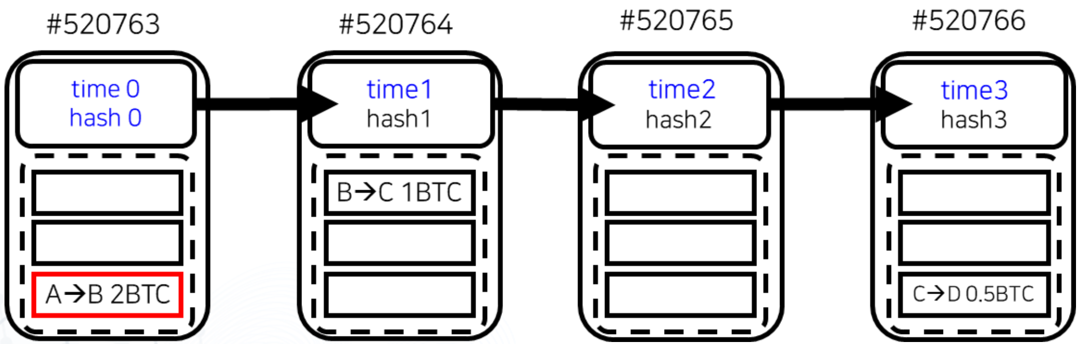
Goal of this lecture note

- Value Transfer over Internet
- Satoshi Nakamoto Cryptography Mailing List
- Brief History of Bitcoin

1 Bitcoin Enables Value Transfer over Internet!



1 Bitcoin Enables Value Transfer over Internet!

- Bitcoin is a P2P network of nodes
  - Attracts P2P nodes.
  - Has them talk to each other via Internet.
  - Has them maintain Blockchain.
  - Has each block time stamped and store transactions.
  - Has the blocks chained together with a hashing function.
  - Has a block include the hash or a summary of the past block.

1 Bitcoin Enables Value Transfer over Internet!

- Bitcoin is a P2P network of nodes
  - Puts PoW to make accidental or intentional block alteration very difficult.
  - Leaves the Blockchain open for public viewing.

1 Bitcoin Enables Value Transfer over Internet!

- Bitcoin has enabled P2P transactions over the Internet!
  - Bitcoin enables, communication of a digital message such as “A gives B a single coin“ works like an in-person transfer of cash over the Internet.
  - Just by sending a digital message over the Internet, one can transfer a real value.
  - Namely, one can transfer a coin, a valuable digital asset, to anyone over the Internet without going through a trusted third party.

1 Bitcoin Enables Value Transfer over Internet!

- Bitcoin works just like an in person cash transfer. But how?
  - Each transaction is verified before being written into Blockchain.
  - Only have the verified transactions recorded in the ledger.
  - Once recorded, every block is sealed in an immutable way that the content cannot be altered.

1 Bitcoin Enables Value Transfer over Internet!

- For example, you want to buy a beer at a convenient store.

Could we do the similar thing with a counter party over the Internet?

1 Bitcoin Enables Value Transfer over Internet!

- A → B 2 BTC
  - Suppose a transaction of coin transfer from A to B
  - The difficulty is obvious in this case.
    - It is easy to make copies since it is a digital message.
    - Double spending attempt can be made
      A → C 2 BTC
  - Alteration of messages can occur
    - A′ instead of A, B′ instead of B, 3 BTC rather than 2 BTC
    - Causes include network errors and frauds

1 Bitcoin Enables Value Transfer over Internet!

- Transaction
  - Thus, there are three parts to a transaction via a message A→B 2BTC

    1. Verification of ownership
    2. Double spending free
    3. Verified transactions are scribed into the Blockchain

1 Bitcoin Enables Value Transfer over Internet!

- Explanation of Transaction

  Solution

  - As the ledger is openly published and shared in the P2P computers in the Internet, any transaction can be verified for the validity of ownership.
  - Being able to refer to Blockchain record anytime and anywhere, double spending can be checked against and deterred.

1 Bitcoin Enables Value Transfer over Internet!

- Explanation of Transaction

  Solution

  - Blockchain is a digital ledger whose content cannot be altered once recorded into it.

    - Blockchain means also a new cryptographic technology which is to resolve the issue how to keep the content of the digital file unaltered once recorded.

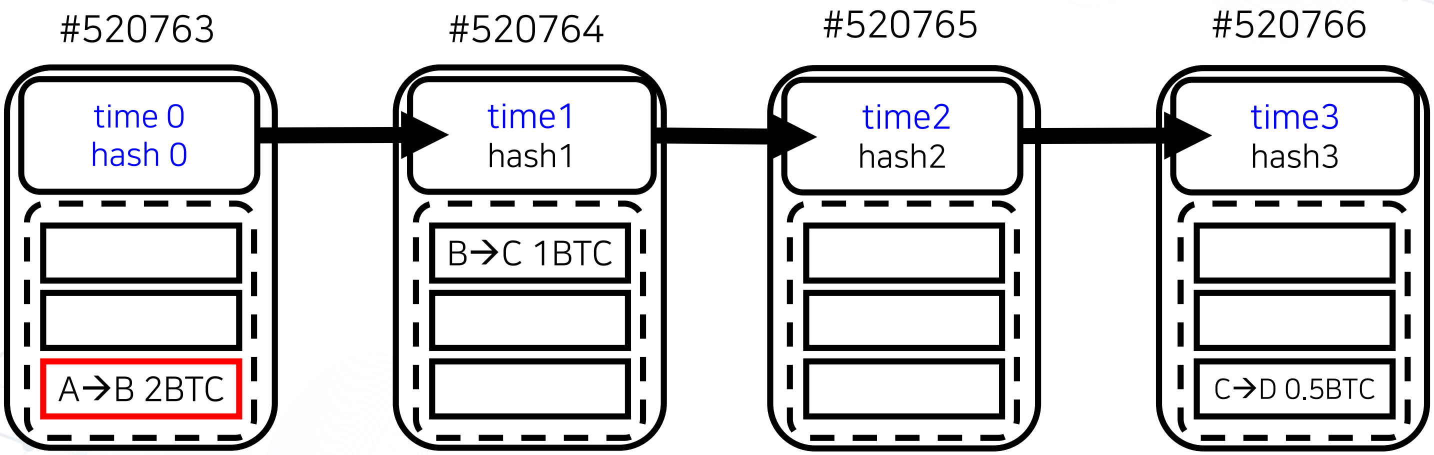
"Blockchain is believed to have many usages beyond currency"

1 Bitcoin Enables Value Transfer over Internet!

- A Blockchain is a digital book with a series of bound pages
  - Time 0: A (Sign of A) gives B two coins.
  - Time 1: B (Sign of B) gives C one coin.
  - Time 2: Empty
  - Time 3: C (Sign of C) gives D 0.5 coin.

GIST

1 Bitcoin Enables Value Transfer over Internet!



GIST

1 Bitcoin Enables Value Transfer over Internet!

- Blockchain is an open ledger in which transactions are recorded
  - What's written inside a block are the transactions and the timestamp of the block.
  - A series of such files connected in order of time is called Blockchain.

1 Bitcoin Enables Value Transfer over Internet!

- Blockchain is an open ledger in which transactions are recorded
  - Namely, Blockchain is a digital ledger with many bound pages.
    - As given above, coin transactions are recorded with time stamps.
    - Taking a look inside this ledger, one can always verify if a party owns enough coin to make a transaction or not.
    - From viewing the ledger, anybody can see how much coin belongs to a party.

1 Bitcoin Enables Value Transfer over Internet!

- Immutability
  - Proof-of-work (PoW) is defined to be a time consuming computational task.
  - A PoW is said to be heavy when the PoW task is very difficult to complete.
  - Bitcoin requires a heavy PoW to be used to make each chain connection.
  - This makes any accidental or intentional block alteration very difficult.

1 Bitcoin Enables Value Transfer over Internet!

- Immutability
  - To complete a PoW task, one computer may have to spend several years to complete a single chain connection task from a particular block to the next block.
  - How can one deceive others without alerting them for alteration?
  - One has to re-do all the PoWs in the chain starting from the very block to which an intentional alteration is made.

1 Bitcoin Enables Value Transfer over Internet!

- Immutability
  - To deceive others, one needs to re-do all the PoW, all the computation already done to the chain very quickly.
  - To make an alteration is thus almost impossible difficult task to do.

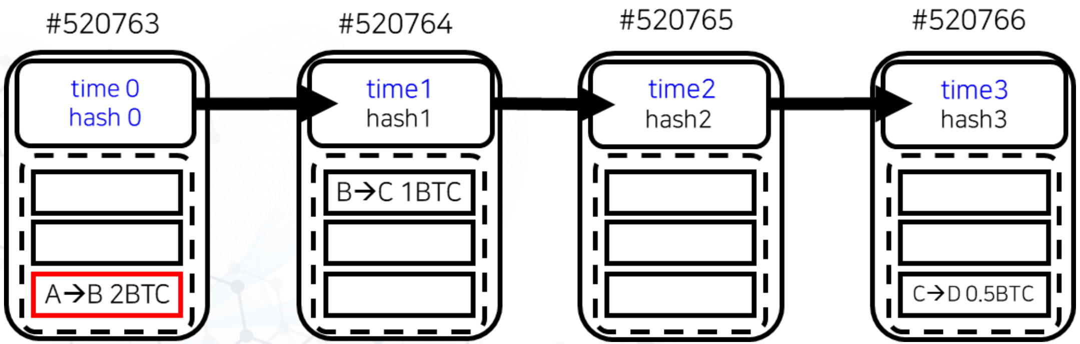
GIST

1 Bitcoin Enables Value Transfer over Internet!

- Anonymity

  But please make no mistake

  - This ledger uses cryptographic hash values and gibberish looking addresses.
  - For example, A above, and B, C, D as well, represents the address of an individual.



1 Bitcoin Enables Value Transfer over Internet!

- Anonymity

  But please make no mistake

  - The coin ownerships are given to these cryptographically addresses.
  - Only can the right person who has the private key to the said address claim the ownership of the coin.

1 Bitcoin Enables Value Transfer over Internet!

- Bitcoin Blockchain Verticals
  - Decentralized
  - Public
  - Immutability
  - Trust
  - Minting coins
  - Anonymity
  - Security

2 Satoshi Nakamoto Cryptography Mailing List

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

The first White Paper

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

2 Satoshi Nakamoto Cryptography Mailing List

- Bitcoin P2P e-cash paper

  NOVEMBER 1, 2008 SATOSHI NAKAMOTO CRYPTOGRAPHY MAILING LIST

  - I've been working on a new electronic cash system that's fully peer-to-peer, with no trusted third party.
  - The paper is available at: http://www.Bitcoin.org/Bitcoin.pdf

2 Satoshi Nakamoto Cryptography Mailing List

- Bitcoin P2P e-cash paper
  - The main properties are Double-spending is prevented with a peer-to-peer network.
  - No mint or other trusted parties.
  - Participants can be anonymous.
  - New coins are made from Hashcash style proof-of-work.
  - The proof-of-work for new coin generation also powers the network to prevent double-spending.

– Satoshi Nakamoto

GIST

2 Satoshi Nakamoto Cryptography Mailing List

- Bitcoin P2P e-cash paper
  - Bitcoin: A Peer-to-Peer Electronic Cash System

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main

The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work

long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

GIST

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Bitcoin P2P e-cash paper
  NOVEMBER 3, 2008 SATOSHI NAKAMOTO CRYPTOGRAPHY MAILING LIST
  - As long as honest nodes control the most CPU power on the network, they can generate the longest chain and outpace any attackers.
  - But they don't. Bad guys routinely control zombie farms of 100,000 machines or more.

– Anonymous

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Bitcoin P2P e-cash paper
  - People I know who run a blacklist of spam sending zombies tell me they often see a million new zombies a day.
  - This is the same reason that hashcash can't work on today's Internet - the good guys have vastly less computational firepower than the bad guys.

– Anonymous

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  NOVEMBER 3, 2008 SATOSHI NAKAMOTO CRYPTOGRAPHY MAILING LIST
  - Thanks for bringing up that point.
  - I didn't really make that statement as strong as I could have. The requirement is that the good guys collectively have more CPU power than any single attacker.

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  - There would be many smaller zombie farms that are not big enough to overpower the network, and they could still make money by generating Bitcoins. The smaller farms are then the “honest nodes” (I need a better term than “honest”).

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  - The more smaller farms resort to generating Bitcoins, the higher the bar gets to overpower the network, making larger farms also too small to overpower it so that they may as well generate Bitcoins too.
  - According to the “long tail” theory, the small, medium and merely large farms put together should add up to a lot more than the biggest zombie farm.

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  - Even if a bad guy does overpower the network, it's not like he's instantly rich. All he can accomplish is to take back money he himself spent, like bouncing a check.

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  - To exploit it, he would have to buy something from a merchant, wait till it ships, then overpower the network and try to take his money back.
  - I don't think he could make as much money trying to pull a carding scheme like that as he could by generating Bitcoins. With a zombie farm that big, he could generate more Bitcoins than everyone else combined.

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  - The Bitcoin network might actually reduce spam by diverting zombie farms to generating Bitcoins instead.

– Satoshi Nakamoto

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Bitcoin P2P e-cash paper
  NOVEMBER 7, 2008 SATOSHI NAKAMOTO CRYPTOGRAPHY MAILING LIST
  - Lengthy exposition of vulnerability of a system to use-of-force monopolies elided.
  - You will not find a solution to political problems in cryptography.

2 Satoshi Nakamoto Cryptography Mailing List

- Re: Re: Bitcoin P2P e-cash paper
  NOVEMBER 7, 2008 SATOSHI NAKAMOTO CRYPTOGRAPHY MAILING LIST
  - Yes, but we can win a major battle in the arms race and gain a new territory of freedom for several years.
  - Governments are good at cutting off the heads of a centrally controlled networks like Napster, but pure P2P networks like Gnutella and Tor seem to be holding their own.

– Satoshi Nakamoto

3 Brief History

- Brief History of Bitcoin
  - Oct. 2009: $1 = 1,309 BTC
  - Feb. 2010: First Bitcoin market
  - May 2010: Pizza day (a pizza = 10,000 BTC)
  - Nov. 2010: Bitcoin $1 Million. $0.5/BTC
  - Feb. 2011: Bitcoin $ 206 Million
  - Mar. 2013: Bitcoin $1 Billion
  - Aug. 2013: Federal Judge Rules Bitcoin is Real Money

3 Brief History

- Brief History of Bitcoin
  - Dec. 2013: China CB, bars financial institutions from handling Bitcoin transactions.
  - Dec. 2014: Microsoft begins accepting Bitcoin payments.

3 Brief History

- ~2010: M. value created, Pizza Day
  - October 2009
    - Bitcoin receives an equivalent value in traditional currencies.
    - The New Liberty Standard established the value of a Bitcoin at $1 = 1,309 BTC.
    - The equation was derived so as to include the cost of electricity to run the computer that created the Bitcoins in the first place.

GIST

3 Brief History

- ~2010: M. value created, Pizza Day
  - February 2010
    - The world's first Bitcoin market is established by the now defunct dollar.

3 Brief History

- ~2010: M. value created, Pizza Day
  - May 2010
    - A programmer living in Florida named Laslo Hanyecz sends 10,000BTC to a volunteer in England, who spent about $25 to order Hanyecz a pizza from Papa John’s.
    - Today that pizza is valued at £1,961,034 and stands as a major milestone in Bitcoin’s history.

3 Brief History

- ~2010: M. value created, Pizza Day
  - November 2010
    - Bitcoin reaches $1 million. Based on the number of Bitcoins in circulation at the time, the valuation leads to a surge in Bitcoin value to $0.50/BTC.

3 Brief History

- 2013: Regulation started, "Bitcoin is money"
  - February 2011
    - Bitcoin reaches parity with the US dollar for the first time.
    - By June each Bitcoin is worth $31 giving the currency a market cap of $206 million.

3 Brief History

- 2013: Regulation started, "Bitcoin is money"
  - March 2013
    - The US Financial Crimes Enforcement Network (FINCEN) issues some of the world's first Bitcoin regulation in the form of a guidance report for persons administering, exchanging or using virtual currency.
    - This marked the beginning of an ongoing debate on how best to regulate Bitcoin.

3 Brief History

- 2013: Regulation started, “Bitcoin is money”
  - March 2013
    - Bitcoin market capitalisation reaches $1b.

3 Brief History

- 2013: Regulation started, "Bitcoin is money"
  - August 2013
    - Federal Judge Mazzant claims: "It is clear that Bitcoin can be used as money" and "It can be used to purchase goods or services" in a case against Trendon Shavers, the so-called 'Bernie Madoff of Bitcoin'.
    - Bloomberg begins testing Bitcoin data on its terminal.
    - Although alternative tickers exist, endorsement from Bloomberg gives Bitcoin more institutional legitimacy.

3 Brief History

- 2013: Regulation started, "Bitcoin is money"
  - December 2013
    - China's central bank bars financial institutions from handling Bitcoin transactions.
    - This ban was issued after the People's Bank of China said Bitcoin is not a currency with "real meaning" and does not have the same legal status as fiat currency.
    - The ban reflects the risk Bitcoin poses to China's capital controls and financial stability.
    - Today China remains the world's biggest Bitcoin trader, with 80% of global Bitcoin transactions being processed in China.

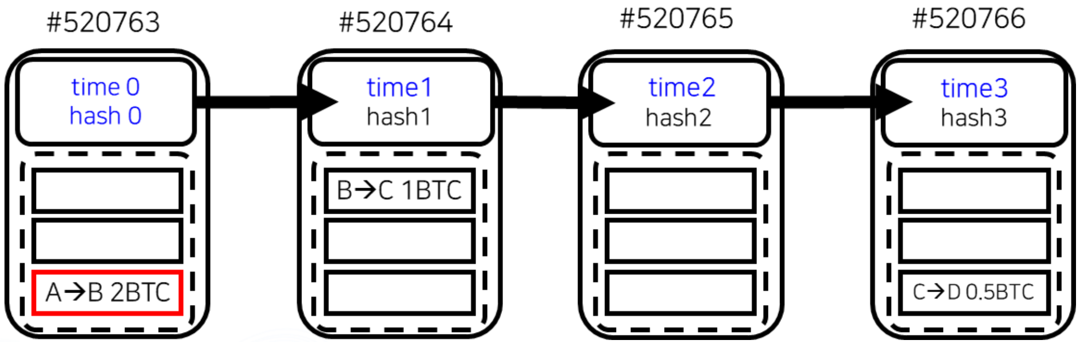
3 Brief History

- 2015: Derivatives, Assets, Payments
  - December 2014
    - Tech giant Microsoft begins accepting Bitcoin payments.

Goal of this lecture note

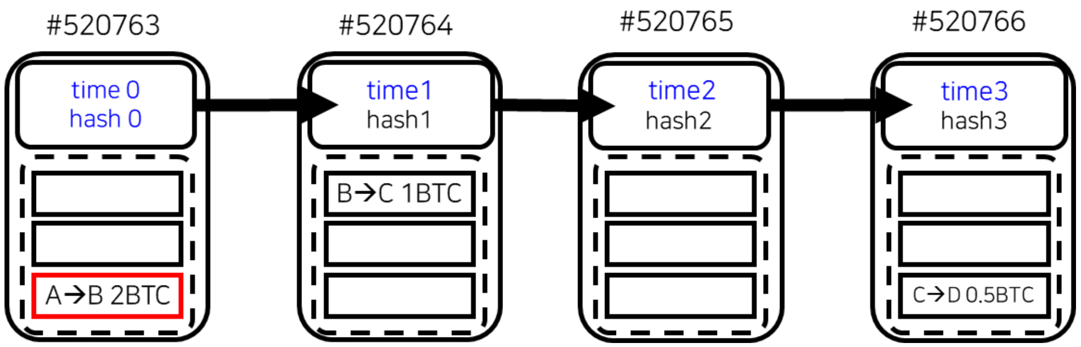
- How to Put Digital Signature to a Message
- Secure Hash Function

1 How to Put Digital Signature to a Message



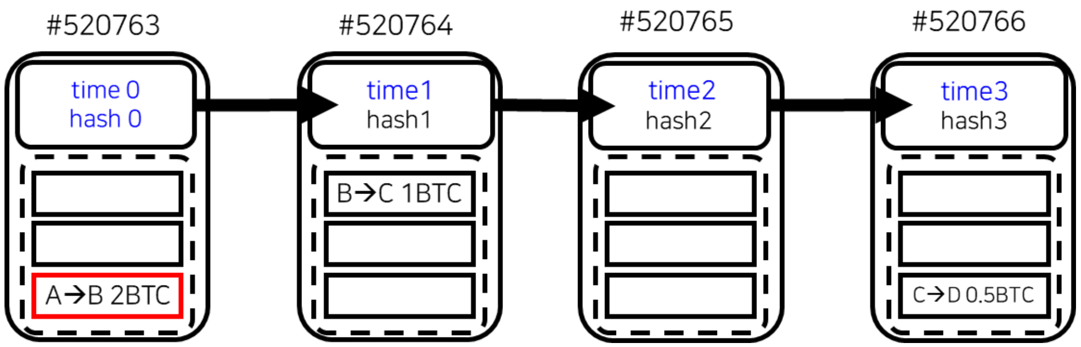
- Time 0: A (Sign of A) gives B two coins
- Time 1: B (Sign of B) gives C one coin
- Time 2: Empty
- Time 3: C (Sign of C) gives D 0.5 coin

1 How to Put Digital Signature to a Message



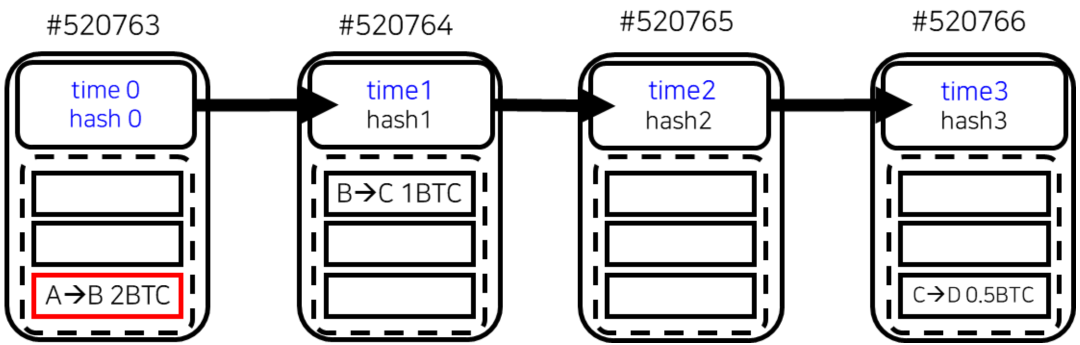
- This is one of the essential charts for understanding how a message transfer to someone can work as a value transfer using a Blockchain.

1 How to Put Digital Signature to a Message



- Namely, the sender shall put his digital signature in order to show the ownership of his coin.

1 How to Put Digital Signature to a Message



- Now we aim to show an example how a digital signature is created.

1 How to Put Digital Signature to a Message

- Public key and private key
  - In cryptography, any person can create as many number of pairs of keys.
  - Each pair comes with a public key and a private key.
  - Encryption
    - With one key, a message can be locked.
  - Decryption
    - With the other key, the locked message can be unlocked.
  - Alice can send Bob a private message.

1 How to Put Digital Signature to a Message

- Generation of a key pair
  - Consider two individuals, Alice and Bob.

Alice generates her keys, Pub_A and Pri_A

Bob does the same, Pub_B and Pri_B

  - Each person keeps the private key in secret, while lets the public key widely known.
  - Using them, one can send a private message and put a digital signature to it.

1 How to Put Digital Signature to a Message

- Encryption and Decryption
  - Define a message m.
  - Define a pair of functions, ENC( )and DEC( ).
  - These functions are publicly known functions.
  - Cyphered message or encrypted message is created with ENC( ) , i.e.,

$$y = \mathrm{ENC}(m, \mathrm{Pub_B})$$

  - Cyphered message can only be deciphered using $\mathrm{Pri_B}$, i.e.,

$$m = \mathrm{DEC}(y, \mathrm{Priv_B})$$

1 How to Put Digital Signature to a Message

- RSA Example of ENC and DEC functions
  - Let e, m and n be known positive integers. Is it easy to find d?

$$(m^e)^d = m \bmod n \quad \text{--- (1)}$$

- Once d known, it is easy to check

$$(m^d)^e = m \bmod n \quad \text{--- (2)}$$

- Let d be private key and e public key.

✓ Modulo란?
대상 숫자의 나머지를 구하는 연산

- Ex) Modulo-3: 5%3=5-3*floor(5/3) = 2.

1 How to Put Digital Signature to a Message

- EX1 Alice would like to send a private message "I love you Bob." to Bob.

	Private key d	Public key e
Alice	d_A	e_A
Bob	d_B	e_B

- Alice encrypts her message m with Bob's public key, i.e., $y = \mathrm{ENC}(m, e_B)$.
- The encrypted message y is transferred to Bob
- Only can Bob decipher encrypted Alice's message, i.e., $m = \mathrm{DEC}(y, d_B)$.

1 How to Put Digital Signature to a Message

- EX2 Alice attaches a digital signature to her encrypted message m sent to Bob.
  - Alice hashes her message m and get $h(m)$.
  - She puts her signature to the digital message m.
  - The digital signature to her message is

$$Sign(m)=h(m)^{d_A}$$

  - Alice uses her pri_key d_A to generate $Sign(m)$.
  - Using Alice's pub_key e_A , Bob recovers $h(m)$ via (2).

1 How to Put Digital Signature to a Message

- EX2 Alice attaches a digital signature to her encrypted message m sent to Bob
  - Using the Alice's message m deciphered from Ex1), Bob generates its hash, $h(m)$.
  - Bob checks if the two hash values match.

GIST

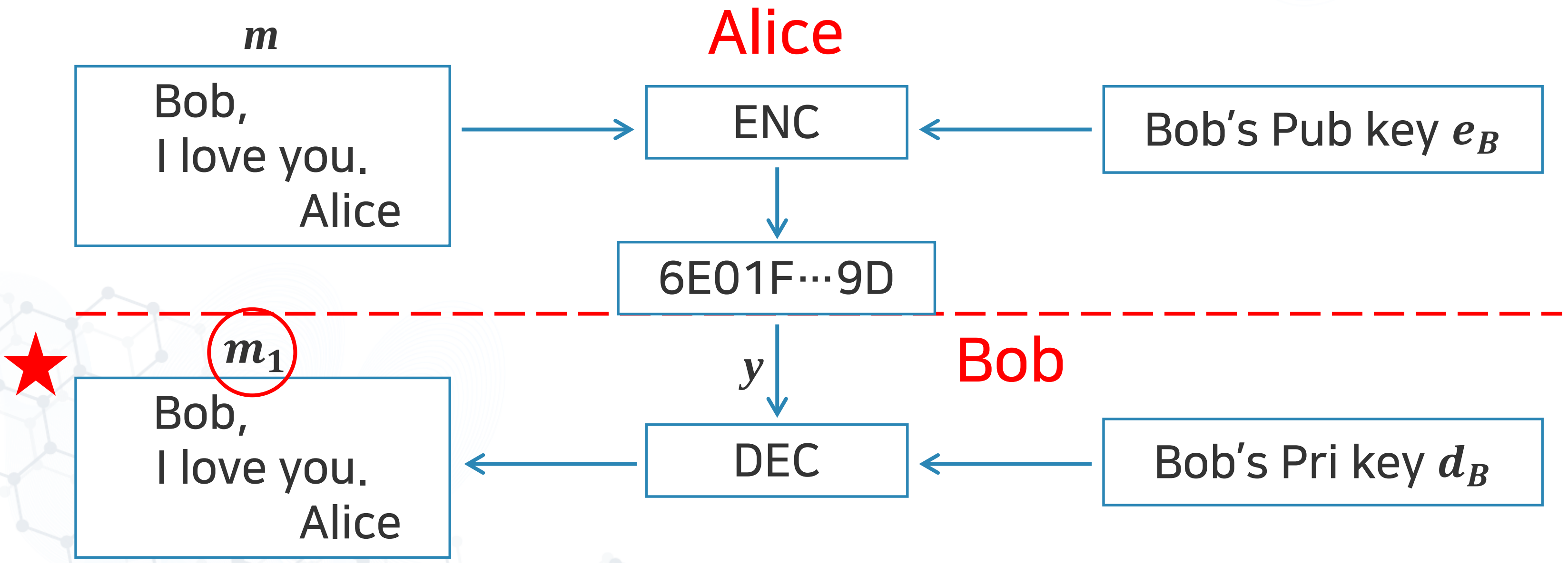
1 How to Put Digital Signature to a Message

- Note
  - Bitcoin does not use RSA but Elliptic Curve Signatures.
  - Our purpose of showing how to put digital signature to a message can be served with RSA as well.

"We may use RSA because it is more familiar to us."

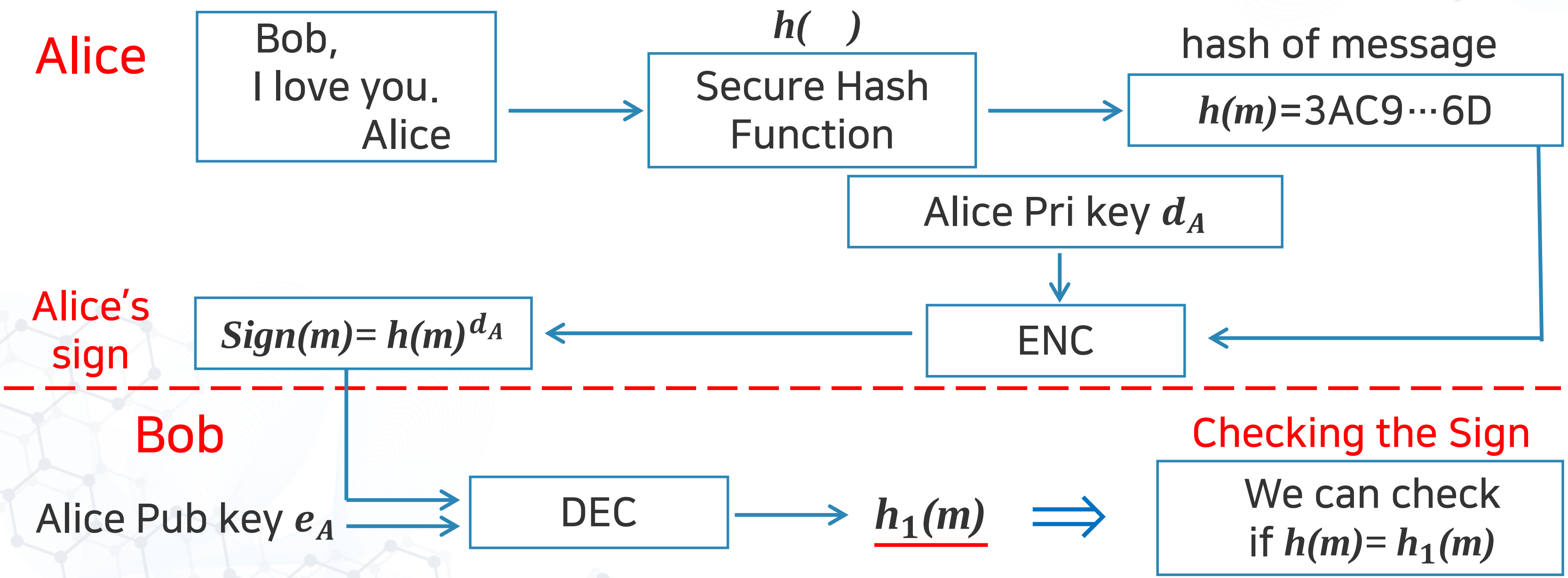
1 How to Put Digital Signature to a Message

- Alice sends a private message to Bob.



1 How to Put Digital Signature to a Message

- Digital Sign and Validation



1 How to Put Digital Signature to a Message

- Let m be "A → B 2 BTC"
  - Alice sends the message m .
  - Alice attaches her digital signature to it.
  - Together it shall look like:

A → B 2BTC Sign_by_A

  - Ownership verification is done by checking the sign.
  - Balance can be checked by looking at the address A.
  - Transaction is complete if this is recorded into the book.

1 How to Put Digital Signature to a Message

- Anonymity
  - A is an address of Bitcoin made from a public key of Alice.
  - B is an address of Bitcoin made from a public key of Bob.

2 Secure Hash Functions

- What is a Hash Function?
  - Definition
    - A hash function is a function, represented with H(input) = output, which takes a text message as its input and gives as its output a fixed number of binary bits.

2 Secure Hash Functions

- What is a Hash Function?
  - Bitcoin uses the Secure Hash Function 256.
  - The length of output bit string is 256.
  - The input to a hash function is a text message or a file.

Ex Input-Output of Hash function H

- Message = [Bob, I love you. Alice.]
- H(Message) = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]
- A string of 64 hexadecimals or a string of 256 bits.

2 Secure Hash Functions

- Conditions for Good Hash Function
  - One way
    With a little change in the input, the output is completely different.
    - Input distance has no relation to output distance.
  - Collision free
    Given $y = H(x)$, finding $x_1 \neq x$ such that $H(x_1) = y$ shall be almost impossible!
  - Collision free stronger
    Finding an input pair of different messages x and x_1 which leads to $H(x) = H(x_1)$ shall be almost impossible!

- x_1 = [Bob, I love you. Alice.]
- $H(x_1)$ = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]

- Illustration of Onewayness

Ex1

- x_1 = [Bob, I love you. Alice.]
- $H(x_1)$ = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]

Ex2

- x_2 = [Bob, I love you. Alice]
- $H(x_2)$ = [B1316ED8BA74AD416C8E966574CD584AD447B8 11B722FB9230C71B047C71B825]

2 Secure Hash Functions

- Illustration of Onewayness

Ex3

- x_3 = [Bob, I loved you. Alice.]
- $H(x_3)$ = [BFDDB00446539D8CF8ECC712E3A8144EDF41A771C0F96560E9EDE3E576CD8FBF]

2 Secure Hash Functions

- Tiny difference → Big difference
  - Note that there is a very small difference between x_1 and x_2.
  - But the difference in the output is huge.
  - This property can be utilized to spot out a tiny alteration made to an original input file.
  - A tiny unnoticeable alteration, and thus is difficult to be detected by human eyes, but can be magnified into easily discernable hash difference.

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 $H(\ )$

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$x \in \mathbf{X}$ → $H(x)$ → $y \in \mathbf{Y}$

x_1=[Bob, I love you. Alice.]

$H(x_1)$ = [2FE442157E2025AB75F3856F092 38E2CD78A3B396BC25F128B95D0 4AD6252634]

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 $H(\ )$

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$x \in \mathbf{X}$ → $H(x)$ → $y \in \mathbf{Y}$

x_2=[Bob, I love you. Alice]

$H(x_2)$ =
[B1316ED8BA74AD416C8E966574CD584AD447B811B722FB9230C71B047C71B825]

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 $H(\ )$

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$x \in X$ → $H(x)$ → $y \in Y$

x_3=[Bob, I loved you. Alice.]

$H(x_3)$ = [BFDDB00446539D8CF8ECC712E3A8144EDF41A771C0F96560E9EDE3E576CD8FBF]

GIST

2 Secure Hash Functions

- SHA-256 Algorithm

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

```
SHA-256(M):
  (* Let M be the message to be hashed *)
  for each 512-bit block B in M do
    W = f_exp(B);
    (* Initialize the registers with the constants. *)
    a = H0; b = H1; c = H2; d = H3; e = H4; f = H5; g = H6; h = H7;
    for i = 0 to 63 do
      (* Apply the 64 rounds of mixing. *)
      T1 = h + Σ1(e) + f_if(e, f, g) + K_i + W_i;
      T2 = Σ0(a) + f_maj(a, b, c);
      h = g; g = f; f = e; e = d + T1; d = c; c = b; b = a; a = T1 + T2;
    (* After all the rounds, save the values in preparation of the next data block. *)
    H0 = a + H0; H1 = b + H1; H2 = c + H2; H3 = d + H3;
    H4 = e + H4; H5 = e + H5; H6 = e + H6; H7 = e + H7;
  (* After all 512-bit blocks have been processed, return the hash. *)
  return concat(H0, H1, H2, H3, H4, H5, H6, H7);
```

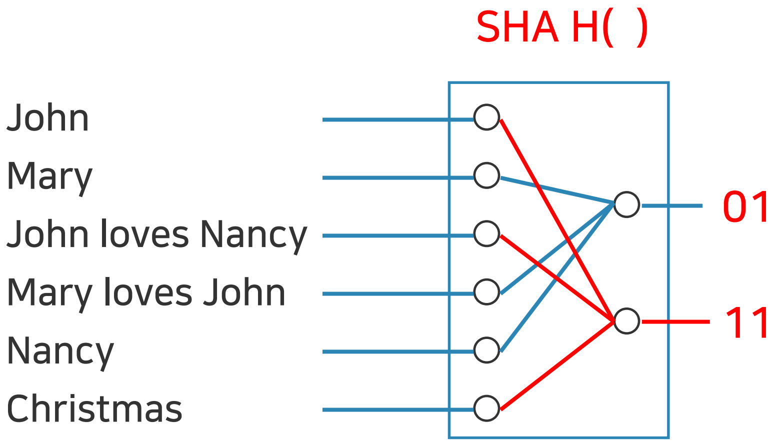
Algorithm 1.3: THE SHA-256 ALGORITHM.

National
Institute of
Standard and
Technology
미국 국립 표준 연구소

Wouter Penardand and Timvan Werkhoven, "On the Secure Hash Algorithm family," *Cryptography in Context,* 2002

2 Secure Hash Functions

- Collision free



Note

H(John)
=H(Christmas)
=H(John loves Nancy)

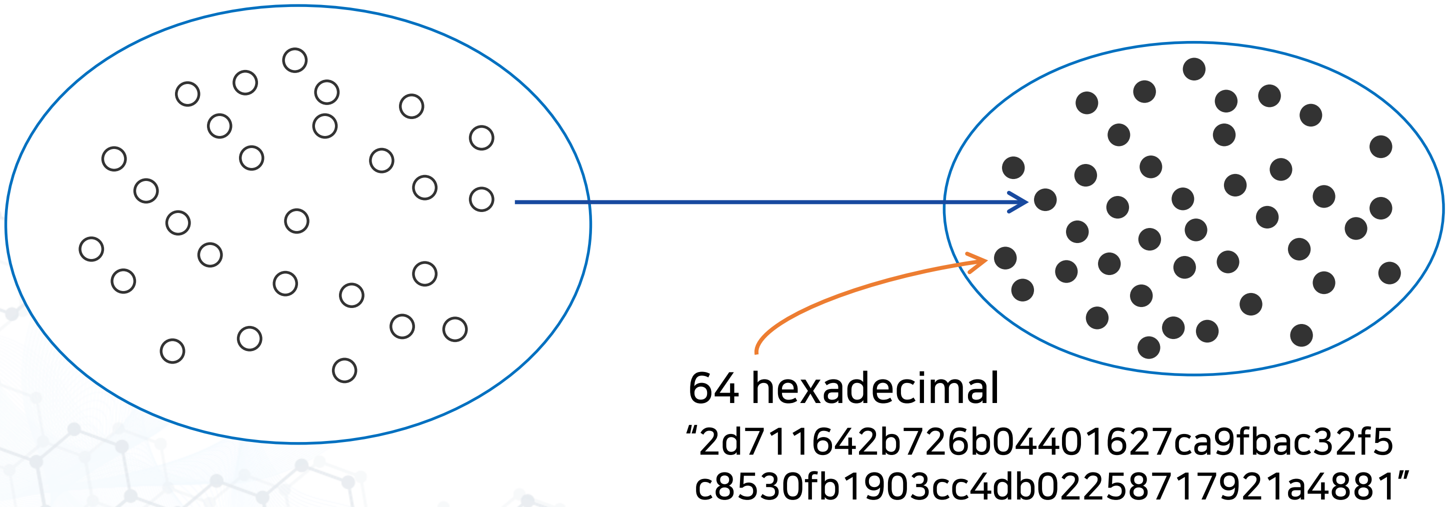
- Collision free implies there surely are collisions but one can hardly encounter one.

2 Secure Hash Functions

- Input-Output of SHA-256, i.e., $H(x) = y$

$X := \{x | x \text{ is a message up to 1 Mbyte in size}\}$

$Y := \{y | y \text{ is a 256bit string}\}$



2 Secure Hash Functions

- Cardinality of the Input file set
  - Bitcoin allows an input file whose size is up to a 1Mbyte.
  - What is the cardinality of the set of all possible input file sets?
    - All possible input files can be enumerated from small files to large files, such as noting, 0, 1, 10, 11, 100, 101, 110, 111, ⋯.
  - Thus, there are $2^{8000000}$ different files.
  - The cardinality of x is about $10^{2400000}$.

2 Secure Hash Functions

- Cardinality of Output Hash Set
  - Each input file produces a 256 bit output.
  - The cardinality of the set of all output hash values is $2^{256} \sim 10^{77}$.
  - For each y in $\mathcal{y}$, how many input files x in X are there such that each $H(x) = y$?

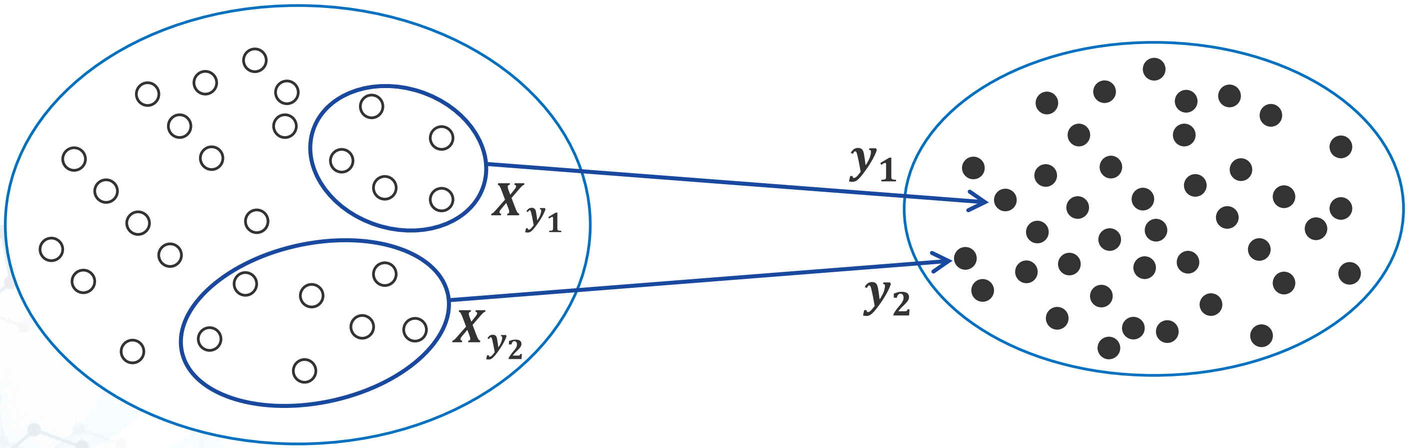
GIST

2 Secure Hash Functions

- Preimage of y is a subset of X

$X_y := \{x \in X | H(x) = y\}$

$Y := \{y | \text{y is a 256bit string}\}$



GIST

2 Secure Hash Functions

- Size of Input Set per Hash Output
  - What is the average size of the input file set whose element leads to the same hash output?
  - For each output hash y in y , the preimage of y can be defined as

$$X_y := \{x : H(x) = y\}$$

  - WLOG, assume the same size for any y_1 and y_2 :

$$|X_{y_1}| = |X_{y_2}|$$

2 Secure Hash Functions

- There are 2^{256} preimage sets.
  - There are 2^{256} distinct y's in y.
  - There are 2^{256} preimages of y in X.
  - These are mutually non-overlapping sets.
  - The size of a preimage of a point y is

$$log_{10}|X_y| = log_{10}\frac{|X|}{|Y|} = 2400000 - 77 = 2399923.$$

2 Secure Hash Functions

- Collisions are abound, but can you find one?
  - Collisions must occur, even abundantly.
  - Consider any two different files x_1 and x_2 in X_y, i.e., the two hashes are the same $H(x_1) = H(x_2)$
  - For any file x_3 in X but not in X_y, we note,

$$H(x_3) \neq y$$

  - What do you mean by Collision Free then?

2 Secure Hash Functions

- What is the meaning of Collision Free?

Small problem

- Suppose the input x is a file of size up to 1 Kilobyte and the SHA output is truncated to 10 bit.
- Bob has found that the input file x_0 has the hash value y_0.

(a) What is the size of the input file set?

(b) What is the size of the output file set?

(c) Bob selects a file x_1 at random from his desktop computer, size smaller than 1 Kilobyte, and runs it thought the truncated to the first 10 bit, say SHA-10.

What is the probability that this output is the same as the first output y_0?

GIST

2 Secure Hash Functions

- Solution 1
  - The set sizes are

$$s_X := log_{10}|X| = log_{10} 2^{8000} = 8e3 \times 0.3010 \sim 2.40e3$$

$$s_Y := log_{10} 2^{10} = 10 \times 0.3010 = 3.01$$

$$s_{X_y} := log_{10} \frac{|X|}{|Y|} = s_X - s_Y \sim 2400 - 3 = 2397$$

2 Secure Hash Functions

- Solution 2
  - Let p_c^1 be the prob. of selecting $x_1 \neq x_0$ leading to hash collision.

$$p_c^1 := Pr\{x_1 : H(x_1) = H(x_0)\} = Pr\{x_1 \in X_y\}$$

$$= \frac{|X_y| - 1}{|X| - 1} \approx \frac{1}{|Y|}$$

GIST

2 Secure Hash Functions

- Solution 3
  - Suppose there were no hash collisions for two x_0 and x_1. Now select another file x_2.
  - Let p_c^2 be the prob. that the hash of x_2 is equal to either of the two previous hashes, leading to hash collision. Find it.

$$p_c^2 := Pr\{x_2 : H(x_2) = H(x_0)\} \cup \{x_2 : H(x_2) = H(x_1)\}$$

$$= Pr\{x_2 \in X_y\} + Pr\{x_2 \in X_{y_1}\}$$

$$= \frac{2(|X_y| - 1)}{|X| - 2} \approx \frac{2}{|Y|}$$

2 Secure Hash Functions

- Solution 3
  - Suppose there were no hash collisions up to three selections of files x_0 , x_1 and x_2 . Now select another file x_3. Let p_c^3 be the prob. that the hash of x_3 is equal to any of the three previous hashes, leading to hash collision. Find it.

$$p_c^3 = Pr\{x_3 \in X_y\} + Pr\{x_3 \in X_{y_1}\} + Pr\{x_3 \in X_{y_2}\}$$

$$= \frac{3(|X_y| - 1)}{|X| - 3} \approx \frac{3}{|Y|}$$

GIST

2 Secure Hash Functions

- Solution 4
  - Suppose there were no hash collisions up to m selections of files x_0 , x_1 and x_{m-1}. Now select an m-th file x_m
  - Let p_c^m be the prob. that the hash of x_m is equal to any of the previous hashes, leading to hash collision. Find it.

$$p_c^m = Pr\{x_m \in X_y\} + \cdots + Pr\{x_m \in X_{y_{m-1}}\}$$

$$= \frac{m(|X_y| - 1)}{|X| - m} \approx \frac{m}{|Y|}$$

2 Secure Hash Functions

- Solution 5
  - The hash collision probability increases as m grows, i.e.,

$$p_c^1 = \frac{1}{1024}$$

$$p_c^2 = \frac{2}{1024}$$

$$p_c^3 = \frac{3}{1024}$$

$$\dots$$

$$p_c^{512} = \frac{512}{1024}$$

2 Secure Hash Functions

- What is the meaning of Collision Free?

Large problem

- Here the input x is a file of size up to 1 Megabyte.
- Bob has found that the input file x_0 has the hash value y_0

  - (a) What is the size of the input file set?
  - (b) He selects a file x_1 at random from his desktop computer and runs it thought SHA-256. What is the probability that this output is the same as the first output y_0?

2 Secure Hash Functions

- Solution
  - The collision probability is so small no matter how many files are selected.

$$p_c^m = Pr\{x_m \in X_y\} + \cdots + Pr\{x_m \in X_{y_{m-1}}\}$$

$$= \frac{m(|X_y| - 1)}{|X| - m} \approx \frac{m}{|Y|} = \frac{m}{10^{77}}$$

2 Secure Hash Functions

- Bitcoin hash cycles per second is huge.
  No collision thus far for 10 years?
  - Bitcoin hash power has reached 10^20 cycles/sec.
    Suppose it's been that way for the past 10 years.
  - What is the probability of collision occurred?
  - Given m=O(10^{29}) distinct hashes generated
    in 10 years,

$$p_c^m = \frac{m}{|Y|} = \frac{10^{29}}{10^{77}} = 10^{-48}$$

2 Secure Hash Functions

- What is the meaning of Collision Free?
  - Size of the hash output set is so huge.
  - One knows there are large number of collisions, but one cannot come across any collision.
  - How larger is this number 10^{77}
    - The number of cells in a human body is O(10^{13}).
    - The number of cells in all human body is O(10^{23}) .
    - The number of stars in the observable universe is O(10^{22}) .
    - The number of atoms in the observable universe is O(10^{80}) .
    - https://en.wikipedia.org/wiki/Large_numbers

GIST

Goal of this lecture note

- Need for Proof-of-Work (PoW)
- PoW Puzzles
- Difficulty Level of Puzzles
- Probability of Mining Success
- AI-IM-To-Po Theory

1 Need for Proof-of-Work (PoW)

- Blockchain
  is a ledger and a technology.
  - A digital file it is.
  - Content can be copied and altered easily.
  - A novel way is to resolve the problem of forgery and unwanted alterations:
    - Each block is summarized.
    - This summary shall be good enough.
    - Only the block with the proof of work included can be connected to the existing chain of blocks.

GIST

1 Need for Proof-of-Work (PoW)

- Blockchain
  - Revolutionary new idea!
    - Any single computer cannot find a good block summary within a given amount of computing time.
    - If the number of computers is large enough and all are simultaneously working on finding good summary of a block, one computer among them can come out successful within the desired time.

GIST

1 Need for Proof-of-Work (PoW)

- Blockchain
  - Revolutionary new idea!
    - A reward is given to this computer which has found a good block summary.
    - Once completed, a new race is set and started again for a new block formation.
    - The more computers are gathered and participate in the race, the safer the system becomes.

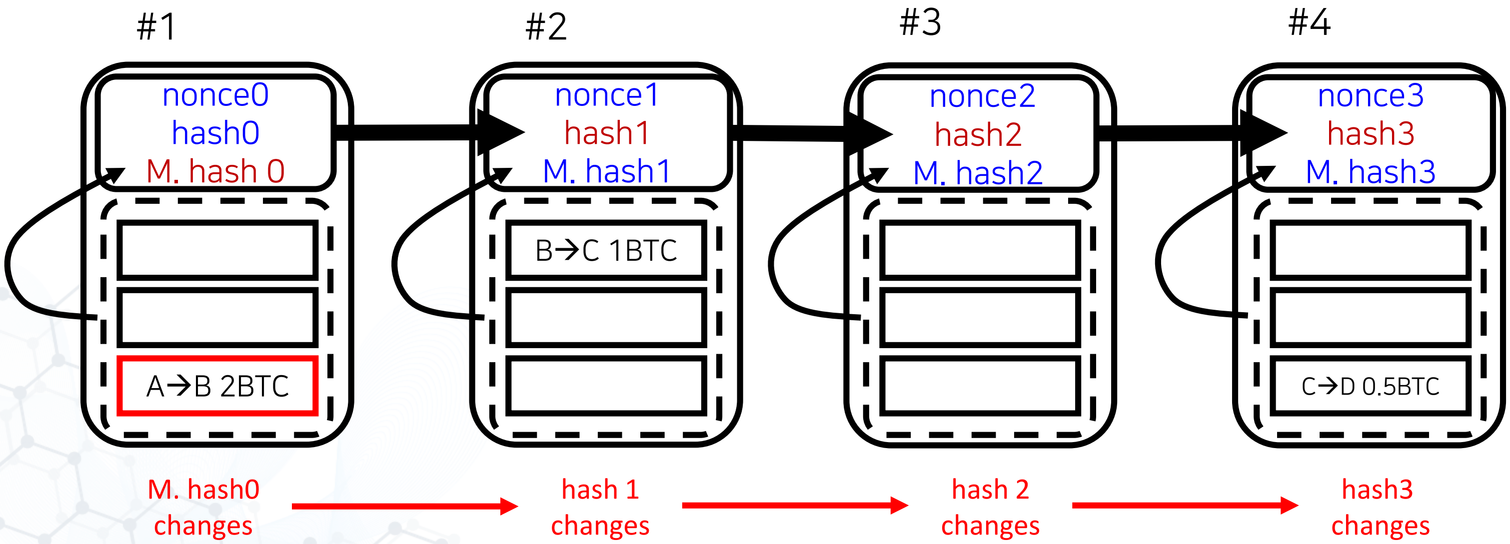
GIST

1 Need for Proof-of-Work (PoW)

- Content in the blockchain cannot be changed.
  - What happens when any alteration is made?
  - Any small alteration is easily noticeable!
  - An unnoticeable change is possible,
    but it requires a complete alteration.
  - The complete job is to redo all the hashes
    of the following blocks.
  - PoW is imposed in each block
    and thus the whole job cannot be made easily.

1 Need for Proof-of-Work (PoW)

- Content in the blockchain cannot be changed, why?



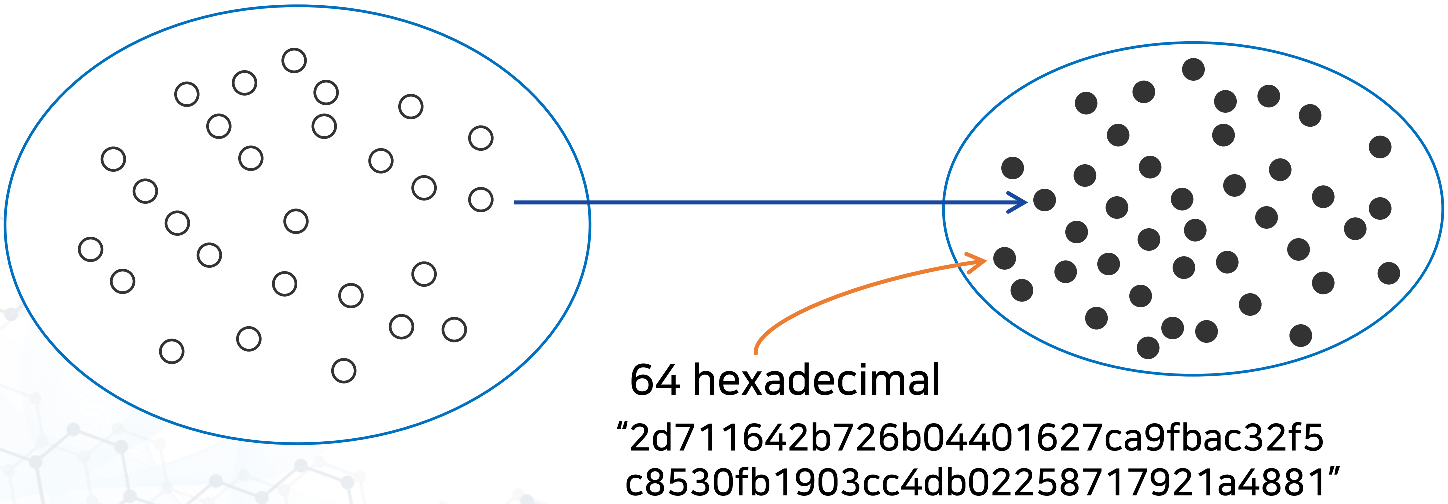
2 PoW Puzzles

- Making PoW puzzles
  - Bitcoin uses SHA256
  - Recall SHA is *oneway* and *collision free*.

2 PoW Puzzles

- SHA256, $F(x) = y$

$X = \{x | x \text{ is a message up to 1 Mbyte in size}\}$ $Y = \{y | y \text{ is a 256bit string}\}$



GIST

2 PoW Puzzles

- Finding Good Block Summary
  - Function F takes x and gives output y

$$y = F(x)$$

  - x is block header (BH), i.e., F(BH) = hash.
  - Then, it can be written as

F(B.H.: nonce) < Target   PoW Ineq.

  - For a block, find a nonce that satisfies the above inequality (Work)
  - Record the nonce in the block header. (Proof)

GIST

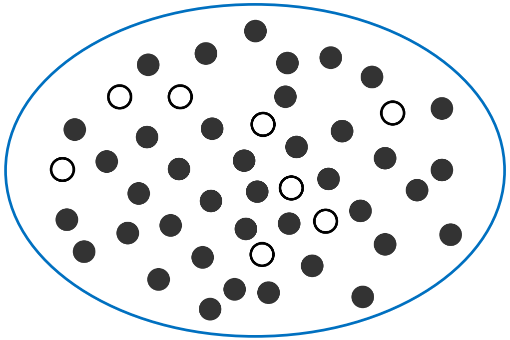
2 PoW Puzzles

- Toy puzzle
  - White and black balls.
  - There are 2^6 black balls.
  - Balls are numbered, i.e., hashes.
  - Let Target be 2^3=8.
  - Pick a nonce and run SHA-256.

Total no. of balls $2^6 = 64$
Target = 2^3 0 0 1 0 0 0
A = {Balls < Target}
2^3 -1 = 7 0 0 0 1 1 1
6 0 0 0 1 1 0
5 0 0 0 1 0 1
...

What is the probability that a while ball is picked?

$$p = 2^3/2^6 = 1/8$$



GIST

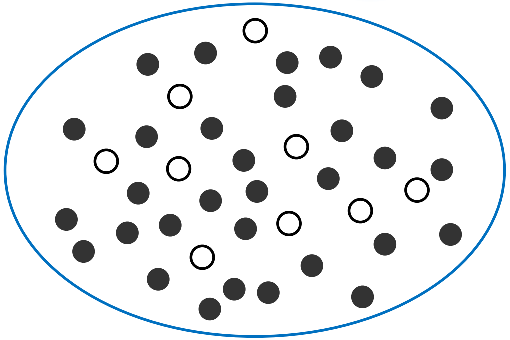
2 PoW Puzzles

- Bitcoin puzzle
  - Hashes are strings of 256 bits.
  - There are 2^{256} hashes in Y.
  - Let Target be $2^{256-16}=2^{240}$.

What is the probability that the hash satisfies the PoW?

$$p = 2^{240}/2^{256}$$
$$= 2^{-16}$$
$$= 1/64000$$

$Y = \{y | y$ is a 256bit string$\}$



White balls are
64 hexadecimals with 4 leading zeros

"00001642b726b04401627ca9fbac32f5
c8530fb1903cc4db02258717921a4881"

3 Difficulty Level of Puzzles

- The probability p that a CPU solves (PoW) in a single cycle, given the first four strings are zeros?
  - Any hash value looks line this: "2d711642b726b04401627ca9fbac32f5c 8530fb1903cc4Db02258717921a4881"
  - A good hash value looks like this: "0000f727854b50bb95c054b39c1fe5c92 e5ebcfa4bcb5dc279f56aa96a365e5a"
    - c = the size of Y the set of all hash values = 2^{256}
    - a = the size of A the set of wanted hash values = $2^{(256-16)} = 2^{240}$
    - p = a/c = $2^{-16} = 1/2^{16} \sim 1/64000$

3 Difficulty Level of Puzzles

- Proof of Work is a ALone IMpossible Together Possible (Al-IM-To-Po) Problem!
  - Let there be a CPU which can take one input and gives one output.
  - What is the probability that this CPU finds a good summary in a single hash cycle?

$$p = a/c = 2^{-16} = 1/64000$$

  - Difficulty of the PoW puzzle can be adjusted by varying the size of a.
  - Thus, p represents a difficulty of the puzzle.

4 Probability of Mining Success

- Given the difficulty p, we aim to find Probability of Mining Success.

GIST

4 Probability of Mining Success

- Definition: Success Random Variable K.
  Let K = 1, 2, 3,⋯, denote the index of the hash at which PoW success occurs.
  - For example, $K = 4$ means that PoW success comes exactly at the 4^{th} hash.
  - This is a random variable since the draw of a successful hash value is a random experiment.

4 Probability of Mining Success

- Definition: Hash Rate of CPU.
  - The hash rate of a CPU is defined as number hashes in a unit time.
  - For example, the hash rate of a CPU which can do 100 hash cycles in 1 second is 100 hashes/sec.

4 Probability of Mining Success

• ASIC Mining Hardware

Bitcoin Mining Hardware Comparison

Pic	Miner	Hash Power	Price	Buy
	Antminer S9	14.0 TH/s	$3,000	
	Antminer R4	8.6 TH/s	$1,000	

출처: https://www.buybitcoinworldwide.com/mining/hardware/

GIST

4 Probability of Mining Success

- Definition: Success Random Variable K.
  Let K = 1, 2, 3,⋯, denote the index of the hash values at which the PoW success occurs
  - What is the probability that this CPU with rate 100 hashes/sec solves PoW in 1 second? Use $p = 10^{-6}$.

$$P_p\{K \le k\} =: P_{Geom}(p, k = 100)$$

$$= p + (1-p)p + \cdots + (1-p)^{k-1} p$$

$$\sim 100{*}p$$

$$= 10^{-4} \quad (2.384\text{e-}5 \text{ exact})$$

GIST

4 Probability of Mining Success

- (PMF) What is the probability that a CPU solves PoW exactly at the k-th hash?

$$
\begin{aligned}
P_{pmf}(p,\ k) &:= P_p\{K \le k\} - P_p\{K \le k-1\} \\
&= P_p\{K = k\} \\
&= p + (1-p)p + (1-p)^2 p + \cdots + (1-p)^{k-1} p \\
&\quad - \left(p + (1-p)p + (1-p)^2 p + \cdots + (1-p)^{k-2} p\right) \\
&= (1-p)^{k-1} p \qquad \text{for any } k = 1, 2, 3, \cdots
\end{aligned}
$$

4 Probability of Mining Success

- Average no. of hashes for a PoW success
  - What is the average number of hashes for a PoW success for a given puzzle difficulty p?

$$
\begin{aligned}
\mathbb{E}\{K\} &= \sum_{k=1}^{\infty} P_{pmf}(p,\ k)\ k \\
&= \sum_{k=1}^{\infty} (1-p)^{k}\ p\ k \\
&= \frac{1}{p} \\
&= 10^{6} \quad \text{[hashes/block]}
\end{aligned}
$$

4 Probability of Mining Success

- $P_{geom}(p, k)$ is the CDF of PoW success in k (hash) hashes.
- Consider the distribution of no success in k hashes.

$$
\begin{aligned}
P_p\{K > k\} &= 1 - P_p\{K \le k\} \\
&= \sum_{j=1}^{k} (1-p)^{j-1} p \\
&= \sum_{j=k+1}^{\infty} (1-p)^{j-1} p \\
&= (1-p)^k \sum_{j=1}^{\infty} (1-p)^{j-1} p \\
&= (1-p)^k
\end{aligned}
$$

GIST

5 AI-IM-To-Po Theory

- Theorem 1. (Alone) The CDF $P_{geom}(p, k)$, the probability of PoW success in k hashes, can be expressed as

$$P_p\{K \le k\} = 1 - P_p\{K > k\}$$
$$= 1 - (1-p)^k.$$

5 AI-IM-To-Po Theory

- Let $P_1(p, k)$ be the probability that a CPU solves a PoW with p in k hashes.

- What is the probability that at least one CPU out of N CPUs finds a good block hash?

GIST

5 AI-IM-To-Po Theory

- Theorem 2. There are N CPUs working independently on the PoW puzzle with difficulty p. The probability P_2 that at least one CPU out of N finds a good block summary in k hashes is given by

$$
\begin{aligned}
P_2(N, p, k) &= \Pr\{\text{at least one CPU success}\} \\
&= 1 - \Pr\{\text{no CPU success}\} \\
&= 1 - [1 - P_1(p, k)]^N
\end{aligned}
$$

GIST

5 AI-IM-To-Po Theory

- Corollary 3. (All Together) There are $N = k$ CPUs which work independently on the PoW puzzle with difficulty p.
  The probability P_{all} that at least one CPU out of N finds a good block hash in a single hash is given by

$$P_{\text{all}}(N{=}k, p) = \mathrm{P}_2(N, p, k{=}1)$$
$$= 1 - \Pr\{\text{no CPU success}\}$$
$$= 1 - [1 - p]^N .$$

$$P_p\{K \le k\} = 1 - P_p\{K > k\}$$
$$= 1 - (1 - p)^k .$$

GIST

5 AI-IM-To-Po Theory

- From the Alone theorem and All-together corollary, one can notice that the distributions are the same, given $N = k$.

$$P_{geom}(p, k) = P_{\text{all}}(N{=}k, p)$$
$$= 1 - \Pr\{\text{no CPU success}\}$$
$$= 1 - [1 - p]^N$$

GIST

5 AI-IM-To-Po Theory

- Let the difficulty of puzzle be given with $p = 10^{-20}$.
- Assume a mining chip with hash rate $R_{chip} = 10^{12}$ hashes/sec.
- Give answers on the average numbers.

1. How many hashes does it take for this chip to make a success?
2. How long does it take for this chip to make a success?
3. How many chips do you need to make a success in a single unit of time?

GIST

5 AI-IM-To-Po Theory

- Let the difficulty of puzzle be given with $p = 10^{-20}$.
- Assume a mining chip with hash rate $R_{chip} = 10^{12}$ hashes/sec.

1. How many cycles does it take for this chip to make a success?

$E\{K\} = 10^{20}$ [hashes/block].

5 AI-IM-To-Po Theory

- Let the difficulty of puzzle be given with $p = 10^{-20}$.
- Assume a mining chip with hash rate $R_{chip} = 10^{12}$ hashes/sec.

2. How long time T_{block} for this chip to make a success?

$$T_{block} = E\{K\} / R_{chip}$$

$$= 10^{20}/10^{12} \text{ [sec/block]}$$

$$= 10^{8} \text{ [sec/block]}$$

$$= 3.15 \text{ [year/block]}$$

GIST

5 AI-IM-To-Po Theory

- Let the difficulty of puzzle be given with $p = 10^{-20}$.
- Assume a mining chip with hash rate $R_{chip} = 10^{12}$ hashes/sec.

3. How many chips do you need to make a success in a second?

$$T_{block} = E\{K\} / (R_{chip} \times N_{chip})$$

$$N_{chip} = E\{K\} / (R_{chip} \times T_{block})$$

$$= 10^{20} / (10^{12} \times 1)$$

$$= 10^{8}$$

$$= 100 \text{ Million}$$

GIST

5 AI-IM-To-Po Theory

- From previous examples, we now understand what we mean by the AI-IM-To-Po theory.
- The Alone-theorem shows that it takes about 3.15 years to a single PoW success, if a single chip is used.
- The All-together corollary indicates that it takes 100 Million such chips working together for a single PoW success.

Goal of this lecture note

- Bitcoin Difficulty
- History of Bitcoin Difficulty
- Geometric vs Exponential Distribution
- Block Generation Speed
- Double Spending Attack Possibility
- Data Immutability

1 Bitcoin Difficulty

- Finding Good Block Summary
  - PoW Inequality is given by

$$F(\text{BH}: nonce) < Target \qquad \text{PoW Ineq}$$

  - Find the first nonce that satisfies PoW Ineq.
  - Record it in to the block header, along with $Target$.
  - $Target$ specifies how difficult the puzzle was.

GIST

1 Bitcoin Difficulty

- Bitcoin Difficulty (D)

  - The aim is to keep the average block generation time be 10 min.
    - Ex) The time span to mine 2016 blocks is set to take 2 weeks.
  - Difficulty is adjusted for every 2016 block.
  - Measure the time span, T [min], during which the past 2016 blocks were mined.
  - Let T_D be 2 weeks [min], i.e., T_D = 2016x10 = 20160 [min].
  - If T is different from T_D, adjust the Difficulty D :

$$D = D_{prev} \times \frac{T_D}{T}$$

In Bitcoin, initial D is set to 1 with 8 leading hexa zeros.

1 Bitcoin Difficulty

- *Target* is defined to be inversely proportional to Difficulty D.
  - The measured time T is used to update Difficulty D.
  - Finally, a new *Target* is thus given by

$$Target = Target_0 \frac{1}{D}$$

  - $Target_0$ is set to $2^{256-32} = 2^{224}$ the maximum allowed target.
  - With $Target = 2^{224}$, all good hashes are smaller than the target and have 32 leading zero bits.

1 Bitcoin Difficulty

- *Target* can be directly updated by combining the two equations:

$$Target = Target_{prev} \frac{T}{T_D}$$

- Target is any real number in the interval from 2^1 to 2^{224}.
- Minimum difficulty is 2^{224}.
- Maximum difficulty is 2^1.

1 Bitcoin Difficulty

- *Target* is inversely proportional to Difficulty.
  - The smaller *Target* is, the more difficult the puzzle is.

1 Bitcoin Difficulty

- What shall be *Target* if all good hashes has 10 leading hexadecimal zeros?
  - 40 binary zeros.
  - Target shall be $2^{256-40} = 2^{216}$.

1 Bitcoin Difficulty

- Conversion from Hashes to Hash Rate req'd.
  - Given a *Target*, one can calculate the number of hashes (avg) to make a single PoW success.
- Let $\mathrm{Log2Target} = \log_2(\mathit{Target})$.
  - Number of hashes needed per success is $2^{256\text{-log2target}}$.
  - The network hash rate req'd to keep 10 min per success is:

$$\text{Hash Rate Req'd} = \frac{2^{256-\log 2\mathrm{Target}}}{600}[\text{hashes / sec}]$$

1 Bitcoin Difficulty

- Given a Target, one can determine the network hash rate.
- Suppose you bring your own mining chip.
- You can determine your chance of winning a puzzle.
- It is the ratio of your hash rate to the total hash rate:

$$= \frac{\text{Your Hash Rate}}{\text{Your Hash Rate} + \text{Network Hash Rate}}$$

1 Bitcoin Difficulty

Example

- Suppose Target is 2^{204}.
- You want to join with 1 Tera hash/sec mining chip.
- What is your chance of winning a block?
  - The network hash power is $2^{256-\log 2 \text{target}}/600 = 2^{52}/600 = 7.51e12$ [hash/sec].
  - The hash rate percentage is:

$$= \frac{\text{Your Hash Rate}}{\text{Your Hash Rate} + \text{Network Hash Rate}}$$

$$= \frac{1.00e12}{1.00e12 + 7.51e12}$$

$$= 11.8\%$$

1 Bitcoin Difficulty

- *Target* specifies how *difficult* the puzzle was.
- It represents the number of hashes needed to solve the puzzle.
- It represents how many number of computers worked together at that time.
- Nonce is the proof.
- Nonce and Target are recorded in the block header along with the time-stamp.
- One can verify if the proof-of-work for the block was correctly done or not.

1 Bitcoin Difficulty

- 블록 높이 516445 비트코인 블록체인 내 깊이 값 513445에서의 블록들

18 Leading Hexadecimal Zeros

요약	
높이	516445 (Main chain)
해시	000000000000000000004758013a1ed70036479f7d5038c19240afc9fd4710832b
이전 차단	
다음 블록	
시각	2018-04-03 12:40:12
수신 시간	
릴레이된 곳	
난이도	3,511,060,552,899.72
Bits	
거래 수	
출력 합계	
예상된 거래량	816.76804565 BTC
크기	1131.349 KB
번역	0x20000000
Merkle Root	5db080790c0433a7ec8c565932ea75fb7347b6873bc404b2e594f797d7762c10
해시 난수	1225863608
블록 보상	
거래 수수료	

해시 000000000000000000004758013a1ed70036479f7d5036c19240afc9fd4710832b

시간 2018년 4월 3일 12시 40분

난이도 3,511,060,522,899.72 → Log2Diff = 41.68
Target = 256 – (32+41.68) = 256 - 73.68 = $2^{182.32}$

Nonce 1225863608

GIST

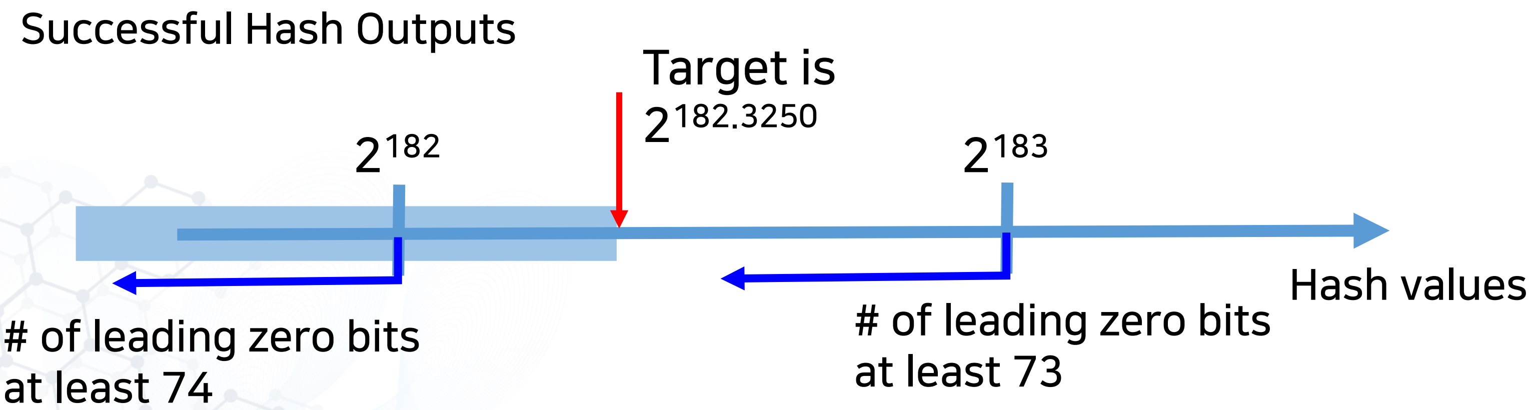
1 Bitcoin Difficulty

- Example of Difficulty and Target
  - Block #516445
  - BlockHash 0000 0000 0000 0000 0047 5801 … … … 832b
    - 18 hex zeros * 4 bits/hex + 1 bit = 72 + 1 = 73 bit zeros
  - Difficulty D is 3,511,060,552,899.7197 = 3.5e12
  - Target is $Target_0$ * (1/Difficulty)
    - Log2(D) = 41.68
    - Target = $2^{224.000}\ 2^{-41.675}$
      $= 2^{182.325}$

GIST

1 Bitcoin Difficulty

Recall PoW Success
is
SHA Hash Output < Target



1 Bitcoin Difficulty

- Network Hash Rate : Block#516445
- With D=3.5e12, the probability p is about $2^{-(32+41.675)} = 2^{-73.6750}$.
- Then, it would take $1/p = 2^{73.6750}$ ~ 1.5080 e22 hashes to mine a single block.
- Dividing it by 10 min = 600 sec, the network hash rate is obtained, 25.1332 Exa hash/sec.

Bitcoin Difficulty

Example

- Calculate no. of Antminer S9s (14 Thps) you need bring to obtain hash power 0.01 %, given the network hash rate is 25 Exa hash/sec.
  - You need to bring at least 179 AS9 chips.

$$\text{Your Hash Rate} \geq \frac{0.01\% \ \ \text{Network Hash Rate}}{100\% - 0.01\%}$$

$$= \frac{1}{9999} 25e18 = 25e14$$

$$= 178.6(14e12)$$

1 Bitcoin Difficulty

Example

- Given the network hash rate is 25 Exa hash/sec, further questions can be asked.
  - What is the least number of mining chips working in the network?
  - How long does it take for a single mining chip to find a good PoW?

2 History of Bitcoin difficulty

- Bitcoin Hash Rate vs Difficulty

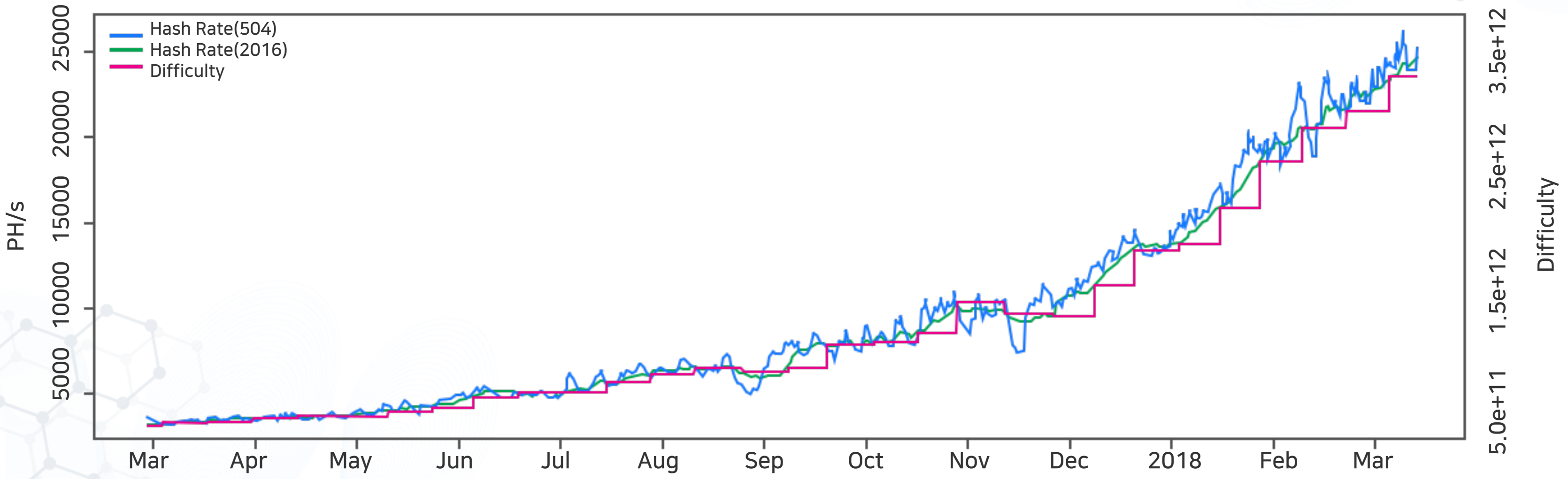
Date	Difficulty	Hash Rate
Apr 01 2018	3,511,060,552,899	25,133,150,415 GH/s
Oct 04 2018	7,454,968,648,263	53,364,744,228 GH/s
Mar 24 2019	6,379,265,451,411	45,664,560,811 GH/s

출처: https://bitcoinwisdom.com/bitcoin/difficulty

GIST

2 History of Bitcoin difficulty

- Bitcoin Hash Rate vs Difficulty (Mar/17 ~ Apr 18)



출처: https://bitcoinwisdom.com/bitcoin/difficulty

3 Geometric vs Exponential Distribution

- Recall the Alone theorem, the probability of PoW success in k hashes is expressed with the per-hash success probability p.
- We now aim to improve it by embedding the concept of time into it.
- Then, we will get the *block generation* speed.
  - Given a unit time one can determine probabilistically the number of PoW successes or the number of blocks formed.

3 Geometric vs Exponential Distribution

- Theorem 1. (Alone) The CDF $P_{geom}(p, k)$, the probability of PoW successes in k hashes, can be expressed for k = 1, 2, 3, ⋯, as

$$P_p\{K \leq k\} = 1 - P_p\{K > k\}$$
$$= 1 - (1 - p)^k$$

3 Geometric vs Exponential Distribution

- To the result of Theorem 1, we aim to put the time into consideration.
- For this, we define a new random variable S.
- *Recall* K is the random time index at which duration the PoW success occurs.

3 Geometric vs Exponential Distribution

- Geometric distribution(p) ~ Exponential distribution(p, T)
  - Let T here be the time-duration per single hash generation.
  - For a fast CPU, T be very small.
    - For example, 1 Tera hash/sec, T = 1e-12 sec/hash.

3 Geometric vs Exponential Distribution

- Geometric distribution(p) ~ Exponential distribution(p,T)
  - Let $S = KT$.
  - Then, S denotes the random time-epoch at which the PoW success occurs.

3 Geometric vs Exponential Distribution

- Geometric distribution(p) ~ Exponential distribution(p, T)
  - Let $S = KT$.
  - Then, S denotes the random time-epoch at which the PoW success occurs.

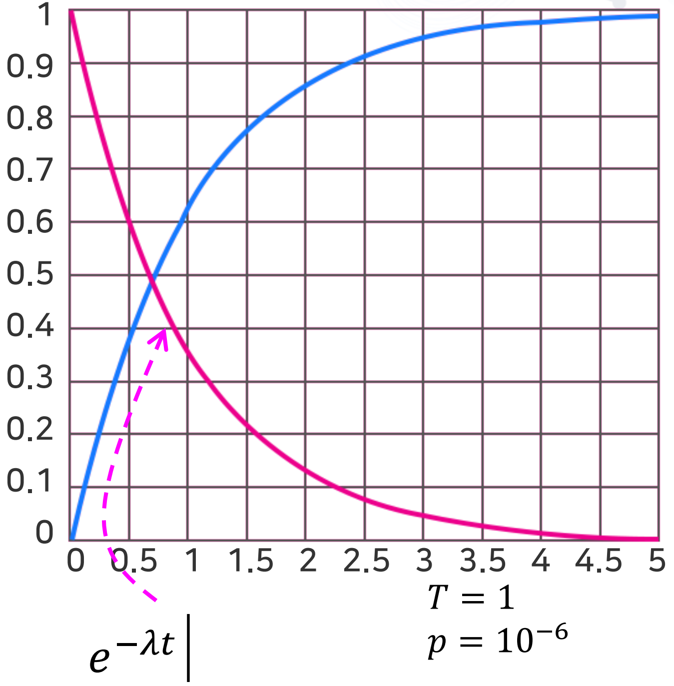
$$\Pr_p\{K \le k\} = \Pr_p\{KT \le kT\}$$
$$=: \Pr_p\{S \le kT\}$$
$$= \Pr_p\{S \le t\}$$

3 Geometric vs Exponential Distribution

- $\Pr(S > t) = e^{-\lambda t}$ where $\lambda = \frac{p}{T}$

$$P_p\{K > k\} = (1-p)^{\frac{1}{T}kT}$$
$$= (1-p)^{\frac{1}{p}\frac{p}{T}kT}$$
$$= \left\{(1-p)^{\frac{1}{p}}\right\}^{\frac{p}{T}kT}$$
$$= e^{-\frac{p}{T}kT}$$
$$= e^{-\lambda t}\Big|_{t = kT}$$

Geometric distribution
& Exponential distribution



GIST

3 Geometric vs Exponential Distribution

- $\Pr(S > t) = e^{-\lambda t}$ where $\lambda = \frac{p}{T}$
- We now aim to determine lambda.
  - Suppose a mining chip with hash rate $R_{chip}=10^{12}$ [hashes/sec].
  - The time duration per hash is 1 pico $T = 10^{-12}$ [sec/hash].
  - We now treat the time is continuous.

GIST

3 Geometric vs Exponential Distribution

- $\Pr(S > t) = e^{-\lambda t}$ where $\lambda = \frac{p}{T}$
- We now aim to determine lambda.
  - Recall the average number of hashes for a PoW success or a block generation is $E\{K\}=1/p=10^{20}$ [hash/block].
  - Thus, lambda's unit is [block/hash]/[sec/hash] = [block/sec].

GIST

3 Geometric vs Exponential Distribution

- Lambda is the block generation speed.
- Recall

$$
\begin{aligned}
T_{block} &= \mathrm{E}\{K\} / R_{chip} \\
&= 10^{20} / 10^{12} \text{ [sec/block]} \\
&= 10^{8} \text{ [sec/block]} \\
&= 3.15 \text{ [year/block]} \\
&= 1/\lambda
\end{aligned}
$$

Thus, lambda is block generation speed!

4 Block Generation Speed

- Network Hash Power vs. Block Generation Speed
  - A Bitcoin network's hash rate is the total mining rate of all online nodes.
  - Suppose the whole network is divided into two pools of computers, say pool A and pool B.
  - The hash rate of pool A is twice that of pool B.
  - What is the block generation speed of A?

4 Block Generation Speed

- Note that

$$\lambda_A + \lambda_B = 1 \quad [\text{block} / 10 \text{ min}].$$

- Since the hash power of pool A is twice that of pool B, the block generation speed of pool A is twice faster than that of pool B, i.e.,

$$\lambda_A = 2\lambda_B .$$

4 Block Generation Speed

- Thus, the block generation speed of B is

$$\lambda_{\mathrm{B}} = 1/3 \text{ [block / 10 min].}$$

- Then, that of A is

$$\lambda_{\mathrm{A}} = 2/3 \text{ [block / 10 min].}$$

5 Double Spending Attack Possibility

- 51% Double Spending Attack and its Possibility
  - Recall our subnet example where Bitcoin network is divided into subnet A and subset B.
  - Suppose the hash power of A is greater than that of B.
  - And, the attacker took the control of A.
  - The honest nodes are in B.
  - In this case, the Double Spending attacks launched by A are possible.
  - The probability of DS success can be calculated exactly.

5 Double Spending Attack Possibility

- Immutable File Keeping Technology
  - It is, according to the Bitcoin white paper, an unlikely event to have such an attacker with a sizeable pool of computers working for him in the network of decentralized and independent participants.

6 Data Immutability

- Proof of Work and Data Immutability
  - Proof of work(작업증명 in Korean) is to have a large set of miners find a solution satisfying the PoW with the given difficulty.
  - The first miner which succeeds in solving it obtains the right to produce a certain amount of new coins minted and paid to himself.

6 Data Immutability

- Proof of Work and Data Immutability
  - It is the key mechanism for enforcing integrity of data stored inside the blockchain.
  - Blockchain can be considered as a very large stone everyone can see!
  - Each and every transaction is checked for validity and scribed into the stone.
  - How can it be done with digital file?

6 Data Immutability

- Proof of Work and Data Immutability

Answer is simple!

- Let a large number of computers work together simultaneously. Let the first computer which is successful at finding a good answer get rewarded.
- Have a new race begin by having the computers work on a new problem (new block) and reward the new winner.
- The proof of work is an evidence that a large number of computers have worked together.
- If any computer, or a group of computers, aims to change the block content, then the same amount of work needs to be redone.

6 Data Immutability

- Immutable File Keeping Technology
  - The problem can almost never be solved alone, but it is designed in such a way that it can be solved within a desired time span when many computers come and compete to find a solution.
  - It also has a means to measure the total amount of work done in probabilistic sense.
  - If the difficulty level of the problem is increased, the number of computers in competition has to increase as well.

6 Data Immutability

- Immutable File Keeping Technology
  - This is used to protect the integrity of the data stored in the Blockchain. Because of the Al-Im-To-Po result, a small group cannot fool the majority.
  - PoW is to find the nonce or the block header (BH) which matches with the block content and have this nonce written into the block header.
  - Why those transactions once scribed inside blockchain are not alterable?
  - The block contents are locked with the nonce.

6 Data Immutability

- Immutable File Keeping Technology
  - When the block content is changed somehow, the content no longer matches with the nonce found.
  - Such blocks are easily detectable and thus a chain containing such block are also easily detectable and thrown away.
  - Thus, anybody who aims to launch an attack of changing the content, the person needs to redo the PoW again and find a new nonce reflecting the changed block content.

6 Data Immutability

- Immutable File Keeping Technology

  But it is not the end

  - The hash value of the previous block, F(block, nonce) in (PoW), is written inside the header of the next block.
  - Blocks are connected in a serial fashion with these hash values.
  - Thus, if an attacker aims to change the content of a block, he has to re-do all the block headers subsequent to the altered one.
  - This requires the attacker to redo all the PoWs for the subsequent blocks.

6 Data Immutability

- Immutable File Keeping Technology
  - Recalling that it is very difficult to find the nonce for a single block alone, it becomes almost impossible for a single computer to find all the nonces again for the subsequent series of blocks.
  - In addition, the honest nodes are continuously making new blocks.
  - Thus, if an attack wants to be successful, he needs to recruit computing resource with a hash power greater than that of honest nodes.

Goal of this lecture note

- Transactions
- Time-Stamp Server
- Estonian Blockchain
- Proof-of-Work

1 Transactions

- Bitcoin Transactions
  - Bitcoin is a chain of signatures.

> "We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin.
> A payee can verify the signatures to verify the chain of ownership."

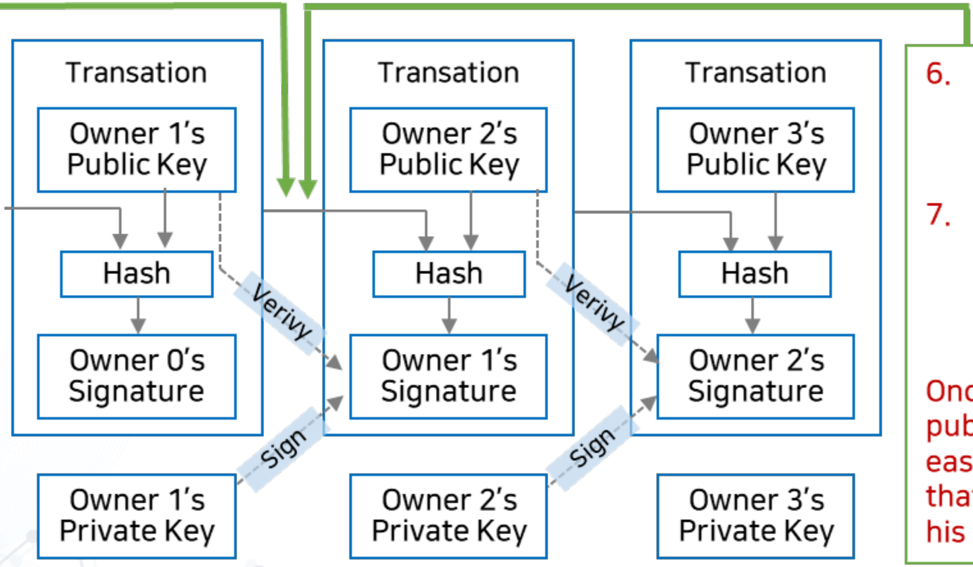
  - In the sequel, quoted sentences in box are from Satoshi.

1 Transactions

- Bitcoin Transactions
  - Bitcoin is a chain of signatures.

2nd Tx box

1. The first TX shows that O1 owns the coin
2. O1 can transfer it to anybody, say O2
3. O1 writes TX 2
4. O1 asks Owner2, the new owner, for his public key
5. O1 hashes the received public key and TX1, and calculates the hash value



6. To show his ownership status, O1 signs the hash value and leaves the signature in TX2
7. Now, anybody can verify O1's signature with O1's pubic key written in TX 1

Once TX 2 recorded and published, anybody can easily see TX 2 and knows that O1 has transferred his coin ownership to O2

GIST

1 Transactions

- Double Spending Problem

"The problem of course is the payee can't verify that one of owners did not double-spend the coin.
A common solution is to introduce a trusted central authority, or mint that checks every transaction for double spending.
After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent.
The problem with the solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank."

GIST

1 Transactions

- Double Spending Problem

"We need a way for the payee to know the previous owners did not sign earlier transactions.
For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend.
The only way to confirm the absence of a transaction is to be aware of all transactions.
In the mint based model, the mint was aware of all transactions and decided which arrived first."

GIST

1 Transactions

- Double Spending Problem

"To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received.
The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received."

[1] W.Dai, "b-money", http://weidai.com/bmoney.txt, 1988.

GIST

2 Timestamp Server

- Timestamp Server
  - The solution Bitcoin propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5].

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements, "In 20th Symposium on Information Theory in the Benelux, May 19993
[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document, " In Journal of Cryptology, vol 3, no 2, pages 00-111, 1991.
[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," in Sequences II : Methods in Communication, Security and Computer Science, pages 329-334, 1993.
[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 28-35, April 1997.

GIST

2 Timestamp Server

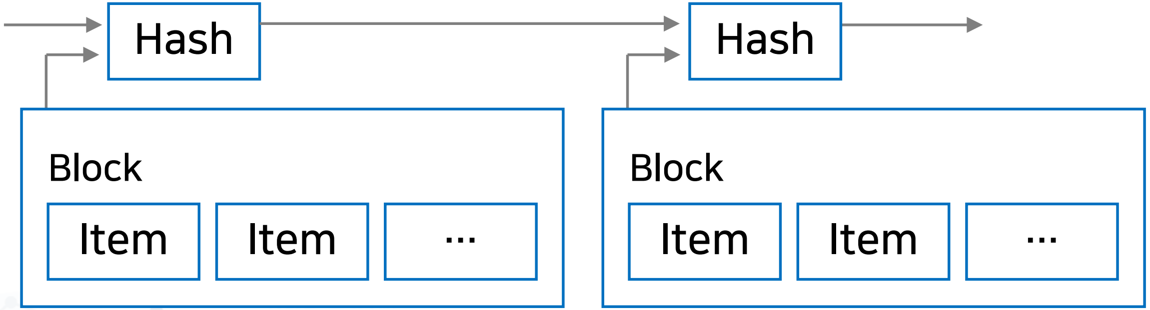
- Timestamp Server

> "The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash.
> Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it."

GIST

2 Timestamp Server

- Timestamp Server



2 Timestamp Server

- Timestamp Server
  - If a timestamp server indicates the existence of hash value at a certain time point, then a legitimate ledger can indeed be made?
    - If hash values only are published while no block contents are published, there will be no issue of scalability, and privacy can be kept since no one other than the parties involved in the transactions can see the content of transactions!
    - But how can one verify for coin ownership and double spending transactions?

GIST

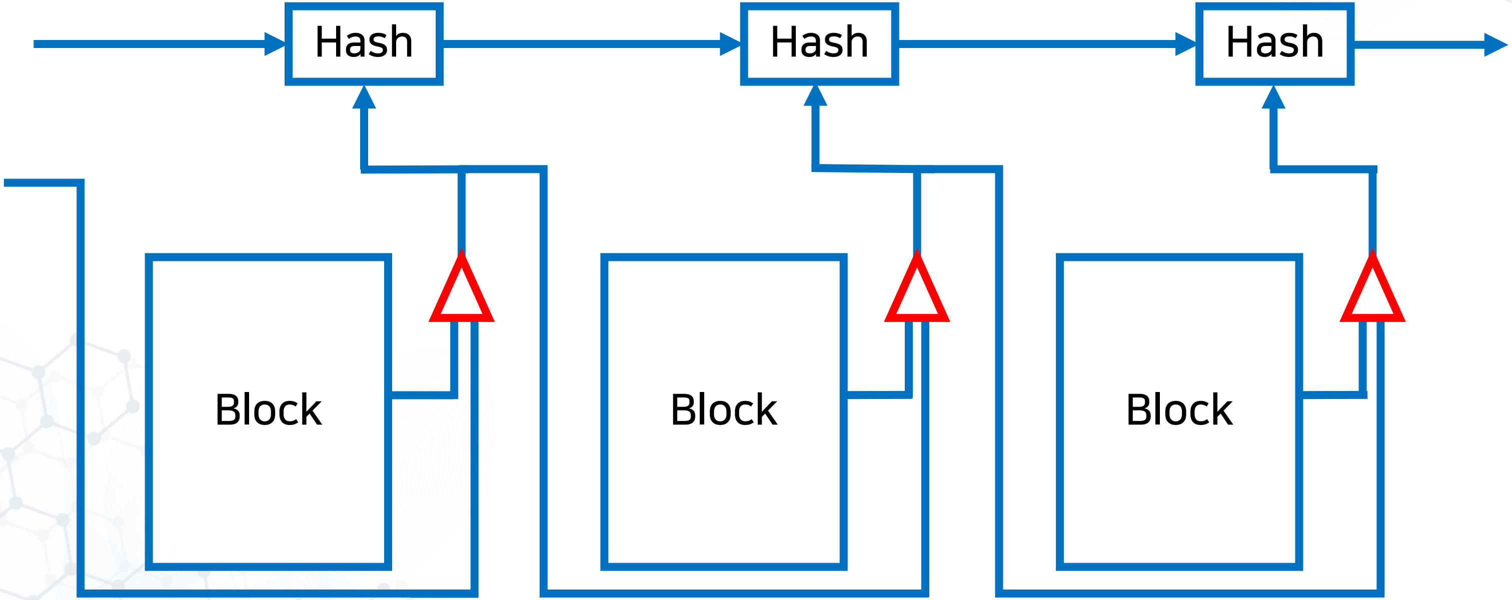
2 Timestamp Server

- Timestamp Server
  - The problem is to decide who should run the timestamp server?
  - If a government runs it, it becomes a private Blockchain (while social terms it is a public chain)!
  - What possible problems are there if it is run by government?

GIST

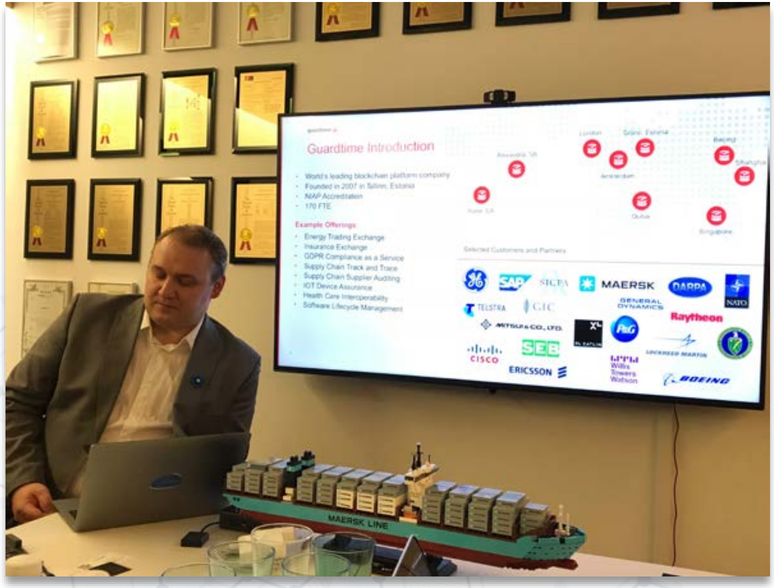
2 Timestamp Server

- Using an off-chain timestamp server



3 Estonian Blockchain

- Estonian Blockchain



전명산, '비트코인 이전에 에스토니아에 블록체인이 있었다.' 공공분야 블록체인 현장 르포_ #7: 에스토니아 정부와 가드타임의 도전. 2018.12.06 (https://www.coindeskkorea.com/33836/)

GIST

3 Estonian Blockchain

- Guardtime publishes its hashes regularly.

MARKETS, 2008. 09. 18. / https://guardtime.com/blog/6-reasons-why-encryption-isnt-working

GIST

3 Estonian Blockchain

- Estonian Blockchain is KSI.
  - "A blockchain is a distributed public record of events; an append-only record of events where each new event is cryptographically linked to the previous. New entries are created using a distributed consensus protocol.
  - This blockchain overcomes two major weaknesses of traditional blockchains, making it usable at industrial scale:" (Guardtime)

출처: guardtime federal, https://www.guardtime-federal.com/ksi/

3 Estonian Blockchain

Scalability

- One of the most significant challenges with traditional blockchain approaches is scalability – they scale at O(n) complexity i.e. they grow linearly with the number of transactions.
- In contrast the E. blockchain scales at O(t) complexity – it grows linearly with time and independently from the number of transactions.

출처: guardtime federal, https://www.guardtime-federal.com/ksi/

3 Estonian Blockchain

Settlement time

- In contrast to the widely distributed crypto-currency approach, the number of participants in KSI blockchain distributed consensus protocol is limited.
- By limiting the number of participants it becomes possible to achieve consensus synchronously, eliminating the need for Proof of Work and ensuring settlement can occur within one second.

출처: guardtime federal, https://www.guardtime-federal.com/ksi/

GIST

3 Estonian Blockchain

Data Privacy

- KSI does not ingest any customer data; data never leaves the customer premises.
- Instead the system is based on one-way cryptographic hash functions that result in hash values uniquely representing the data, but are irreversible such that one cannot start with the hash value and reconstruct the data – data privacy is guaranteed at all times.

출처: guardtime federal, https://www.guardtime-federal.com/ksi/

GIST

4 Proof-of-Work

- Proof-of-Work

"To implement a distributed timestamp server on a peer-to peer basis, Bitcoin uses a proof-of work system similar to Adam Back's Hashcash[6], rather than newspaper or Usenet posts.
The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits."

[6] A. Back, "Hashcash – a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

GIST

4 Proof-of-Work

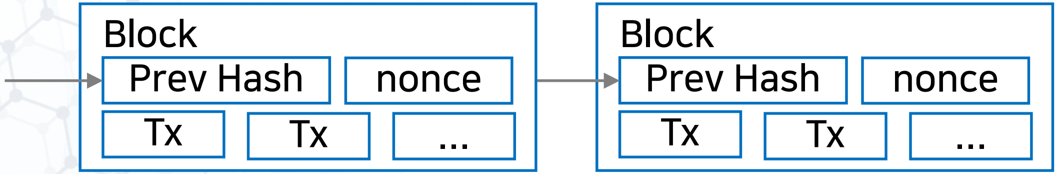
- Proof-of-Work

"The average work required is exponential in the number of zero bits required and can be verified by executing a single hash."
"For the timestamp network, Bitcoin implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits."

4 Proof-of-Work

- Proof-of-Work

> "Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it."



GIST

4 Proof-of-Work

- Proof-of-Work

> "The proof-of work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote."

4 Proof-of-Work

- Proof-of-Work

> "The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains."

GIST

4 Proof-of-Work

- Proof-of-Work

“To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes.”
“We will show that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.”

4 Proof-of-Work

- Proof-of-Work

"To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of work difficulty is determined by a moving average targeting an average number of blocks per hour.
If they're generated too fast, the difficulty increases."

GIST

4 Proof-of-Work

- Proof-of-Work
  - Aim to make a timestamp Server in a P2P network

Why?

- Not to rely on any central authority.
- Central authority such as banks and states.
- Within a nation, the state government may run the timestamp server.
- But for trades overseas, P2P across different nations is needed.

4 Proof-of-Work

- Proof-of-Work
  - Solution?
    - Distributed timestamp P2P server network.
    - Distributed, thus, it is difficult to maintain the integrity of data.
    - To keep the integrity of data, PoW system is proposed!

Goal of this lecture note

- Network
- Blockchain Scalability
- Block Header
- Consensus
- Payment and Change
- Privacy

1 Network

- Network
  - The steps to run the network are as follows

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

GIST

1 Network

- Network

"Nodes always consider the longest chain to be the correct one and will keep working on extending it.
If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first."

GIST

1 Network

- Network

"In that case, they work on the first one they received, but save the other branch in case it becomes longer.
The tie will be broken when the next proof-of work is found and one branch becomes longer, the nodes that were working on the other branch will then switch to the longer one."

1 Network

- Network

“New transaction broadcasts do not necessarily need to reach all nodes.
As long as they reach many nodes, they will get into a block before long.
Block broadcasts are also tolerant of dropped messages.
If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.”

1 Network

- Network
  - Are there any guarantee for transactions to be included into blocks?
  - With a large incentive(tx fee), a tx can be put on high priority.
  - But if the production rate of txs is higher than the service rate, then there must be some transactions not to end up in the blockchain.

2 Blockchain Scalability

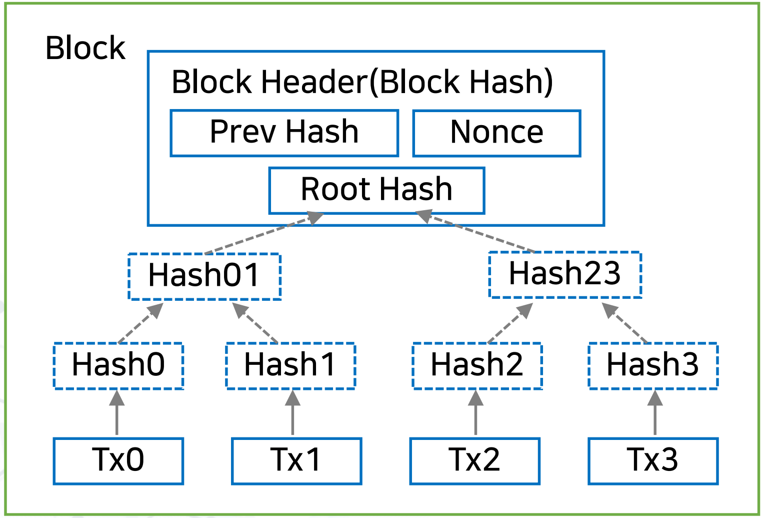
- Reclaiming Disk Space

"Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree[7][2][5], with only the root included in the block's hash.
Old blocks can then be compacted by stubbing off branches of the tree.
The interior hashes do not need to be stored."

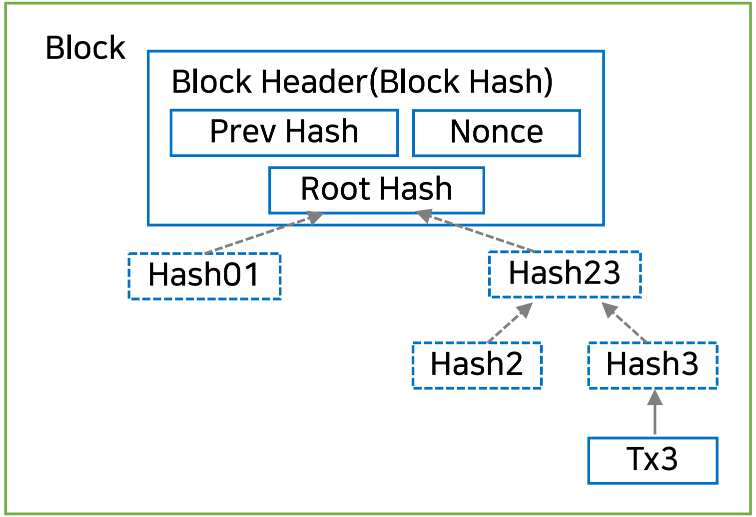
GIST

2 Blockchain Scalability

- Blockchain Scalability
  - Reclaiming Disk Space



Transaction Hashed in a Merkle Tree



After Pruning Tx0-2 from Block

GIST

2 Blockchain Scalability

- Reclaiming Disk Space

"A block header with no transactions would be about 80 bytes.
If we suppose blocks are generated every 10 minutes, 80 bytes*6*24*365=4.2MB per year.
With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even is the block headers must be kept in memory."

GIST

2 Blockchain Scalability

- Blockchain Scalability
  - Use Merkle tree and save disk space.
  - Save the blockhash in the header.
  - Those tree branches recording past transactions are erased but the hash values are kept.
  - 80 byte Blockheader

(1) Prev hash
: 256 bit = $2^8 = 2^{5*(2\text{^}3)} = 2^5$ Bytes = 32 Bytes
(2) Roothash = 32 Bytes
(3) Nonce = 4 Bytes = 32 bit
(4) Time
(5) Difficulty
(6) version

GIST

3 Block Header

- 80 Byte Block Header

Bytes	Name	Data Type	Description
4	version	int32_t	Indicates which set of block validation rules to follow.
32	previous block header hash	char[32]	A SHA256(SHA256()) hash in internal byte order of the previous block's header. This ensures no previous block can be changed without also changing this block's header.
32	merkle roothash	char[32]	A SHA256(SHA256()) hash in internal byte order. The merkle root is derived from the hashes of all transactions included in this block, ensuring that none of those transactions can be modified without modifying the header.

출처:: https://bitcoin.org/en/developer-reference#block-headers

GIST

3 Block Header

- 80 Byte Block Header

Bytes	Name	Data Type	Description
4	time	uint32_t	The block time is a Unix epoch time when the miner started hashing the header (according to the miner). Must be strictly greater than the median time of the previous 11 blocks. Full nodes will not accept blocks with headers more than two hours in the future according to their clock.
4	nBits	uint32_t	An encoded version of the target threshold this block's header hash must be less than or equal to. See the nBits format described below.
4	nonce	uint32_t	An arbitrary number miners change to modify the header hash in order to produce a hash less than or equal to the target threshold. If all 32-bit values are tested, the time can be updated or the coinbase transaction can be changed and the merkle root updated.

출처:: https://bitcoin.org/en/developer-reference#block-headers

GIST

4 Consensus

- It is a way to resolve a conflict.
- Longest chain is trusted
  - Simplified Payment Verification

> "It is possible to verify payments running a full network node.
> A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in."

GIST

4 Consensus

- Longest chain is trusted
  - Simplified Payment Verification

> "He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it."

5 Payment and Change

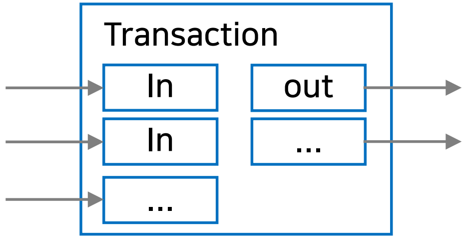
- Payment and changes
  - Combining and Splitting Value

"Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender."

GIST

5 Payment and Change

- Payment and changes
  - Combining and Splitting Value



How to get the change?

GIST

6 Privacy

- Privacy, by Anonymous Pub Key
  - Privacy

> "The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes the method, but privacy can still be maintained by breaking the flow of information in another place by keeping public keys anonymous."

GIST

6 Privacy

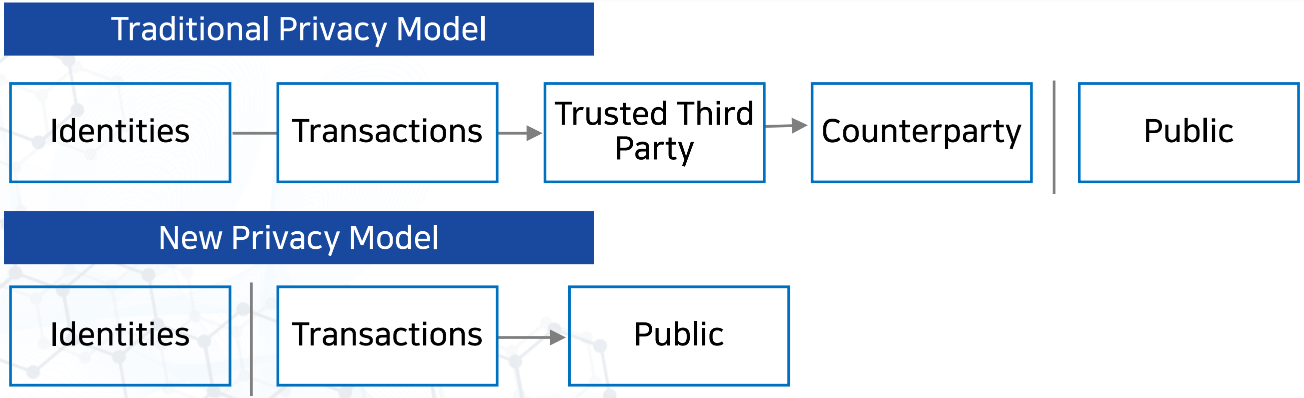
- Privacy, by Anonymous Pub Key
  - Privacy

> "The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone."

6 Privacy

- Privacy, by Anonymous Pub Key
  - Privacy

> "This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the 'tape', is made public, but without telling who the parties were."



GIST

6 Privacy

- Privacy, by Anonymous Pub Key
  - Privacy

> "As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner.
> Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner.
> The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner."

GIST

6 Privacy

- Privacy, by Anonymous Pub Key
  - Blockchain is published.
  - Privacy is maintained
    by keeping public key anonymous!
  - Additional privacy
    by using new public key per transaction!

Goal of this lecture note

- Attacker vs Honest Nodes
- From Hash Rate Ratio to Mining Probabilities
- Number of Blocks Minded during an Interval is Poisson
- Double Spending Attack
- Gambler's Ruin Problem
- Attack Success Probability

1 Attacker vs Honest Nodes

- Recall honest network guards the blockchain.
- Honest network's hash rate is published in the blockchain in the form of Target.
- Block generation speed is 1 block/600 sec.
- Suppose attacker's hash rate is slightly greater than honest network's.
- Then, the attacker can launch a 51% attack.
- We aim to calculate the probability of Double Spending success.

1 Attacker vs Honest Nodes

- From Target, the hash rate of honest network can be obtained.
- Lambda is 1 block/10 min.
- Given attacker's hash rate, attacker's lambda can be determined.
- Once we obtained the two parameters, given a new block mined, we can assign probability to which network the new block belongs.

2 From Hash Rate Ratio to Mining Probabilities

- Suppose Honest network hash rate R_H = 30 E hash/sec.
- Attacker's hash rate R_A = 10 E hash/sec.
  - Let p be the probability that given a new block is formed, the new block belongs to Honest chain.
  - Let q be the probability that given a new block is formed, the new block belongs to Attacker's chain.

GIST

2 From Hash Rate Ratio to Mining Probabilities

- Overall hash rate = 40 E hash/sec.
- Overall block generation speed is (4/3) block/10-min.

$$\lambda_{all} = \lambda_H + \lambda_A$$

- $\lambda_H = 1$ block/10-min
- $\lambda_A = \frac{1}{3}$ block/10-min

GIST

2 From Hash Rate Ratio to Mining Probabilities

- Each time a new block is formed, it belongs either to the Attacker's chain or to the Honest chain.
- The probability is given by

$$q = \frac{\lambda_A}{\lambda_H + \lambda_A}$$

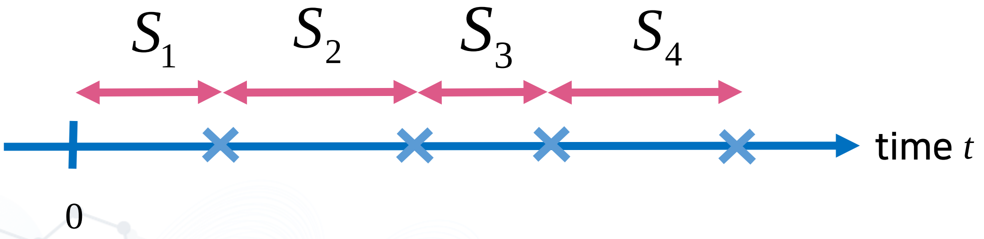
(Note also $\frac{q}{p} = \frac{R_A}{R_H}$)

$$p = \frac{\lambda_H}{\lambda_H + \lambda_A}$$

$$p + q = 1$$

3 Number of Blocks Minded during an Interval is Poisson

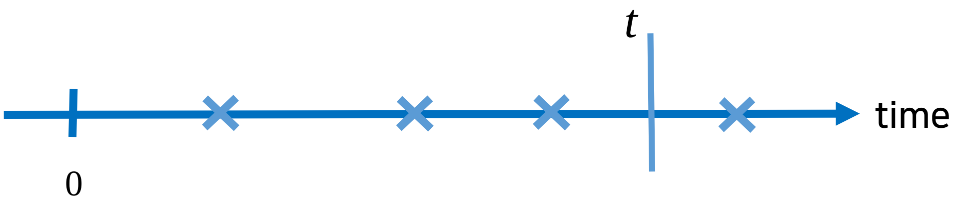
- We now aim to know the distribution of number of blocks generated within a given time $t > 0$.



$$T_k^S := \sum_{i=1}^{k} S_i$$

3 Number of Blocks Minded during an Interval is Poisson

- We now aim to know the distribution of number of blocks generated within a given time $t > 0$.



$$P_\lambda\{k \text{ blocks in interval } t\} = e^{-\lambda t}\frac{(\lambda t)^k}{k!}$$

$$k = 1,2,3,\ldots$$

GIST

4 Double Spending Race Attack

- **Definition** Double Spending Race Attack
  - Suppose A is the attacker.
  - B is the recipient.
  - B waits for z blocks. (Block confirmation)
  - Honest network's hash rate R_H
  - Attacker's hash rate R_A

GIST

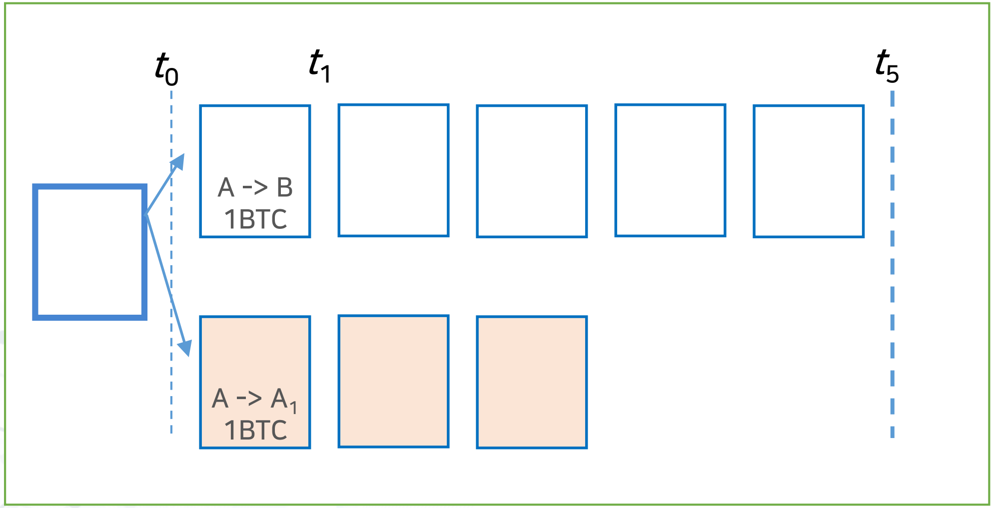
4 Double Spending Race Attack

- **Definition** Double Spending Race Attack
  - Let $z = 5$ be block confirmation number.
  - A announces a TX showing A sends B 1 BTC at time t_0.
  - This TX gets into a block (1 confirmation) at t_1.
  - B waits until he gets 5th confirmation which occurs at t_5.
  - A starts preparation in secret for his double spend attack at t_0.
  - Namely, A grows its own chain.
    His chain has replaced the TX A ->B $1\mathrm{BTC}$ with a fake TX, A -> A_1 $1\mathrm{BTC}$. A_1 is another public key of A.
  - At t_5, A has mined 3 blocks and needs to decide if he continues to grow his own chain or not.

GIST

4 Double Spending Race Attack

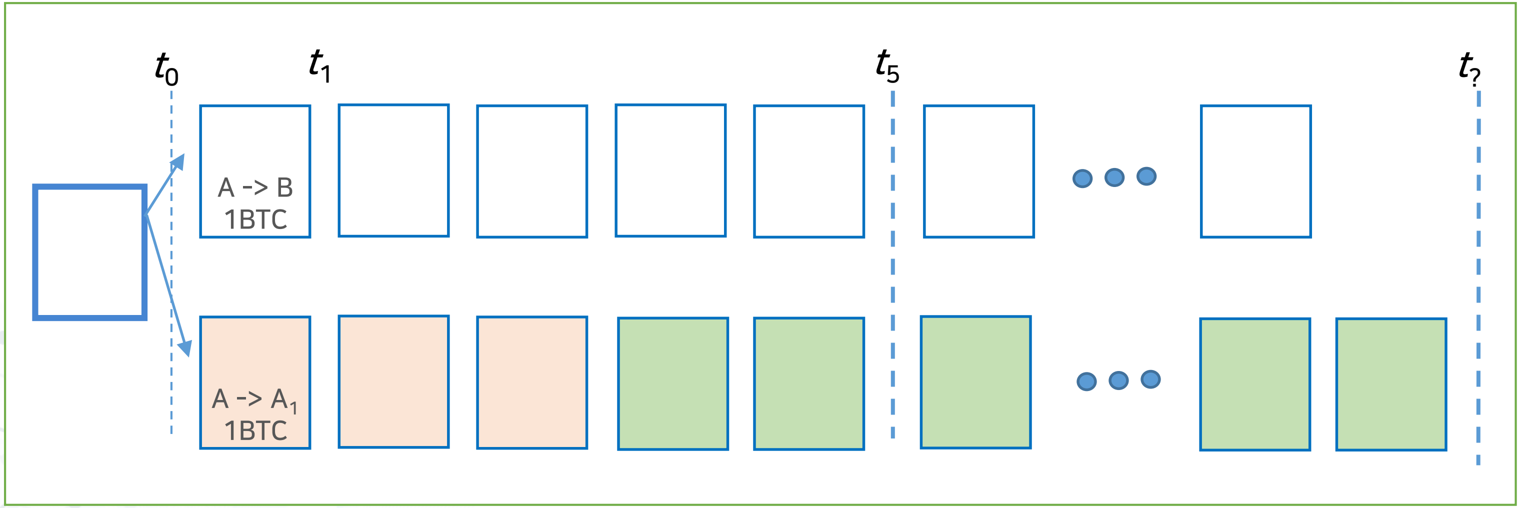
- Double Spending Race Attack: Race begins.



GIST

4 Double Spending Race Attack

- Double Spending Race Attack: Success



Chain is announced!

GIST

4 Double Spending Race Attack

- **Definition** Double Spending Race Attack
  - The probability calculation has two phases.
  - First phase is the time interval in which the honest node mines z blocks.
    - Assume that the attacker has added k blocks to his chain.
    - Attacker's chain is thus $z - k$ blocks behind the honest chain.

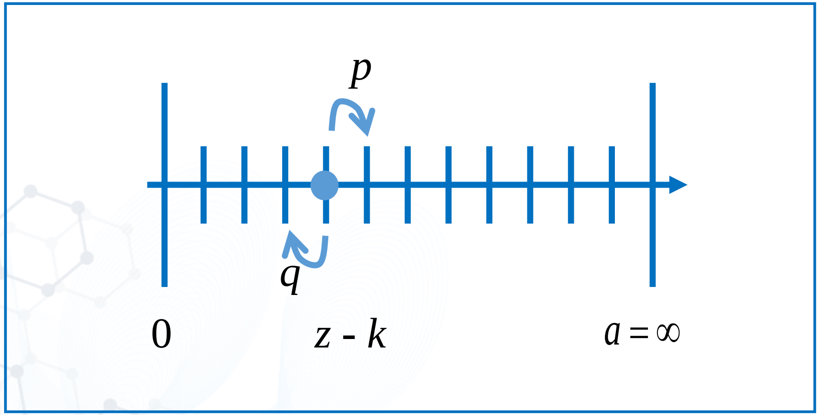
GIST

4 Double Spending Race Attack

- **Definition** Double Spending Race Attack
  - The probability calculation has two phases.
  - First phase is the time interval in which the honest node mines z blocks.
  - Second phase begins at the end of the first phase.
    - We aim to calculate the probability that the attacker catches up with the honest chain.

4 Double Spending Race Attack

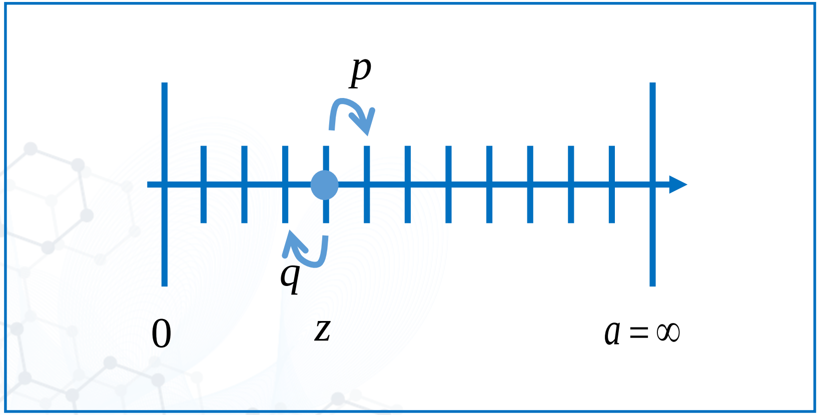
- Race begins with $z - k$ blocks behind.
  - When a new block mined belongs to the attacker with prob. q, move left.



GIST

5 Gambler's Ruin Problem

- Gambler's Ruin Problem
  - Feller's Gambler ruin result(Feller, vol.1, page 347)
  - Let z be the starting asset of the Gambler.



GIST

5 Gambler's Ruin Problem

- Feller's Gambler ruin result(Feller, vol.1, page 347)
  - There is a gambler who wins a dollar with probability p and loses with probability q in a game, i.e., $p + q = 1$.
  - Gambler starts with z dollars.
  - Gambler plays the game repeatedly against the dealer who has $a - z$ dollars, i.e., $a \geq z$.

GIST

5 Gambler's Ruin Problem

- Feller's Gambler ruin result(Feller, vol.1, page 347)
  - The probability q_z of the gambler's ultimate ruin (loses all his money).
  - Let p_z the probability of the gambler's ultimate winning.
  - Note $p_z + q_z = 1$.

GIST

5 Gambler's Ruin Problem

- Attack on the Mining Pool of Bitcoin and How to avoid?
  - Figure 1: The Gambler's Ruin Problem
    - The gambler starts with z dollars and the dealer with $a - z$ dollars.
    - Gambler wins a trial with probability p and loses with $q = 1 - p$.

1 After the first trial, the gambler's fortune is either increased by 1, $z+1$, or decreased by 1, z -1
Thus, we have

$$q_z = pq_{z+1} + qq_{z-1} \text{ for } 0 < z < a \qquad (1.1)$$

(with $q_0 = 1$ and $q_a = 0$)

GIST

5 Gambler's Ruin Problem

- Attack on the Mining Pool of Bitcoin and How to avoid?
  - Figure 1: The Gambler's Ruin Problem

2 Solving the difference equation Eq. (1.1), the result is obtained as

$$q_z = \frac{(q/p)^a - (q/p)^z}{(q/p)^a - 1} \qquad (1.2)$$

GIST

5 Gambler's Ruin Problem

- Attack on the Mining Pool of Bitcoin and How to avoid?
  - Figure 1: The Gambler's Ruin Problem

3 Letting $a \to \infty$,

$$q_z = \lim_{a\to\infty} \frac{(q/p)^a - (q/p)^z}{(q/p)^a - 1} = \lim_{a\to\infty} \frac{1-(q/p)^z (q/p)^{-a}}{1-(q/p)^{-a}} = \lim_{a\to\infty} \frac{1-(q/p)^{z-a}}{1-(q/p)^{-a}} = \begin{cases} 1 & if\ q \geq p \\ (q/p)^z & if\ q < p \end{cases} \quad (1.3)$$

GIST

5 Gambler's Ruin Problem

- During z blocks added by the honest nodes, the number of blocks k mined by the attacker is Poisson.
- Given $z-k$ blocks behind, the attack can catch up in 2nd phase.
- Let $z \rightarrow z-k$ in (1.3).

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{Bmatrix} (q/p)^{(z-k)} & if\ k \leq z \\ 1 & if\ k > z \end{Bmatrix}$$

GIST

5 Gambler's Ruin Problem

- Given z blocks added by the honest nodes, what is the average number of blocks mined by the attacker?
- The ratio is $z : p = \ ? : q$.

$$\lambda = z\frac{q}{p}$$

GIST

6 Attack Success Probability

- Gambler's ruin(z) → Replace $z = z - k$ for Attack Success Probability $(q, z-k)$

$$\sim \sum_{k=0}^{\infty} \begin{Bmatrix} (q/p)^{z-k} & k < z \\ 1 & k \geq z \end{Bmatrix} \text{Poisson}(\lambda = zq/p)$$

λ is the average number of blocks that the attacker mines in z unit of time.

$$= \sum_{k=0}^{\infty} \begin{Bmatrix} (q/p)^{z-k} & k < z \\ 1 & k \geq z \end{Bmatrix} \frac{(zq/p)^{k} e^{-zq/p}}{k!}$$

GIST

6 Attack Success Probability

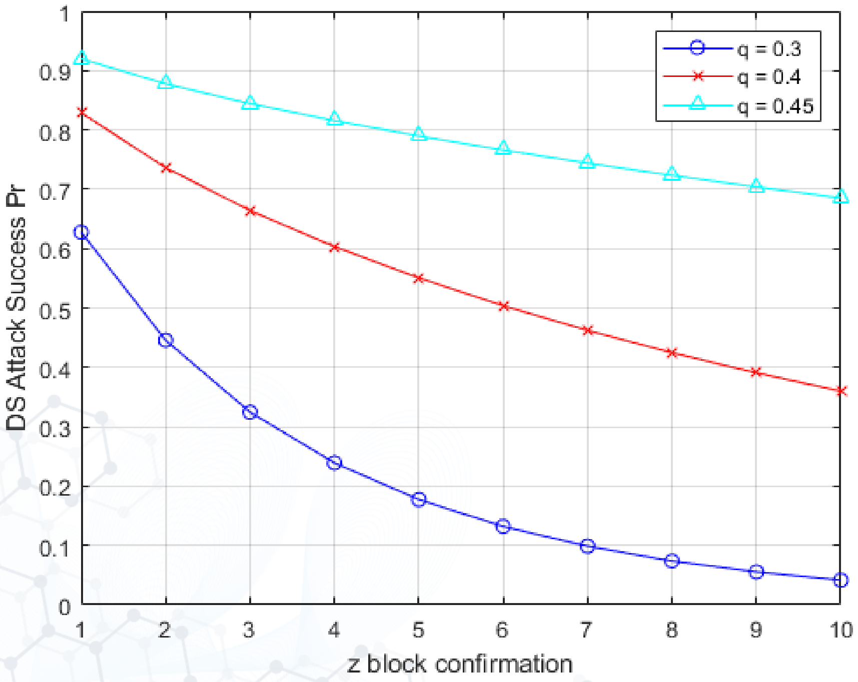
- Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

- Converting to C code...

GIST

6 Attack Success Probability



6 Attack Success Probability

- Double Spending Attacks are possible even if hash rate of the attacker does not overpower (51% attack) that of the honest network.
- The DS success probability decreases rapidly with diminishing q.
- DS success probability decreases rapidly with growing z.

Goal of this lecture note

- Bitcoin Networks
- Pre-cursors to Bitcoin
- Proof of Work-the Monopoly Problem
- Proof of X schemes
- Summary of Altcoins

1 Bitcoin Networks

- Test Bitcoin Network
- Main Bitcoin Network

GIST

1 Bitcoin Networks

- Experimental Test Bitcoin Network
  - Testnet runs the same code as the mainnet does, but can be run as an experiment.
  - One can change the protocol and runs one’s own bitcoin with
    - New free coins
    - Faster block generations time
    - Different issuance schedule
    - Difficulty

1 Bitcoin Networks

- Joining and Maintaining the network
  - Every peer in the Bitcoin network aims to maintain a minimum of 8 connections and a maximum 125 connections.
  - Peers listen on port 8333 for inbound connections.

GIST

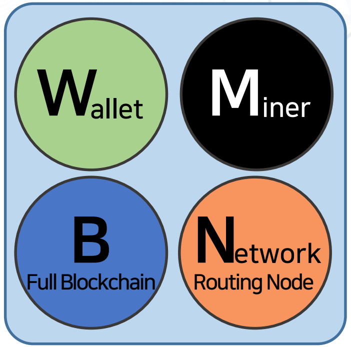
1 Bitcoin Networks

- Nodes Types and Roles
  - While nodes in the Bitcoin network are equal, they may take on different roles depending on the functionality they are supporting.
  - A bitcoin node is a collection of functions such as routing, blockchain database, mining, and wallet services.

Andreas M. Antonopoulos,『Mastering Bitcoin』, O’reilly, 2010

1 Bitcoin Networks

- Nodes Types and Roles
  - Each node has the routing function to participate in the network.
  - Full node has all four functions.
  - Wallet node has W and N.
  - Miner node has M, B, and N.
  - Full blockchain node has B and N.



Andreas M. Antonopoulos, 『Mastering Bitcoin』, O’reilly, 2010

2 Pre-cursors to Bitcoin

- PoW is a gold, or a coin.
- Hashcash (02')
- RPOW (03') is a centralized currency.
- B-money (98') is a decentralized currency.
- Karma (03') is a distributed currency.
- BitGold (05') is a distributed currency.

2 Pre-cursors to Bitcoin

- Hashcash by Adam Back
  - Proof-of-work used to limit email spam. (97')
  - PoW with a num. of high zero bits is a token. (02')
- Reusable POW (03') by Hal Finny is a centralized currency.
  - A server issues a coin in return for a PoW.
  - Coins are reusable and transferrable.
  - The server checks the validity.
- B-money (98') by Wei Dai
  - Uses PoW money and a set of servers (decentralized) for validation, and assumes unjammable broadcast channel.

GIST

2 Pre-cursors to Bitcoin

- Karma by Vishnumurthy et al.(03') is a distributed currency.
  - A bank set keeps track of coins in file sharing.
  - Coin creation is adjusted considering inflation and deflation.
- BitGold by Nick Szabo (05')
  - Metalic gold vs Bitgold
  - Suggested to chain the proof-of-work. (uses the last entry to create new puzzle and adjust difficulty)
  - But relied on IP addresses and thus vulnerable to Sybil attack.

2 Pre-cursors to Bitcoin

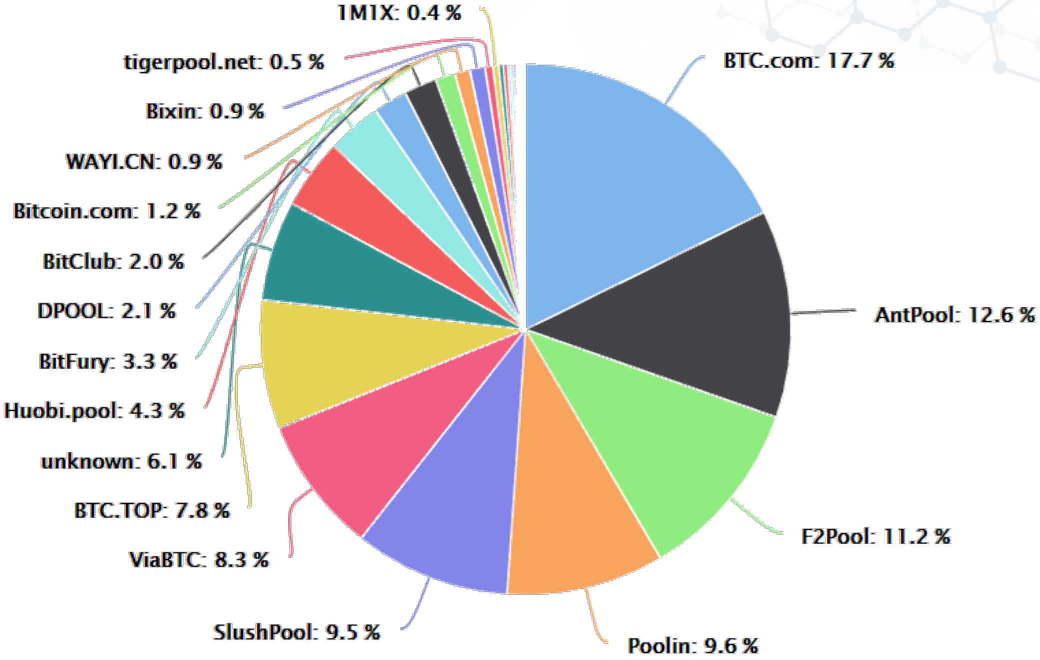
- Bitcoin
  - Inflation and deflation
  - Sybil attacks
  - Proof-of-work
  - One CPU one vote

3 Proof of Work-the Monopoly Problem

- Miners are rational profit seekers.
- They strive to make as much profit as possible.
- Mining operations are highly parallelizable, GPU mining quickly replaced CPU mining:
- FPGAs did GPUs.
- ASICs did FPGAs.

3 Proof of Work-the Monopoly Problem

- Proof of Work, any alternative?
  - PoW, monopolized today.
  - Handful of mining sites dominating.
  - The trust has been degraded.
  - No more one CPU one vote.
  - Rational profit seeking miners use ASICs now.



https://btc.com/ (기간: 2018. 09 ~ 2019. 09)

3 Proof of Work-the Monopoly Problem

- Items to consider for new PoW
  - One way is to diversify the puzzles and change them over time.
  - Considerations for new puzzles.
    - A puzzle should be difficult to solve but very easy to check.
    - The puzzle should be resistant to attacks.
    - Solution to the puzzle for a block should not be reusable.
    - Puzzle difficulty should be adjustable.
    - Anyone with a CPU who wishes to participate should be able to join.
    - Consensus must eventually be reached; there must be a common rule to resolve forks and to determine the main blockchain.

4 Proof of X Schemes

- Proof of Stake (PoS)/Delegated PoS
- Proof of Activity
- Proof of Publication

GIST

4 Proof of X Schemes

- Proof of Stake
  - Give higher PoW chance to a node with a higher stake (more coins).
  - Good: No high energy consumption
  - Bad: Rich gets richer problem
  - What if the node stays off line?
    - Delegated PoW

Reference: See page 2114, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 3, THIRD QUARTER 2016

GIST

4 Proof of X Schemes

- Proof of Stake based on Coin Age
  - *Coin age* is no. coins times the holding period.
    - Implemented in Peercoin (peercoin.net).
  - The difficulty of PoW is individually determined, inversely proportional to one's *coin age*.
  - If one finds a solution, one's *coin age* is reset.
  - Slowly increasing the chances of solving the puzzle next time.

Reference: See page 2114, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 3, THIRD QUARTER 2016

4 Proof of X Schemes

- Proof of Stake
  - In contrast to PoW, where the longest block chain survives, *coin age* PoS declares the block chain with the highest total sum of *destroyed coin age* as the main chain.
  - An attacker must hold a huge amount of coins.

GIST

4 Proof of X Schemes

- Proof of Stake
  - Good: Energy consumption is minimized.

[229] N. Houy, “It will cost you nothing to ‘kill’ a proof-of-stake cryptocurrency,”
Econ. Bull., vol. 34, no. 2, pp. 1038–1044, 2014.

GIST

4 Proof of X Schemes

- Proof of Stake
  - Bad
    - Coin age accumulates even when the node is not connected to the network.
    - Come online for reward go offline afterwards.
    - The lacking of sufficient number of online nodes, may facilitate attacks.

4 Proof of X Schemes

- Proof of Activity
  - In [234] the author notes higher activity produces a healthier economy.
  - Key Idea is to reward active peers.
  - Let a fresh coin accumulate age faster.
  - It is thus a combination of proof of work and proof of stake.

[234] L. Ren, "Proof of stake velocity: Building the social currency of the digital age," Tech. Rep., Apr. 2014 [Online]. Available:http://www.reddcoin.com/papers/PoSV.pdf

GIST

4 Proof of X Schemes

- Proof of Activity
  - Hybrid of PoW and PoS.
  - Good: Saving in energy consumption
    - for p2p file sharing, e.g. BitTorrent.
  - Bad: Uses PoW. Thus still use a lot of energy. Uses PoS; coin hoarders still have higher chances of accumulating more rewards.

[234] L. Ren, “Proof of stake velocity: Building the social currency of the digital age,” Tech. Rep., Apr. 2014 [Online]. Available:http://www.reddcoin.com/papers/PoSV.pdf

GIST

4 Proof of X Schemes

- Proof of Publication
  - Documents and timestamps are hashed and secured by private key of the timestamping server.
  - But the server can easily backdate documents by hashing and signing a previous timestamp.
  - Linked chain of timestamps and use of a set of servers can prevent this problem.
  - But the approach comes with the premise of trusting the set of timestamping servers.
  - Thus, Sybil attacks shows up again.

4 Proof of X Schemes

- Proof of Publication
  - Recall the time-stamp server of the bitcoin white paper!
  - Bitcoin provides a secure distributed timestamping service, with an accuracy of about 10 min.

4 Proof of X Schemes

- Proof of Publication
  - Bitcoin can be used as a timestamping service.
  - Use cases include coin tosses [238], lotteries [239], or decentralized poker [240].
  - Multi Party Computing works without a central entity.

[238] A. Back and I. Bentov, "Note on fair coin toss via bitcoin," Computing Research Repository, Tech. Rep. abs/1402.3698, 2014.
[239] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in Proc. IEEE 35th Symp. Secur. Privacy (SP'14), May 2014, pp. 443–458.
[240] R. Kumaresan, T. Moran, and I. Bentov, "How to use bitcoin to play decentralized poker," in Proc. ACM 22nd Conf. Comput. Commun. Secur. (CCS'15:), Oct. 2015, pp. 195–206.

GIST

5 Summary of Altcoins

- Table IV – Summary of Altcoins and Extensions

	Approach	Distinct Feature (incl. References)	Sec.
Precursor	B-Money	Mining reward proportional to proof of work difficulty; requires a broadcast channel [7]	II-B, V-D, V-E
	Bit Gold	Chained proof of work [10]; Byzantine-resilient quorum [13]	III-B, V-D, V-E
	Karma	Distributed currency maintained by a bank set [8]	V-E
	RPOW	Centralized (reusable) proof of work exchange/ bank [9]	V-E
Altcoins	Bitshares (BTS)	Delegated proof of stake [231]	V-F
	Bytecoin (BCN)	Implements CryptoNote [190], which aims for unlinkable and untraceable transactions	V-C, V-E
	Counterparty (XCP)	Colored coin; used proof of burn	V-H, V-H
	Cryptonite (XCN)	Implements the mini block chain scheme [127]	IV-D
	Dash (DASH)	Formerly known as Darkcoin; implements native CoinJoin-like transactions [178]	V-C
	Dogecoin (DOGE)	Block payload holds TXIDs only; fast block generation	IV-D, V-E
	Litecoin (LTC)	Uses scrypt [214] to foster distributed power among miners	V-E
	Mastercoin (MSC)	Colored coin; exodus address	V-H
	Nextcoin (NXT)	Entirely proof of stake based	V-F
	Peercoin (PPC)	Identified coin age as alternative measure; proof of stake [227]	V-F
	Primecoin (XPM)	Proof of work with intrinsic value i. e. prime chains [218]	V-E
	Reddcoin (RDD)	Proof of stake velocity [234]	V-E
	RSCoin	Centrally controlled money supply with distributed verification [126]	IV-D
	Ripple (XRP)	Implements a novel Byzantine agreement protocol [200]	V-D
	Zerocash	Full-fledged altcoin, carrying on the ideas of Zerocoin [189]	V-C

IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 3, THIRD QUARTER 2016

5 Summary of Altcoins

- Table IV – Summary of Altcoins and Extensions

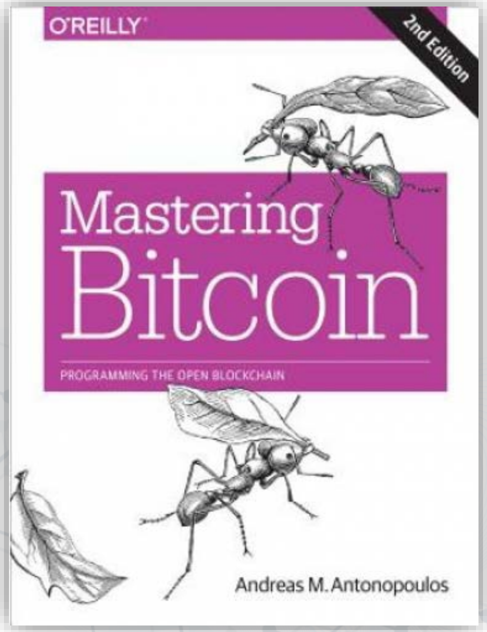
	Approach	Distinct Feature (incl. References)	Sec.
Altchains	Bitmessage	Secure messaging service [145]	IV-G
	Ethereum (Ether)	Turing complete smart contract processing [44], [45]	II-E
	Namecoin (NMC)	Key-value storage; realizes decentralized domain name coordination [143]	IV-G
	Permacoin	Decentralized file storage; proposes proof of retrievability [100]	V-E
Protocols / Extensions	CoinJoin	Uses multi-signature transactions to enhance privacy [160]	V-C
	CoinShuffle	Decentralized protocol to coordinate CoinJoin transactions [180]	V-C
	CoinSwap	Enables P2P-based trustless mixing [41]	V-C
	CommitCoin	Secure timestamping protocol [40]	V-H
	Mini block chain	Identifies individual block chain components [127]	IV-D
	Mixcoin	Mixing with accountability [174]	V-C
	Zerocoin	Unlinkable and untraceable transactions by employing zero knowledge proofs [187]	V-C

IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 3, THIRD QUARTER 2016

Goal of this lecture note

- Mastering Bitcoin
- Elliptic Curve Signatures
- Bitcoin Addresses
- Unspent Transaction Outputs (UTXOs)

1 Mastering Bitcoin



Refer to M.B. for materials:

1. Elliptic Curve Signatures
2. Transactions
3. Scripts
4. OP Codes
5. Example Scripts
6. Smart Contracts

『Mastering Bitcoin』, Antonopoulos, Andreas M., O'Reilly Media, 교보문고 제공

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithms
  - Additions and multiplications on some curves.
  - Fifteen curves defined in a NIST standard.
  - But Bitcoin uses the curves def'd in `Secp256k1`.
  - Asymmetric cryptography, pub and priv keys.
  - A public key is used to give a Bitcoin address.
  - A private key is to sign the transfer of right.

[http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm]
http://www.secg.org/sec2-v2.pdf

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Public domain info

1. Use a designated hash function $H(*)$
2. A curve is collection of the roots of $y^2 = x^3 + ax + b$ over a finite field $F(p)$ with prime p.
3. $G = (x, y)$, a point on the curve.
4. n the multiplicative order of G.

[http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm]

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Key Generation

Out: k (private key), K (public key)

1. Select an integer k in [0, n-1].
2. Compute $K = k\,G$.
3. K and G ~ points on the curve
4. The key-pair is (k, K).

Results: Alice's pair (k_A, K_A) and Bob's pair (k_B, K_B).

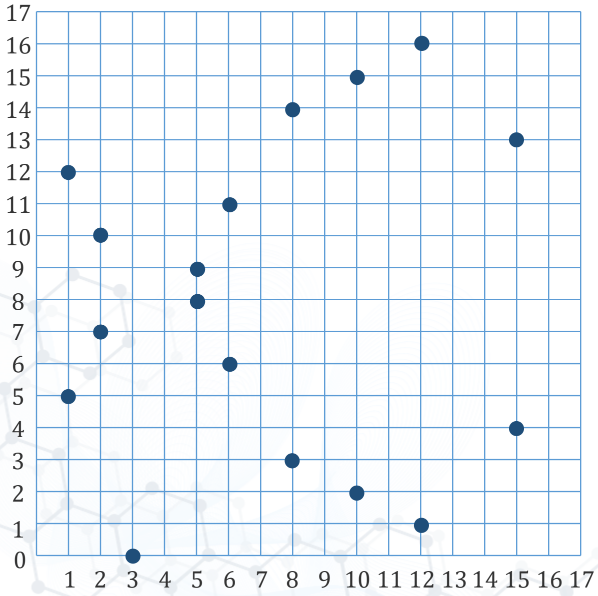
It is an asymmetric cryptography.

[http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm]

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Elliptic Curve



- The points are the roots (x, y) of the curve equation defined by:

$$y^2 = x^3 + 7 \bmod 17$$

Figure 4-3. Elliptic Curve Cryptography: Visualizing an elliptic curve over $F(p)$, with $p = 17$

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - How many points are on the curve?
    - Observation:
      - For each x, there are 0, 1, or 2 possible y-point(s).
      - There are total 17 $(x,\ y)$-points.
    - Facts:
      - The set of finite points on the curve forms a *group* which is closed under a binary operation.

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Addition of any two points on elliptic curve
    - There are three cases:
      Case 1) Adding two points where x_1 neq to x_2:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$s = \frac{y_2 - y_1}{x_2 - x_1}$$

$$x_3 = s^2 - x_1 - x_2$$

$$y_3 = s \cdot (x_1 - x_3) - y_1$$

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Elliptic Curve Cryptography (ECC)
    - There are three cases:
      Case 2) Adding two points where $x_1 = x_2$ and $y_1 = y_2$

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$s = (3x_1^2 + a)/2y_1$$

$$x_3 = s^2 - x_1 - x_2$$

$$y_3 = s \cdot (x_1 - x_3) - y_1$$

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Elliptic Curve Cryptography (ECC)
    - There are three cases:
      Case 3) Adding two points where $x_1 = x_2$ and $y_1 = -y_2$

$$(x_1, y_1) + (x_1, -y_1) = O$$

The identity element

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Example to find a point on a curve
    - Let p = 17.
    - Let the curve be $y^2 = x^3 + 7 \bmod 17$.
    - Find a point on the curve

      Let $x = 3$. Then $y = ?$

      $y^2 = 27 + 7 = 34 = 0$

      $y^2 = 0$

      $y = 0$
    - Thus, (3, 0) is a point on the curve.

GIST

2 Elliptic Curve Signatures

Anaconda Powershell Prompt

```
>>>
>>> p = 17
>>> x = 3
>>> y_square = (x**3 + 7)%p
>>> y_square
0
>>>
```

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Example to find a point on a curve
    - Let us continue to find another point.
    - This time, let us start with an y element.
    - Let $y = 12$ and find x.

$$
\begin{aligned}
y^2 &= 12^2 \\
&= 144 - \text{floor}(144/17)\text{x}17 \\
&= 8 \\
x^3 + 7 &= 8 \\
x^3 &= 1 \\
x &= 1
\end{aligned}
$$

    - Thus, (1, 12) is a point on the curve.

GIST

2 Elliptic Curve Signatures

Anaconda Powershell Prompt

```
>>> p = 17
>>> y = 12
>>> y_square = y**2
>>> y_square
144
>>> y_square = y_square%p
>>> y_square
8
>>> x_3rd_power = (y_square - 7)%p
>>> x_3rd_power
1
>>>
```

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - Let us add two points.

Given two points $(x_1, y_1) = (3, 0)$ and $(x_2, y_2) = (1, 12)$.

Find $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$.

Note this is Case 1.

$$s = \frac{y_2 - y_1}{x_2 - x_1} = \frac{12 - 0}{1 - 3} = -6 = 11$$

$$x_3 = s^2 - 3 - 1 = 121 - 4 = 117\%17 = 15$$

$$y_3 = s(x_1 - x_3) - y_1 = 11(3 - 15) - 0 = 11(5) = 4$$

$(x_3, y_3) = (15, 4)$

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - A scalar multiplication example
    - Take any point $P = (x, y)$ on the curve and multiply it by a scalar k.
    - The resulting point can be obtained by adding P k times, i.e.,

$$kP = P + P + \ldots + P$$

2 Elliptic Curve Signatures

- We may use Python for computations.
  - A point $P(x, y)$ is point on the `secp256k1` curve.
  - You can check our results using Python.

```
Anaconda Powershell Prompt
(base) PS C:\Users\Heung-No Lee> python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> p = 17
>>> x = 1
>>> y = 5
>>> (x**3+7-y**2)%p
0
>>>
```

2 Elliptic Curve Signatures

- We may use Python libraries at github.
  - One example is https://github.com/vbuterin/pybitcointools.
  - It offers `pybitcointools` library which allows us to generate and display keys and addresses.
  - The other one is at https://github.com/warner/python-ecdsa which offers ECDSA implementation in Python.

2 Elliptic Curve Signatures

- From private key k, obtain public key by $K = k*G$.
  - A 256 bit string is shown as 64 hexadecimal string.

$k =$ 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD

$G = (x, y) =$ (55066263022277343669578718895168534326250603453777594175500187360389116729240,
32670510020758816978083085130507043184471273380659243275938904335757337482424)

  - Multiply the private key k with the generator point G to obtain the public key K.

$K =$ 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD $*G$

$K =$ (x, y)

where,

$x =$ F028892BAD...DC341A

$y =$ 07CF33DA18...505BDB

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - We now know how to generate keys.
  - Next is how to sign and validate it.

http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - SignGenerate

In m the message, Alice's private key k_A
Out Alice' signature (r, s)

1. Calculate the message hash $e=H(m)$
2. Let z be the L_n leftmost bits of e where L_n is the bit length of the group order n
3. Select an integer d from $[1, n-1]$
4. Calculate the curve point $(x_1, y_1)=dG$
5. Calculate $r=x_1 \bmod n$, If $r=0$, go to step 3
6. Calculate $s= k_A^{-1}(z+rk_A) \bmod n$, If $s=0$, go to step 3
7. The signature is the pair (r, s)

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - IsSignatureValid

In m a message, Alice's signature (r, s), and K_A
Out Valid or invalid

1. Verify if K_A is a valid curve point as follows:
   1. Check to see if K_A is not equal to the identity element O
   2. Check to see if K_A lies on the curve
   3. Check that $n \times K_A = O$
2. Verify that r and s are integers in $[1, n-1]$
   If not, the signature is invalid
3. Calculate the message hash $e = H(m)$

[출처: http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm]

GIST

2 Elliptic Curve Signatures

- Elliptic Curve Digital Signature Algorithm
  - IsSignatureValid

In m the message, Alice's signature (r, s), and K_A
Out Valid or invalid

4. Let z be the L_n leftmost bits of e where L_n is the bit length of the group order n
5. Calculate $w = s^{-1} \bmod n$
6. Calculate u_1=z w mod n and $u_2=r * w \bmod n$
7. Calculate the curve point $(x_1, y_1)= u_1 * G + u_2 * Q_A$
   If $x_1, y_1=0$, then the signature is invalid
8. The signature is valid if $r = x_1 \bmod n$, invalid otherwise

[출처: http://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm]

3 Bitcoin Addresses

- An example Bitcoin Address is *1thMjrt546nngXqyPEz532S8fLwbozud8.*
  - BTCs belong to a Bitcoin address.
  - We aim to know how they are generated.
  - An address is generated from a public key.
  - It goes through several mappings such as `SHA256`, `RIPEMD160`, and `Base58Check`.

GIST

3 Bitcoin Addresses

- Making a Bitcoin address from a public key
  - Private key k (32 bytes)
  - Public key $K = G * k$
    - Uncompressed one is 65 bytes $(0x04 + x + y)$.
    - Compressed one is 33 bytes
      ($0x02+ x$, use 02 for even y; $0x03+x$ *for odd* y).
  - Public Key Hash = RIPEMD160(SHA256(K))
    - 160 bit (20 byte)
  - Base58Str
    = Base58Check(PKH + 4Byte_checksum)

Ex 1PMycacnJaSqwwJqjawXBErnLsZ7RkXUAs

GIST

3 Bitcoin Addresses

- What is Base58Check and why?
  - Base58Check is mapping a PKH into a more readable format.
  - Base58 is similar to Base64 but with 6 characters removed.
  - Base64 uses A-Z, a-z, 0-9, + and /.
  - Removed are +, /, 0, O, I and l.
  - These symbols are prone to confusion.
  - A Bitcoin address is of between 27 and 34 characters long!

GIST

3 Bitcoin Addresses

- Base58 Value-to-Character Mapping Table

Value	Character	Value	Character	Value	Character	Value	Character
0	1	1	2	2	3	3	4
4	5	5	6	6	7	7	8
8	9	9	A	10	B	11	C
12	D	13	E	14	F	15	G
16	H	17	J	18	K	19	L
20	M	21	N	22	P	23	Q
24	R	25	S	26	T	27	U
28	V	29	W	30	X	31	Y
32	Z	33	a	34	b	35	c
36	d	37	e	38	f	39	g
40	h	41	i	42	j	43	k
44	m	45	n	46	o	47	p
48	q	49	r	50	s	51	t
52	u	53	v	54	w	55	x
56	y	57	z				

GIST

3 Bitcoin Addresses

- Example of Base58Check Mapping

$$
\begin{aligned}
12437_{10} &= 3\text{x}58^2 + 40\text{x}58^1 + 25 \\
&= 3\ 40\ 25_{58} \\
&= 4hS_{58}
\end{aligned}
$$

GIST

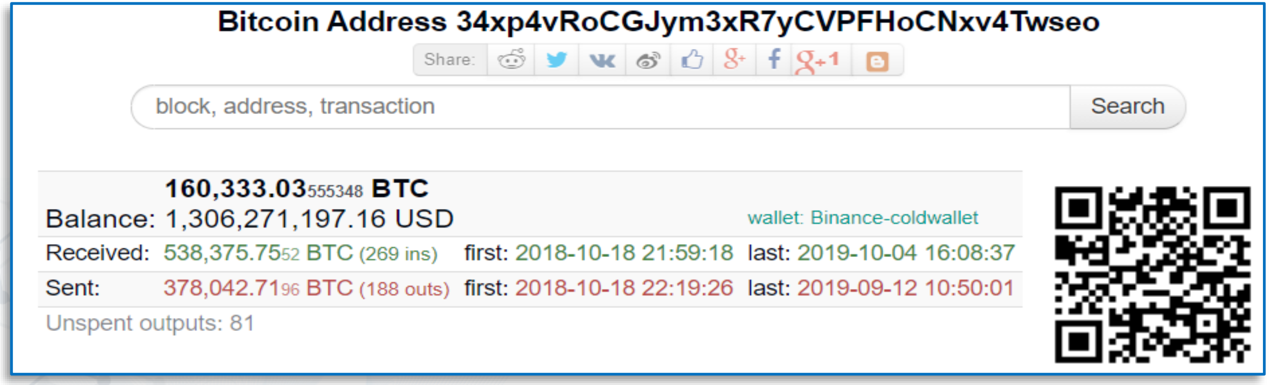
3 Bitcoin Addresses

- A version prefix is appended to Base58Str
  - Table 4-1. Version Prefixes

Type	Version prefix (hex)	Base-58 prefix
Bitcoin Address	0×00	1
Pay-to-Script-Hash Address	0×05	3
Bitcoin Testnet Address	0×6F	m or n
Private Key WIF	0×80	5, K or L
BIP38 Encrypted Private Key	0×0142	6P
BIP32 Extended Public Key	0×0488B21E	xpub

3 Bitcoin Addresses

- The richest Bitcoin address on 2019/10/14 is
  34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo
- It holds 160,333.03 BTCs.



https://bitinfocharts.com/bitcoin/address/34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo

GIST

4 Unspent Transaction Outputs (UTXOs)

- UTXO is an unspent transaction output.
- Given an address, one can obtain all the UTXOs belonging to that address by going through the ledger.
- We are interested in

  Creating, signing and submitting Transactions based on UTXOs.

GIST

4 Unspent Transaction Outputs (UTXOs)

- How to obtain UTXOs?
  - When you download/install Bitcoin core, you run the Bitcoin client.
  - Mastering Bitcoin has a detailed procedure for installation (see Ch.3)
  - One can use the Bitcoin client to find all the UTXOs.
  - The command `listunspent` can list out all UTXOs which belong to address.
  - Once UTXOs are figured out, they can be spent.

『Mastering Bitcoin』, Antonopoulos, Andreas M., O'Reilly Media

GIST

4 Unspent Transaction Outputs (UTXOs)

- UTXOs
  - First, use the `listunspent` command to show all the unspent confirmed outputs to each address in our wallet.

```
$ bitcoin-cli listunspent

[
    {
        "txid" : "9ca8f969bd3ef5ec2a8685660fdbf7a8bd365524c2e1fc66c309acbae2c14ae3",
        "vout" : 0,
        "address" : "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
        "account" : "",
        "scriptPubKey" : "76a91407bdb518fa2e6089fd810235cf1100c9c13d1fd288ac",
        "amount" : 0.05000000,
        "confirmations" : 7
    }
]
```

4 Unspent Transaction Outputs (UTXOs)

- UTXOs
  - When you want to spend an `UTXO`, you make a transaction in which an `UTXO` is used as an input by referring to the previous `txid` and `vout` index.
  - You need to create a new transaction that will spend the 0th `vout` of the `txid` `9ca8f0...` as its input and assign it to a new output address.

GIST

4 Unspent Transaction Outputs (UTXOs)

- Closer look at a UTXO with `txid 9ca8..., vout0`
  - Use the `gettxout` command.
  - Transaction outputs are always referenced by `txid` and `vout`, and they are the parameters we pass to `gettxout`.

GIST

4 Unspent Transaction Outputs (UTXOs)

- Closer look at `txid 9ca8... vout0`

```
$ bitcoin-cli gettxout 9ca8f969bd3ef5ec2a8685660fdbf7a8bd365524c2e1fc66c309acbae2c14ae3 0
```



```
{
    "bestblock" : "0000000000000001405ce69bd4ceebcdfdb537749cebe89d371eb37e13899fd9",
    "confirmations" : 7,
    "value" : 0.05000000,
    "scriptPubKey" : {
        "asm" : "OP_DUP OP_HASH160 07bdb518fa2e6089fd810235cf1100c9c13d1fd2\
         OP_EQUALVERIFY OP_CHECKSIG",
        "hex" : "76a91407bdb518fa2e6089fd810235cf1100c9c13d1fd288ac",
        "reqSigs" : 1,
        "type" : "pubkeyhash",
        "addresses" : [
            "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL"
        ]
    },
    "version" : 1,
    "coinbase" : false
}
```

GIST

4 Unspent Transaction Outputs (UTXOs)

- Closer look at `txid 9ca8…, vout0`
  - What we see above is the output that has 0.05 BTC to our address `1hvz....`
  - To spend this output we shall create a new transaction.
  - For this, we need to get an address to which we will send the money:

GIST

4 Unspent Transaction Outputs (UTXOs)

- Making a new transaction
  - There is a Bitcoin client command `createrawtransaction`.
  - It can be used to generate a raw transaction.
  - Suppose you want to make a new transaction
    - A payment of 0.030 BTC to a recipient with address 1LTz9⋯1cP.
    - A change of 0.015 BTC is given back to an address of yours, 1Bts8⋯2Ps.
    - The rest, 0.050 – 0.030 – 0.015 = 0.005 BTC, is given to miners as TX fee.

GIST

4 Unspent Transaction Outputs (UTXOs)

TXID 7957a35⋯f18

In0:		Vout0:	
TXID 9ca8…ae3	→	ScriptPK1	0.030 BTC
vout 0		Vout1:	
Sign		ScriptPK2	0.015 BTC
0.050 BTC			

GIST

4 Unspent Transaction Outputs (UTXOs)

- Each TX is locked. To unlock, you need the private key.
  - 시간 1: A's Signature (Key)→ B (Locked to B) 2BTC.
  - 시간 2: B's Signature (Key)→ C (Locked to C) 1BTC.
  - 시간 3: C's Signature (Key)→ D (Locked to D) 0.5BTC.

4 Unspent Transaction Outputs (UTXOs)

- Making a new transaction
  - Inputs given to `createrawtransaction` include:
    - UTXO's TXID vout 0
    - 1LTz9…1cP 0.030 BTC
    - 1Bts8…2Ps 0.015 BTC
  - Then, a chuck of script code is generated.

Goal of this lecture note

- Bitcoin Script
- Tables of OP Codes
- Easy Script
- Pay-to-Public Key Hash (P2PKH) Script
- Multisignature and Smart Contracts Scripts

GIST

1 Bitcoin Script

- Bitcoin Script
  - Bitcoin uses a scripting language for transactions.
  - A script is simple, stack-based, and processed from left to right.
  - It is intentionally not Turing-complete, with no loops.
  - A script is a list of instructions.
  - The payer locks the vout value to a payee's public address.
  - The payee unlocks the lock by providing the signature.

GIST

1 Bitcoin Script

- Bitcoin Script
  - Payer uses a lock script to lock the `vout` value to a destination Bitcoin address and payee uses an unlock script to spend it.
    1. The `vout` value transferred to a destination address mapped from a public key is locked into the locking script, and
    2. A signature is embedded in the unlocking script which proves the ownership of the private key corresponding to the locked value.
  - Further reading from https://en.bitcoin.it/wiki/Script

GIST

1 Bitcoin Script

- See if scriptSig unlocks scriptPubKey!
  - *Script Construction (Unlock+Lock)*
    - The locking script is called a *scriptPubKey*, because it contains a public key or a Bitcoin address.
    - The unlocking script is called *scriptSig* because it contains a digital signature.
    - When a correct unlocking script is provided to the locking script, the execution of the complete script comes out TRUE.
    - Then, the provider of scriptSig can spend the value.

GIST

1 Bitcoin Script

- Pay to Public Key Hash
  - 시간 1: A's Sign (Priv. Key)→ Lock to Pub. Key of B 2.0BTC.
  - 시간 2: B's Sign (Priv. Key)→ Lock to Pub. Key of C 1.0BTC.
  - 시간 3: C's Sign (Priv. Key)→ Lock to Pub. Key of D 0.5BTC.

Unlocking Script (scriptSig)	+	Locking Script (scriptPubKey)
<sig> <PubK>		DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
Sign to PubK		Generate_Sign with PubKHash

1 Bitcoin Script

- Values provided by users are given in < >.
- DUP, HASH160, EQUALVERIFY, CHECKSIG are Operations.

Unlocking Script (scriptSig)	+	Locking Script (scriptPubKey)
<sig> <PubK>		DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
Sign to PubK		Generate_Sign with PubKHash

GIST

2 Tables of OP Codes

- Table C-7. Cryptographic and Hashing Operations

Symbol	Value(hex)	Description
OP_RIPEMD160	0xa6	Return RIPEMD160 hash of top item
OP_SHA1	0xa7	Return SHA1 hash of top item
OP_SHA256	0xa8	Return SHA256 hash of top item
OP_HASH160	0xa9	Return RIPEMD160(SHA256(x)) hash of top item
OP_HASH256	0xaa	Return SHA256(SHA256(x)) hash of top item
OP_CODESEPARATOR	0xab	Mark the beginning of signature-checked data

Appendix C of Mastering Bitcoin

GIST

2 Tables of OP Codes

- Table C-7. Cryptographic and Hashing Operations

Symbol	Value(hex)	Description
OP_CHECKSIG	0xac	Pop a public key and signature and validate the signature for the transaction's hashed data, return TRUE if matching
OP_CHECKSIGVERIFY	0xad	Same as CHECKSIG, then OP_VERIFY to halt if not TRUE
OP_CHECKMULTISIG	0xae	Run CHECKSIG for each pair of signature and public key provided. All must match. Bug in implementation pops an extra value, prefix with OP_NOP as workaround
OP_CHECKMULTISIGVERIFY	0xaf	Same as CHECKMULTISIG, then OP_VERIFY to halt if not TRUE

Appendix C of Mastering Bitcoin

GIST

2 Tables of OP Codes

- Table C-3. Stack Operations

Symbol	Value(hex)	Description
OP_TOALTSTACK	0x6b	Pop top item from stack and push to alternative stack
OP_FROMALTSTACK	0x6c	Pop top item from alternative stack and push to stack
OP_2DROP	0x6d	Pop top two stack items
OP_2DUP	0x6e	Duplicate top two shack items
OP_3DUP	0x6f	Duplicate top three shack items
OP_2OVER	0x70	Copies the third and fourth items in the stack to the top
OP_2ROT	0x71	Moves the fifth and sixth items in the stack to the top
OP_2SWAP	0x72	Swap the two top pairs of items in the stack
OP_IFDUP	0x73	Duplicate the top item in the stack if it is not 0
OP_DEPTH	0x74	Count the items on the stack and push the resulting count

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-3. Stack Operations

Symbol	Value(hex)	Description
OP_DROP	0x75	Pop the top item in the stack
OP_DUP	0x76	Duplicate the top item in the stack
OP_NIP	0x77	Pop the second item in the stack
OP_OVER	0x78	Copy the second item in the stack and push it on to the top
OP_PICK	0x73	Pop value N from top, then copy the Nth item to the top of the stack
OP_ROLL	0x7a	Pop value N from top, then move the Nth item to the top of the stack
OP_ROT	0x7b	Rotate the top three items in the stack
OP_SWAP	0x7c	Swap the top three items in the stack
OP_TUCK	0x7d	Copy the top item and insert it between the top and second item

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-6. Numeric Operators

Symbol	Value(hex)	Description
OP_1ADD	0x8b	Add 1 to the top item
OP_1SUB	0x8c	Subtract 1 from the top item
OP_2MUL	0x8d	Disabled (Multiply top item by 2)
OP_2DIV	0x8e	Disabled (Divide top item by 2)
OP_MEGATE	0x8f	Flip the sign of top item
OP_ABS	0x90	Change the sign of the top item to positive
OP_NOT	0x91	If top item is 0 or 1 boolean flip it, otherwise return 0
OP_ONOTEQUAL	0x92	If top item is 0 return 0, otherwise return 1
OP_ADD	0x93	Pop top two items, add them and push result

Appendix C of Mastering Bitcoin

GIST

2 Tables of OP Codes

- Table C-2. Conditional Flow Control

Symbol	Value(hex)	Description
OP_NOP	0x61	Do nothing
OP_VER	0x62	Halt – Invalid transaction unless found in an unexecuted OP-IF clause
OP_IF	0x63	Execute the statements following if top of stack is not 0
OP_NOTIF	0x64	Execute the statements following if top of stack is 0
OP_VERIF	0x65	Halt – Invalid transaction
OP_VERMPTIF	0x66	Halt – Invalid transaction
OP_ELSE	0x67	Execute only if the previous statements were not executed
OP_ENDIF	0x68	Ends the OP_IF, OP_NOTIF, OP_ELSE block
OP_VERIFY	0x69	Check the top of the stack, Halt and Invalidate transaction if not TRUE

Appendix C of Mastering Bitcoin

3 Easy Script

- Example script: 2 + 3 = 5

SCRIPT

2	3	ADD	5	EQUAL

EXECUTION POINTER

STACK

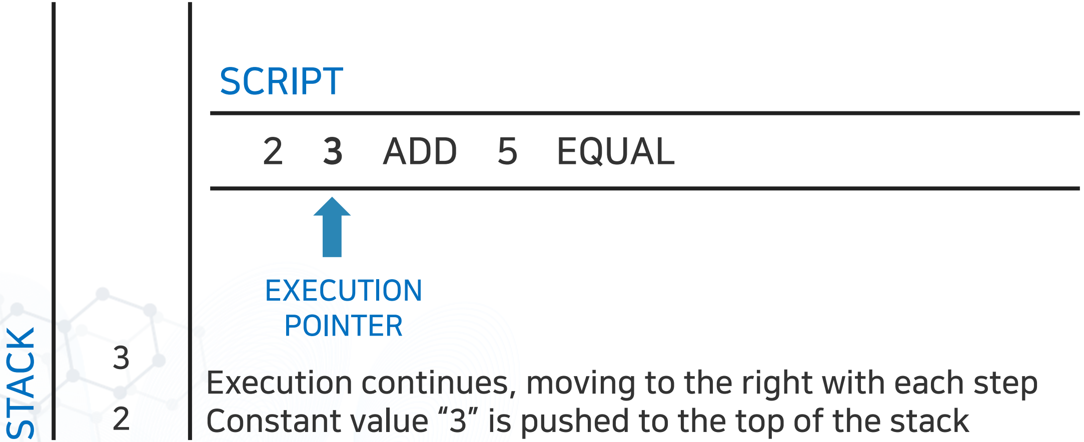
2

Execution starts from the left
Constant value “2” is pushed to the top of the stack

GIST

3 Easy Script

- Example script: 2 + 3 = 5



Execution continues, moving to the right with each step
Constant value “3” is pushed to the top of the stack

GIST

3 Easy Script

- Example script: 2 + 3 = 5

SCRIPT

2 3 **ADD** 5 EQUAL

EXECUTION POINTER

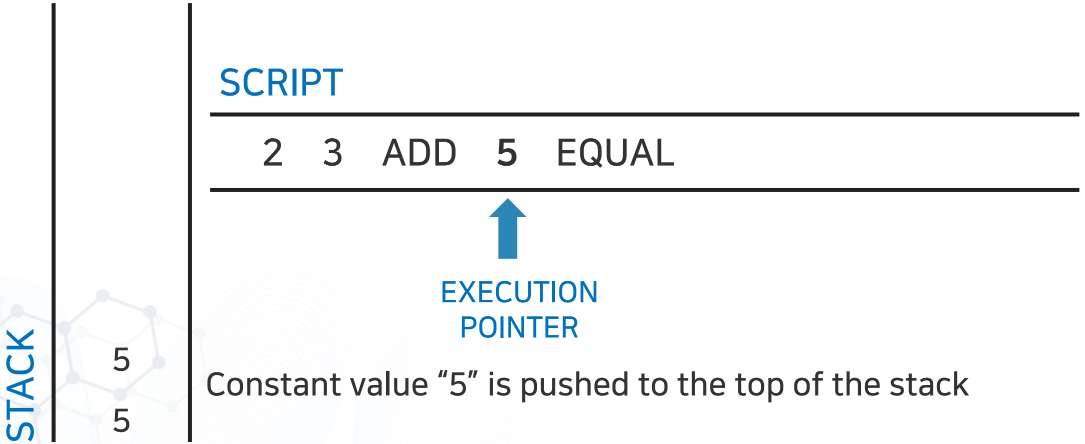
STACK

5

Operator ADD pops the top two items out of the stack and adds them together (3 add 2));
then Operator ADD pushes the result (5) the top of the stack

3 Easy Script

- Example script: 2 + 3 = 5



3 Easy Script

- Example script: 2 + 3 = 5

SCRIPT

2 3 ADD 5 **EQUAL**

EXECUTION POINTER

STACK

TRUE

Operator EQUAL pops the top two items out of the stack and compares the values (5 and 5)
and if they equal, EQUAL pushes TRUE (TRUE=1) to the top of the stack

GIST

3 Easy Script

- Unlock + Lock Pair, shows a proof of ownership
  - Use a part of the arithmetic example script as the locking script:

    3 OP_ADD 5 OP_EQUAL

  - Which can be satisfied by a transaction containing an input with the unlocking script:

    2

  - Put them together, we have the complete script.

    2 3 OP_ADD 5 OP_EQUAL

  - This pair will produce an outcome of TRUE.

4 P2PKH Script

- Now let us make a more realistic pair focusing on B.
  - 시간 1: A's Sign (Priv. Key)→ Lock to Pub. Key of B 2.0BTC.
  - 시간 2: B's Sign (Priv. Key)→ Lock to Pub. Key of C 1.0BTC.
  - the signature.

Unlocking Script (scriptSig) + Locking Script (scriptPubKey)

<sig> <PubK>

DUP HASH160 <PubKHash of B> EQUALVERIFY CHECKSIG

Sign to PubK

Check_Sign with PubKHash

GIST

4 P2PKH Script

- P2PKH of B
  - Unspent value belongs to Pay to Public Key Hash(P2PKH) script.

```
OP_DUP OP_HASH160 <Public Key Hash of B> OP_EQUAL OP_CHECKSIG
```

  - Unlocking script is a digital sign created by corresponding private key.

```
<sig of B>  <PubK of B>
```

4 P2PKH Script

- Locking script with a single <input>
  - One input, four operations
  - OP_DUP: duplicate
  - OP_Hash160(x)= RIPEMD(SHA256(x))
  - <Public Key Hash of B>
  - OP_EQUAL: return TRUE if the two top most values are equal
  - OP_CHECKSIG: checks to see if the provided sign and pubkey are valid

4 P2PKH Script

- Locking script with <input>

SCRIPT

<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG

EXECUTION POINTER

Sign_A

STACK

<sig>

Execution starts
Value <sig> is pushed to the top of the stack.

Figure 5-3.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

GIST

4 P2PKH Script

- Locking script with <input>

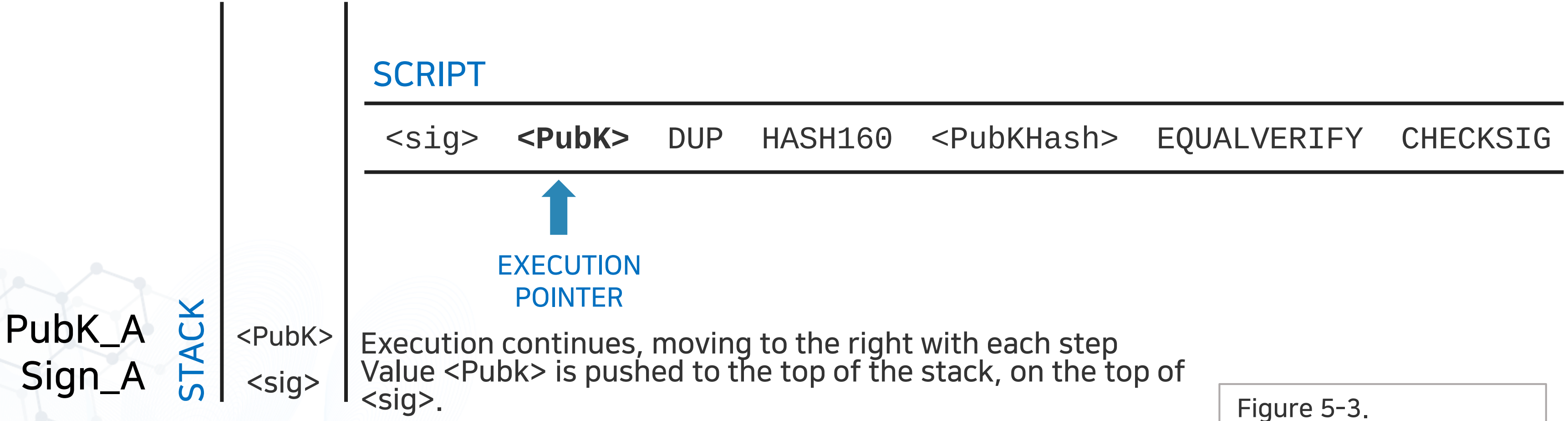
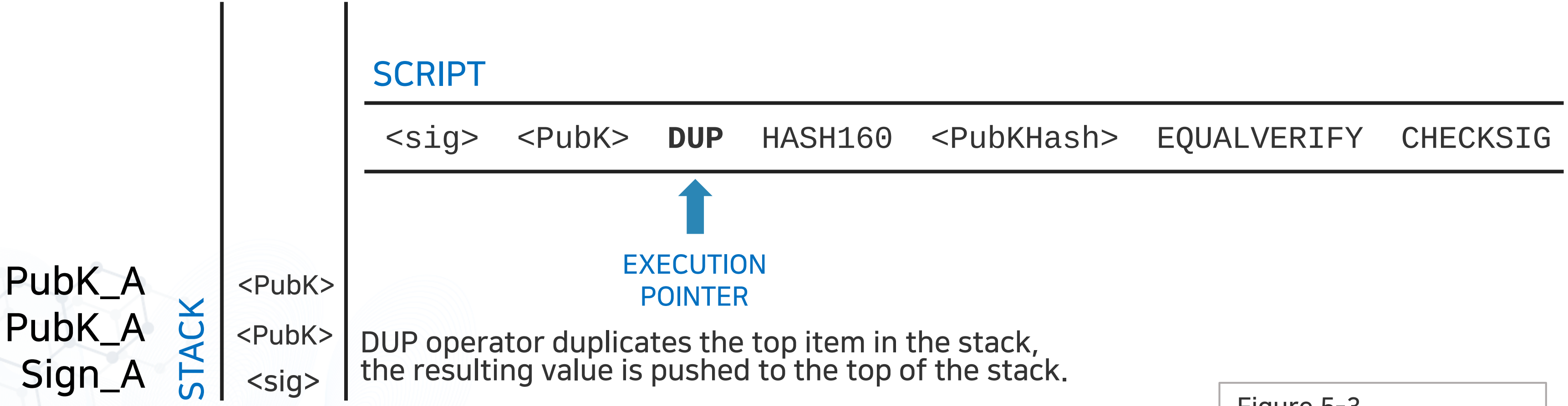


Figure 5-3.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

GIST

4 P2PKH Script

- Locking script with <input>



DUP operator duplicates the top item in the stack, the resulting value is pushed to the top of the stack.

Figure 5-3. Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

4 P2PKH Script

- See if two PubKH_As match

PubKH_A
PubK_A
Sign_A

STACK

<PubK Hash>
<PubK>
<sig>

SCRIPT

<sig> <PubK> DUP **HASH160** <PubKHash> EQUALVERIFY CHECKSIG

EXECUTION POINTER

HASH160 operator hashes the top item in the stack with RIPEMD160(SHA256(PubK)).
the resulting value (PubKHash) is pushed to the top of the stack.

4 P2PKH Script

- See if the two PubKH_As match

PubKH_A
= ab3813c
PubKH_A
PubK_A
Sign_A

STACK

<PubK Hash>
<PubK Hash>
<PubK>
<sig>

SCRIPT

<sig>	<PubK>	DUP	HASH160	**<PubKHash>**	EQUALVERIFY	CHECKSIG

EXECUTION POINTER

The value PubKHash from the script is pushed on top of the value PubKHash calculated previously from the HASH160 of the PubK.

4 P2PKH Script

- Check Signature

SCRIPT

<sig> <PubK> DUP HASH160 <PubKHash> **EQUALVERIFY** CHECKSIG

EXECUTION POINTER

STACK

PubK_A <PubK>
Sign_A <sig>

The EQUALVERIFY operator compares the PubKHash encumbering the transaction with the PubkHash calculated from the user's <Pubk>. If they match, both are removed and execution continues.

Figure 5-4.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 2 of 2)

GIST

4 P2PKH Script

- Recall `SignGenerate` and `isSignatureValid` routines
  - m = {TXID, output [n] = {value, a locking script with PKH_A}}
  - Sign_A = SignGenerate (m, k_A);
  - isSignatureValid(m, Sign_A, PK_A) = TRUE/False

4 P2PKH Script

- Check Signature

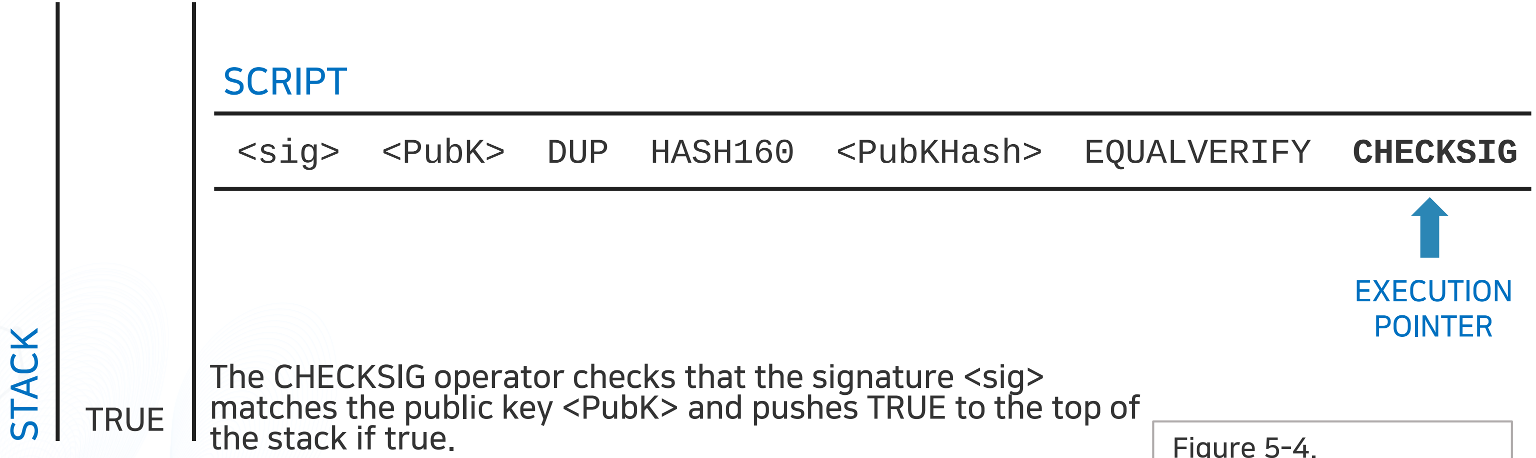


Figure 5-4. Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 2 of 2)

GIST

5 Multisignature and Smart Contracts Scripts

- Other Scripts
  - Pay to Public Key (P2PK), introduced in the Bitcoin white paper.
  - Pay to Public Key Hash (P2PKH), used in the code by Satoshi Nakamoto.
  - Pay to Script Hash (P2SH), introduced winter of 2012.
    - These Bitcoin addresses are beginning with 3.
    - Hash of a script is the beneficiary.
    - It can be used for a `multisignature` script.
      - M out of N keys are needed to spend the value.
      - Useful for joint accounts

5 Multisignature and Smart Contracts Scripts

- Bitcoin uses scripts for Smart Contracts
  - There are many different possibilities that can be expressed with this scripting language.
  - Smart contracts can be programmed in to code which expresses more complex conditions for spending and how these conditions can be satisfied by unlocking scripts.
  - *This language allows for a nearly infinite variety of conditions to be expressed.*
  - *This is how bitcoin gets the power of "programmable money." (Mastering Bitcoin)*

GIST

5 Multisignature and Smart Contracts Scripts

- Bitcoin does not allow any loop for stable operations.
- Ethereum does.
    - `Jump` and `JumpTo` are used in the list of OP codes.
    - https://github.com/crytic/evm-opcodes.
- Bitcoin is more prudent and focuses on safety.

Goal of this lecture note

- Blockchain Core
- Program Package
- Python Blockchain Core

1 Blockchain Core

- We aim to study what a blockchain core is.
  - Cryptocurrency is built on a program suite.
  - The suite is to form and maintain a ledger in a P2P computer network.
  - The best way to learn is to develop it from scratch.
  - We need to install SW package and do a little bit of coding.

GIST

1 Blockchain Core

- A simple Python core is written.
  - This code controls a node.
  - It can have nodes interact with each other.
  - A group of such nodes can support a cryptocurrency system.

1 Blockchain Core

- List of things we aim to do:
  - Run the core at a group of nodes,
  - Have nodes register their neighbors,
  - Have nodes generate new transactions,
  - Have nodes mine new blocks,
  - Have nodes reach consensus, and show
- This network can maintain a blockchain.

GIST

1 Blockchain Core

- Define node discovery routines:
  - Be aware of neighbors
  - Give my list of addresses upon requests
  - Listen to chains and transactions announcements and get them from neighbors

1 Blockchain Core

- Define what a block is:
  - BH: Previous Hash, Merkle Root Hash, Timestamp, Nonce,
    Version, Difficulty
  - BB: Transactions, Tree Structure
- Make the genesis block.

1 Blockchain Core

- Define transaction generation routines
  - Generate keys and addresses.
  - Make a new transaction:
    - Find UTXOs
    - Get destination addresses
    - Make a locking script per each address
    - Make TXID
  - Announce TX.
  - Track the TXs issued until fully confirmed.
    - check to see if TXs are included in the main chain.
    - re-issue those TXs not included in the main chain.

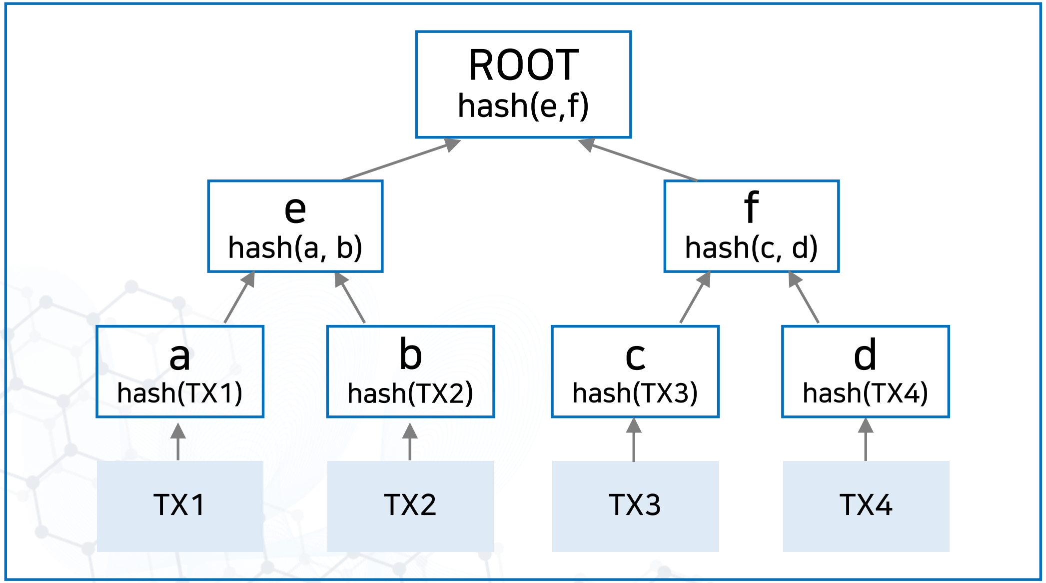
1 Blockchain Core

- Define a transaction verification routine
  - Get a TX and validate it.
  - Input UTXOs with locking scripts.
  - See if each sign unlocks the lock.
  - Check output values in the locking scripts.
  - See if the balance is enough.
  - Verify TXID = hash(inputs, outputs).

GIST

1 Blockchain Core

- Define Merkle root hash routine
  - Binary hash tree of TXIDs



GIST

1 Blockchain Core

- Define a block verification routine.
  - Verify each transaction.
  - Verify the Merkle root hash.
  - Verify the hash of the BH.
    - Put the prev. block header into SHA-256 and see if it satisfies the difficulty level.
    - The difficulty level cannot be forged since it is included in the block header.

1 Blockchain Core

- Define difficulty change routine.
  - Change `Target` periodically.

1 Blockchain Core

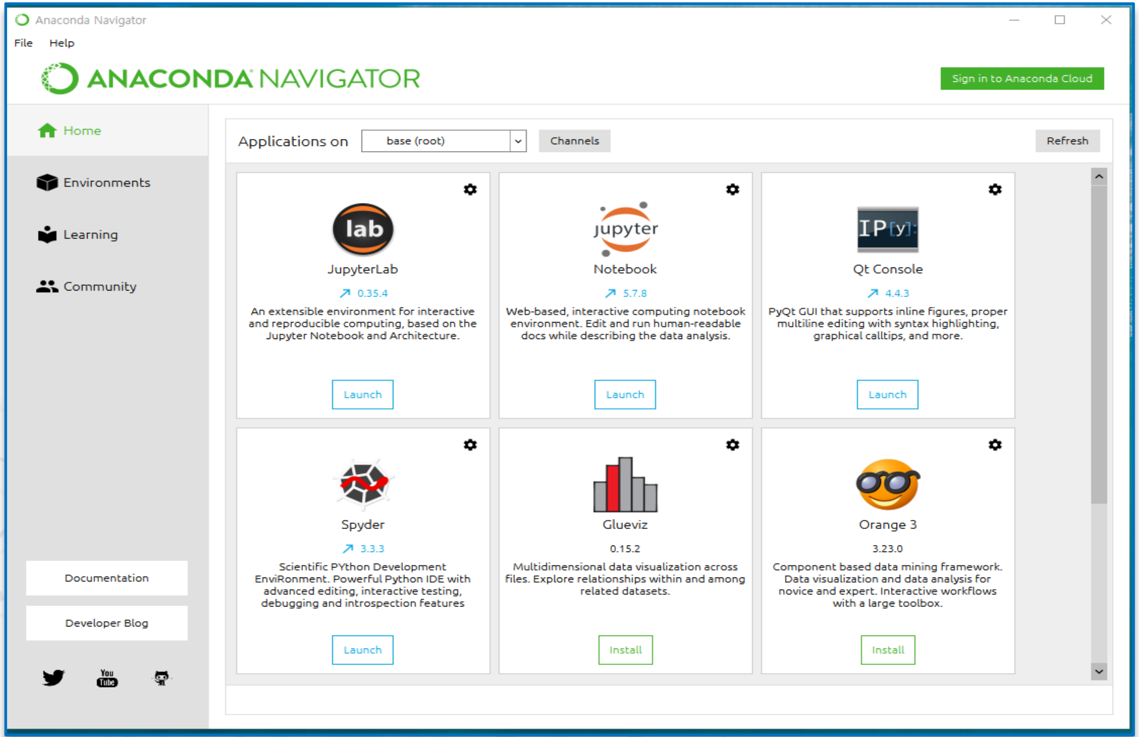
- Define a mining routine.
  - Collect announced transactions.
  - Get the longest chain from the neighbors.
  - Validate the imported chain.
    - Verify the blocks.
    - Verify the sequence of proofs.
  - Form a new block by finding a good nonce.
  - Announce the new chain ASAP.

2 Program Package

- *Anaconda*
  - https://www.anaconda.com/distribution/
  - Free OS *Python*
  - *Spyder*
    - *Write python code and run*
- *FLASK*
  - Use it to write an API in *Python*
- *Postman*
  - Use it to test APIs.
  - https://www.getpostman.com/downloads/

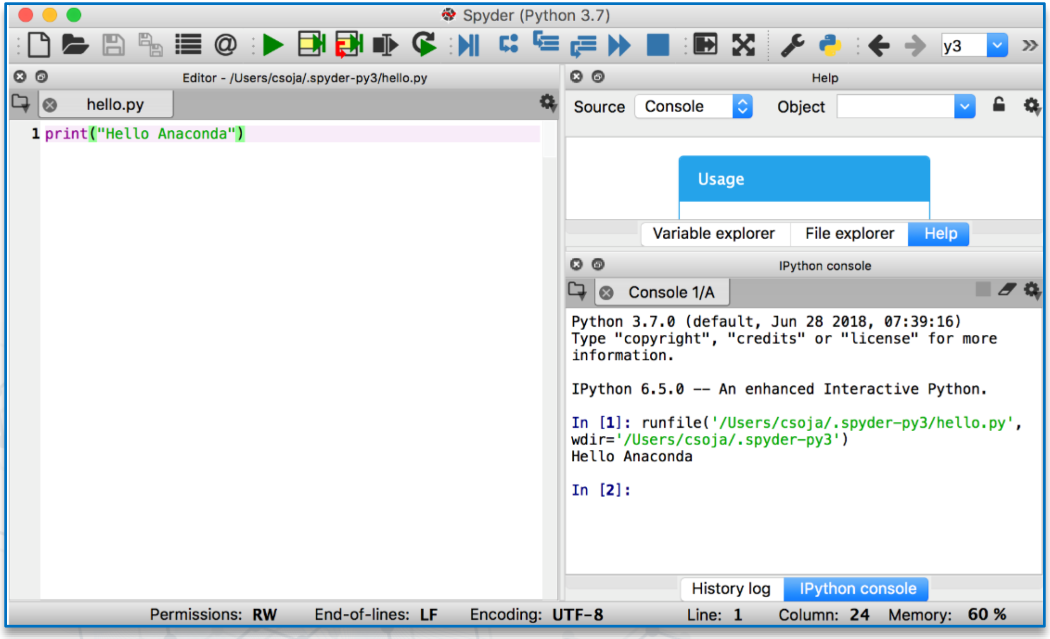
GIST

2 Program Package



2 Program Package

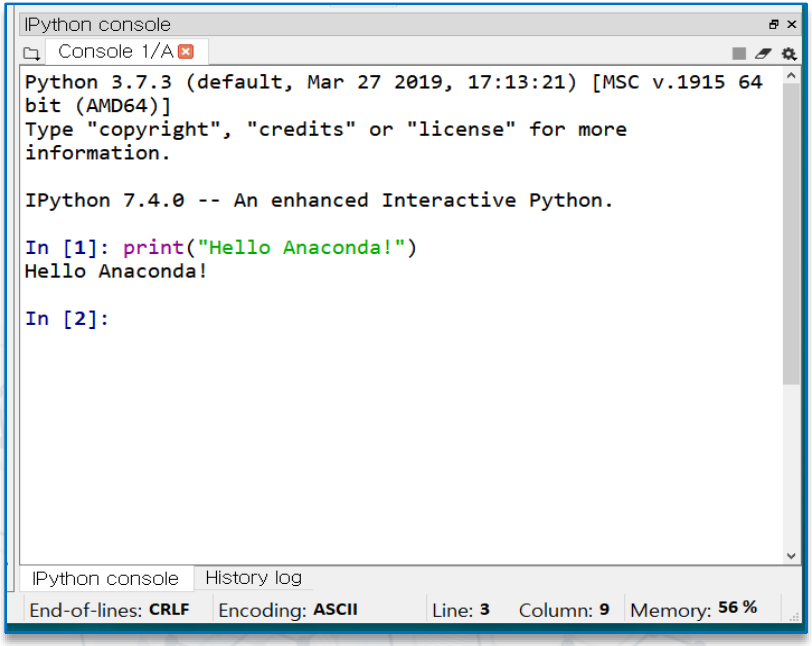
- Edit and run Python at Spyder IDE



출처: https://docs.anaconda.com/anaconda/user-guide/getting-started/

2 Program Package

- Open an Anaconda terminal and run a Python program.



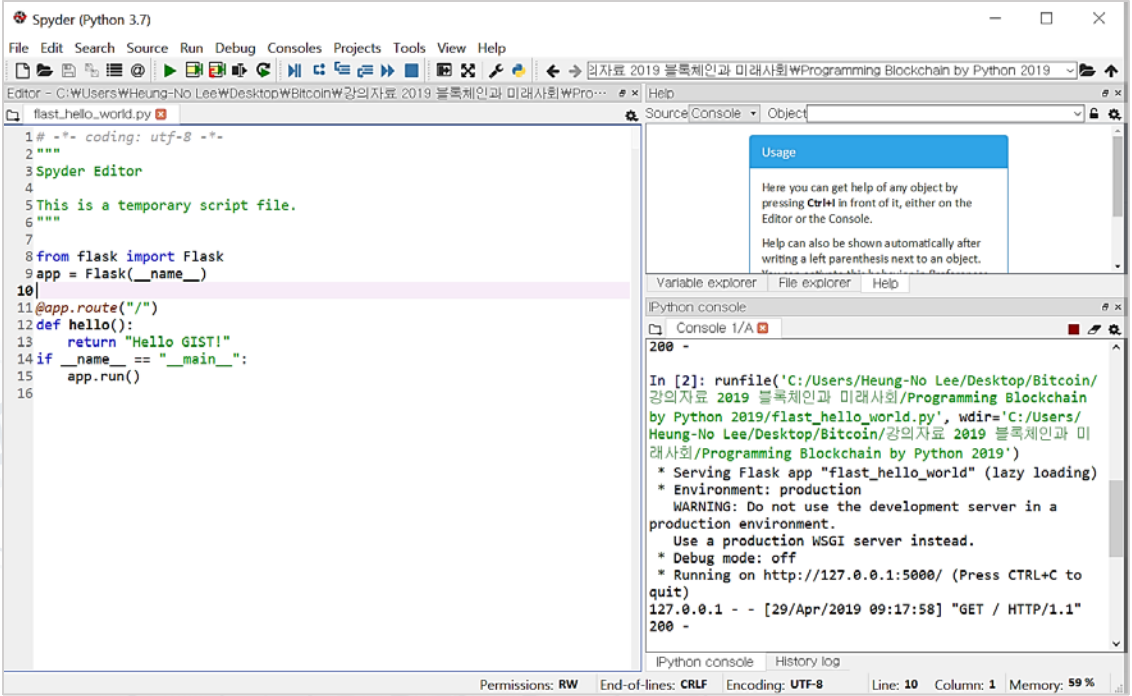
2 Program Package

- FLASK
  - Flask is a micro web development tool written in Python.
  - The following code shows a simple web application that prints "Hello World!":

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
        return "Hello World!“

if __name__ == "__main__":
        app.run()
```

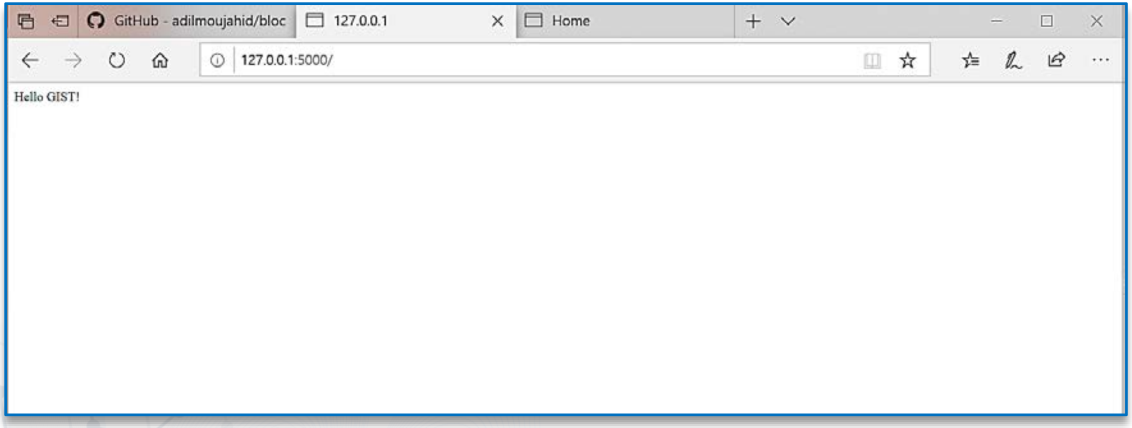
2 Program Package

- Write the First FLASK code



2 Program Package

- Confirmation



3 Python Blockchain Core

- Download the Python blockchain files.
  - https://github.com/infonetGIST/Blockchain_lecture
- There are three kinds of Python files:
  - blockchain.py,
  - miner1.py, miner2.py, miner3.py
  - trader.py

GIST

3 Python Blockchain Core

- Python code for a Blockchain
  - Open up `blockchain.py` file in Spyder
  - It defines the blockchain class under which all core routines are defined.
  - It is only 531 lines long (17 def's and 9 app's)

Chaining	Mining
Resolving the chain by its length	Posting TXs (limited to itself)

3 Python Blockchain Core

- Blockchain.py

Import libraries

Declare Blockchain class

Define Flask app's

```
Import

class Blockchain:

# Instantiate the Node
app = Flask(__name__)

# Instantiate the Blockchain
blockchain = Blockchain()
```

3 Python Blockchain Core

- Import libraries

```
from threading import Thread, Event
import time
from flask import Flask, jsonify, request
import requests
import hashlib
import json
from urllib.parse import urlparse
from uuid import uuid4
import random
```

GIST

3 Python Blockchain Core

- class blockchain has 17 definitions

```
def __init__(self):
def register_node
def valid_chain
def resolve_conflicts
def new_block
def new_transaction
def update_transactions
def is_valid_TX
def check_current_TXs_validity
def update_awaiting_TX
def make_published_TXID_list
```

3 Python Blockchain Core

- class blockchain has 17 definitions

```
def announcement
def mine
def last_block
def hash
def proof_of_work
def valid_proof
```

GIST

3 Python Blockchain Core

```
class Blockchain:
    def __init__(self):
        self.current_transactions = []
        self.awaiting_transactions =[]
        self.chain = []
        self.nodes = set()
        self.published_transactions_ID = []
        self.mining_reward_address='0'
        self.MY_NODE_ADDRESS='0'
        # Generate a globally unique address for this node
        self.node_identifier = str(uuid4()).replace('-', '')
        # node_identifier = hex(random.randrange(1, 9999999))
        self.interrupt_flag=False

        dummy_block = {
            'index': 0,
            'timestamp': 0,
            'transactions': [],
            'proof': 0,
            'previous_hash': 0,
        }
        Ini_proof = self.proof_of_work(mining_time=0, last_block=dummy_block)
        self.new_block(previous_hash='0', mining_time=0, proof=Ini_proof)
```

3 Python Blockchain Core

- Take a look at some def's under blockchain class

```
    def register_node(self, address):
        """
        Add a new node to the list of nodes

        :param address: Address of node. Eg. 'http://192.168.0.5:5000'
        """

        parsed_url = urlparse(address)
        if parsed_url.netloc:
            self.nodes.add(parsed_url.netloc)
        elif parsed_url.path:
            # Accepts an URL without scheme like '192.168.0.5:5000'.
            self.nodes.add(parsed_url.path)
        else:
            raise ValueError('Invalid URL')
```

GIST

3 Python Blockchain Core

```
    def new_block(self, mining_time, proof, previous_hash):
        """
        Create a new Block in the Blockchain

        :param proof: The proof given by the Proof of Work algorithm
        :param previous_hash: Hash of previous Block
        :return: New Block
        """

        block = {
            'index': len(self.chain) + 1,
            'timestamp': mining_time,
            'transactions': self.current_transactions,
            'proof': proof,
            'previous_hash': previous_hash or self.hash(self.chain[-1]),
        }

        # Reset the current list of transactions
        self.current_transactions = []

        self.chain.append(block)
        self.make_published_TXID_list()
        return block
```

3 Python Blockchain Core

```python
    def mine(self):
        # We run the proof of work algorithm to get the next proof...
        last_block = self.last_block
        mining_time = time.time(),
        randomSTR = str(uuid4()).replace('-', '')
        self.new_transaction(
            sender="Coinbase transaction",
            recipient=self.mining_reward_address + '  #' + randomSTR,
            amount=1,
        )
        proof = self.proof_of_work(mining_time, last_block)

        # We must receive a reward for finding the proof.
        # The sender is "0" to signify that this node has mined a new coin.

        # Forge the new Block by adding it to the chain
        if proof==0:
            del self.current_transactions[-1]
        else:
            previous_hash = self.hash(last_block)
            block = self.new_block(mining_time, proof, previous_hash)
            print("Mining success!")
            self.announcement()
```

GIST

3 Python Blockchain Core

```python
def proof_of_work(self, mining_time, last_block):

    last_proof = last_block['proof']
    if len(self.chain) == 0:
        last_hash = '0'
    else:
        last_hash = self.hash(last_block)

    proof = 0
    test_block = {
        'index': len(self.chain) + 1,
        'timestamp': mining_time,
        'transactions': self.current_transactions,
        'proof': proof,
        'previous_hash': last_hash or self.hash(self.chain[-1]),
    }
    while self.valid_proof(test_block) is False:
        if self.interrupt_flag:
            blockchain.interrupt_flag = False
            return 0

        proof += 1
        test_block = {
            'index': len(self.chain) + 1,
            'timestamp': mining_time,
            'transactions': self.current_transactions,
            'proof': proof,
            'previous_hash': last_hash or self.hash(self.chain[-1]),
        }

    return proof
```

3 Python Blockchain Core

- Flask app's
  - GETs and POSTs
    - Mine a block
    - Post a TX
    - Make a chain
    - Get transactions
    - Get chain updates
    - Register a node
    - Make consensus
    - Shut down

```
@app.route('/mine', methods=['GET'])
def mine():

@app.route('/transactions/new', methods=['POST'])
def new_transaction():

@app.route('/chain', methods=['GET'])
def full_chain():

@app.route('/get_transactions', methods=['GET'])
def full_transactions():

@app.route('/get_awaiting_transactions')
def awaiting_transactions():
```

GIST

3 Python Blockchain Core

- Flask app's
  - GETs and POSTs
    - Mine a block
    - Post a TX
    - Make a chain
    - Get transactions
    - Get chain updates
    - Register a node
    - Make consensus
    - Shut down

```
@app.route('/get_updates')
def receiving_longest_chain_and_update_TX_list():

@app.route('/nodes/register', methods=['POST'])
def register_nodes():

@app.route('/nodes/resolve', methods=['GET'])
def consensus():


@app.route("/shutdown")
def shutdown()
```

Goal of this lecture note

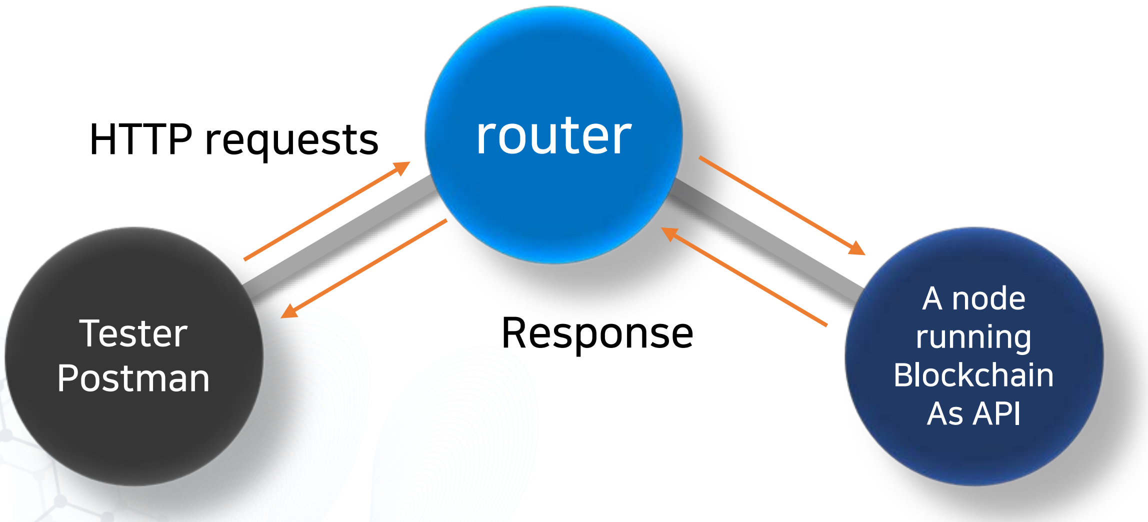
- Running and Testing Blockchain API
- Blockchain Internet
- Six Node Blockchain Network

1 Running and Testing Blockchain API

- Aim to test the core.
- Run the core at a single node.
- Step-by-step testing each routine
  - Register its neighbors.
  - Generate new transactions
  - Mine new blocks (mint coins)
  - Blocks are chained using PoWs.
  - Difficulty level of PoW is changed with leading number zeros.

GIST

1 Running and Testing Blockchain API



GIST

1 Running and Testing Blockchain API

- Have API running core at 127.0.0.21:2000

Append this code inside blockchain.py

```
if __name__ == '__main__':
    from argparse import ArgumentParser

    parser = ArgumentParser()
    parser.add_argument('-p', '--port', default=5000, type=int, help='port to listen on')
    args = parser.parse_args()
    port = args.port

    app.run(host='127.0.0.1', port=2000)
```

At the console

```
In [3]: runfile('C:/Users/Heung-No Lee/
Desktop/Bitcoin/MooC 강의/블록체인 Python/
blockchain_homework_python/
blockchain_core.py', wdir='C:/Users/Heung-No
Lee/Desktop/Bitcoin/MooC 강의/블록체인 Python/
blockchain_homework_python')
 * Serving Flask app "blockchain_core" (lazy
loading)
 * Environment: production
   WARNING: Do not use the development server
in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:2000/ (Press
CTRL+C to quit)
```

Now, we can use Postman to interact with this API!

GIST

1 Running and Testing Blockchain API

- Post a transaction

```
@app.route('/transactions/new', methods=['POST'])
def new_transaction():
    values = request.get_json()

    # Check that the required fields are in the POST'ed data
    required = ['sender', 'recipient', 'amount']
    if not all(k in values for k in required):
        return 'Missing values', 400

    # Create a new Transaction
    index = blockchain.new_transaction(values['sender'], values['recipient'], values['amount'])

    response = {'message': f'Transaction will be added to Block {index}'}
    return jsonify(response), 201
```

GIST

1 Running and Testing Blockchain API

- Post a transaction
  - Use Postman
  - Put the following JSON script into the Body
  - Select “raw” in the window
  - Select JSON(application/json) from the pull down menu

GIST

1 Running and Testing Blockchain API

- Post a transaction

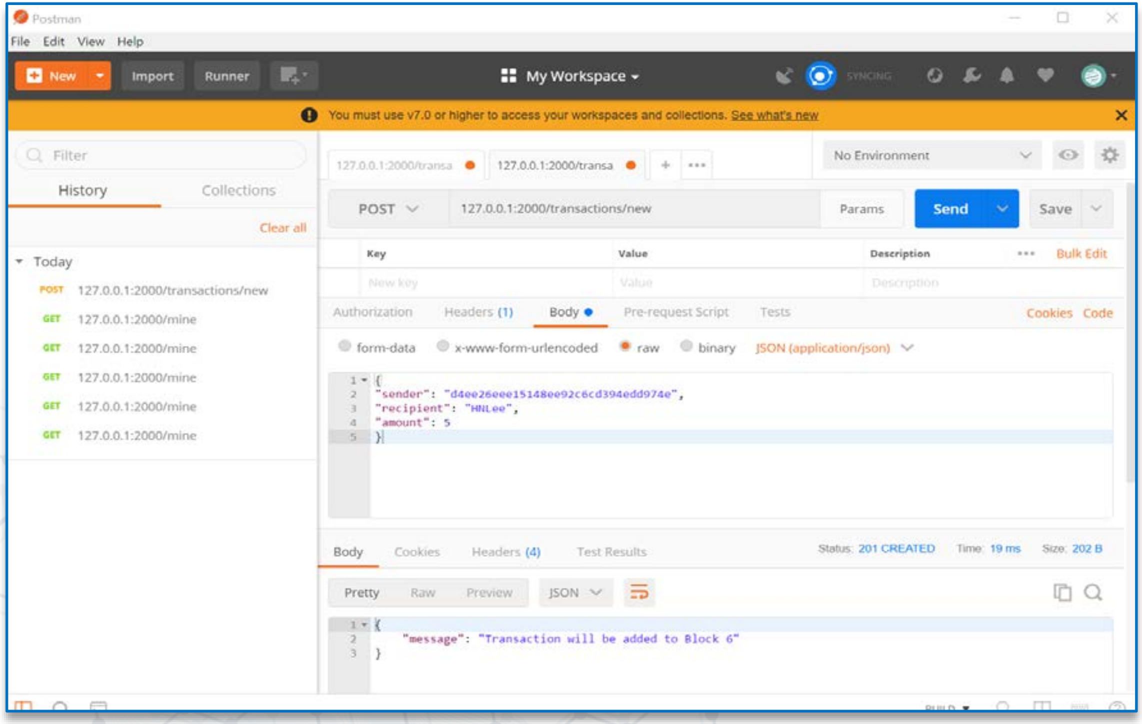
{
"sender": "d4ee26eee15148ee92c6cd394edd974e",
"recipient": "HNLee",
"amount": 5
}

- If successful, you will see this message:

{
"message": "Transaction will be added to Block 2"
}

GIST

1 Running and Testing Blockchain API



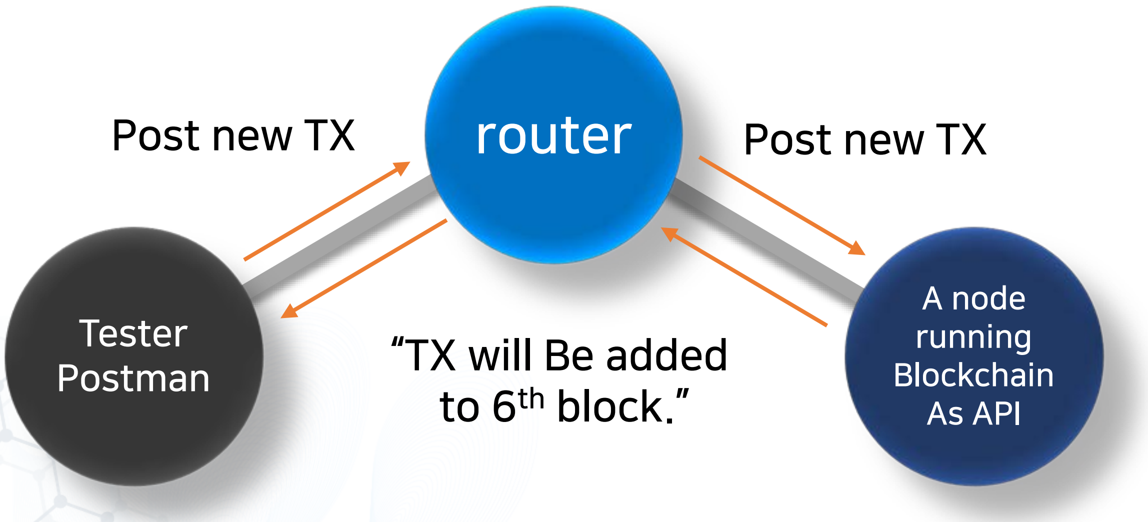
GIST

1 Running and Testing Blockchain API

```
In [3]: runfile('C:/Users/Heung-No Lee/
Desktop/Bitcoin/MooC 강의/블록체인 Python/
blockchain_homework_python/
blockchain_core.py', wdir='C:/Users/Heung-No
Lee/Desktop/Bitcoin/MooC 강의/블록체인 Python/
blockchain_homework_python')
 * Serving Flask app "blockchain_core" (lazy
loading)
 * Environment: production
   WARNING: Do not use the development server
in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:2000/ (Press
CTRL+C to quit)
127.0.0.1 - - [17/Oct/2019 22:10:50] "POST /
transactions/new HTTP/1.1" 201 -
```

GIST

1 Running and Testing Blockchain API



GIST

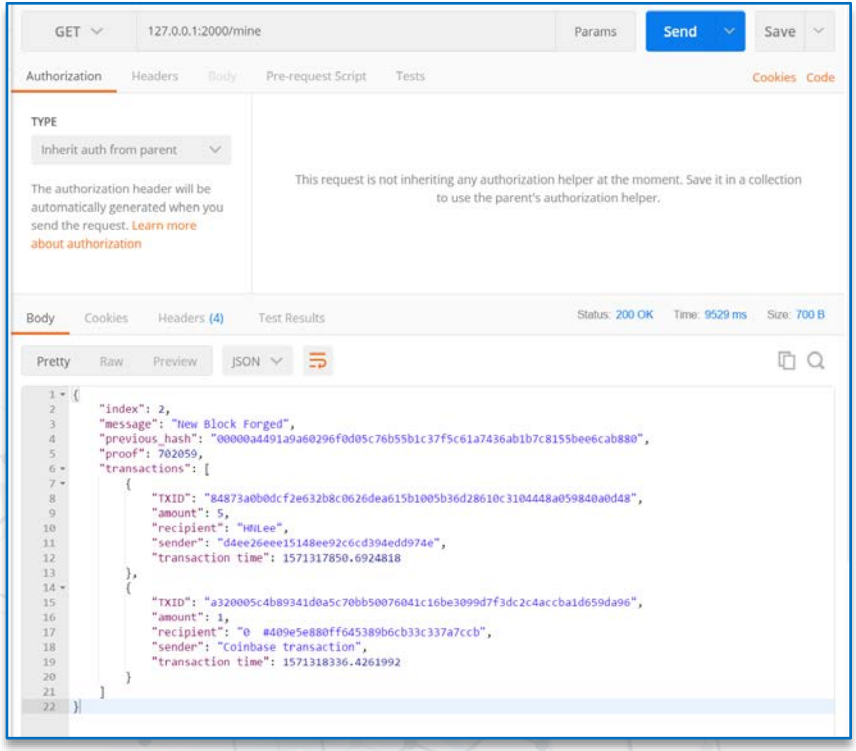
1 Running and Testing Blockchain API

- Mine a block

```
@app.route('/mine', methods=['GET'])
def mine():
    # We run the proof of work algorithm to get the next proof...
    last_block = blockchain.last_block
    proof = blockchain.proof_of_work(last_block)

    # We must receive a reward for finding the proof.
    # The sender is "0" to signify that this node has mined a new coin.
    blockchain.new_transaction(
        sender="0",
        recipient=node_identifier,
        amount=1,
    )

    # Forge the new Block by adding it to the chain
    previous_hash = blockchain.hash(last_block)
    block = blockchain.new_block(proof, previous_hash)

    response = {
        'message': "New Block Forged",
        'index': block['index'],
        'transactions': block['transactions'],
        'proof': block['proof'],
        'previous_hash': block['previous_hash'],
    }
    return jsonify(response), 200
```

GIST

1 Running and Testing Blockchain API



GIST

1 Running and Testing Blockchain API

- Chain the blocks

```
@app.route('/chain', methods=['GET'])
def full_chain():
    response = {
        'chain': blockchain.chain,
        'length': len(blockchain.chain),
    }
    return jsonify(response), 200
```

GET 127.0.0.1:2000/chain Params Send


```
{
    "chain": [
        {
            "index": 1,
            "previous_hash": "0",
```

```
        {
            "index": 2,
            "previous_hash": "00009ad060e077fe0fd0eb436bad33f141122074947ff876877ec0e284b316bb",
            "proof": 126362,
            "timestamp": [
                1570432319.5437605
```

```
        {
            "index": 3,
            "previous_hash": "00009813ca2decd07bd52ccdf203524062c82f8148df4f0c08d127e0e85df9f1",
            "proof": 237836,
            "timestamp": [
                1570432341.2409103
            ],
            "transactions": [
                {
                    "TXID": "8c8e0eaed39f72a500c34608804ac0cc40d337aac8cc5060f700e9aca6492b59",
                    "amount": 1,
                    "recipient": "0  #d41937e6688f4bcf9d1cd70026e17a72",
                    "sender": "Coinbase transaction",
                    "transaction time": 1570432341.2409103
                }
            ]
        },
```

GIST

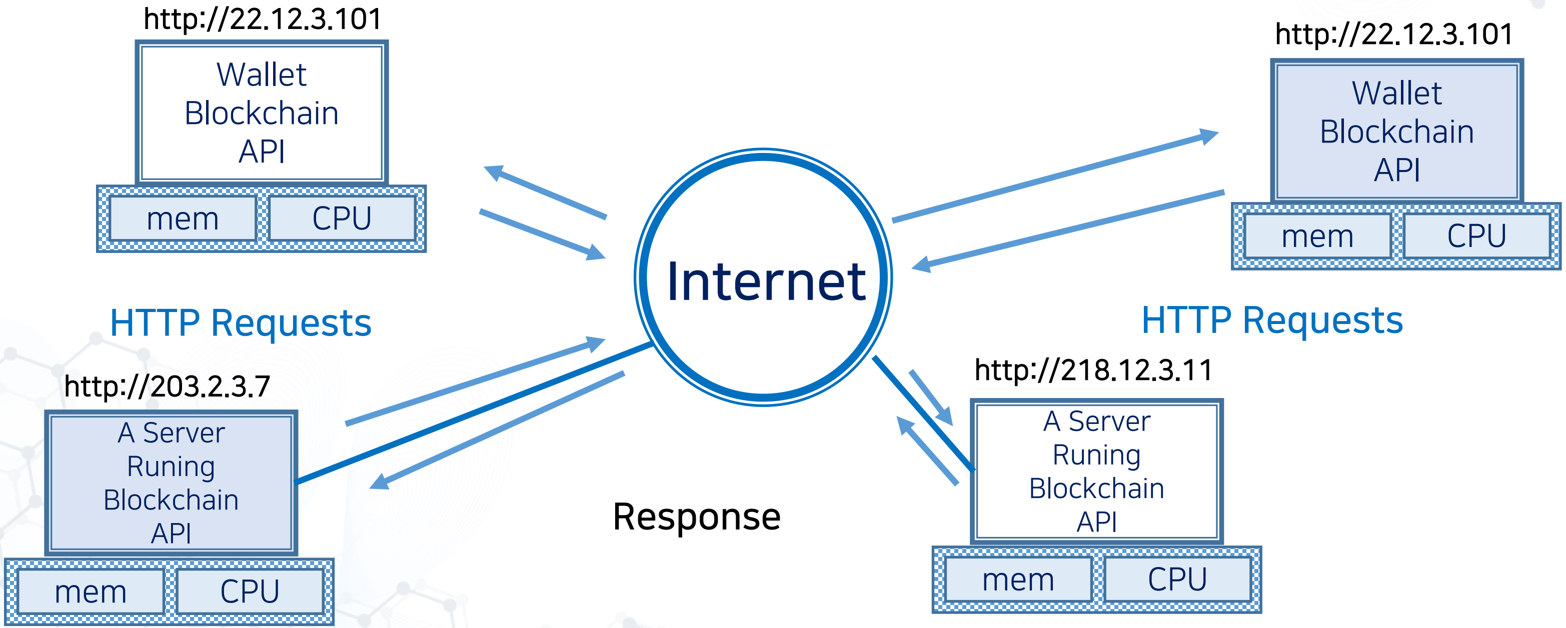
1 Running and Testing Blockchain API

- We can test out others as well
  - Get transactions
  - Get chain updates
  - Register a node
  - Make consensus
  - Shut down
- Step-by-step guidance at https://infonet.gist.ac.kr/

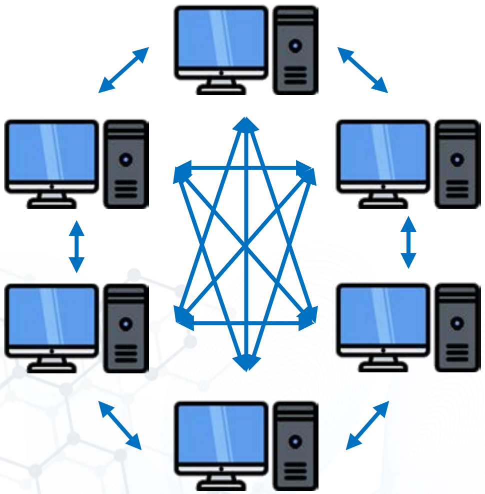
2 Blockchain Internet

- Any node which downloads the core can serve as a P2P node.
- Collection of these nodes form a blockchain internet.
  - Wallet holders
  - Miners
  - Full nodes

- Blockchain Internet



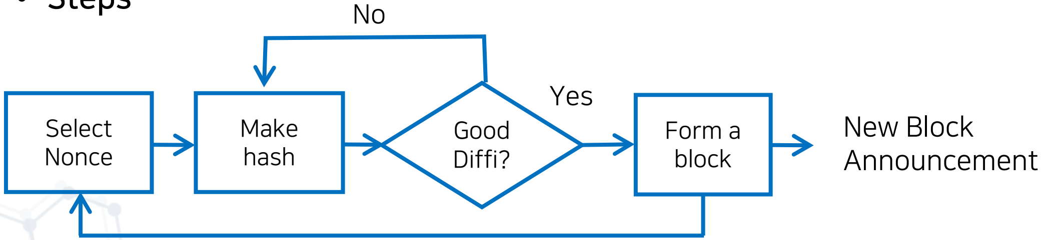
3 Six Node Blockchain Network



- Simple Chain Structure

	Block 1	Block 2	Block 3
Index	8	9	10
Nonce	66808	254479	157752
Body	Transactions : {'Sender' :' 0', 'Receipient' : '8911...', 'Amount' : 50} Num_zeros : 5	Transactions : {'Sender' :' 0', 'Receipient' : '6a06...', 'Amount' : 50}, {'Sender' :' 89a4...', 'Receipient' : '27b6......', 'Amount' : 50}, {'Sender' : ...} Num_zeros : 5	Transactions : {'Sender' :' 0', 'Receipient' : '6e30...', 'Amount' : 50} Num_zeros : 5
Previous hash	00000c567d49cfd12272e...	00000f562087e3a5c75776...	00000dabc94224d86c5076...
Current hash	00000f562087e3a5c75776...	00000dabc94224d86c5076...	00000g922087e3a5c24376...

- Steps



GIST

3 Six Node Blockchain Network

- Aim is to show how blockchain works.
- Simplicity is the key
  - BH: index (block height), nonce, prev_hash, num_zeros
  - BB: TXs
- Change difficulty by leading number of zeros.
- Networking and longest chain consensus
  - Each node asks for the longest chain upon starting.
  - Listens TXs and get them to its new block.
  - Listens new chain announcements.
  - Compares and adopts the longest valid chain.

GIST

3 Six Node Blockchain Network

- The five miner nodes running `miner.py` are
  - Node 1 (IP: 172.26.16.41) is a mining node.
  - Node 2 (IP: 172.26.16.66) is a mining node.
  - Node 3 (IP: 172.26.16.43) is a mining node.
  - Node 4 (IP: 172.26.16.42) is a mining node.
  - Node 5 (IP: 172.26.16.32) is a mining node.
- The trader node running `trader.py` is
  - Node 6 (IP: 203.237.54.101) which is the transaction generating node
- The port number is 5000 for all nodes.

GIST

3 Six Node Blockchain Network

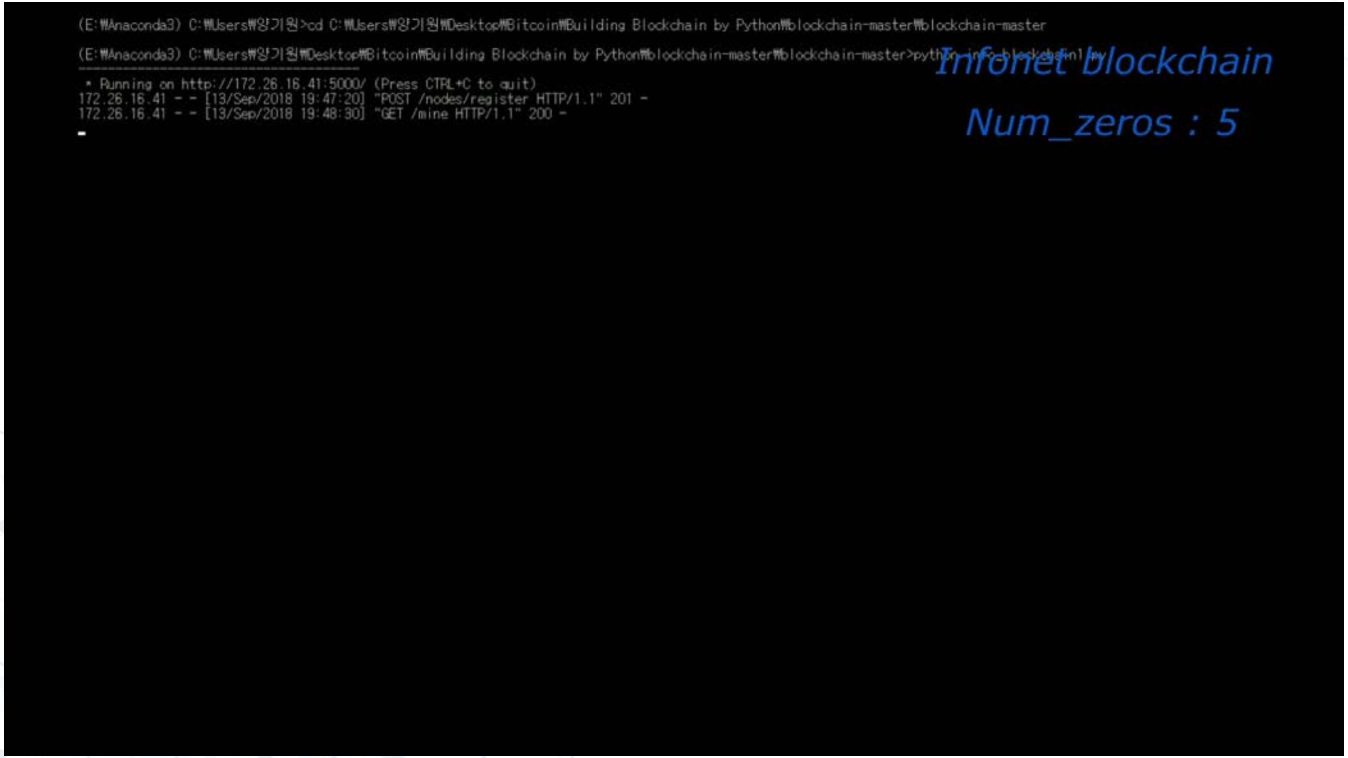
- We ran the experimental set up and captured it into a video file.
- The captured video is the feed ran at the console of Node 1.
  - Open up an Anaconda console at Node 1
  - Move to the directory in which the blockchain core file is located.
  - Run blockchain with the command:

```
>python blockchain.py
```

    - Make sure the core is running at the console.
    - Node 1 is now a blockchain server in this small network.
- Other nodes shall be started off as well with the same procedure.
- This is not shown in this console since this console is at Node 1.

GIST

3 Six Node Blockchain Network



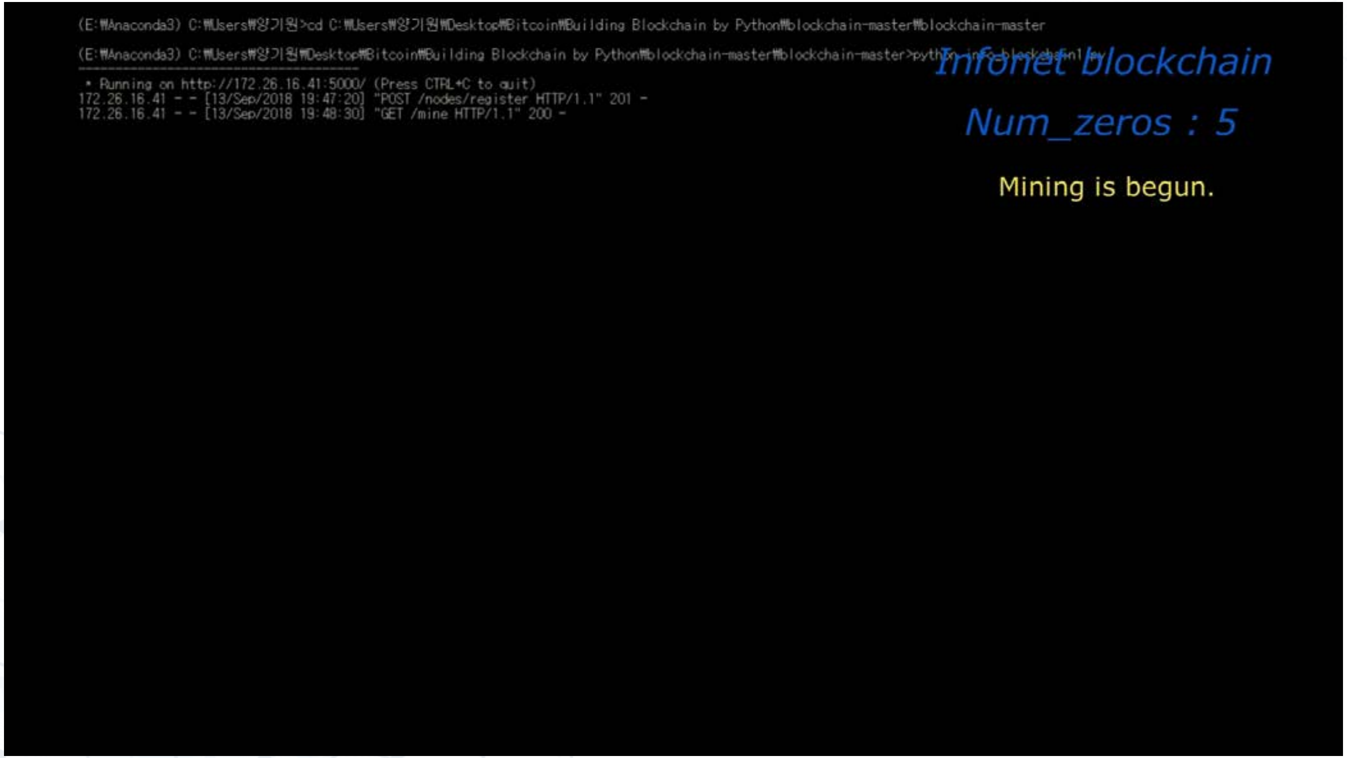
GIST

3 Six Node Blockchain Network

- First, other nodes get registered as neighbors of Node 1.
- Node 1 (IP: 172.26.16.41) starts mining!
  - As soon as it has started mining, it first aims to gather all the chains from its neighbors.

GIST

3 Six Node Blockchain Network



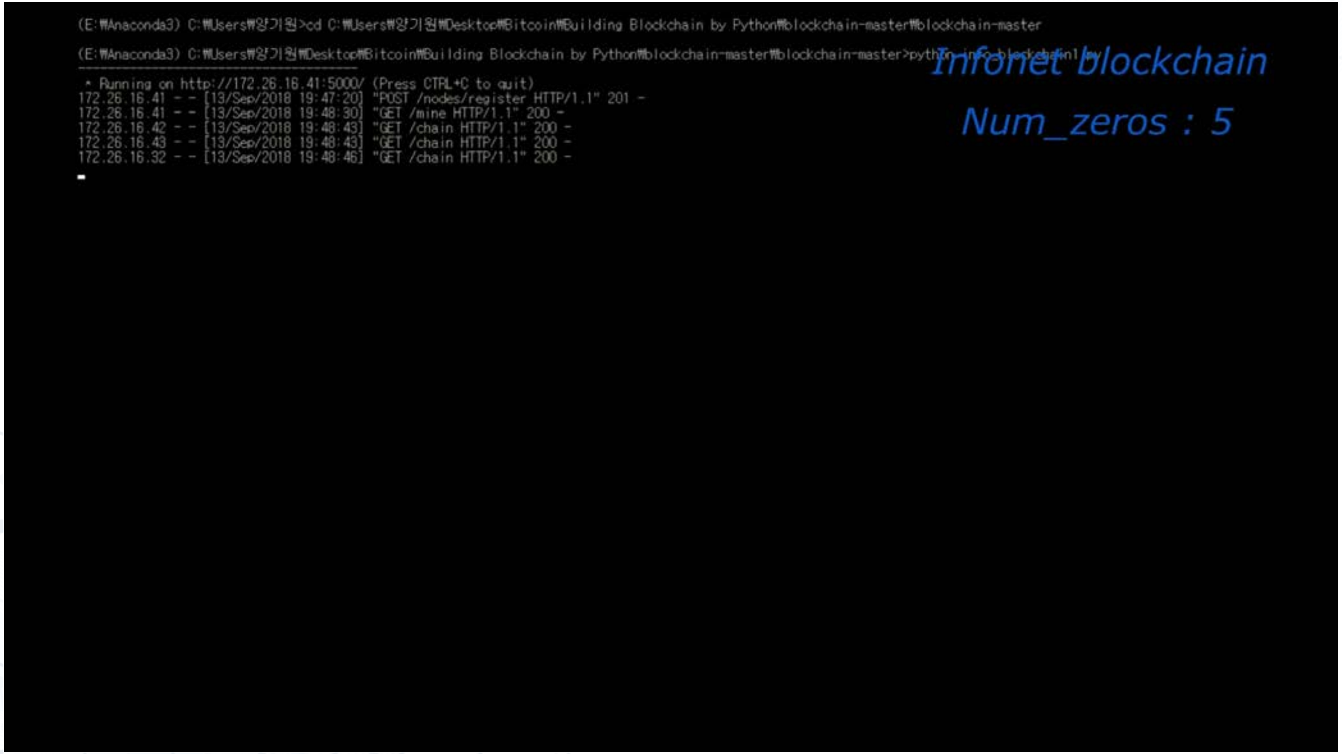
GIST

3 Six Node Blockchain Network

- Node 1 requests to get chains from its neighbors,
  - that of Node 4 (IP: 172.26.16.42),
  - that of Node 3 (IP: 172.26.16.43), and
  - that of Node 5 (IP: 172.26.16.32).

GIST

3 Six Node Blockchain Network

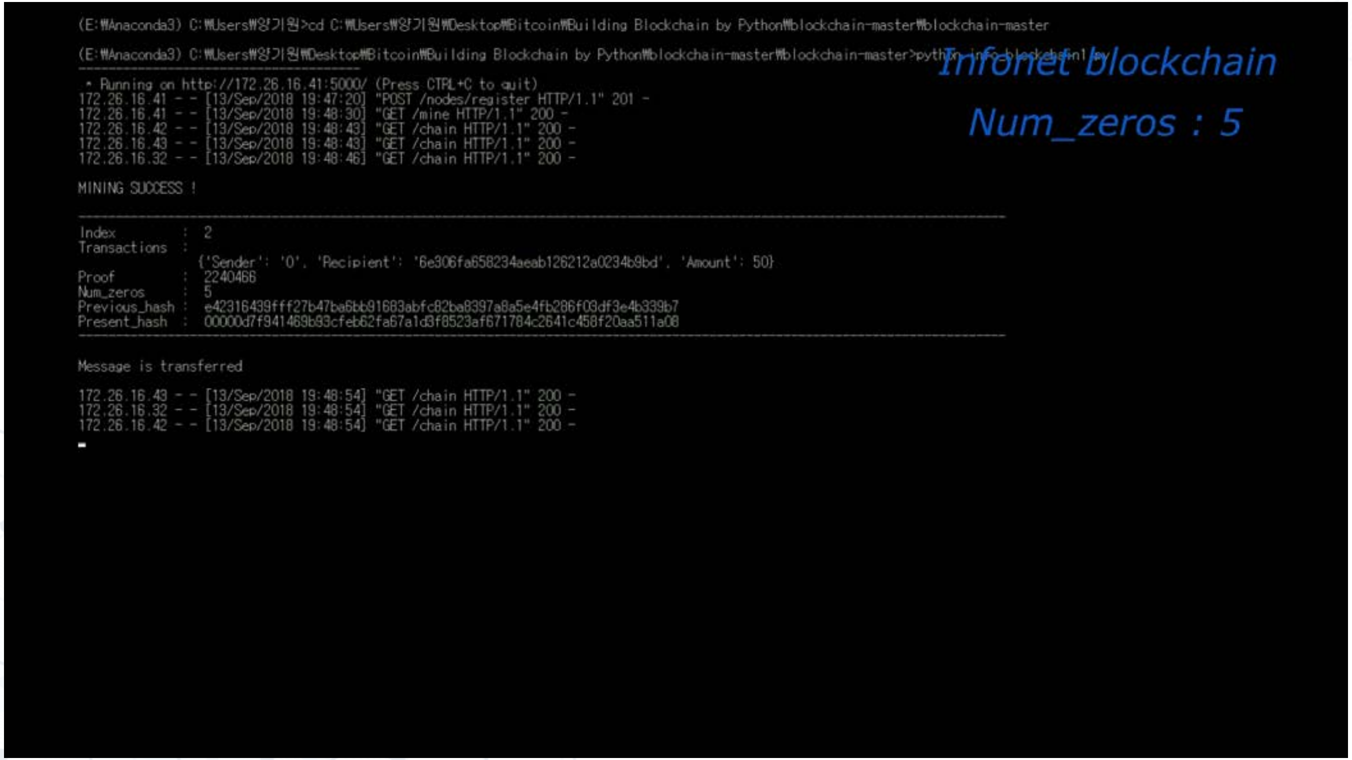


3 Six Node Blockchain Network

- Node 1 announces its mining success to neighbors.
  - Other nodes stop mining their current block, accept this chain and start mining again aiming to grow this adopted chain.

GIST

3 Six Node Blockchain Network



GIST

3 Six Node Blockchain Network

- Node 4 announces the third block mining success.
  - Adopting it by other nodes follows.
- Again, Node 4 mines the 4th block.
- The 5th block is mined by Node 3.
- Node 6 generates a transaction.
- It is included in the 6th block which is mined by Node 2.

GIST

3 Six Node Blockchain Network

```
172.26.16.43 - - [13/Sep/2018 19:48:54] "GET /chain HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:48:54] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:48:54] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:48:57] "GET /chain HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:48:58] "GET /chain HTTP/1.1" 200 -

Node4 has mined!

172.26.16.42 - - [13/Sep/2018 19:48:58] "GET /message HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  3
Transactions   :
                 {'Sender': '0', 'Recipient': '8911a636c7904d92abcb87ef428743bc', 'Amount': 50}
Proof          :  317538
Num_zeros      :  5
Previous_hash  :  00000d7f941469b93cfeb62fa67a1d3f8523af671784c2641c458f20aa511a08
Present_hash   :  000002acb0eeb935849cc8ae7844e65e2f0119f2f350c5aff5287f6efc2ef7dc
--------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:48:58] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:48:58] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:48:59] "GET /chain HTTP/1.1" 200 -

Node4 has mined!

172.26.16.42 - - [13/Sep/2018 19:48:59] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:48:59] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  4
Transactions   :
                 {'Sender': '0', 'Recipient': '8911a636c7904d92abcb87ef428743bc', 'Amount': 50}
Proof          :  39250
Num_zeros      :  5
Previous_hash  :  000002acb0eeb935849cc8ae7844e65e2f0119f2f350c5aff5287f6efc2ef7dc
Present_hash   :  000004dfeb5dc6c80ade4aa947abeb9a8fac899ce1b8f7837b1f0e9062244488
--------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:48:59] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:00] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -

Node3 has mined!

172.26.16.43 - - [13/Sep/2018 19:49:01] "GET /message HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:02] "GET /chain HTTP/1.1" 200 -
```

Infonet blockchain

Num_zeros : 5

GIST

3 Six Node Blockchain Network

- Node 6 generates a transaction.
  - It is included in the 6th block which is mined by Node 2.
- In each block, the first TX is the coinbase TX, and mining reward of 50 coins is paid to miner's address.

GIST

3 Six Node Blockchain Network

```
Previous_hash :  000002acb0eeb935849cc8ae7844e65e2f0119f2f350c5aff5287f6efc2ef7dc
Present_hash  :  000004dfeb5dc6c80ade4aa947abeb9a8fac899ce1b8f7837b1f0e9062244488
--------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:48:59] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:00] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -

Node3 has mined!

172.26.16.43 - - [13/Sep/2018 19:49:01] "GET /message HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:01] "GET /chain HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:02] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  5
Transactions   :
                  {'Sender': '0', 'Recipient': '3e4d91d9ca6a434a828d846a8caba5bf', 'Amount': 50}
Proof          :  258150
Num_zeros      :  5
Previous_hash  :  000004dfeb5dc6c80ade4aa947abeb9a8fac899ce1b8f7837b1f0e9062244488
Present_hash   :  000007f905c10024beaea27e804dfb7c96049ca100440c9b73a90c3a2ba7e41b
--------------------------------------------------------------------------------

172.26.16.43 - - [13/Sep/2018 19:49:02] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:49:02] "GET /chain HTTP/1.1" 200 -
203.237.54.101 - - [13/Sep/2018 19:49:02] "POST /transactions/get HTTP/1.1" 201 -

Node2 has mined!

172.26.16.166 - - [13/Sep/2018 19:49:05] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:05] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:06] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  6
Transactions   :
                  {'Sender': '0', 'Recipient': '0219783336594a3894a6c0b7dbe54eef', 'Amount': 50}
                  {'Sender': '89a4bcbb9032a62d33cefc2c392ab122', 'Recipient': '27b63aa197bc10efff46871af2d3771e', 'Amount': 10}
Proof          :  305059
Num_zeros      :  5
Previous_hash  :  000007f905c10024beaea27e804dfb7c96049ca100440c9b73a90c3a2ba7e41b
Present_hash   :  000001d6c656590a4baae37379b4cfa40aa470d9fdf18e18be8d372ae4731bd5
--------------------------------------------------------------------------------

172.26.16.166 - - [13/Sep/2018 19:49:06] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:49:06] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:06] "GET /chain HTTP/1.1" 200 -
```

Infonet blockchain

Num_zeros : 5

A new transaction is created and sent to miners.

The created transaction is saved into the sixth block.

GIST

3 Six Node Blockchain Network

- This continues…
- 7th Block minded by Node 4.
- 8th Block minded by Node 4.
- 9th Block minded by Node 5.

- 16th block has two transactions.

GIST

3 Six Node Blockchain Network

```
Previous_hash :  00000f8476e73cfbbb030897b142d21574c86eff8c2f02fcad210e8986310113
Present_hash  :  0000041eb81d5b7b4fa1d115a0dddf14081ede36e642091fb25693d6426d57ee
------------------------------------------------------------------------------------------

172.26.16.43 - - [13/Sep/2018 19:49:39] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:39] "GET /chain HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:39] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:49:39] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:40] "GET /chain HTTP/1.1" 200 -

Node4 has mined!

172.26.16.42 - - [13/Sep/2018 19:49:40] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:40] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:49:40] "GET /chain HTTP/1.1" 200 -

------------------------------------------------------------------------------------------
Index         :  15
Transactions  :
                 {'Sender': '0', 'Recipient': '8911a636c7904d92abcb87ef428743bc', 'Amount': 50}
Proof         :  160306
Num_zeros     :  5
Previous_hash :  0000041eb81d5b7b4fa1d115a0dddf14081ede36e642091fb25693d6426d57ee
Present_hash  :  0000059eb6412bb992f92996f6ad9d7a9b455b24a794b0b5a69adca1d9ce0ba2
------------------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:49:41] "GET /get_block HTTP/1.1" 200 -
203.237.54.101 - - [13/Sep/2018 19:49:41] "POST /transactions/get HTTP/1.1" 201 -
172.26.16.42 - - [13/Sep/2018 19:49:41] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:49:43] "GET /chain HTTP/1.1" 200 -

Node3 has mined!

172.26.16.43 - - [13/Sep/2018 19:49:43] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:49:43] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:49:43] "GET /chain HTTP/1.1" 200 -

------------------------------------------------------------------------------------------
Index         :  16
Transactions  :
                 {'Sender': '0', 'Recipient': '3e4d91d9ca6a434a828d846a8caba5bf', 'Amount': 50}
                 {'Sender': '89a4bcbb9032a62d33cefc2c392ab122', 'Recipient': '27b63aa197bc10efff46871af2d3771e', 'Amount': 18}
Proof         :  264378
Num_zeros     :  5
Previous_hash :  0000059eb6412bb992f92996f6ad9d7a9b455b24a794b0b5a69adca1d9ce0ba2
Present_hash  :  000009c4f60d032a75a27b8f4d297449c7a8c10adbc196b277a9d75ade4cc7fa
------------------------------------------------------------------------------------------

172.26.16.43 - - [13/Sep/2018 19:49:44] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:49:44] "GET /chain HTTP/1.1" 200 -
```

Infonet blockchain

Num_zeros : 5

GIST

3 Six Node Blockchain Network

- This continues…until 30^{th} block.
- From 31^{st} block, difficulty is changed to Num_zeros = 6.

GIST

3 Six Node Blockchain Network

- It takes avg. 4.7 sec to mine a block at 5 leading zeros.
  - This is hexadecimal zeros.
  - Thus, one more zero means 16 x longer.
- The expected time to mine a block is
  - 16 x 4.7 = 75 sec per block.
  - It will now take more than a minute.

GIST

3 Six Node Blockchain Network

```
                     {'Sender': '0', 'Recipient': '3e4d91d9ca6a434a828d846a8caba5bf', 'Amount': 50}
                     {'Sender': '89a4bcbb9032a62d33cefc2c392ab122', 'Recipient': '27b63aa197bc10efff46871af2d3771e', 'Amount': 23}
Proof          :  153240
Num_zeros      :  5
Previous_hash  :  00000d57a85ae8d3948943d093cca21a8cfcde2966f95c731a8046b504e5e640
Present_hash   :  00000ffa2924453f60408a10973baa99670fc77ba738e914e84208ddb44b73b7
--------------------------------------------------------------------------------------------------

172.26.16.43 - - [13/Sep/2018 19:50:16] "GET /get_block HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:16] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:16] "GET /chain HTTP/1.1" 200 -

Node5 has mined!

172.26.16.32 - - [13/Sep/2018 19:50:16] "GET /message HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:16] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:16] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------------------------
Index          :  28
Transactions   :
                     {'Sender': '0', 'Recipient': '6a06496cd61242ecae8051b649464361', 'Amount': 50}
Proof          :  10126
Num_zeros      :  5
Previous_hash  :  00000ffa2924453f60408a10973baa99670fc77ba738e914e84208ddb44b73b7
Present_hash   :  000003fe8af957e9fd0390a5369ee234c0b35ce17a8f895b9a6d8dbbd1bd59d0
--------------------------------------------------------------------------------------------------

172.26.16.32 - - [13/Sep/2018 19:50:16] "GET /get_block HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:50:17] "GET /chain HTTP/1.1" 200 -

Node2 has mined!

172.26.16.166 - - [13/Sep/2018 19:50:20] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:50:20] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:20] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------------------------
Index          :  29
Transactions   :
                     {'Sender': '0', 'Recipient': '0219783336594a3894a6c0b7dbe54eef', 'Amount': 50}
Proof          :  390320
Num_zeros      :  5
Previous_hash  :  000003fe8af957e9fd0390a5369ee234c0b35ce17a8f895b9a6d8dbbd1bd59d0
Present_hash   :  00000f6677b9e118e68e1c1c44790df1a7b0a2935f36df55a8584f31c2f057b1
--------------------------------------------------------------------------------------------------

172.26.16.166 - - [13/Sep/2018 19:50:21] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:21] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:21] "GET /chain HTTP/1.1" 200 -
```

Infonet blockchain

Num_zeros : 5

3 Six Node Blockchain Network

- The difficulty level was posted by node 101 and changed to 6 leading zeros.
  - Notice this at time 02:13.
  - Note that the 31st block, mined by Node, has 6 leading zeros.

GIST

3 Six Node Blockchain Network

```
Proof          :  390320
Num_zeros      :  5
Previous_hash  :  000003fe8af957e9fd0390a5369ee234c0b35ce17a8f895b9a6d8dbbd1bd59d0
Present_hash   :  00000f6677b9e118e68e1c1c44790df1a7b0a2935f36df55a8584f31c2f057b1
--------------------------------------------------------------------------------

172.26.16.166 - - [13/Sep/2018 19:50:21] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:21] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:21] "GET /chain HTTP/1.1" 200 -

Node2 has mined!

172.26.16.166 - - [13/Sep/2018 19:50:23] "GET /message HTTP/1.1" 200 -
203.237.54.101 - - [13/Sep/2018 19:50:23] "POST /difficulty/get HTTP/1.1" 201 -
172.26.16.32 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  30
Transactions   :
                 {'Sender': '0', 'Recipient': '02197833365594a3894a6c0b7dbe54eef', 'Amount': 50}
Proof          :  158121
Num_zeros      :  5
Previous_hash  :  00000f6677b9e118e68e1c1c44790df1a7b0a2935f36df55a8584f31c2f057b1
Present_hash   :  0000033df6c17049076a4c2d71e59813a4b24e830105ebacfc794d2c0f89fad2
--------------------------------------------------------------------------------

172.26.16.166 - - [13/Sep/2018 19:50:23] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:24] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -

Node4 has mined!

172.26.16.42 - - [13/Sep/2018 19:50:27] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index          :  31
Transactions   :
                 {'Sender': '0', 'Recipient': '8911a636c7904d92abcb87ef428743bc', 'Amount': 50}
Proof          :  347355
Num_zeros      :  6
Previous_hash  :  0000033df6c17049076a4c2d71e59813a4b24e830105ebacfc794d2c0f89fad2
Present_hash   :  000000446494187aeaf8aaf6d10650b508fc8f4229681cbad9d434b75dcc77d9
--------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:50:27] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:28] "GET /chain HTTP/1.1" 200 -
```

Infonet blockchain

Num_zeros : 6

Num_zeros is changed to 6.

GIST

3 Six Node Blockchain Network

- The 31st block was a luck and mined quick.
- But it takes avg. more than a min. now.
  - Since it takes longer to mine a block, many TXs are posted, but not included.

GIST

3 Six Node Blockchain Network

```
172.26.16.166 - - [13/Sep/2018 19:50:23] "GET /message HTTP/1.1" 200 -
203.237.54.101 - - [13/Sep/2018 19:50:23] "POST /difficulty/get HTTP/1.1" 201 -
172.26.16.32 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index         :  30
Transactions  :
                {'Sender': '0', 'Recipient': '0219783336594a3894a6c0b7dbe54eef', 'Amount': 50}
Proof         :  158121
Num_zeros     :  5
Previous_hash :  00000f6677b9e118e68e1c1c44790df1a7b0a2935f36df55a8584f31c2f057b1
Present_hash  :  0000033df6c17049076a4c2d71e59813a4b24e830105ebacfc794d2c0f89fad2
--------------------------------------------------------------------------------

172.26.16.166 - - [13/Sep/2018 19:50:23] "GET /get_block HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:23] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:24] "GET /chain HTTP/1.1" 200 -
172.26.16.166 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -

Node4 has mined!

172.26.16.42 - - [13/Sep/2018 19:50:27] "GET /message HTTP/1.1" 200 -
172.26.16.32 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -
172.26.16.43 - - [13/Sep/2018 19:50:27] "GET /chain HTTP/1.1" 200 -

--------------------------------------------------------------------------------
Index         :  31
Transactions  :
                {'Sender': '0', 'Recipient': '8911a636c7904d92abcb87ef428743bc', 'Amount': 50}
Proof         :  347355
Num_zeros     :  6
Previous_hash :  0000033df6c17049076a4c2d71e59813a4b24e830105ebacfc794d2c0f89fad2
Present_hash  :  000000446494187aeaf8aaf6d10650b508fc8f4229681cbad9d434b75dcc77d9
--------------------------------------------------------------------------------

172.26.16.42 - - [13/Sep/2018 19:50:27] "GET /get_block HTTP/1.1" 200 -
172.26.16.42 - - [13/Sep/2018 19:50:28] "GET /chain HTTP/1.1" 200 -
203.237.54.101 - - [13/Sep/2018 19:50:30] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:50:34] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:50:43] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:50:48] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:50:52] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:50:57] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:01] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:04] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:08] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:13] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:18] "POST /transactions/get HTTP/1.1" 201 -
203.237.54.101 - - [13/Sep/2018 19:51:23] "POST /transactions/get HTTP/1.1" 201 -
```

Infonet blockchain

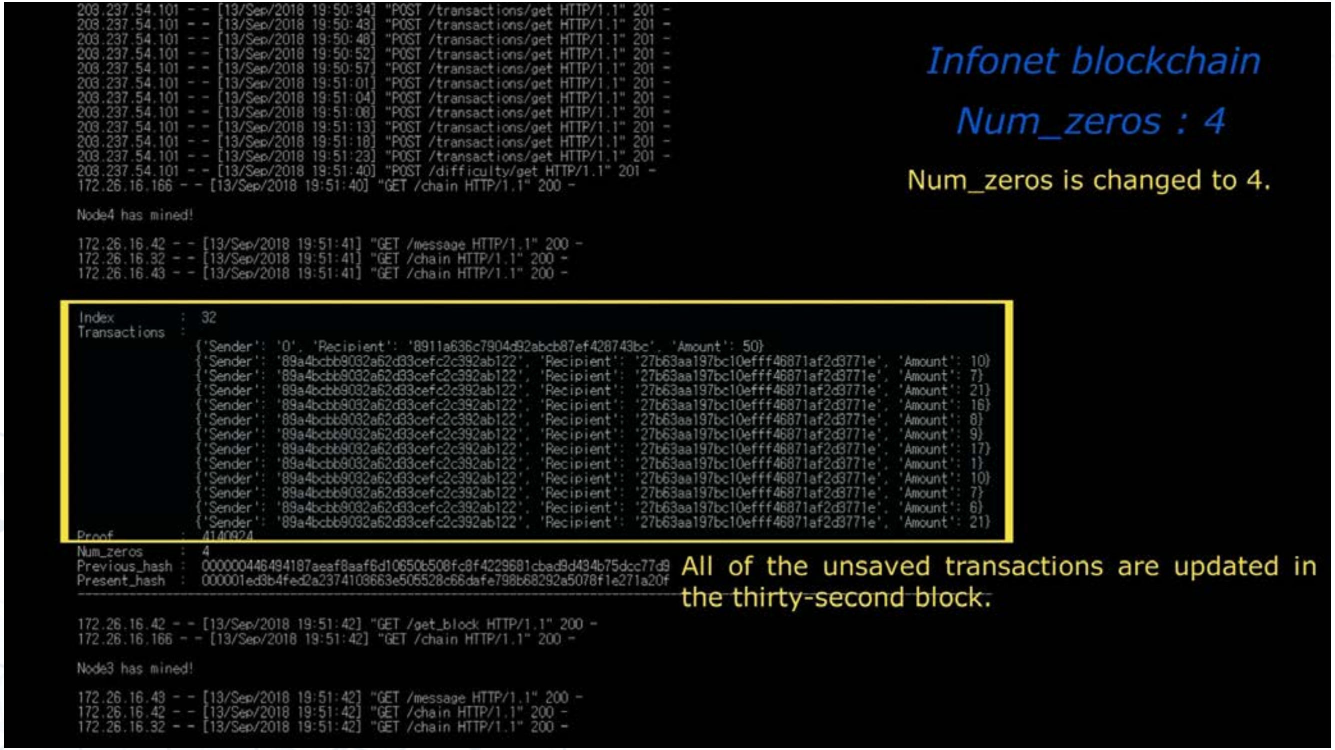
Num_zeros : 6

3 Six Node Blockchain Network

- On 03:28, node101 posts a difficulty change message and changes it to 4 leading zeros.
- The 32nd block contains all the awaiting transactions.

GIST

3 Six Node Blockchain Network

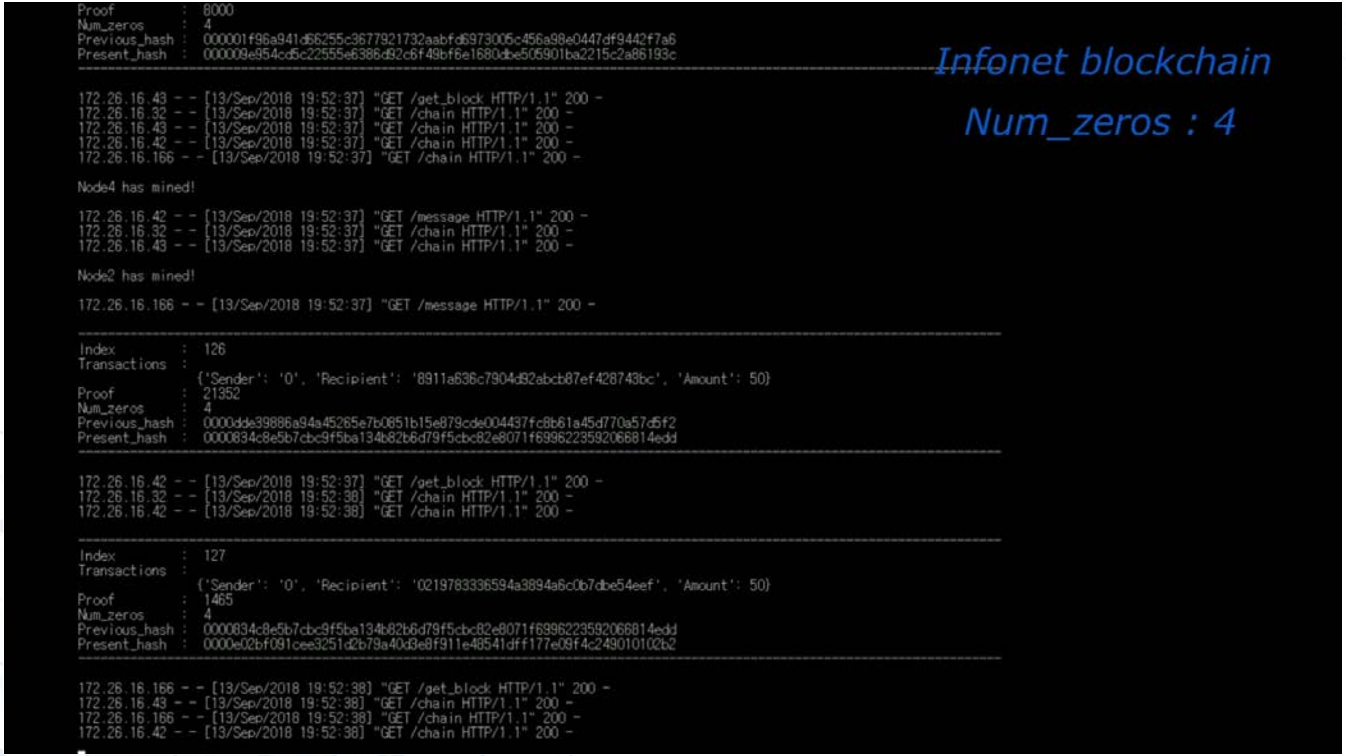


GIST

3 Six Node Blockchain Network

- After that, blocks are mined very quick.
  - About 100 blocks mined for 1 minute, we see.

3 Six Node Blockchain Network



GIST

3 Six Node Blockchain Network

- On the time of 04:36, right after 130th block was mined, the difficulty level is changed back to 5 leading hex zeros.
  - Note the difficulty change Post by Node 101.
- This continues till the end.

GIST

3 Six Node Blockchain Network

- Lessons
  - Python and Flask can be used to program P2P blockchain suite.
  - P2P computers exchange messages, blocks, and commands.
  - They grow the blockchain ledger.
  - Block generation speed can be set fast or slow by changing difficulty.

Infonet blockchain

Num_zeros : 5

```
(E:₩Anaconda3) C:₩Users₩양기원>cd C:₩Users₩양기원₩Desktop₩Bitcoin₩Building Blockchain by Python₩blockchain-master₩blockchain-master

(E:₩Anaconda3) C:₩Users₩양기원₩Desktop₩Bitcoin₩Building Blockchain by Python₩blockchain-master₩blockchain-master>python info_blockchain1.py
---------------------------------------
 * Running on http://172.26.16.41:5000/ (Press CTRL+C to quit)
172.26.16.41 - - [13/Sep/2018 19:47:20] "POST /nodes/register HTTP/1.1" 201 -
172.26.16.41 - - [13/Sep/2018 19:48:30] "GET /mine HTTP/1.1" 200 -
```

Goal of this lecture note

- Bitcoin and Ethereum
- Problems of PoW
- Trilemma vs. DeSecure Strategy
- DeSecure Blockchains
- ECCPoW
- Open Source DeSecure Project
- Impact of DeSecure Blockchains

1 Bitcoin and Ethereum

- Bitcoin's Ideals
  - BTC is the first global digital currency of people which works beyond national boundaries.
  - Ideals around BTC are
    - Decentralization
    - Reforming Wall street
    - Unbundling big corporations
    - Reduction of inequality

1 Bitcoin and Ethereum

- Ethereum's Ideals
  - ETH is a world decentralized computing platform.
  - Programming smart contracts is easier.
  - One can make DApps.
  - One can create tokens in 20 minutes.
  - People can make Decentralized Autonomous Organizations (DAO).

2 Problems of PoW

PoW is fundamental.
But there are problems.
Let us fix its problems and use it.

GIST

2 Problems of PoW

- Complaints today
  - PoW based blockchains are most secure;

    But they are …

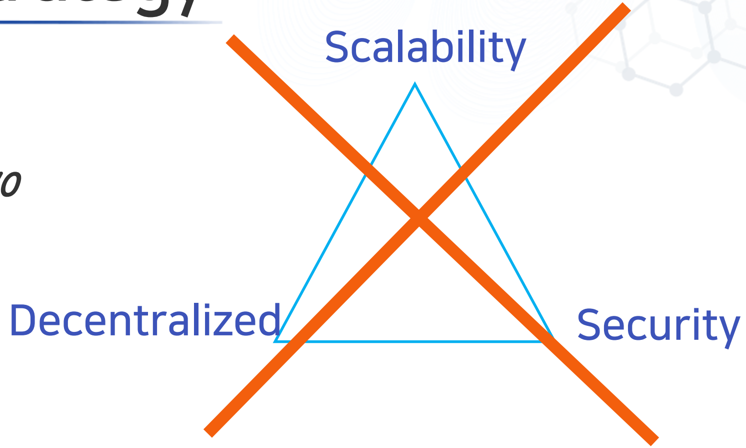
  - Spending too much energy in mining
  - Re-centralized
  - Said to be too slow, not supporting speedy transactions

3 Trilemma vs. DeSecure Strategy

- Blockchain Trilemma?

"blockchain systems can only at most have two of the following three properties"

- Vitalik Buterin

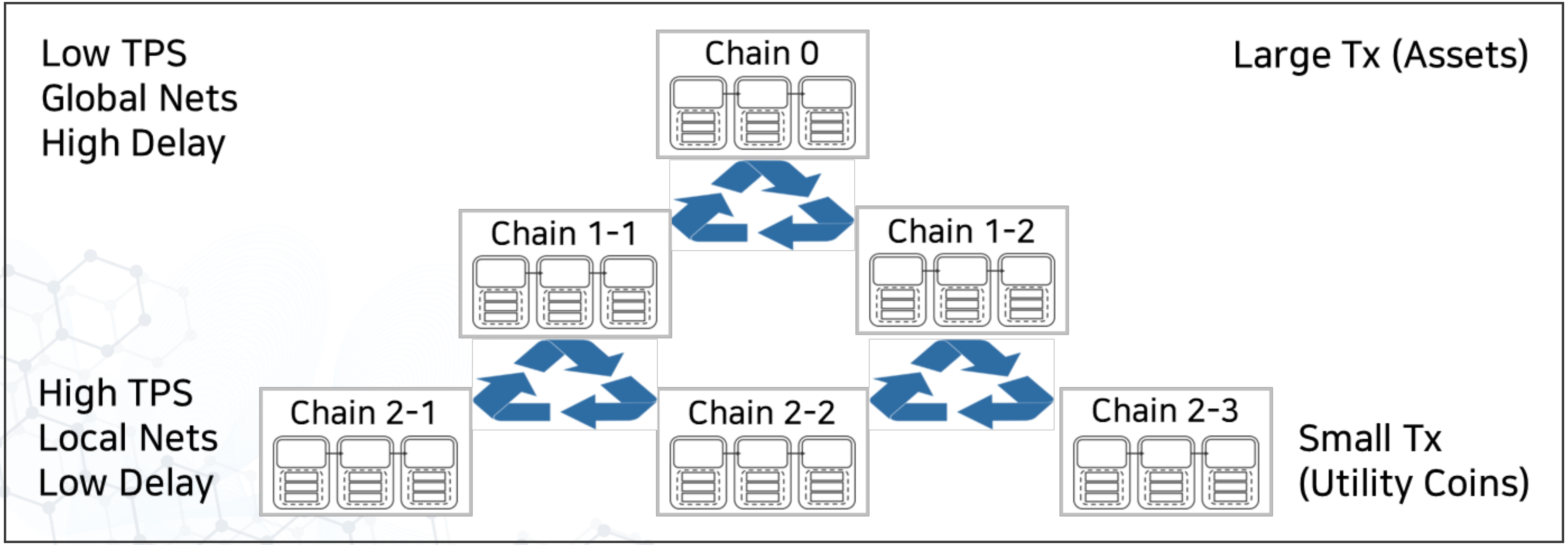


Wrong approach!
- Not in a single blockchain, can it be achieved!
- We shall promote many decentralized secure (DeSecure) blockchains to achieve scalability!

출처: https://github.com/ethereum/wiki/wiki/Sharding-FAQ

3 Trilemma vs. DeSecure Strategy

- Provision of DeSecure chains is to solve Scalability issue!
  - Global chains → national chains → local chains → diff. applications



4 DeSecure Blockchains

- We aim to approach these two issues with DeSecure blockchains.
  - Anti-ASIC ECC PoW
  - Ecosystem of DeSecure blockchains
- DeSecure blockchains use novel Error-Correction Code PoW.
- We aim to provide two DeSecure blockchains, ETH-ECC and BTC-ECC.

4 DeSecure Blockchains

- They Have Sought Alternatives to PoW, BUT

	Pros	Cons	Coins within top 50 rank
PoW (Proof-of-Work)	• Strong security - Difficult to produce - Easy to verify	• Extreme computing power • 51% attacks • Transaction speed / Transaction throughput	Bitcoin, Ethereum
PoS (Proof-of-Stake)	• Energy & hardware efficiency • Much more expensive 51% attacks	• Recentralization • The rich-get-richer • “Noting at stake” problem	Qtum, Stratis

4 DeSecure Blockchains

- They Have Sought Alternatives to PoW, BUT

	Pros	Cons	Coins within top 50 rank
DPoS (Delegated PoS)	• Scalability and speed • Energy & hardware efficiency • Encouraging good behavior by realtime voting	• Recentralization • DDoS attacks • Double Spending	EOS, NEO
PoA (Proof-of-Activity)	• Decentralization - Validators are randomly selected	• Computing power • Recentralization • The rich-get-richer	decred

4 DeSecure Blockchains

- Existing Scalability Solutions
  - *DeSecure Blockchain aims to resolve the re-centralization problem without sacrificing the decentralization and secureness!*

Type	DeSecure	Bitcoin		Ethereum	
Name	Multi-level, multiple chains	Seg-Wit	Lightening Network	Plasma	Sharding
구현	ECCPoW 기반 독립체인들을 여러 계층으로 묶음	블록 데이터 구조를 변경하여 구현	오프체인 거래 진행 최종 결과값을 메인 블록체인에 기록	하부 체인 생성 거래 진행 후 최소한의 기록만 메인 블록체인 기록	블록체인의 DB에 해당하는 스테이트를 여러 샤드로 분할, 분리 처리

4 DeSecure Blockchains

- Existing Scalability Solutions
  - *DeSecure Blockchain aims to resolve the re-centralization problem without sacrificing the secureness!*

Type	DeSecure	Bitcoin		Ethereum	
Name	Multi-level, multiple chains	Seg-Wit	Lightening Network	Plasma	Sharding
장점	서로 다른 블록체인 연결해 다양한 기능과 역할 구현	쉽게 구현이 가능함	결제 속도 제고 즉각적인 완결성 수수료 절감	수수료 절감	트랜잭션 처리 속도 증가

GIST

4 DeSecure Blockchains

- Existing Scalability Solutions
  - *DeSecure Blockchain aims to resolve the re-centralization problem without sacrificing the secureness!*

Type	DeSecure	Bitcoin		Ethereum	
Name	Multi-level, multiple chains	Seg-Wit	Lightening Network	Plasma	Sharding
단점	No single chain solution/ 생태계필요	트랜잭션 처리속도 증가 효과 미비	오프체인 거래기록 없음	Full노드 만 플라즈마 사용 가능	S/W 복잡도 상승

5 ECCPoW

- We aim to Replacing SHA-PoW with ECC-PoW!

Blockchain Core Program

Three key parts

1. Web server interface n
- Node registration, get
- Full node or light node
- Communication amon

2. Wallet for TX generati
- Make private and publ neighbor, check to see

3. Consensus Mechanism
- Data: Genesis block +
- Protocol: consensus, l
- Mining: Get the longes mempool and form a l block header, and atta

Program Suite
- C++, Python, Go, Java, Fi
- Download and run, then you ha

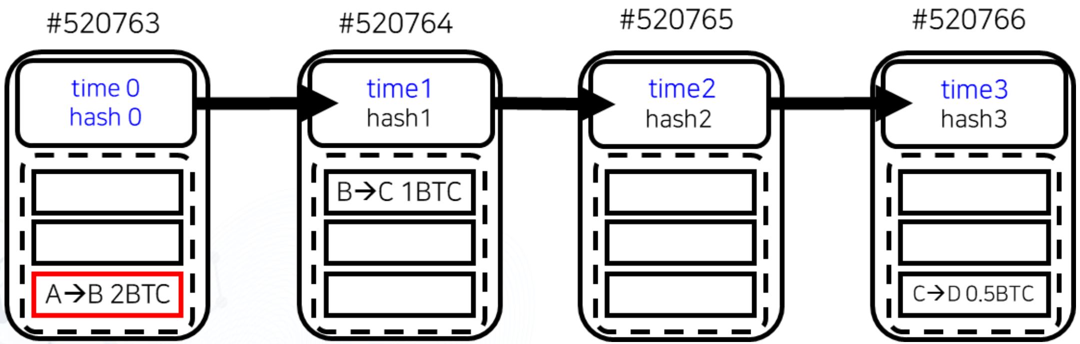
Consensus Engine

3. Consensus Mechanism

- Data: Genesis block + regular blocks, one block every 10 min, block-size 1Mbyte
- Protocol: consensus, block header, difficulty level adjustment, …
- Mining: Get the longest chain, validate it and all transactions within it, get transactions from mempool and form a block, run SHA repeatedly until you hit a good hash, put the proof into the block header, and attach the proofed block to the longest chain, and make announcement ASAP.

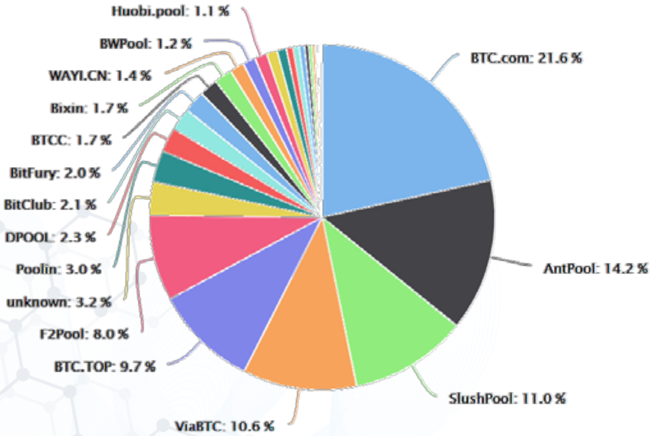
5 ECCPoW

- Pow is fundamental to OPEN blockchains.
  - What happens when any alteration is made?



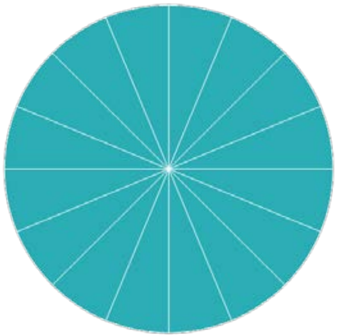
5 ECCPoW

- ECC-PoW aims to resolve Recentralization Issue.
  - ASIC → Mining Moguls → Discourage Average Miners
  - Prone to Collusion, Censorship



Recentralized

Decentralized again



1. ASIC resistant
2. Vulnerability to DS attacks reduced

- Item to consider a new PoW!
  - A new puzzle generation system is capable of varying puzzles from block to block with the following properties:
    - P1: Easy to verify but difficult to prove
    - P2: Robust to detect block modification attacks
    - P3: Controllable in changing the difficulty level
    - P4: Open to anyone with a CPU
    - P5: Unfixed and changeable from block to block
  - The re-centralized problem can be resolved thanks to P5.

- Novel Error Correction Codes PoW (ECCPoW)
  - There are many one-way functions in inverse problems
    - Error Correction Codes
    - Sparse-Signal Recovery
    - Space-Time Coding
    - Sphere-Decoding
  - In these problems, encoding is easy but decoding is controllablly time-consuming!

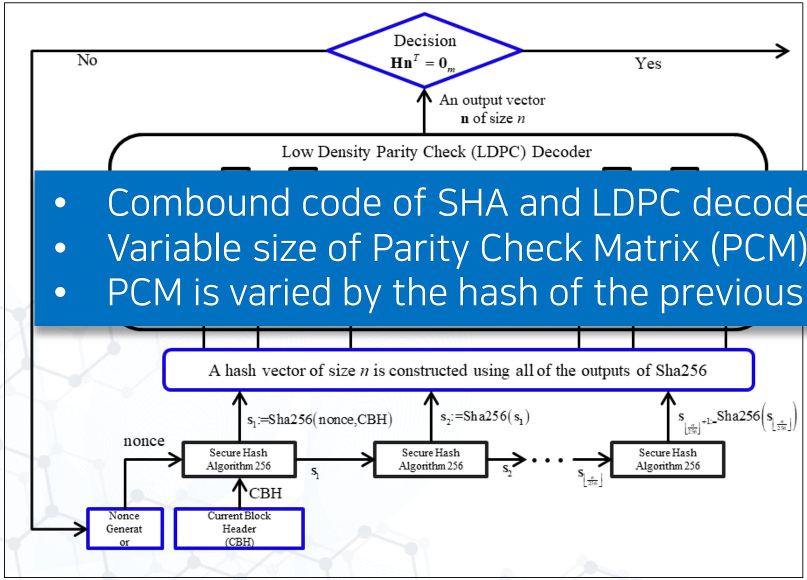
5 ECCPoW

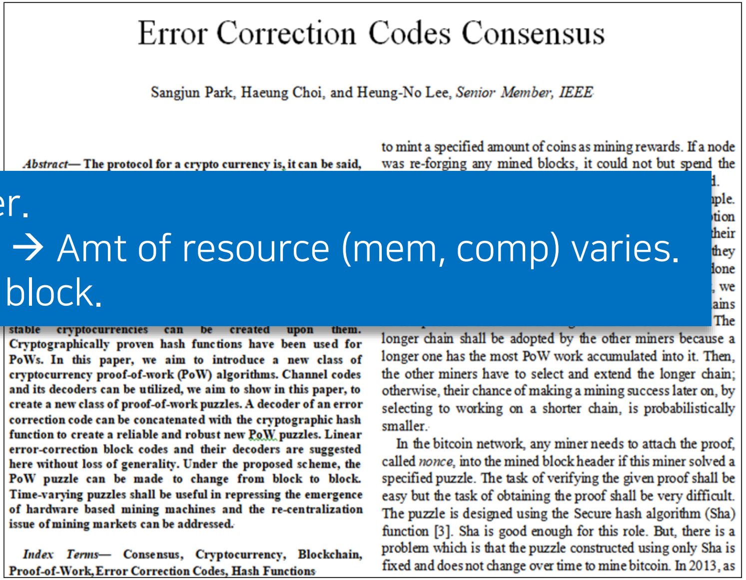
- Novel Error Correction Codes PoW (ECCPoW)
  - We combine a Error Correcting Code framework with SHA-xxx.

  - The decision of mining success is made with the output of the above decoder.

5 ECCPoW

- Novel ECCPoW Consensus is proposed!
  - ECCPoW 합의 엔진





Error Correction Codes Consensus

Sangjun Park, Haeung Choi, and Heung-No Lee, *Senior Member, IEEE*

***Abstract*— The protocol for a crypto currency is, it can be said,**

stable cryptocurrencies can be created upon them. Cryptographically proven hash functions have been used for PoWs. In this paper, we aim to introduce a new class of cryptocurrency proof-of-work (PoW) algorithms. Channel codes and its decoders can be utilized, we aim to show in this paper, to create a new class of proof-of-work puzzles. A decoder of an error correction code can be concatenated with the cryptographic hash function to create a reliable and robust new PoW puzzles. Linear error-correction block codes and their decoders are suggested here without loss of generality. Under the proposed scheme, the PoW puzzle can be made to change from block to block. Time-varying puzzles shall be useful in repressing the emergence of hardware based mining machines and the re-centralization issue of mining markets can be addressed.

***Index Terms*— Consensus, Cryptocurrency, Blockchain, Proof-of-Work, Error Correction Codes, Hash Functions**

to mint a specified amount of coins as mining rewards. If a node was re-forging any mined blocks, it could not but spend the

The longer chain shall be adopted by the other miners because a longer one has the most PoW work accumulated into it. Then, the other miners have to select and extend the longer chain; otherwise, their chance of making a mining success later on, by selecting to working on a shorter chain, is probabilistically smaller.

In the bitcoin network, any miner needs to attach the proof, called *nonce*, into the mined block header if this miner solved a specified puzzle. The task of verifying the given proof shall be easy but the task of obtaining the proof shall be very difficult. The puzzle is designed using the Secure hash algorithm (Sha) function [3]. Sha is good enough for this role. But, there is a problem which is that the puzzle constructed using only Sha is fixed and does not change over time to mine bitcoin. In 2013, as

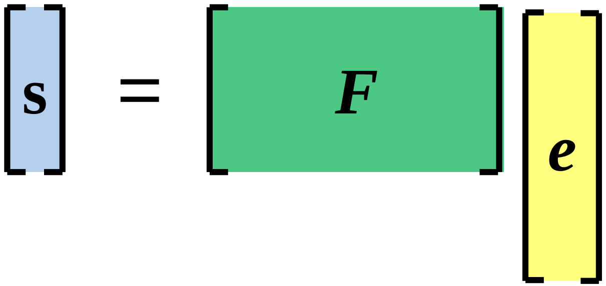
- Combound code of SHA and LDPC decoder.
- Variable size of Parity Check Matrix (PCM) → Amt of resource (mem, comp) varies.
- PCM is varied by the hash of the previous block.

5 ECCPoW

- Error Correction Code
  - Transmitter and receiver uses a codebook.
  - In a codebook, there are codewords.
  - Transmitter sends a message.
  - Message goes through channel.
  - Errors are induced.
  - Receiver gets the erroneous message.
  - Decoder aims to find a nearest codeword.

- Decoder uses memory and computer to run and find a codeword.

- Block code, encoder and decoder



$S \in GF(q)^{M\times 1}$

$F \in GF(q)^{M\times N}$

$e \in GF(q)^{N\times 1}$

Encoder: Given e, find $s = Enc(e, G)$

Decoder: Given s, find $\hat{e} = Enc(s, F)$

5 ECCPoW

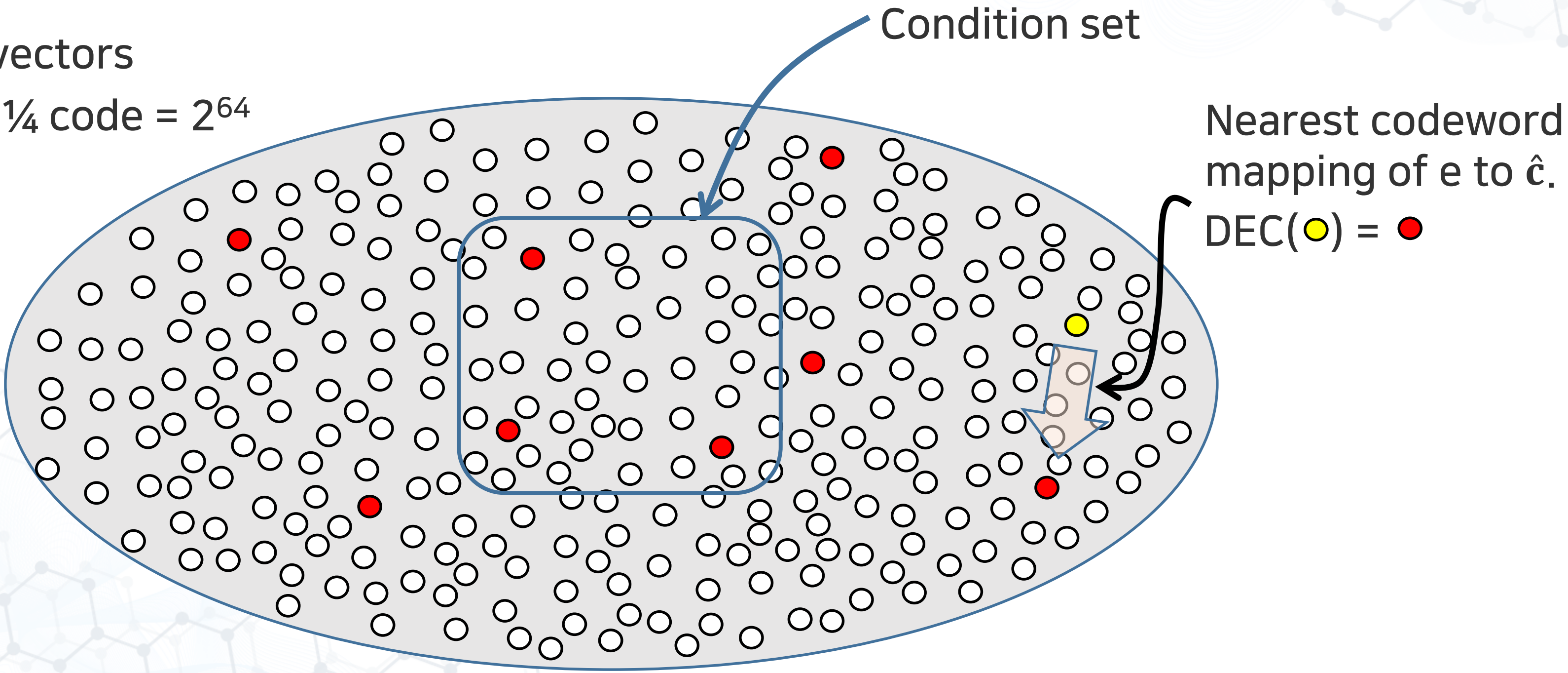
- Decoder
  - SHA output is input to the decoder.
  - Decoder treats it as erroneous message and produces either a codeword or non-codeword.
  - We use the low-density parity-check (LDPC) code and its message passing decoder.
  - We change the matrix F to change the puzzle.

5 ECCPoW

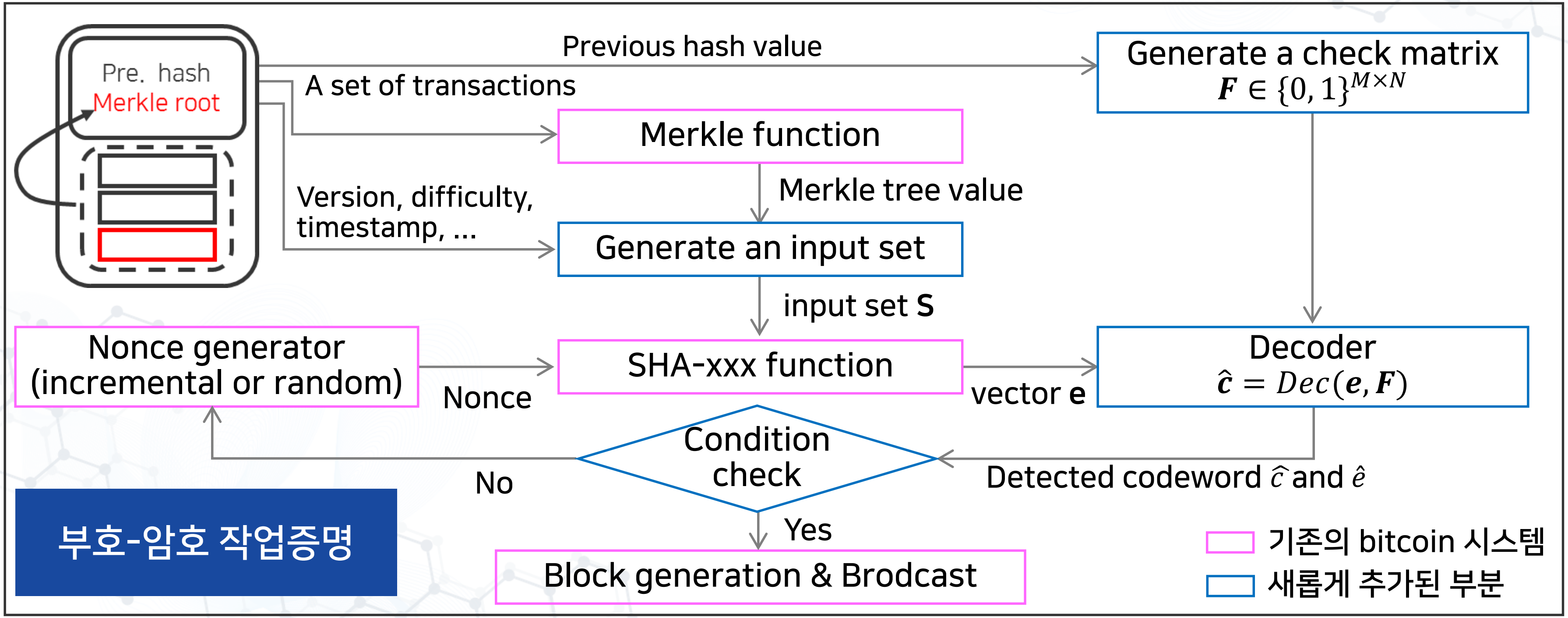
- Geometrical Explanation

○ 2^{256} vectors

● Rate ¼ code = 2^{64}

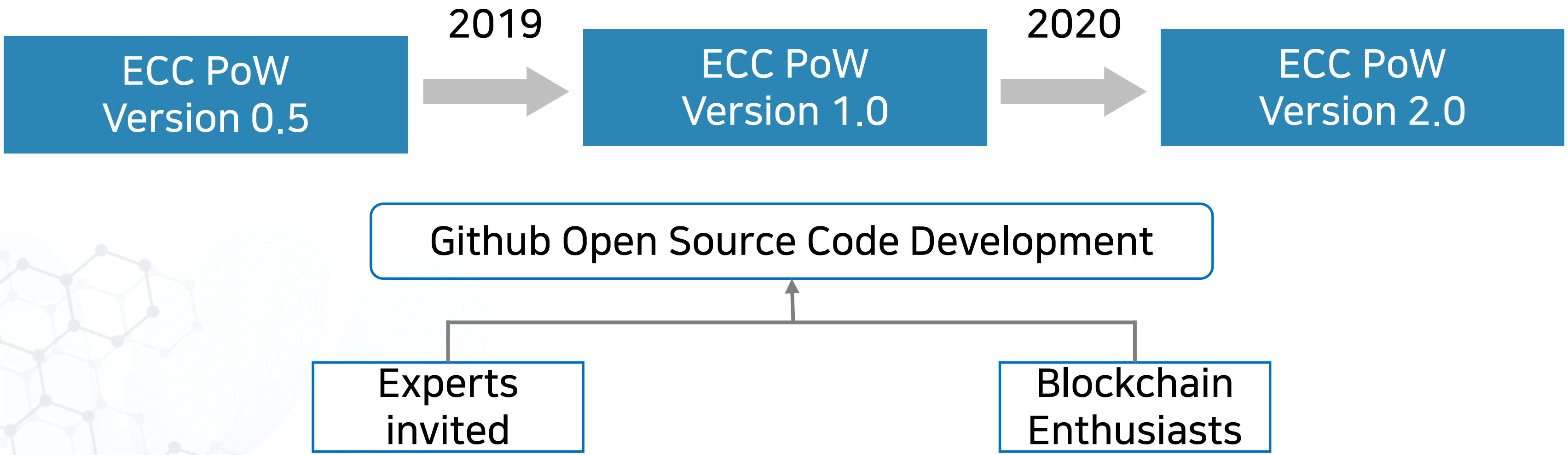


- Diagram of ECCPoW



6 Open Source DeSecure Project

- DeSecure Blockchain Release Plan



GIST

6 Open Source DeSecure Project

- DeSecure is Open Project.
  - 공식홈페이지: https://desecure.org/
  - BTC-ECC explorer: http://13.209.97.152/blocks
  - BTC-ECC Github: https://github.com/cryptoecc/bitcoin_ECC
  - ETH-ECC: https://github.com/cryptoecc/go-ethereum_ECC/tree/eccpow-1.9
  - DeSecure Blockchain 관련 논문
    https://infonet.gist.ac.kr/?page_id=6832

7 Impact of DeSecure Blockchains

- Impact on Safe Start
  It is easier to start a new blockchain network.
  - Today there are mining equipment renting sites.
  - A new borne blockchain network needs to grow, but a newbie is much more vulnerable to 51% attacks.
  - New blockchain networks with ECCPoW do not suffer from such problems since there are no mining equipment available for ECCPoW.

GIST

7 Impact of DeSecure Blockchains

- Impact on Standardization
  One can make multiple blockchain networks
  - Make the first blockchain network by running ETH-ECC over a network (Pusan ETH)
  - Make the second blockchain network by running BIT-ECC over other network (Gwangju BIT)
  - Make the third blockchain network by running ETH-ECC over another network (Seoul ETH)
  - Make the fourth blockchain network by running BIT-ECC over yet another network (Global BIT)

7 Impact of DeSecure Blockchains

- Impact on Standardization
  One can make multiple blockchain networks
  - Each cryptocurrency is independent with its own genesis block and random starting seed and can be adjusted sufficiently strong for its regional requirement in the sense of scalability, security and decentralization.
  - These blockchains are inter-connected at the local, regional, and national, transnational level.

GIST

7 Impact of DeSecure Blockchains

- Impact on Resolving the Scalability Trilemma
  - Each DeSecure blockchain is already very strong in decentralization.
  - Each DS blockchain is flexible enough to provide various settings of transaction speeds and security levels.
    - Regional DeSecure networks can be set to work very fast, i.e. allowing up to 10s of thousands of TXs per sec.
    - National DeSecure networks can be set sufficiently fast for covering inter-regional transactions.
    - Transnational DeSecure networks shall be set to work slow due to large delays.

GIST

7 Impact of DeSecure Blockchains

- Impact on Resolving the Scalability Trilemma
  - All these blockchains started up with its own seed and decentralized levels are mutually independent and each one can be set to work at the required level of security and speed to serve its purpose.
  - All these DeSecure blockchains can be inter-connected via distributed value-exchange networks.

7 Impact of DeSecure Blockchains

- Impact : New PoW

PoW is problem. Yes.
But it is not the inherent to PoW.
It is the fixedness and simplicity of the PoW puzzle.
ECCPoW is time-varying and grow very complex.

GIST

7 Impact of DeSecure Blockchains

- Impact on Deterrence to ASICs:
  The complexity of ECCPoW puzzles can be set to grow very large.
  - ECCPoW is a computer algorithm!
  - Thus it is not impossible to find a hardware acceleration solution for it.
  - But it comes with boundless cost to memory and computing resource.

7 Impact of DeSecure Blockchains

- Impact on Energy Spending:
  - Deterrence to hardware acceleration offers a blockchain network with small hash rate requirement.
  - Ordinary people can join.
  - One-cpu one-vote possible again