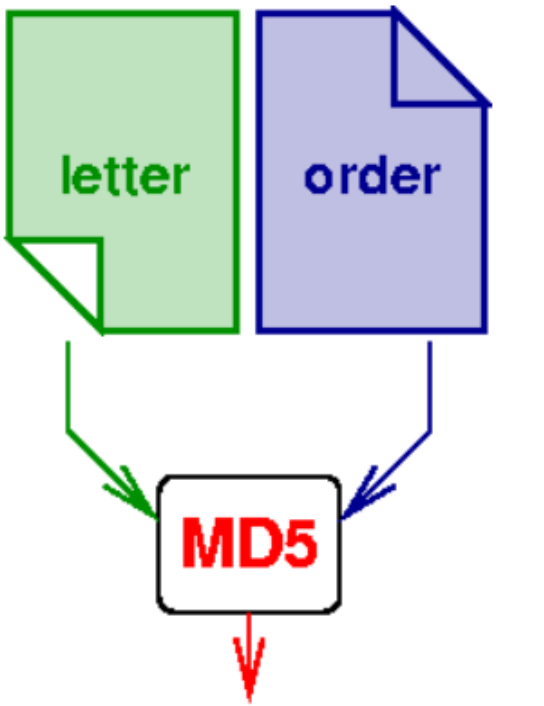
Cryptographic hash functions

- Desired properties
  - **Deterministic**: For the same input, you will always get the same output
  - **Efficient**: Quickly computed
  - **Preimage resistance** – Infeasible to determine input from output (e.g. for a given `h`, it is hard to find `x`)
  - **Second preimage resistance** (basis for immutability in blocks) – for a given input `x1`, it is hard to find a different input `x2` such that `H(x1)=H(x2)`
  - **Collision resistance** – it is hard to find any pair `x1, x2` such that `H(x1)=H(x2)`
  - **Avalanche effect** (basis for proof-of-work in mining) – a 1-bit change in input `x` causes each output bit in `h` to flip with probability ½ (sometimes called a pseudo-random function)

Beware of broken schemes

- MD5 (1992) – Merkle Damgard
- Collision resistance broken since 2004
- Second pre-image resistance broken since 2010
  - Example from: https://web.archive.org/web/20100327141611/http://th.informatik.uni-mannheim.de/people/lucks/HashCollisions/

    - One to display the letter of recommendation, and
    - a second one, an order from Caesar to grant Alice some kind of a security clearance.



Other broken schemes

- SHA (1993) – Broken, don't use
- SHA-1 (1995) – Fixes SHA, but collisions have been found (2017)
  - Don't use for new projects
- Replaced by …

SHA-2

- Secure Hash Algorithm 2
  - Designed by NSA
  - Published in 2001
  - Digest size 224, 256, 384, or 512 bits
  - Current cryptanalysis: Pretty good; OK for now
- Used in Bitcoin

```
H(x) = SHA256(SHA256(x))
```

keccak

- Winner of the SHA-3 competition sponsored by NIST to replace SHA-1 and SHA-2
  - https://keccak.team/keccak.html
  - Competition started in 2007
  - Ended in 2012 (after Bitcoin deployed)
- Sponge function that generates hashes of arbitrary length
  - https://keccak.team/sponge_duplex.html
- Basis of various NIST-approved SHA-3 implementations
  - e.g. SHA3-224, SHA3-256, SHA3-384, SHA-512
- Used in Ethereum

Two uses for hashes in a blockchain

- Use #1: Ensure integrity of a block
  - Hash signature changes if data changes
  - Second pre-image resistance makes it difficult to find another input `x2` that maps to the same hash value as original input `x1`

Demo

- https://anders.com/blockchain/hash

Two uses for hashes in a blockchain

- Use #2: Mining blocks
  - Implement rate-limits
    - On number of blocks added to a blockchain (to avoid double-spending problem and to bound the size of the ledger)
    - On amount of currency (to restrict supply and reduce inflation)
      - New currency (coinbase) only issued to miners via a block reward
  - Recall definitions from last class
    - ….consistent storage system secured by economic incentive
  - Specific example
    - A valid block must come with a nonce, when combined with the block data, results in a hash with a certain number of leading 0s
    - Hash function treated as a random function!
      - Brute-force search by incrementing nonce and checking block hash
      - Probability of a bit in a hash flipping should be 50% if any bit is changed in the block!

Demo

- Manually find a nonce that produces a hash with one leading 0, given data "mine me"
  - https://anders.com/blockchain/block
- Change the nonce without clicking on "Mine"
- What is the smallest nonce that gives you a leading 0?
- Questions
  - How many hashes on average would it take to find one with 2 leading 0s?
  - How many hashes on average would it take to find one with 4 leading 0s?
- Use the "Mine" button to find one with 4 leading 0s
  - Repeat on multiple distinct blocks to validate estimate

Exercise: Mining

- Visit https://anders.com/blockchain/block
  - Set Block # = 20191002 (Today's date)
  - Set Data = "Blockchain" (without the quotes)
  - Repeatedly change the nonce and "Mine" to try to find a nonce that results in a hash which starts with 5 zeros
  - Example
    - Nonce = 2023497392383
    - Hash = 000006cefee87….

Exercise: Current BTC work function

- Visit https://blockchain.com/explorer
- Find the current number of leading 0s that a successfully mined block must have
  - Tuned to a 10 minute block time with current hardware (mostly run in China)

What is stored in the block?

- Currency transactions
  - Bitcoin transfers from one address to another (Shared Ledger)
- Program execution state transitions
  - Ethereum Virtual Machine (Shared State Machine)
  - Smart contracts running live, long-running programs
- Asset ownership
- Data itself (e.g. documents, images)
  - Expensive!
- Hashes of data
  - Factom, Bitcoin commitments to documents stored off-chain
  - Stamp.io
    - Place document hashes onto blockchain
    - Produce actual content to prove ownership if required
    - https://youtu.be/GkmHnc-5OyY

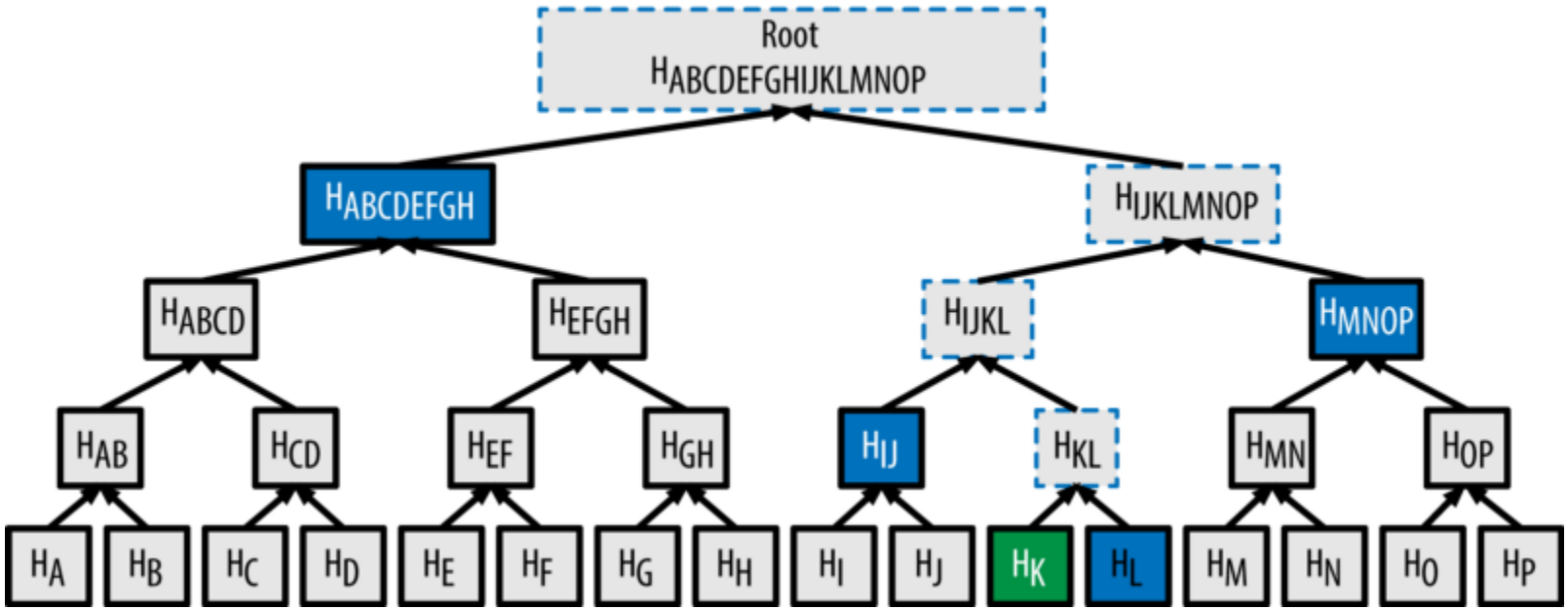
But ...

- Want to validate a single transaction in a block with thousands of transactions
  - Must go through all transactions to generate blockhash
  - Slow if validation extended to many transactions (Bitcoin blockchain currently > 200GB of data)
  - Motivates different techniques to improve performance

Merkle Tree

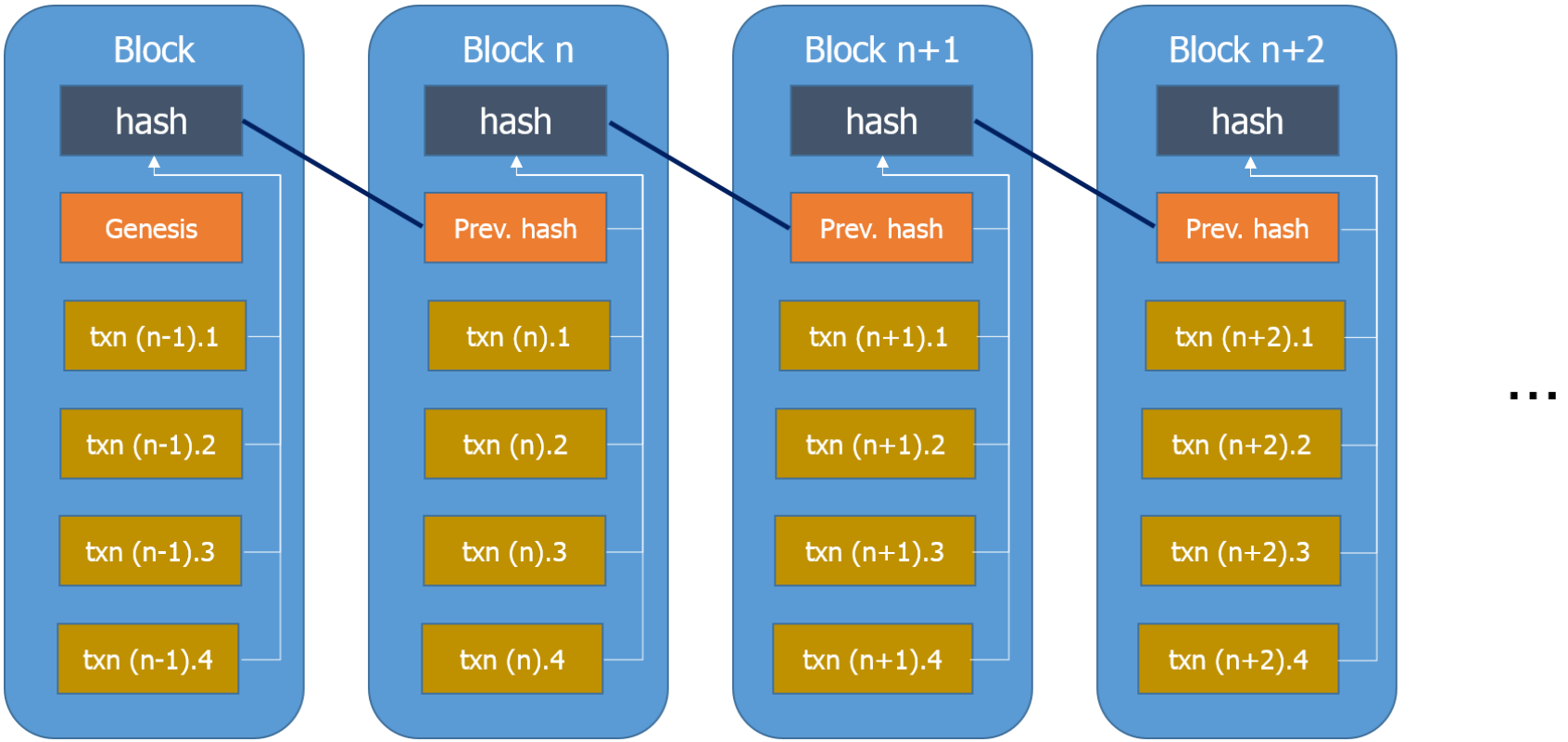
- Immutability of transactions within block
- Tree of hashes to verify one piece of data without verifying entire log
  - Efficiently prove integrity and validity of K by checking from the root

    $H_{ABCDEFGHIJKLMNOP}$ => $H_{ABCDEFGH}$ $H_{IJKLMNOP}$
    $H_{IJKLMNOP}$ => H_{IJKL} H_{MNOP}
    H_{IJKL} => H_{IJ} H_{KL} => H_K
    H_{KL} => H_K H_L
  - Second pre-image resistance property prevents replacement of K



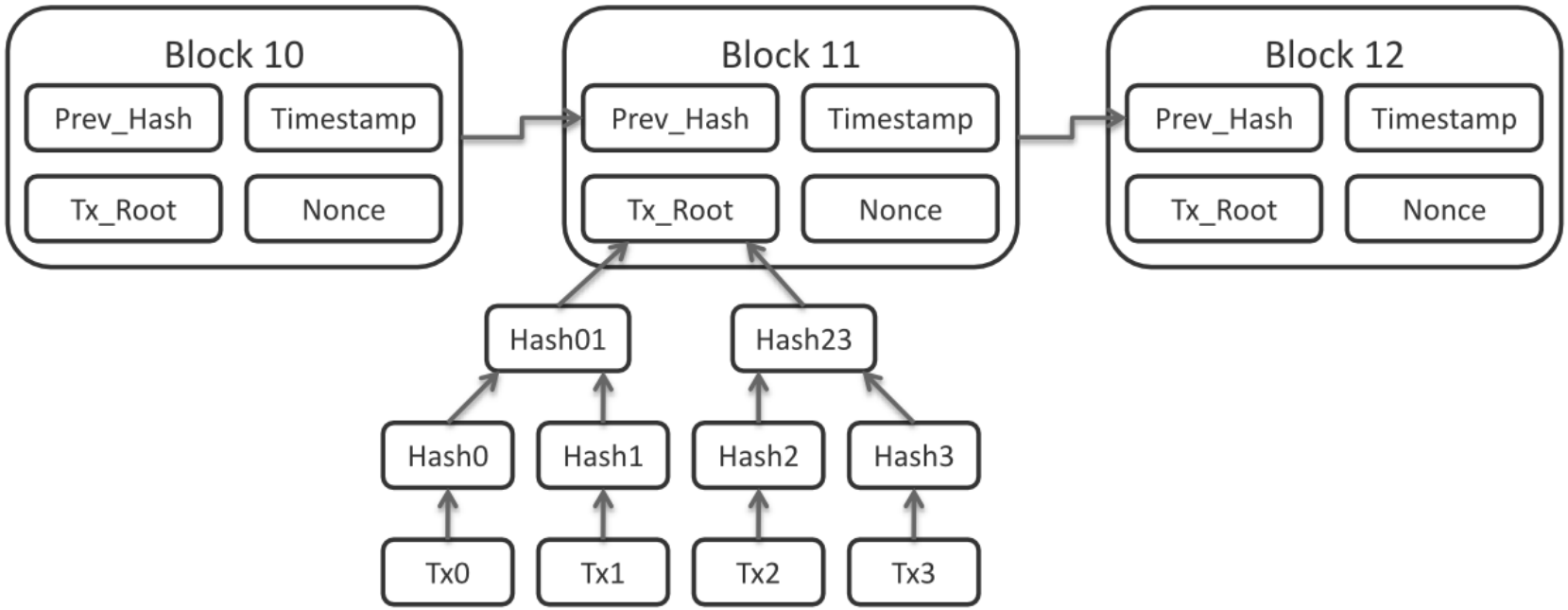
Chaining

- Immutability across blocks in "blockchain"
  - Merkle-Damgard compression construction applied at block level
  - Hash of previous block used as input to hash of the next one
    - Tampering with Block n invalidates subsequent hashes



Put together

- Merkle tree + hash chaining

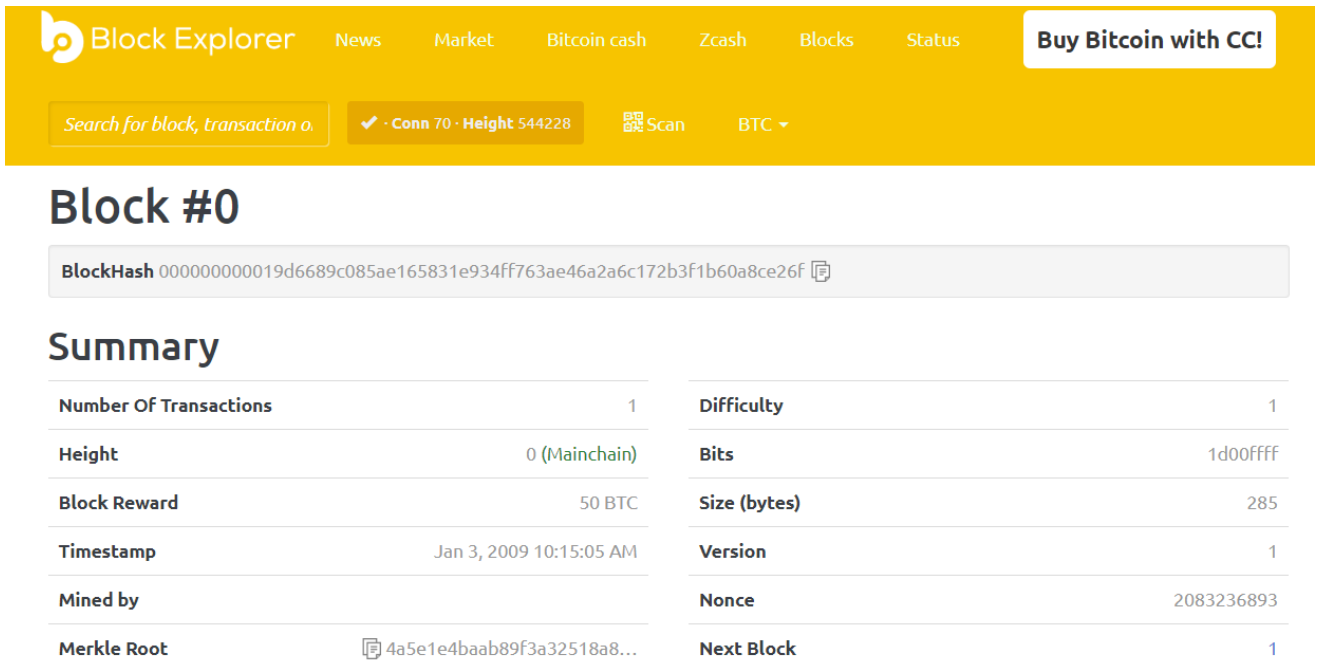


Demo

- https://anders.com/blockchain/blockchain
  - Prev hash used to bind current block to preceding block
  - Tampering with one block invalidates subsequent blocks in chain
- Adversary would need to re-mine all subsequent blocks to "modify" the ledger
- Blocks deeper in the chain are harder to tamper with

Explorers

- Rewriting history is *hard*
- Blocks effectively immutable
- Can navigate blockchain on a number of sites
  - bitcoin.info, blockexplorer.com, etherscan.io, etherchain.org



Demo: Putting things together (play along)

- https://bc.oregonctf.org/blockchain
- Modify transaction
  - Invalidates blockhash as well as the signature of the transaction
- Fix hash chain
  - Miners can mine block to fix hash chain
  - But, can not fix broken signature
- Nodes programmed to reject all blocks with invalid signatures
  - Miners would never mine a block with an invalid signature since they would get no credit for it

- Provides the basis on which trust is built