Bitcoin

Networking

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 2 July 2014
Common/Reports/bitcoin.tex, r905

Contents

Symmetric Key Cryptography

- Encryption algorithm E(), decryption algorithm D()
- Users A and B share a secret key K_{AB}
- Encrypting plaintext, P, with a key K_{AB}, produces ciphertext C, e.g. $C = \mathrm{E}(K_{AB}, P)$
- Decrypting ciphertext with the correct key will produce the original plaintext. The decrypter will be able to recognise that the plaintext is correct (and therefore the key is correct). E.g. $P = \mathrm{D}(K_{AB}, C)$
- Decrypting ciphertext using the incorrect key will *not* produce the original plaintext. The decrypter will be able to recognise that the key is wrong, i.e. the decryption will produce unrecognisable output.
- Examples: AES, DES, 3DES, RC4, ...

Public Key Cryptography

- Each user has a pair of keys, public (PU) and private (PR).
- One key from the pair is used for encryption, the other is used for decryption. Decryption will only be successful when using the other key from the keypair.
- Confidentiality: user A encrypts message M using destinations public key, e.g. $C = \mathrm{E}(PU_B, M)$
  - Only B has the private key to decrypt
- Signature: user A encrypts message M using its own private key, e.g. $C = \mathrm{E}(PR_A, M)$
  - Only A has the private key to encrypt, others can verify the message came from A
- Examples: RSA, DSA, ElGamal, Diffie-Hellman, Elliptic curve

Hash Functions

- A cryptographic hash function, H(), takes a variable sized input message, M, and produces a fixed size, small output hash, h, i.e. $h = \mathrm{H}(M)$.
- Given a hash value, h, it is impossible to find the original message M.
- Given a hash value, h, it is impossible to find another message M' that also has a hash value of h.
- It is impossible to find two messages, M and M', that have the same hash value.
- Examples: MD5, SHA1, SHA2, RIPEMD, ...

Digital Signatures

- A digital signature of a message M is the hash of that message encrypted with the signers private key, i.e. $S = \mathrm{E}(PR, \mathrm{H}(M))$
- An entity receiving a message with an attached digital signature knows that that message originated by the signer of the message.
- Examples: combine RSA, DSA with MD5, SHA1, SHA2

Random Numbers

- Pseudo-random number generators (PRNG) are deterministic algorithms that produce a sequence of effectively true random numbers
- The outputs of encryption algorithms and cryptographic hash functions are pseudo-random numbers

Attacks

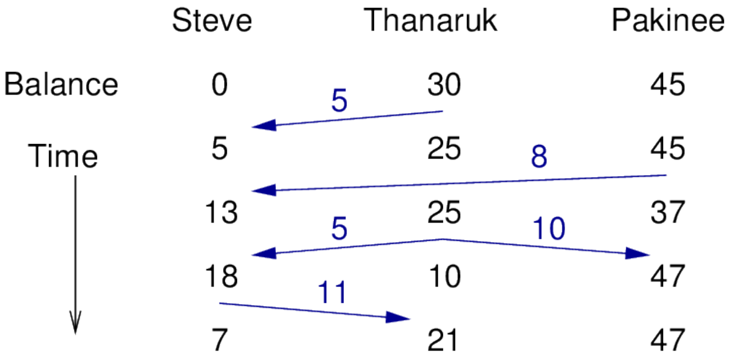
- All agorithms used in cryptography, e.g. encryption/decryption algorithms, hash functions, are public.
- An attacker knows which algorithm is being used, and any public parameters of the algorithm.
- An attacker can intercept any message sent across a network.
- An attacker does not know secret values (e.g. symmetric secret key K_{AB} or private key PR_A).
- Brute force attacks requiring greater than 2^{80} operations are impossible.

Contents

Normal Banking

- Bank keeps record of balances and transactions for all accounts
- Transaction: input account, amount, output account
- Banks are a centralized system
- Can we implement as a decentralized system?

Example Transactions and Balances



Example Viewed as List of Transactions

Opening balances: Steve=0; Thanaruk=30; Pakinee=45

TXN1. in: Thanaruk; out: Steve, 5

TXN2. in: Pakinee; out: Steve, 8

TXN3. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10

TXN4. in: Steve; out: Thanaruk, 11

Special Transactions: Creating Money

All users start with 0

TXN1. out: Thanaruk, 30 (coinbase transaction)

TXN2. out: Pakinee, 45 (coinbase transaction)

TXN3. in: Thanaruk; out: Steve, 5

TXN4. in: Pakinee; out: Steve, 8

TXN5. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10

TXN6. in: Steve; out: Thanaruk, 11

Decentralized Transaction Log

- Instead of the bank storing the transaction log, it is distributed to all users
  - Users submit transaction to the network of all users
  - Network verifies transaction (with respect to previous transactions in log)
  - Network adds verified transactions to the log
- How to ensure all users agree upon same transaction log, i.e. reach consensus?
- Can users change past transactions in the log?
- Can users double-spend?

Verifying New Transactions

All users start with 0

TXN1. out: Thanaruk, 30

TXN2. out: Pakinee, 45

TXN3. in: Thanaruk; out: Steve, 5

TXN4. in: Pakinee; out: Steve, 8

TXN5. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10

TXN6. in: Steve; out: Thanaruk, 11

New transaction submitted:

TXN7. in: Pakinee; out: Thanaruk, 15

Network accepts TXN7

Verifying New Transactions

All users start with 0

TXN1. out: Thanaruk, 30

TXN2. out: Pakinee, 45

TXN3. in: Thanaruk; out: Steve, 5

TXN4. in: Pakinee; out: Steve, 8

TXN5. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10

TXN6. in: Steve; out: Thanaruk, 11

TXN7. in: Pakinee; out: Thanaruk, 15

New transaction submitted:

TXN8. in: Steve; out: Thanaruk, 10

Network rejects TXN8

How To Prevent Users Changing Past Transactions?

All users start with 0

TXN1. out: Thanaruk, 30

TXN2. out: Pakinee, 45

TXN3. in: Thanaruk; out: Steve, 5

TXN4. in: Pakinee; out: Steve, 8

TXN5. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10

TXN6. in: Steve; out: Thanaruk, ~~11~~ 8

TXN7. in: Pakinee; out: Thanaruk, 15

New transaction submitted:

TXN8. in: Steve; out: Thanaruk, 10

Network accepts TXN8

How To Prevent Users Changing Past Transactions?

Ensure verifying transactions requires significant effort and transactions depend on previous transactions

- Modifying a previously verified transactions requires the attacker to:
  1. Re-verify that modified transactions AND
  2. Re-verify all transactions since that modified transaction

Contents

Bitcoin

- Bitcoin is a payment system, cryptocurrency, digital currency, . . .
- Unit of currency/accounts is bitcoin or BTC
- Payments are recorded as transactions (TXN)
- Public ledger records transactions
- New submitted transactions are verified by users before being added to public ledger
- Users are identified using public keys

Public Keys and Hashes

- Each user generates one or more public/private keypairs (private key: 256 bits; public key: 512 bits $++$)
  - Steve: (PU_S, PR_S)
  - Thanaruk: (PU_T, PR_T)
  - Pakinee: $(PU_{P1}, PR_{P2}); (PU_{P2}, PR_{P2}); \ldots$
- Public keys are hashed (using SHA256 and RIPEMD160) to produce 160 bit values
- Public key hash is Base58 encoded to produce Bitcoin address

Transactions

- Inputs: identify where the BTC came from
  - txnid and output index of previous transactions
  - Senders public key
  - Transaction data signed with senders private key
- Outputs: identify where the BTC is going to
  - amount and public key hash of recipient
- txnid is hash of transaction information (inputs, outputs)
- Transactions must spend all of input
  - Any left over is considered transaction fee
  - Change can be sent to yourself
- Unspent Transaction Output (UTXO) is considered current balance

Example Transactions

All users start with 0

TXN1. out1: Thanaruk, 30

TXN2. out1: Pakinee, 45

TXN3. in: Thanaruk; out1: Steve, 5; out2: Thanaruk, 25

TXN4. in: Pakinee; out1: Steve, 8; out2: Pakinee, 37

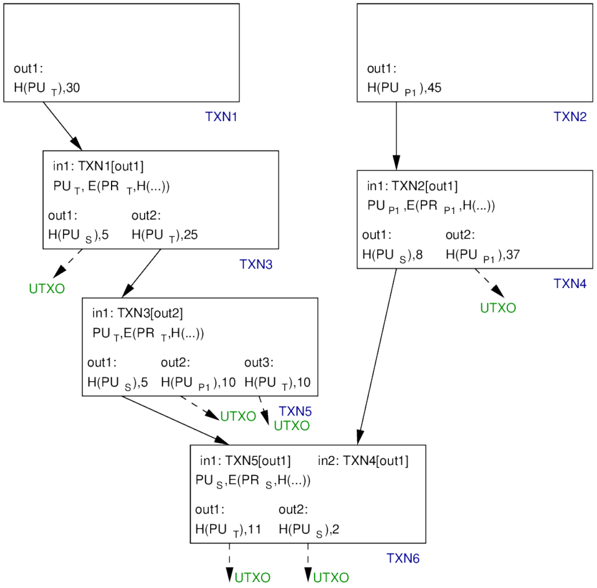
TXN5. in: Thanaruk; out1: Steve, 5; out2: Pakinee, 10; out3: Thanaruk, 10

TXN6. in: Steve; out1: Thanaruk, 11; out2: Steve, 7

Transaction TXN3

- in1: TXN1[out1]
- out1: $H(P_S)$, 5
- out2: $H(P_T)$, 25
- full public key (in1): PU_T
- signature (in1): $\mathrm{E}(PR_T, H(transdata))$

Example Bitcoin Transactions



Verifying Transactions

- Sender broadcasts transaction to Bitcoin network
- Others in network validate the transaction:
  - Hash of include public key matches hash in output of previous transaction
  - Signature is correct
- To add new transactions to public ledger, transactions are included in blocks

Blocks and Block Chains

- One or more new transactions are grouped into a block, which also includes header:
  - previous block ID
  - transaction IDs
  - root of Merkle tree
  - difficulty factor
  - counter
- Block ID is created from SHA256 hash of block header
- Each block points to previous block, forming block chain
- Block chain is the public ledger (i.e. complete log of all transactions)
- Challenge: ensuring all users agree on same block chain

How To Prevent Users Changing Past Transactions?

Ensure verifying transactions requires significant effort and transactions depend on previous transactions

- Modifying a previously verified transactions requires the attacker to:
  1. Re-verify that modified transactions AND
  2. Re-verify all transactions since that modified transaction

Proof of Work

- A user that groups transactions to create a block must proof that they did significant (computational) work in doing so
- Users that create new blocks are called miners
- What is the proof of work?
  - SHA256 hash has 2^{256} possible values
  - Difficulty factor (DF) is known in network (currently about 2^{34})
  - Block ID (SHA256 hash of block header) must be:

$$blockid \leq \frac{2^{256}}{DF \times 2^{32}}$$

  - Keep changing *counter* in block header until small enough block ID is found

How Many Attempts to Find Block ID?

- Assume DF $= 2^{34}$, then $blockid \leq 2^{190}$
- Set $counter = 0$ and calculate SHA256 hash of block header
- SHA256 produces effectively random output: 2^{256} possible values
- Probability $blockid \leq 2^{190}$: $2^{190}/2^{256} = 2^{-66}$
- Set $counter = 1$ and try again ...
- On average, need $2^{66} = 7 \times 10^{19}$ attempts to find $blockid \leq 2^{190}$
- How fast is current hardware in calculating SHA256 hash?
  - About 1×10^{17} hashes per second
  - Require 7×10^{2} seconds or 11 minutes

Difficulty Factor

- Bitcoin designed so takes approx. 10 minutes to mine each block
- Duration depends on: difficulty factor and hardware speed
- Every 2016 blocks (2 weeks), difficulty factor is automatically adjusted based on time taken to mine those blocks
- As hardware capabilities increases, difficulty factor increases (and vice versa)

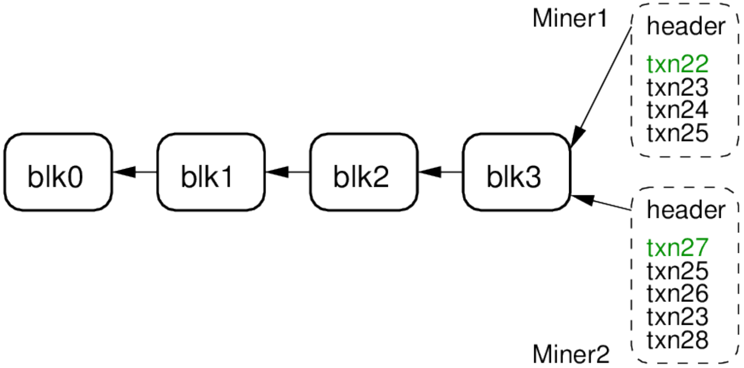
Block Mining

- Individuals and (more commonly) groups of users compete in mining blocks (why?)
- When a new block is mined, the user broadcasts to the network
- Two or more users may mine (and broadcast) the next block at same time
- Which block is added to the block chain?
- Users normally select first block they receive
- When a longer fork is created, it is selected and other fork transactions are orphaned and need to be re-verified

Why Mine Blocks?

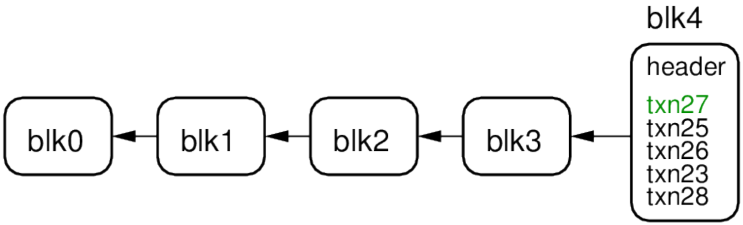
- Mining blocks to add to block chain costs money (hardware, electricity)
- Why mine blocks for others?
  1. A single coinbase transaction is included in each block of transactions
     - Coinbase transaction: no input, only one output
     - The miner is the output
     - Current reward: 25 BTC per block, halves every 210,000 blocks
  2. Some transactions include fees
     - Transaction fee = (sum of input) - (sum of outputs)
     - Miner collects the fee
     - Transaction fees motivate miners to verify the transaction

Block Chain: Mining By Two Miners



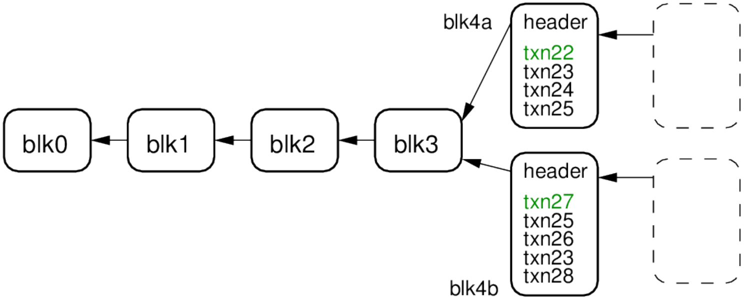
- Multiple miners are working on create next block (which points to blk3)
- Miners choose set of transactions from pool of unverified

Block Chain: Miner 1 Wins



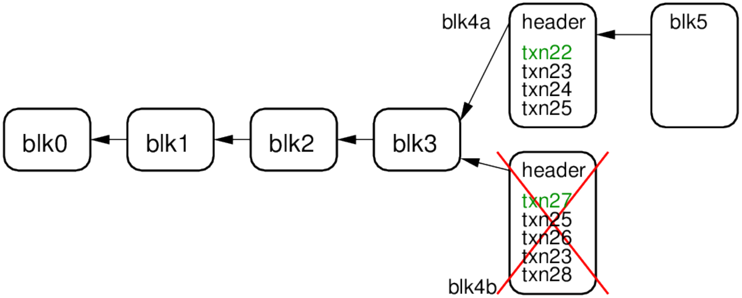
- Assuming miner 1 created block first, it is added to chain
- Miner 2 gives up, starts working on new block, that points to blk4

Block Chain: Blocks Mined at Same Time



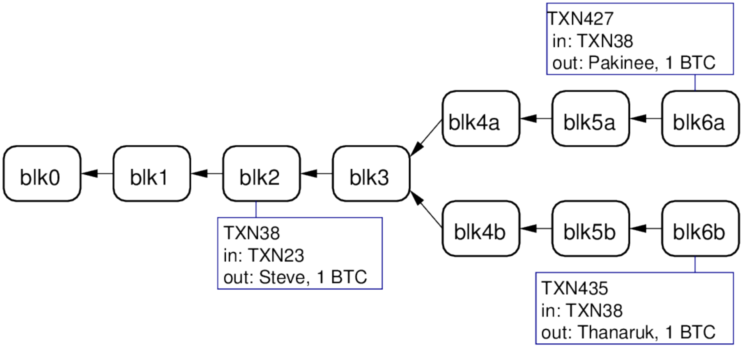
- Some times blocks are created at same time
- Some users will accept one block, other users the other block: a fork in the chain is created

Block Chain: Longest Chain is Used



- Eventually one fork will be longer than the other
- Users select the longest fork
- All transactions from the short fork and now considered unverified

Block Chain: Double Spending



- Double Spending is possible with forks
- To avoid, transactions should not be considered confirmed until multiple subsequent blocks have been mined

Contents

Confirming Bitcoins

A transaction is broadcast with you as output. When can you be confident that you have the bitcoins?

- Coinbase transactions cannot be spent for at least 100 blocks
- Normal transactions are often trusted after 6 blocks: highly unlikely that transaction will be rejected

Anonymity

- All transactions are recorded and publicly available
- But transactions identify users by (hash of) public key
- Users may use multiple public keys
- Connecting public keys to actual users may be hard

Other Bitcoin Concepts

Wallet	File (or program) that stores private key(s) and allows management of keys
Bitcoin network	P2P network of computers running Bitcoin software; supports broadcasting of transactions and blocks