Bitcoin Mining

CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

Proof of Work

- Alice wants to prove to Bob that she did certain amount of works (e.g., no. of operations or time)
  - Bob gives Alice a target
  - Alice finds a solution for this target
  - Bob verifies this solution based on this target

- Alice needs certain time to find a solution (time consuming, but still possible); With a solution and a target, it is very easy for Bob to verify.

Proof of Work

- Alice has a message m, wants to give it to Bob
- Bob tells Alice to find a nonce n s.t.
  - H(m||n) <= target
  - Otherwise he will not accept message m

- Enumerating all possible nonces is the only way to for Alice find a solution
  - Since hash function is hard to invert

Example

- Bob tells Alice to find a nonce n s.t.
  - H(m||n) <= x/2;

- Example: Given m = **uc**, Alice chooses n = **a**
  - H(**uc**||**a**) > x/2, **a** is not a solution;
- Alice tries n = **b**
  - H(**uc**||**b**) > x/2, **b** is not a solution;
- Alice tries n = **c**
  - H(**uc**||**c**) <= x/2, **c** is a solution, return

H(*)	000	001	010	011	100	101	110	111

- Difficulty: the no. of starting 0 bits in a target
- The no. of hash operations at Alice increases exponentially with difficulty

Target	Difficulty	Hashes at Alice	Hashes at Bob
x=111	0	1	1
x/2= 011	1	2	1
x/4=001	2	4	1
x/8=000	3	8	1

Proof of Work

- Given difficulty d, target t = $x/2^d = 2^{L-d}$
- For message m and nonce n,
  - Pr[H(m||n) <=t] = $1/2^d$
- Average 2^d hashes to find a solution

- Example: difficulty d = 8, $x = 2^{20}$, L = 20
  - target t = $x/2^d = 2^{20}/2^8 = 2^{12}$
    - 00000 00011 11111 11111
  - Average 2^d = 256 hashes to find solution

Proof of Work

- Given difficulty d, target t = 2^{L-d}
- On average, 2^{d} hashes to find a solution
- SHA256 on a laptop: 200,000 hashes/second

Difficulty	No. of Hashes	Time
20	1,048,576	5.2 secs
22	4,194,304	21 secs
24	16,777,216	84 secs
26	67,108,864	?? secs
28	268,435,456	?? secs

Proof of Work

- Given difficulty d, target t = 2^{L-d}
- On average, 2^{d} hashes to find a solution
- SHA256 on a laptop: 200,000 hashes/second

Difficulty	No. of Hashes	Time
20	1,048,576	5.2 secs
22	4,194,304	21 secs
24	16,777,216	84 secs
26	67,108,864	335 secs
28	268,435,456	1,342 secs

Proof of Work

- Given difficulty d, target t = 2^{L-d}
- On average, 2^{d} hashes to find a solution
- Practice: increase to 400,000 hashes/second

Difficulty	No. of Hashes	Time (200,000)	Time (400,000)
20	1,048,576	5.2 secs	?? secs
22	4,194,304	21 secs	?? secs

Proof of Work

- Given difficulty d, target t = 2^{L-d}
- On average, 2^{d} hashes to find a solution
- Practice: increase to 400,000 hashes/second

Difficulty	No. of Hashes	Time (200,000)	Time (400,000)
20	1,048,576	5.2 secs	2.6 secs
22	4,194,304	21 secs	10.5 secs

Applications of PoW

- Defer Denial of Service Attacks:
  - Limit the no. of service requests that an attacker can submit

- Bitcoin Mining:
  - Decide which miner will add the next block to the current blockchain

DDoS Attack

- Basic setting: a server can only take a certain number of requests in a short time
  - Due to limited memory, bandwidth, etc.

- Denial-of-Service (DDoS) Attack
  - Attacker sends a large no. of legitimate service requests to a server in a short time
  - A server cannot handle those requests in a short time, server goes down…

Example of DDoS Attack

- Example: If UC server receives more than 30 requests within 1 min from a user/attacker, then UC server will go down

- Legitimate user Alice submits 5 requests every min
- Attacker submits 40 requests within 1 min

- UC server goes down due to this DDoS attack

Defer DDoS with PoW

- Use PoW to defer Denial-of-Service (DDoS)
  - Server uses a counter c to record the number of request that a user/attacker has submitted
  - Server uses counter c as difficulty d
  - Server sets target as $t = 2^{L-c}$
  - For each new request r, a user/attacker is asked to submit a solution s such that
    - E.g., $H(r||s) <= t = 2^{L-c}$
  - If solution is invalid, then server ignores request r

Defer DDoS with PoW

- Use PoW to defer Denial-of-Service (DDoS)
  - Server uses counter c as difficulty d of PoW

- No. of hashes (i.e., time) to find a solution for the next request is 2^c, increases exponentially with no. of requests
  - Attacker cannot submit a large number of requests within a short time.
  - Only minor impacts to legitimate users

- Example: UC server goes down if receives >30 requests within 1 min from one user/attacker
- User/attacker needs to compute 2^c hashes for each new request, c is counter

- Minor impact on Alice (200,000 hashes/sec)
  - 1st request, no hash, no extra time;
  - 2nd request, 2 hashes, 1 micro sec;
  - 3rd request, 2^2 hashes, 2 micro secs;
  - 4th request, 2^3 hashes, 4 micro secs;
  - 5th request, 2^4 hashes, 8 micro secs;
- Total overhead: only 15 micro secs

- Example: UC server goes down if receives >30 requests within 1 min from one user/attacker
- User/attacker needs to compute 2^c hashes for each new request, c is counter

- Defer DDoS attack (200,000 hashes/sec)
  - First 20 requests, 2^{20} hashes in total, 5 secs;
  - 21st request, 2^{20} hashes, 5 secs;
  - 22nd request, 2^{21} hashes, 11 secs;
  - 23rd request, 2^{22} hashes, 21 secs;
  - 24th request, 2^{23} hashes, 42 secs;
- Attacker cannot submit >30 requests within 1min

DDoS Attack

- With PoW, DDoS attack may still work if
  - Attacker has enormous hashing power

- Example: UC server goes down if receives >30 requests within 1 min from one user/attacker
- User/attacker needs to compute 2^c hashes for each new request, c is counter

- DDoS attack (200,000,000,000 hashes/sec)
  - 21st request, 2^{20} hashes, 5 micro secs;
  - 22nd request, 2^{21} hashes, 11 micro secs;
  - 23rd request, 2^{22} hashes, 21 micro secs;
  - 24th request, 2^{23} hashes, 42 micro secs;
- Attacker can submit 30 requests within 1 sec, finish DDoS attack easily

DDoS Attack

- With PoW, DDoS attack may still work if
  - Attacker controls a huge number of machines

- Example:
  - Attacker controls 1,000 machines
  - Counter is still 1 for each machine
    - PoW is not effective yet
  - 1,000 requests arrive same time at UC server
  - UC server goes down again…

Bitcoin Mining

- Adding new blocks to blockchain is called mining

- There are many miners in the network, how to decide which miner generates the next block?
  - Each miner is given a same "Puzzle"
  - Each miner selects its own block from current pending trans pool
  - Who finds a solution for "Puzzle" first based on its block will add its block to blockchain

PoW in Bitcoin

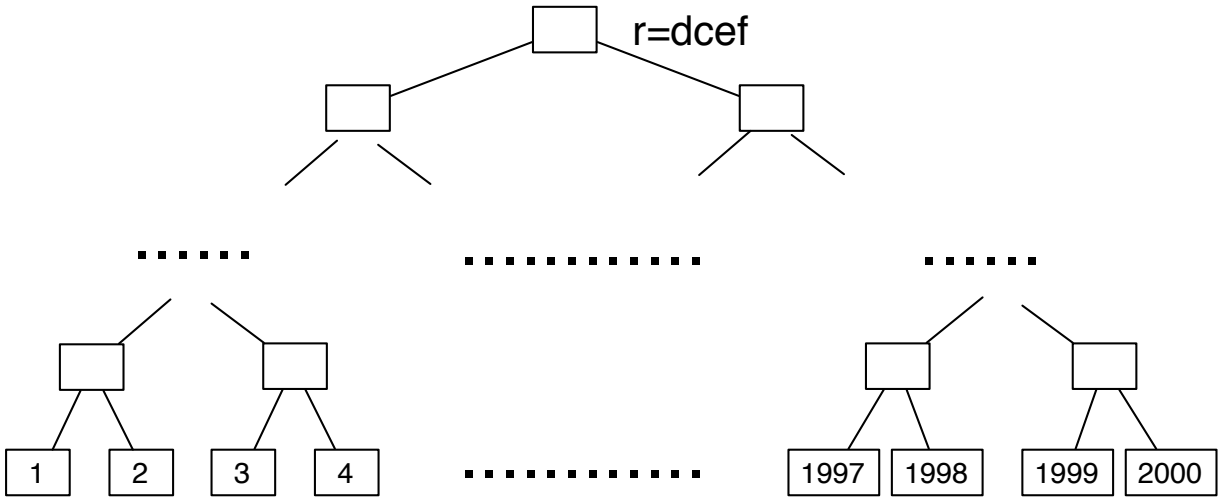
- Each miner is given a same PoW target
- Who finds a PoW solution first based on its block will add its block to blockchain.
- This PoW solution can be verified by other miners, once verified, this block is added to the blockchain, everyone will start to find a solution for the next block.

- Example: Alice and Bob are two miners
- There are 10,000 pending trans in the network
  - Alice has 8,000 pending trans locally
  - Bob has 9,000 pending trans locally
  - Some trans are not reached Alice or Bob yet

- Block No. 12345 is just added to blockchain, the current target is t = 00001111 for all miners
  - All miners start to create next block
  - Alice forms a block with 2,000 trans (about 1MB)
  - Bob forms a block with 1,800 trans (about 1MB)
  - Alice's block and Bob's block are different

- Example: Alice and Bob are two miners
- Current block length: 12345,
- Blockchain hash value: h=0xabcd
- Current PoW target: t = 00001111

  - Alice forms a block with 2,000 trans (about 1MB)
  - Alice builds a Merkle tree based on those 2,000 trans, computes a root hash r=0xdcef



- Example: Alice and Bob are two miners
- Current block length: 12345,
- Blockchain hash value: h=0xabcd
- Current PoW target: t = 00001111

  - Given h=0xabcd, r=0xdcef, and target t, Alice starts to find a solution s, s.t.

$$H(h \,||\, r \,||\, s) <= t$$

  - If Alice finds a solution before others, she broadcasts her block with (r, s).

- Example: Alice and Bob are two miners
- Current block length: 12345,
- Blockchain hash value: h=0xabcd
- Current PoW target: t = 00001111

  - Others verify (r, s), with current (h, t), if this (r, s) passes, this block with the 2,000 trans selected by Alice will be added as block No. 12346.

  - Update hash value as h = H(h || r || s)
  - Block No. 12346 & new h are broadcasted
  - All miners start to build next block

- Example: Alice and Bob are two miners
- Current block length: 12345,
- Blockchain hash value: h=0xabcd
- Current PoW target: t = 00001111

  - Alice and Bob are both finding a PoW solution
  - In PoW, enumerating all possible nonces is the only way to find a solution

  - Who finds a solution first depends hashing power
    - Alice 200 hashes/sec; Bob 100 hashes/sec
    - In theory, Alice will find a solution first

- Example: Alice and Bob are two miners
- Current PoW target: t = 00001111; difficulty is 4
- Average 2^4 = 16 hashes to find a solution
- Alice 200 hashes/sec; Bob 100 hashes/sec

  - If Alice & Bob choose exactly the same block
    - same set of trans, same order of trans
  - Alice & Bob have exactly same PoW problem
  - If Alice & Bob use same approach to find solution
    - Same initial nonce, same enumeration (n++)
    - E.g., both need 14 hashes to find solution
  - Then, Alice will definitely win

- Example: Alice and Bob are two miners
- Current PoW target: t = 00001111; difficulty is 4
- Average 2^4 = 16 hashes to find a solution
- Alice 200 hashes/sec; Bob 100 hashes/sec

  - Alice & Bob have different pending trans
  - Alice & Bob select 2 different blocks
    - different sets of trans, different orders of trans
  - Same target, but 2 different PoW problems
  - Different enumeration methods
    - E.g., Alice: n=n+1; Bob: n=n+2
  - Both could win; Alice has a better chance

- Example: Alice and Bob are two miners
- Current PoW target: t = 00001111; difficulty is 4
- Average 2^4 = 16 hashes to find a solution
- Alice 200 hashes/sec; Bob 100 hashes/sec

  - On average, both need 16 hashes to find solution
  - Bob could still win sometimes

Block	Alice' Hashes	Alice's Time	Bob's Hashes	Bob's Time
12345	22	0.11 s	10	0.10 s
12346	15	0.075 s	20	0.20 s
12347	11	0.055 s	17	0.17 s

- Example: Alice and Bob are two miners
- Current PoW target: t = 00001111; difficulty is 4
- Average $2^4 = 16$ hashes to find a solution
- Alice 200 hashes/sec; Bob 100 hashes/sec

  - Total hashing power in Bitcoin network
    - T = 200 + 100 = 300 hashes/sec
  - Outputs of hash func. are uniformly distributed
  - Probability that Alice can win
    - $p_A = 200/T = 2/3$
  - Probability that Bob can win
    - $p_B = 100/T = 1/3$

- Example: Alice and Bob are two miners
- Alice 200 hashes/sec; Bob 100 hashes/sec
  - Total hashing power in Bitcoin network
    - T = 200 + 100 = 300 hashes/sec
  - Probability that Alice can win
    - p_A = 200/T = 2/3 = 0.67

- Alice increases to 400 hashes/sec, Bob is the same
  - Total hashing power in Bitcoin network
    - T = 400 + 100 = 500
  - Probability that Alice can win
    - p_A = 400/T = 4/5 = 0.80

- Example: Alice and Bob are two miners
- Alice 200 hashes/sec; Bob 100 hashes/sec
  - Total hashing power in Bitcoin network
    - T = 200 + 100 = 300 hashes/sec
  - Probability that Alice can win
    - p_A = 200/T = 2/3 = 0.67

- Alice 400 hashes/sec, Bob 200 hashes/sec
  - Total hashing power in Bitcoin network
    - T = 400 + 200 = 600
  - Probability that Alice can win
    - p_A = 400/T = 2/3 = 0.67

Mining Competition

- Every miner realizes that increasing hashing power could increase its probability to win

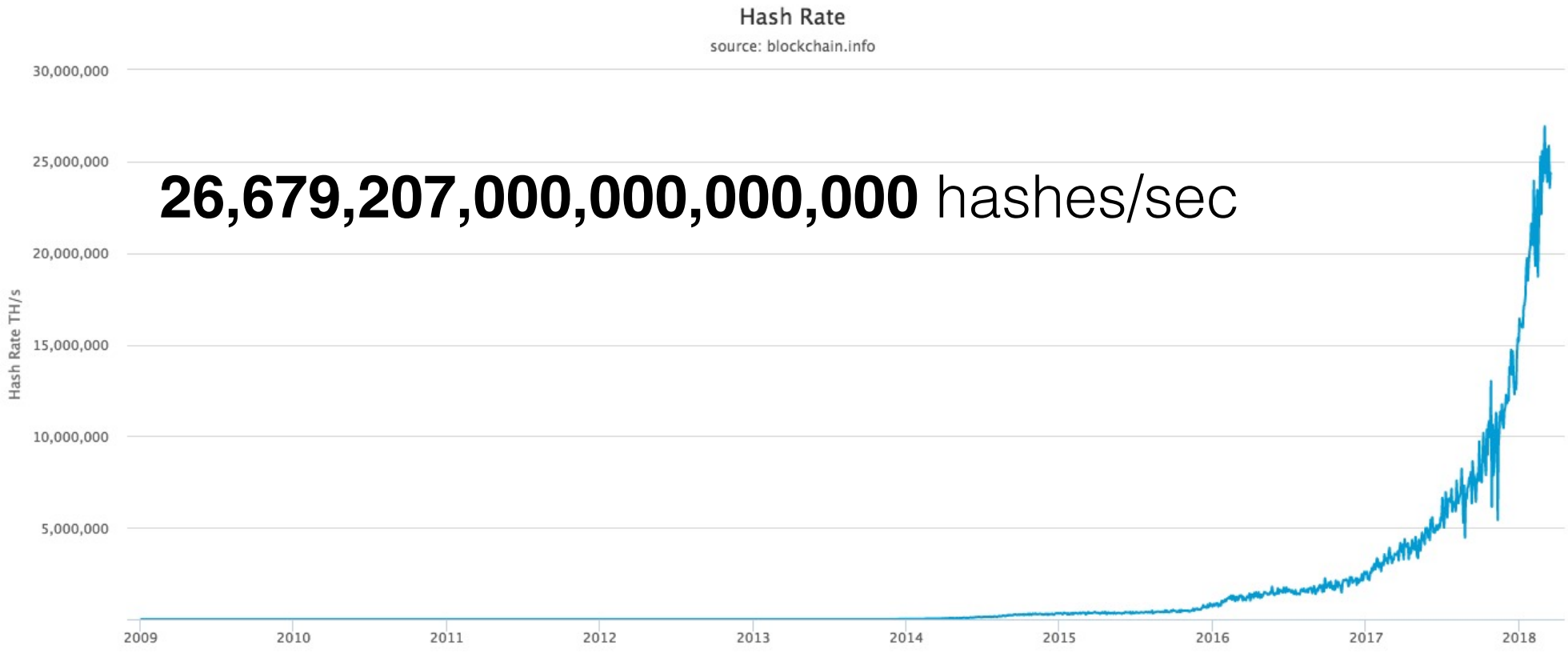
- Hashing power competition
  - CPU (2011, 2012) —> GPU (2013) —> Hardware (2015) —> Mining Station (2016, 2017)

AntMiner S7

Avalon6

Mining Competition

- All users increase hashing power
- The total hashing power increases exponentially



26,679,207,000,000,000,000 hashes/sec

Mining Competition

- If you buy a $2,000 PC to mine bitcoins yourself
  - 400,000 hashes/sec
  - About $1*10^{-15}$

- But other miners are like this…

Mining Competition

- You weapons are like this…

- Other miners' weapons are like this..

Mining Group

- New Hope: Individual miners form mining groups in order to have a better chance to win.

- Alice, Bob and Charlie form a mining group
- Alice: 400 hashes/sec; Bob: 200 hashes/sec; Charlie: 1000 hashes/sec;
- Group: 1600 hashes/sec

- A mining group may have several thousand miners

Mining Group

- Members in a group try to solve a same PoW problem with different subsets of randoms.
- If one member finds a solution, this group gets transaction fee and new bitcoins

- All members share this reward based on hashing power: Alice: 0.25; Bob 0.125; Charlie: 0.625
- David who organizes this group will get some reward as well

Mining Group