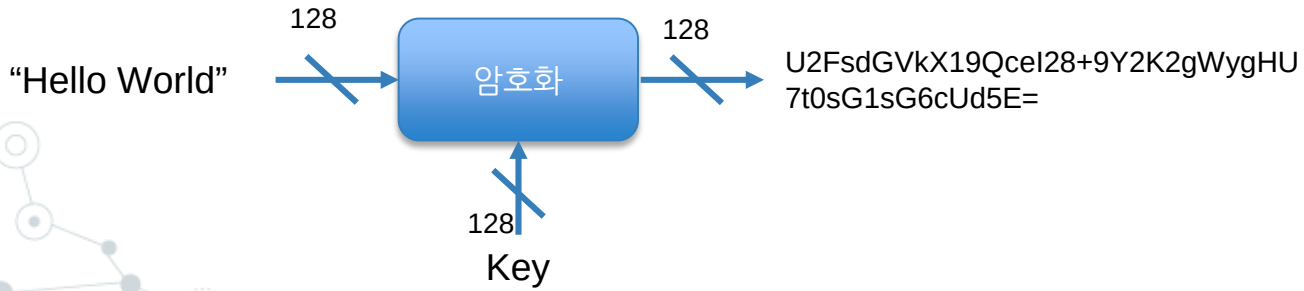
암호기술의 이해

해시 알고리즘

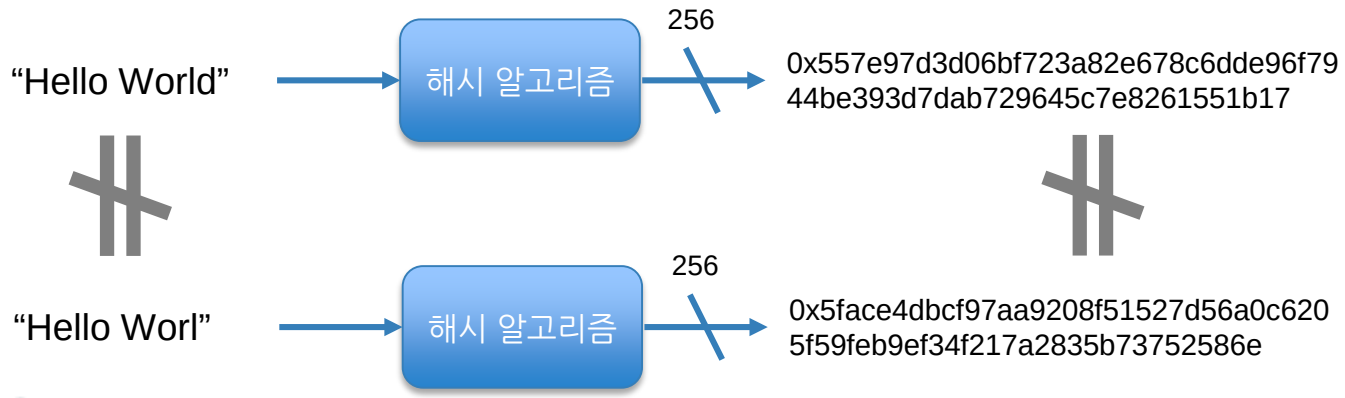
암호란?

- 다른 공격자에게 메시지의 기밀성(confidentiality)를 유지하기 위한 기술
- 암호 알고리즘의 종류
  - 대칭키 암호 알고리즘 (Symmetric Cipher, Block Cipher, etc.)
    - 암호화를 위한 키와 복호화를 위한 키가 동일한 암호 알고리즘
    - AES, SEED, ARIA, DES, TDES, etc..
  - 비대칭키 암호 알고리즘 (Asymmetric Cipher, Public Key Encryption, etc)
    - 암호화와 복호화의 키가 다른 암호 암호 알고리즘
    - RSA, ECC, etc..
  - 해시 알고리즘 (SHA-1/2/3, MD5)
    - 메시지를 일정 길이(20바이트, 32바이트, etc)로 축약하는 암호 알고리즘



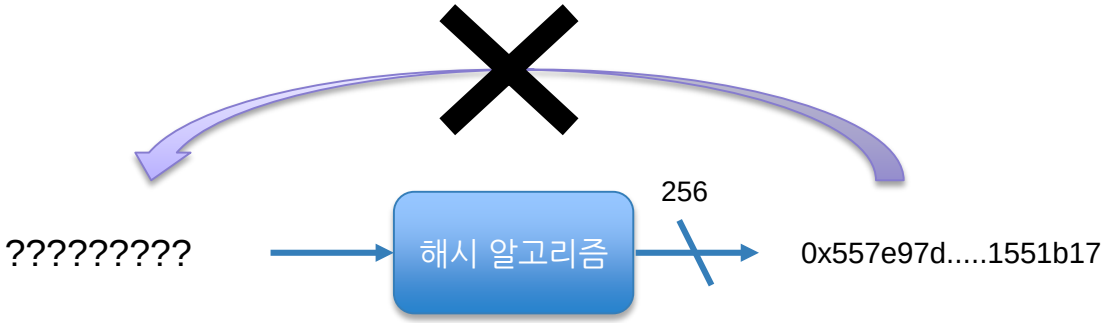
해시 알고리즘 (SHA)

- SHA : Secure Hash Algorithm
  - 미국 NIST(표준연구소)에서 표준화한 해시 알고리즘 명칭
  - 단방향 함수(One-way Function)
  - 현재 SHA-1, SHA-2. SHA-3까지 있음
    - 2008년 10월에 SHA-3 Competition 알고리즘 제출 마감
    - 2012년 10월에 SHA-3 Winner 알고리즘으로 Keccak 으로 선발(이더리움에서 사용)
    - 2015년 8월 Padding의 일부분 변경하여 SHA-3 표준화 완료(FIPS-202)
    - 이더리움에서의 해시 함수, Keccak과 SHA3는 다름(NIST에서 일부 수정)

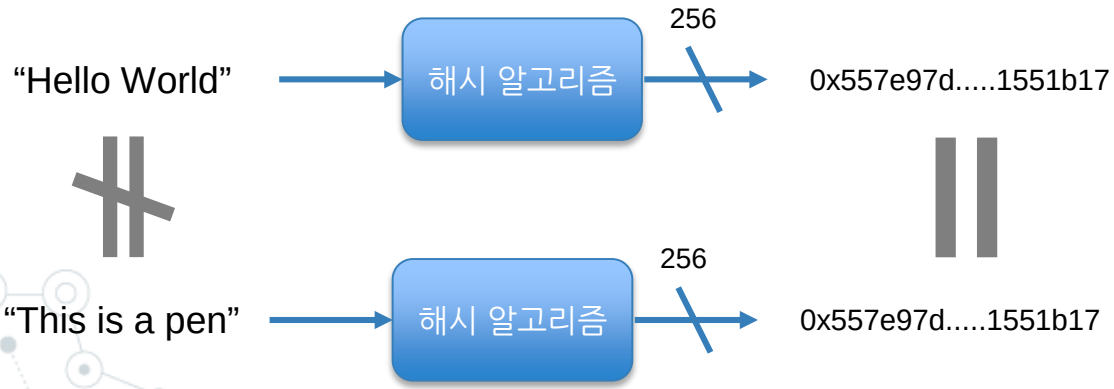


해시 알고리즘 (SHA)

- 단방향 함수(preimage resistance)
  - 역으로 원래 메시지 입력값을 찾는 것은 불가능
  - 역상이 존재하지 않음 (출력=해시값은 랜덤)



- 동일한 해시값을 가지는 두개의 다른 입력값 x (collision resistance)



해시 알고리즘 (SHA)

◎ 해시값이 일정 조건을 만족하도록 입력값을 찾는 것은 가능함

◎ 해시값이 0xFFFF....FF(최대값) 이하일 경우의 입력값 찾기??
- 모든 입력값이 해당

◎ 해시값이 0x7FFF...FF(최대값/2) 이하일 경우의 입력값 찾기??
- 약 50%의 확률로 입력값 찾는 것이 가능함

마이닝(Mining)과정은 아주 어려운 문제를 푼다라고 함

해시 알고리즘 (SHA)

- 주사위 2개를 던졌을 때에 합이 12이하가 나오는 경우 찾기 시간(던지는 횟수) : 하
  - 모든 경우에 해당

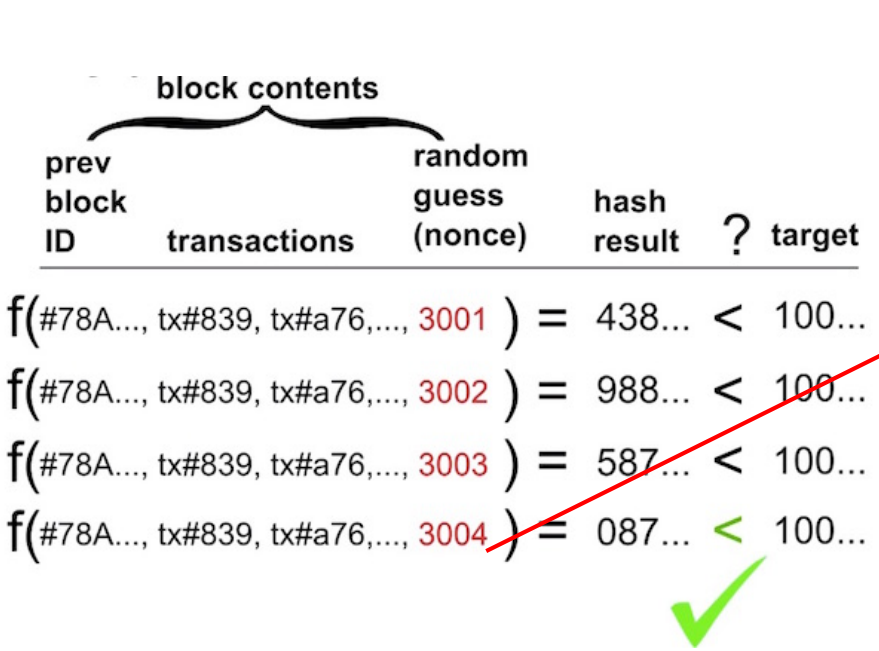
- 주사위 2개의 합이 3이하가 나오는 경우 찾기 시간(던지는 횟수) : 상
  - (1,1), (2,1), (1,2)로 3/36=1/12=8.3%의 확률

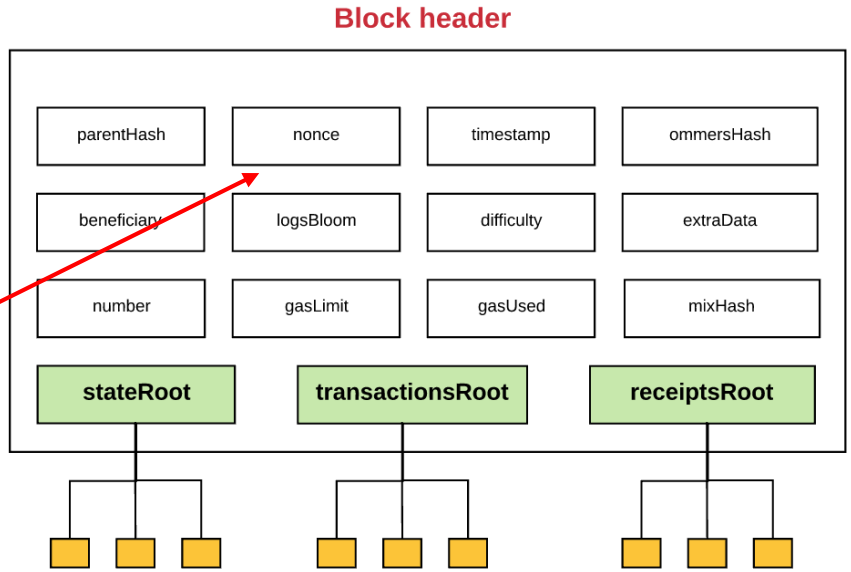
- 주사위 2개의 합이 5이하가 나오는 경우 시간(던지는 횟수) : 중
  - (1,1),(1,2),(1,3),(1,4), (2,1),(2,2),(2,3), (3,1),(3,2), (4,1)로 10/36=5/18=27.7%

난이도(Difficulty)는 조건(3이하, 5이하 등)에 해당
-> 난이도에 따라 걸리는 시간이 다름

마이닝(Mining)

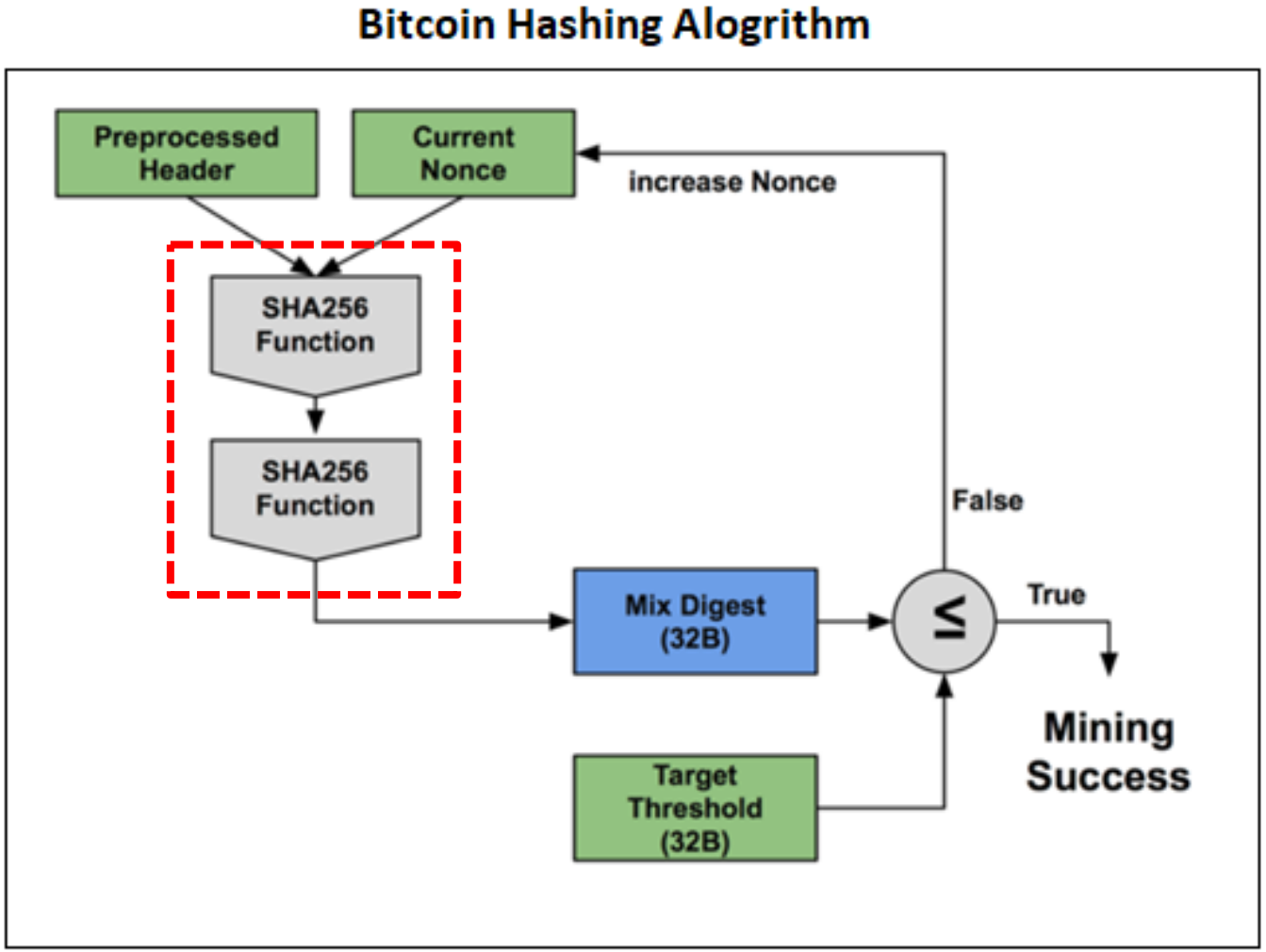
- PoW에서 블록을 생성하기 위한 Nonce값을 구하는 과정





마이닝(Mining)

◎ 비트코인 해시 함수(마이닝)



전용 마이닝 H/W

https://www.amazon.co.jp/Antminer-4-73TH-Bitcoin-Miner-AntMiner/dp/B014OGCP6W

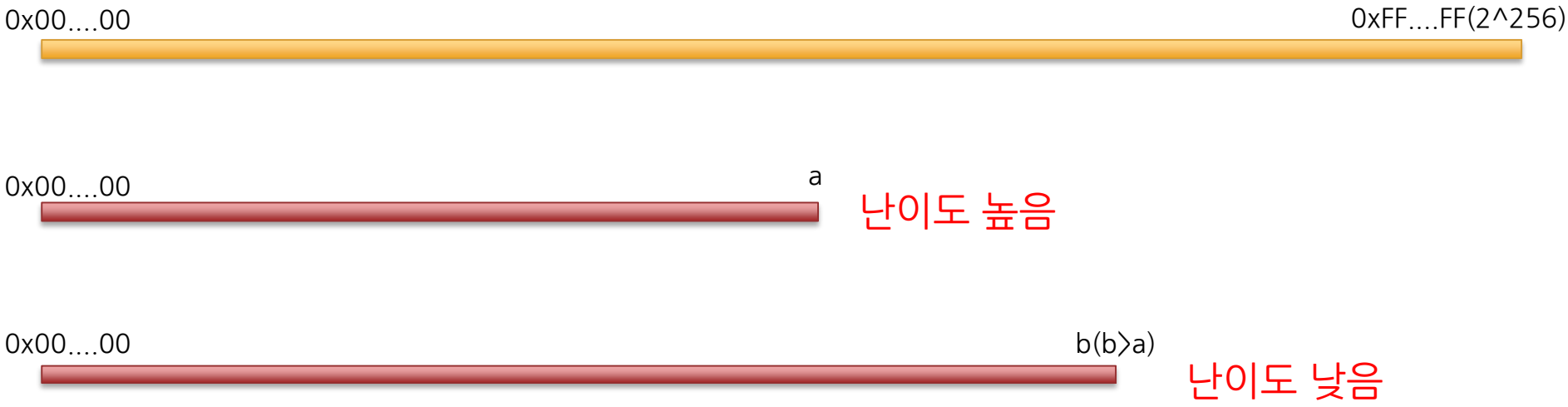
약250만원

마이닝(Mining)

◎ 이더리움 해시 함수(마이닝)

마이닝(Mining)

- 채굴기 : Nonce값을 고속으로 구하기 위한 장비
- 연산 속도(whattomine.com)
  - GPU 1070 6대 = 180Mh/s (4.28$/24h)



마이닝(Mining)

◎ 마이닝풀 : 단독으로 nonce값을 찾기 어려워 여러 채굴기를 하나로 묶어서 공동으로 채굴하는 방식

0x00....00 ~ 0xFF....FF(2^256)

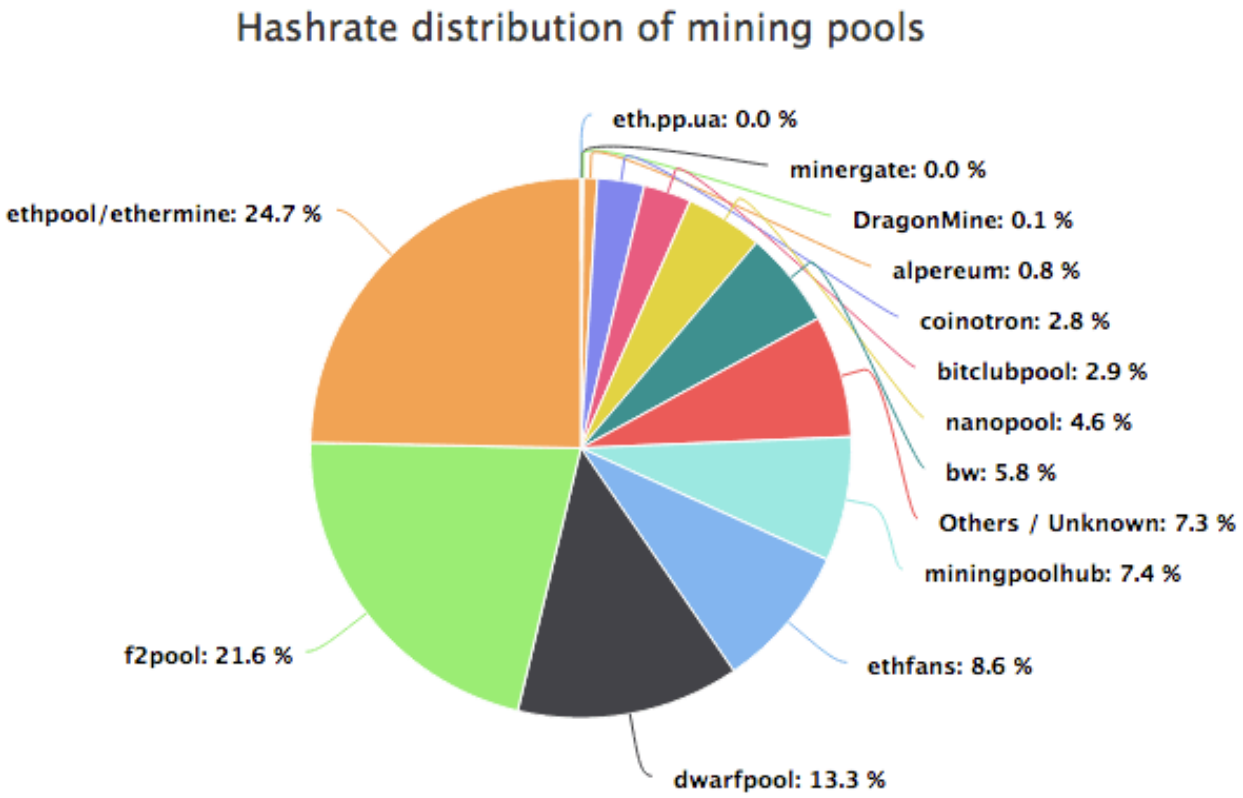
0x00....00 ~ a : 난이도 높음(이더리움 난이도)

0x00....00 ~ b(b>a) : 난이도 낮음

낮은 난이도의 해답을 찾다 보면(accepted share로 표현)
높은 난이도(이더이룸 난이도)를 찾게됨

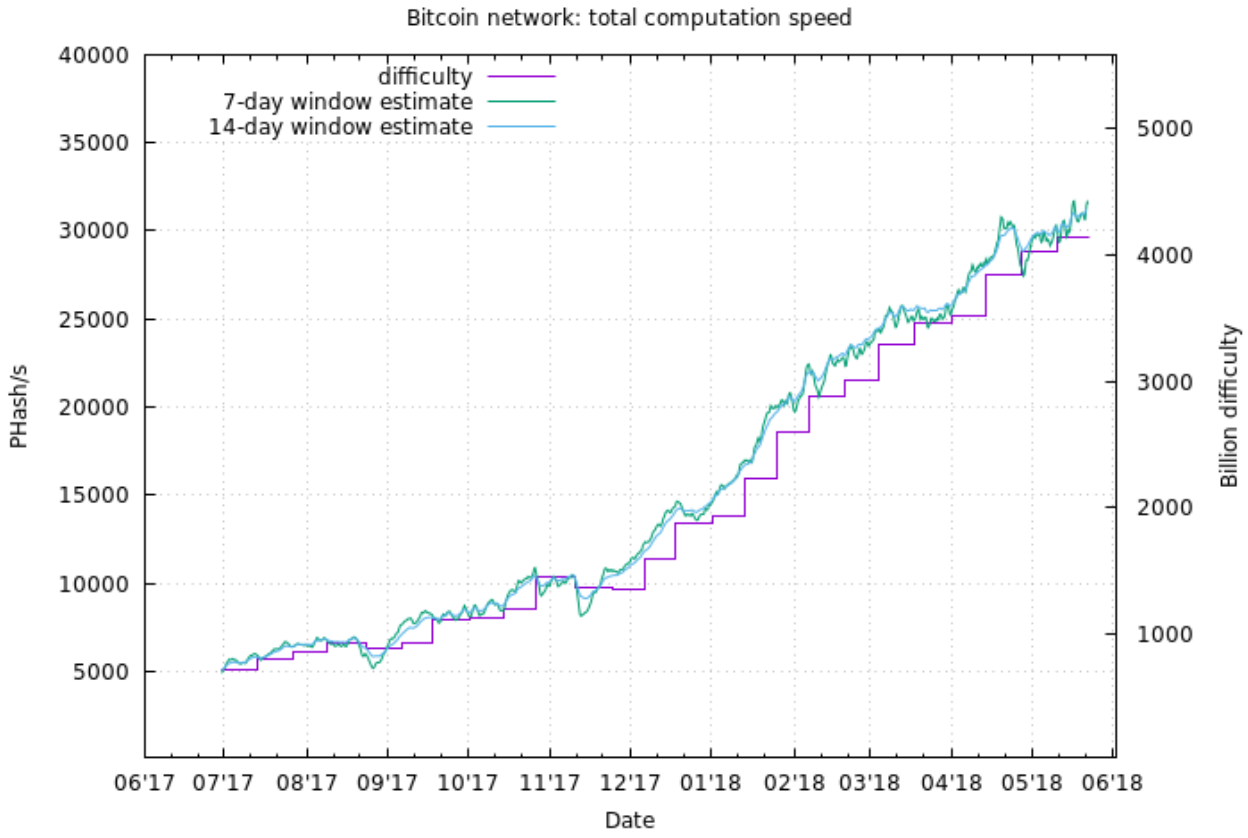
마이닝(Mining)

- 마이닝풀 운영자의 Centralized 방식
  - 어떠한 tx를 포함하여 block을 만들어 nonce를 찾을지를 결정



난이도(Difficulty)

- 난이도를 조정하여 블록을 생성하는(Nonce를 찾기까지 걸리는 시간)을 조정 가능함
  - 비트코인 : 10분, 이더리움 : 15초...



난이도(Difficulty)

- 이더리움의 경우 3가지 버전에 따라 난이도 계산
  - Fontier -> Homestead -> Metropolis(Byzantium)
  - consensus.go에 기술됨


```
// CalcDifficulty is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time
// given the parent block's time and difficulty.
func CalcDifficulty(config *params.ChainConfig, time uint64, parent *types.Header) *big.Int {
        next := new(big.Int).Add(parent.Number, big1)
        switch {
        case config.IsByzantium(next):
                return calcDifficultyByzantium(time, parent)
        case config.IsHomestead(next):
                return calcDifficultyHomestead(time, parent)
        default:
                return calcDifficultyFrontier(time, parent)
        }
}
```

난이도(Difficulty)

- **Frontier**
  - block_diff = parent_diff + parent_diff // 2048 * (1 if block_timestamp - parent_timestamp < 13 else -1) + int(2**((block.number // 100000) - 2))
  - 각 블록간의 시간차이가 기준 시간(13초)에 따라 얼마나 다른 가에 따라 조정

- **Homestead**
  - adj_factor = max(1 - ((timestamp - parent.timestamp) // 10), -99)
  - child_diff = int(max(parent.difficulty + (parent.difficulty // BLOCK_DIFF_FACTOR) * adj_factor, min(parent.difficulty, MIN_DIFF)))
  - parent difficulty도 같이 고려

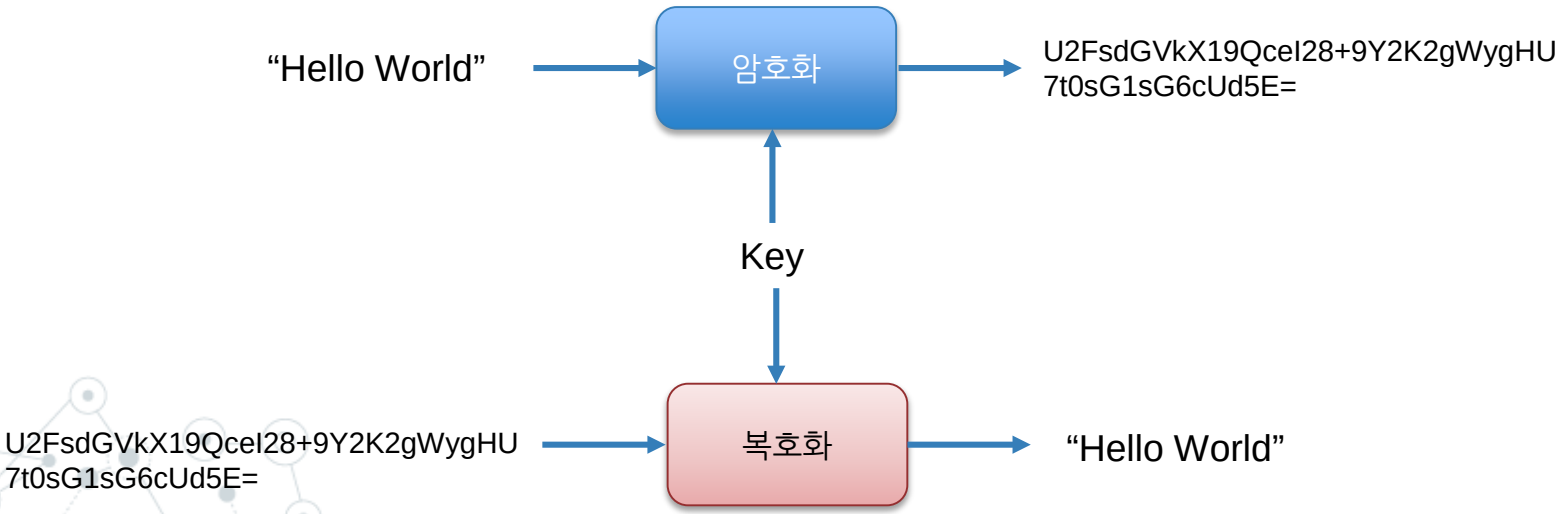
- **Metropoliz(Byzantium)**
  - adj_factor = max(1 + len(parent.uncles) - ((timestamp - parent.timestamp) // 9), -99)
  - child_diff = int(max(parent.difficulty + (parent.difficulty // BLOCK_DIFF_FACTOR) *adj_factor, min(parent.difficulty, MIN_DIFF)))
  - uncle 블록도 고려하여 계산

대칭키/비대칭키 암호 알고리즘

대칭키 암호 알고리즘

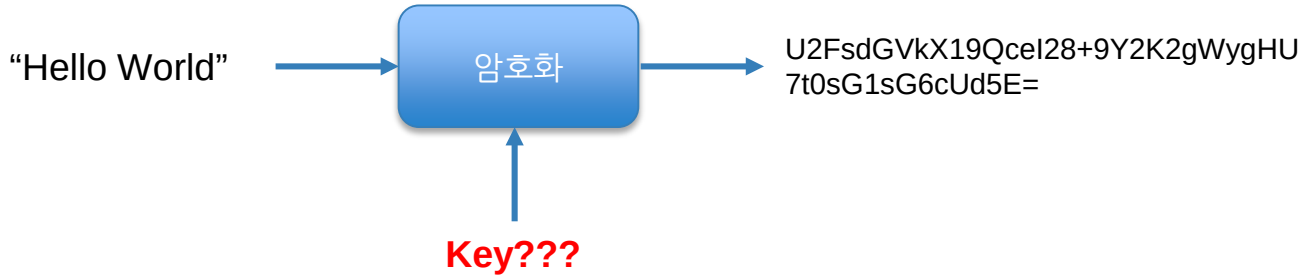
- 암호화 및 복호화에 동일한 키를 사용
- 입력은 일정 블록 사이즈(16바이트)로만 입력이 가능
  - 입력값이 16바이트 이하일 경우는 16바이트로 만들어서 암호화 처리(Padding)
  - 출력값(암호문)에서 다시 복호화 후에 16바이트로 만들기 전으로 변경(un-Padding)

- 키 사이즈는 128/196/256비트 사용이 가능(AES의 경우)
  - 암호문으로부터 키를 모르면 평문 계산이 매우 심하게 어려움
  - 대칭키 암호 알고리즘은 키 사이즈에 의존



대칭키 암호의 안전성

- 암호 알고리즘 해킹 = 키를 찾는 방법
  - 평문과 암호문이 주어졌을 때에 해당하는 키를 찾는 방법
  - 모든 가능한 경우의 키를 하나하나 대입해서 찾음(Brute-force)



Key	암호문
0x00000000000000000000000000000000	U2FsdGVkX18aIhuYXhx65zi9aT+vUldHeieD1X5OUbY=
0x00000000000000000000000000000001	U2FsdGVkX19XbLqOizZwgG+KQ+IkZcpJqByeSA4jPSU=
0x00000000000000000000000000000002	U2FsdGVkX19XasfafggGG+KQ+IkZcpJqByeSA4jPSU=
.....	.....
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	U2FsdGVkX1+oBEy6oHZiaetmsVvJb9/s8vu62YCqaRU=

대칭키 암호의 안전성

- 안전성 = 키 사이즈(모든 키의 조합의 수)
  - 128비트일 경우에 모든 조합의 수는 2^128로 약 300000000000000000000000000000000000(3x10^38)
  - 한 대의 PC의 처리 속도 : 10^9(10억)/초
  - 3x10^38 / 10^9 = 3x10^29초 -> 약 10^23년
  - 1년 31,536,000초
  - 지구 역사 46억년(4.6x10^9)

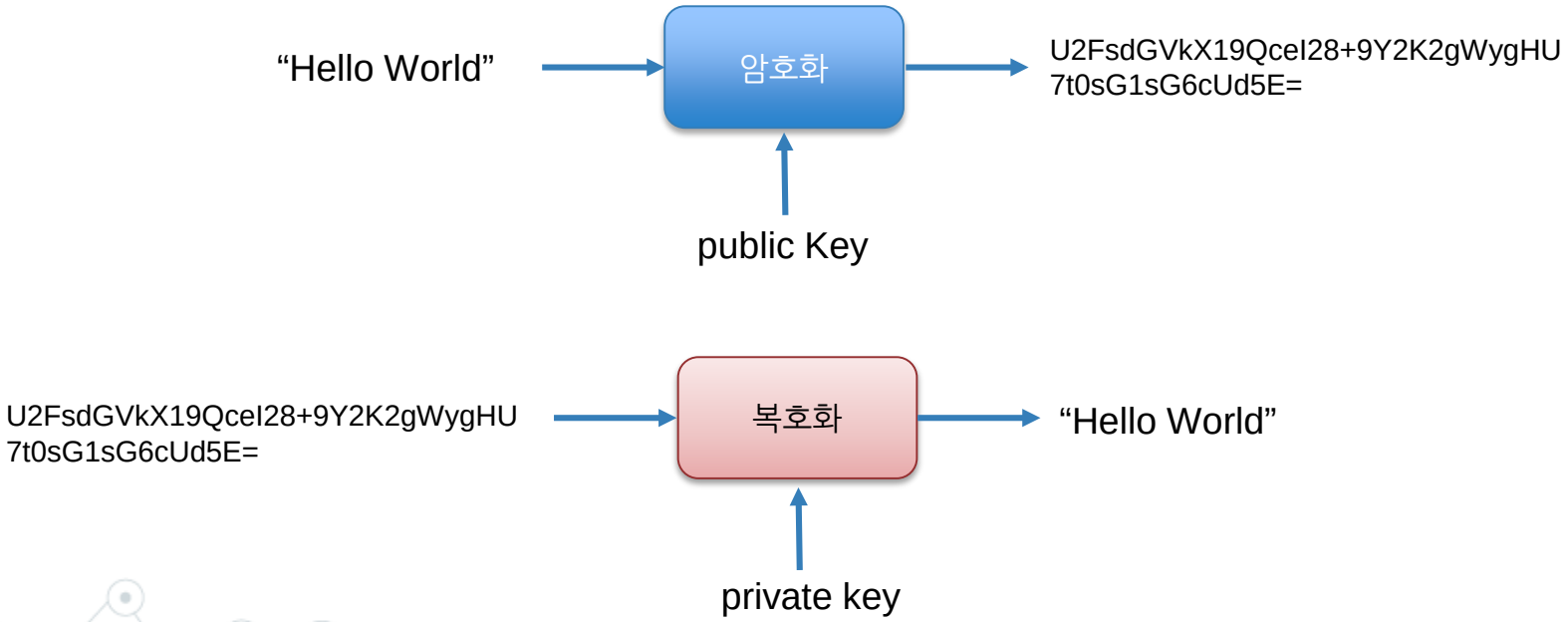
전세계의 인구(70억명)이 평균 10대의 PC를 가지고 나눠서 돌렸을 때에 1.4 x 10^12(14000억) 년이 걸림

Q. 신문지를 100번 겹쳐서 접으면 몇 cm로 될까??

A. 2^100 x 0.01cm = 1.2x10^23

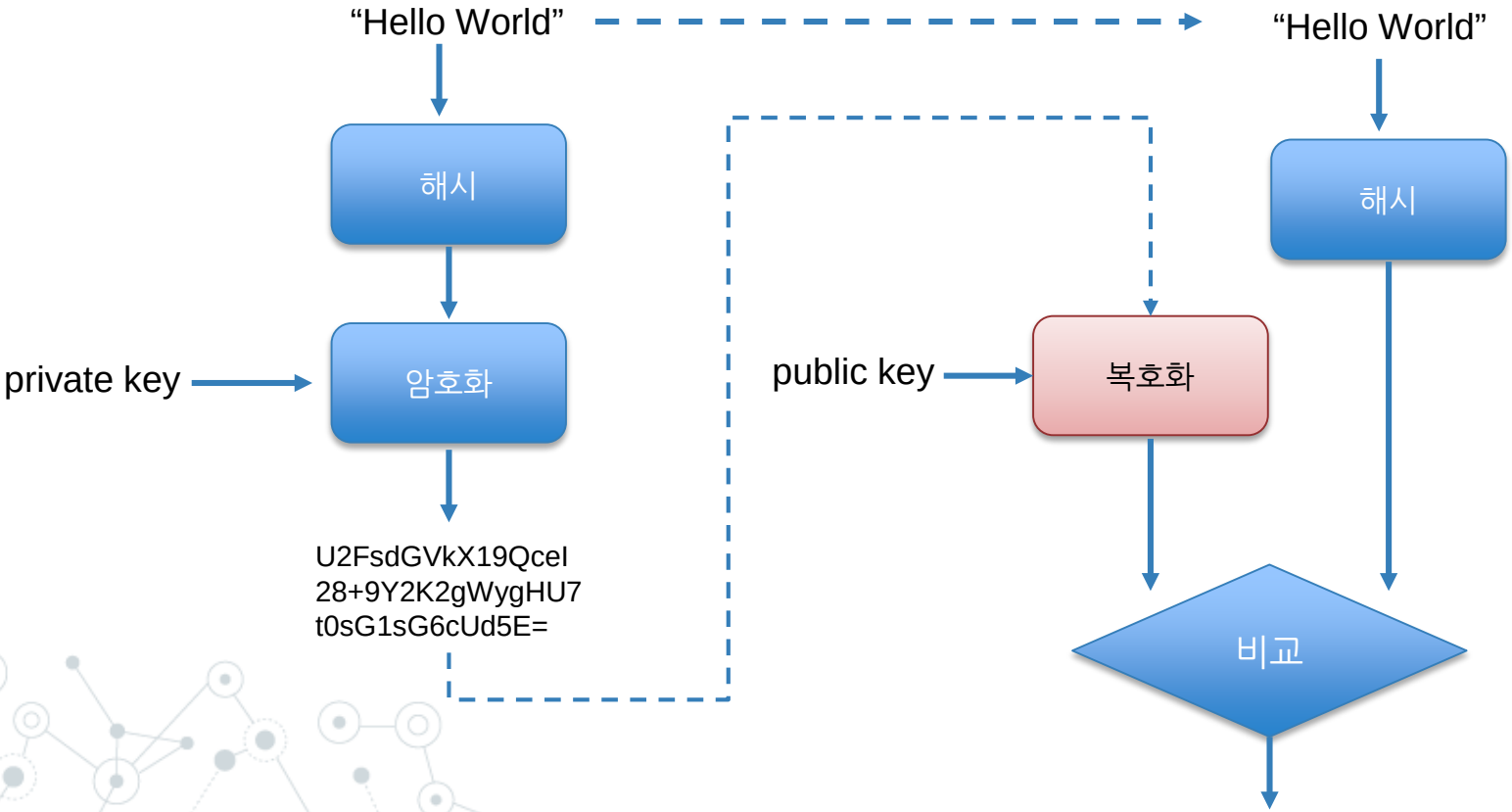
비대칭키 암호 알고리즘

- 암호화와 복호화에 다른 키를 사용
  - 공개키 : 모두에게 공개된 키
  - 개인키 : 공개키와 쌍을 이루는 개인키로 비공개(secret)



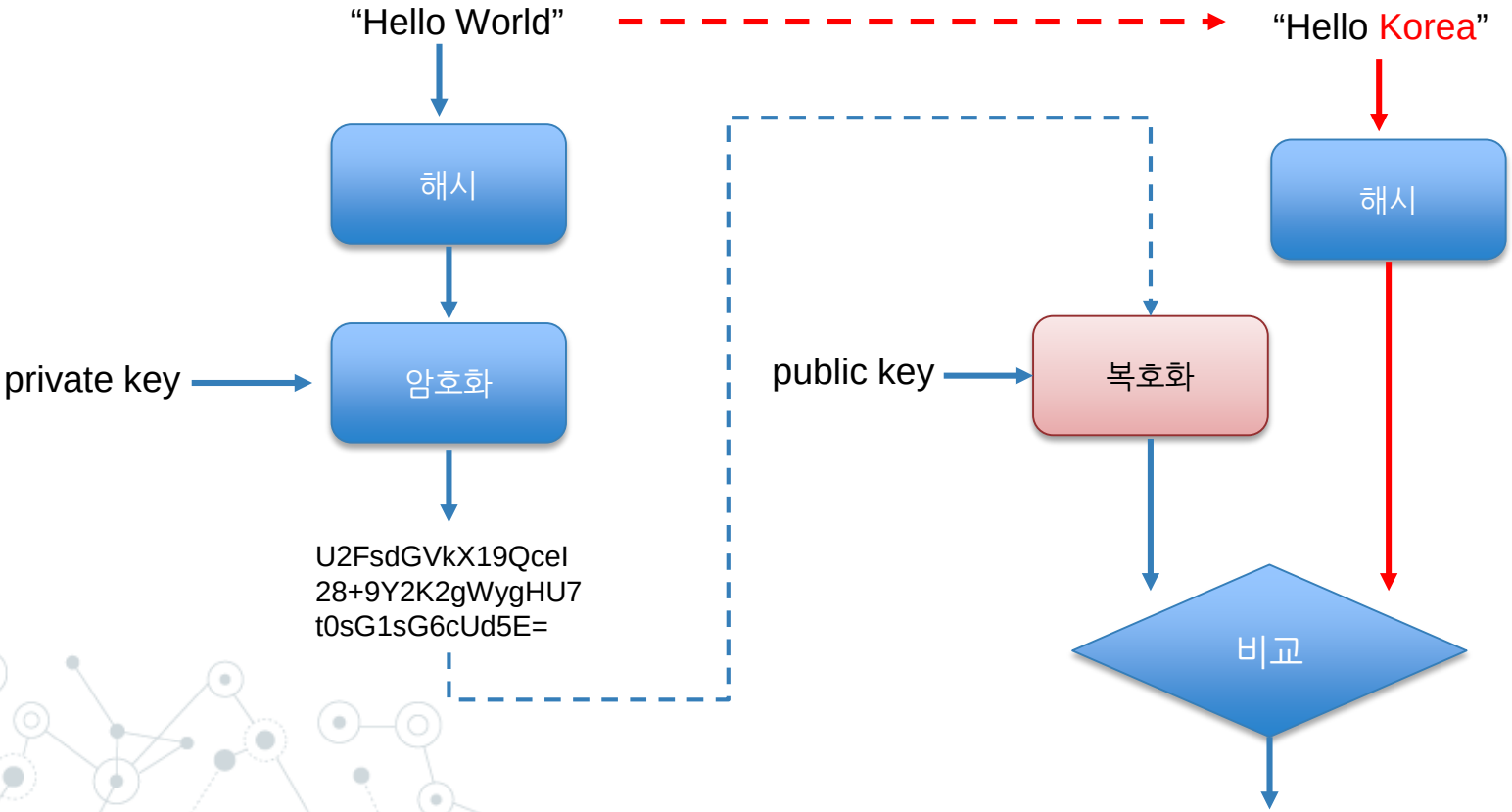
전자서명(Digital Signature)

- 암호화에 개인키, 복호화에 공개키를 사용
  - 보내는 사람(서명자)가 맞는지 확인
  - 보내는 메시지의 내용이 변경이 되지 않고 전달된 것인지 확인



전자서명(Digital Signature)

- 암호화에 개인키, 복호화에 공개키를 사용
  - 보내는 사람(서명자)가 맞는지 확인
  - 보내는 메시지의 내용이 변경이 되지 않고 전달된 것인지 확인



이더리움에서의 전자서명

- 모든 트랜젝션에 전자서명이 포함됨(ECDSA)
  - 트랜젝션 : 송금 데이터 등 이더리움 네트워크에 보내는 데이터
  - e.g. {from: 0xaaf..ff, to:0x04abff..ab, value:1 “ether”}

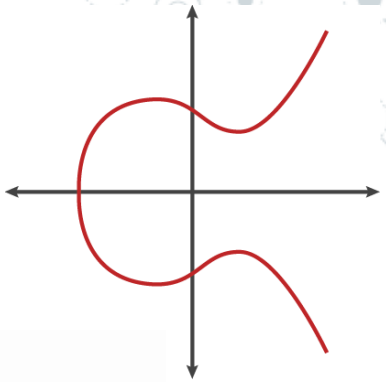
TxHash:	0xa01d8096d04d9aba3722079d058e74f3230a3a223d17977239c8815bfcc0ac6e
TxReceipt Status:	Success
Block Height:	2971210 (319105 block confirmations)
TimeStamp:	48 days 3 hrs ago (Apr-05-2018 01:02:25 AM +UTC)
From:	0xcb81068b05ce376122401abd5aba6d0c830faaee
To:	0x106eba62614c049198dc94d243761ad94672668a
Value:	10 Ether ($0.00)
Gas Limit:	21000
Gas Used By Txn:	21000
Gas Price:	0.00000002 Et
Actual Tx Cost/Fee:	0.00042 Ether

10이더 송금

Raw Transaction Data

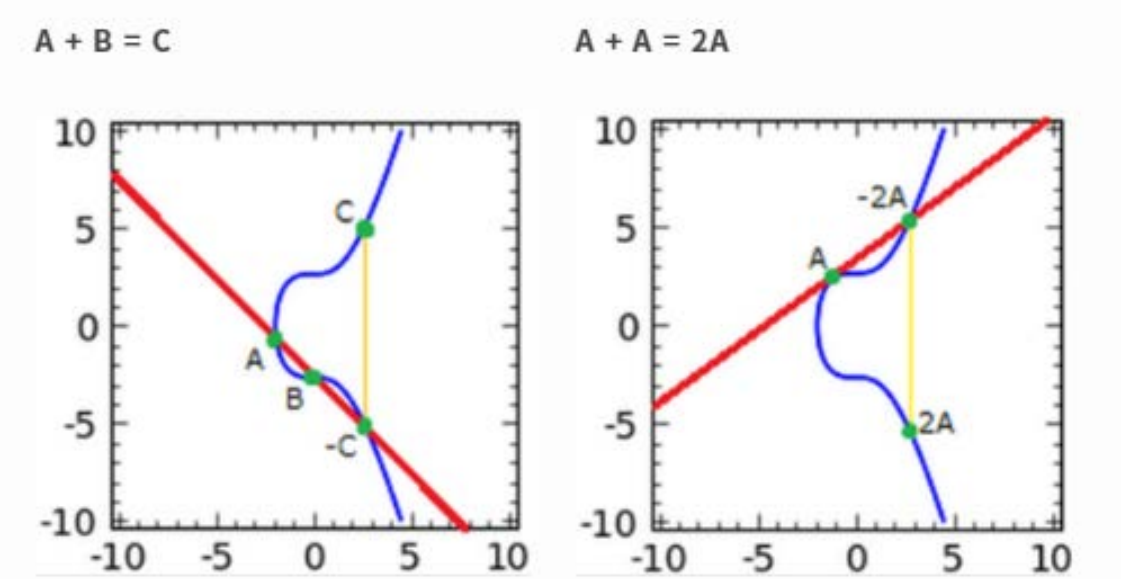
```
> eth.getTransaction("0xa01d8096d04d9aba3722079d058e74f3230a3a223d17977239c8815bfcc0ac6e")
{
  blockHash: "0x224e51449e2ce85340ada58ed928442a1b4b3bbb387732a3c342bc22bb5bc49b",
  blockNumber: 2971210,
  from: "0xcb81068b05ce376122401abd5aba6d0c830faaee",
  gas: 21000,
  gasPrice: 20000000000,
  hash: "0xa01d8096d04d9aba3722079d058e74f3230a3a223d17977239c8815bfcc0ac6e",
  input: "0x",
  nonce: 7,
  r: "0x9e4b314f5364a971607d49dd9db55d444f027a7cb44655f370a153c14ab4ab7c",
  s: "0x5410f03fab516d14eb0246a1ca9f7d5c3d9737b98a3e0db10915c26758803ed8",
  to: "0x106eba62614c049198dc94d243761ad94672668a",
  transactionIndex: 63,
  v: "0x29",
  value: 10000000000000000000
```

Elliptic Curve 암호 시스템



◎ 비대칭 암호 알고리즘(비밀키, 공개키)

$$y^2 \equiv x^3 + a \cdot x + \mathrm{b} \mod \mathrm{p}\,, where\ all\ pairs\ (x, y) \in GF(p)$$



Addition Doubling

$$T = dP$$

private key : d

public key : T, P

ECDSA

- Elliptic Curve Digital Signature Algorithm

$$y^2 \equiv x^3 + a \cdot x + \mathrm{b} \mod \mathrm{p}$$

public key : $(\mathrm{a}, \mathrm{b}, \mathrm{p}, \mathrm{q}, \mathrm{A}, \mathrm{B})$, where $\mathrm{B} = \mathrm{dA}, 0 < \mathrm{d} < \mathrm{q}$

private key : (d)

- 서명 생성
  - K_e : 임의의 수
  - $h(x)$: 해시함수
  - x_R: 포인트 R의 x좌표
  - A : 타원곡선상의 포인트

$$0 < \mathrm{K_e} < \mathrm{q}$$
$$R = K_e A$$
$$r = x_R$$
$$s = (h(x) + d \cdot r) \cdot K_e^{-1} \; mod \; q$$

- 서명 검증
  - 공개키 **B**가 필요함

$$\mathrm{w} \equiv s^{-1} \; mod \; q \; 0 < \mathrm{K_e} < \mathrm{q}$$
$$u_1 \equiv w \cdot h(x) \; mod \; q$$
$$u_2 \equiv w \cdot r \; mod \; q$$
$$P = u_1 A + u_2 B$$
$$check \; x_p = r \text{ or not}$$

ECDSA

- 비트코인 및 이더리움 : **secp256k1 파라미터 사용**

$$y^2 \equiv x^3 + 7 \mod p$$

private key : $a = 0, b = 7, p =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
A = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
q = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C

- Transaction의 서명 검증(공개키가 있어야함)
  - 이더리움 지갑 주소가 공개키는 아님(비트코인도 마찬가지)
    - 추후에 양자컴퓨터의 등장 등으로 공개키 -> 개인키 추정이 불가능 하도록 설계
  - 공개키의 해시값의 일부를 나타낸 값이 지갑 주소임

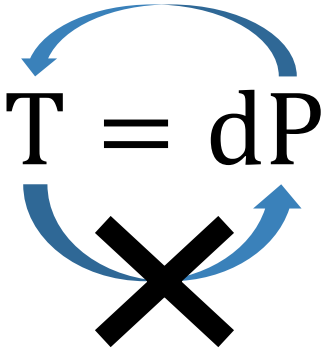
- 하지만, signature(r,s) 및 이더리움 지갑 주소로부터 공개키 계산이 가능
  - 서명 검증이 가능함

$$B = r^{-1}(sR - h(x) \cdot A) \ \ or \ \ r^{-1}(sR' - h(x) \cdot A)$$

$$\begin{aligned} B &= r^{-1}(sR - h(x) \cdot A) \\ &= r^{-1}\{(h(x) + d \cdot r) \cdot K_e^{-1} \cdot R - h(x) \cdot A\} \\ &= r^{-1} \cdot h(x) \cdot K_e^{-1} \cdot R + d \cdot K_e^{-1} \cdot R - r^{-1} \cdot h(x) \cdot A \\ &= r^{-1} \cdot h(x) \cdot K_e^{-1} \cdot K_e \cdot A + d \cdot K_e^{-1} \cdot R - r^{-1} \cdot h(x) \cdot A \\ &= d \cdot K_e^{-1} \cdot R \\ &= d \cdot K_e^{-1} \cdot K_e \cdot A = d \cdot A \end{aligned}$$

정리

- 대칭키 암호 알고리즘
  - 암호화 및 복호화에 동일한 키를 사용함
  - 2^128, 2^256 brute-force는 현실적으로 불가능
- 비대칭키 암호 알고리즘
  - 암호화 및 복호화에 다른 키를 사용함
  - 비대칭키 기반 서명 알고리즘(ECDSA)는 개인키로 서명 생성, 공개키로 서명 검증
- 해시 알고리즘
  - 출력값(해시값)은 유니폼한 랜덤성을 가짐(2^256)
  - 다른 두개의 입력값에 대해서 다른 두개의 해시값이 나옴



private key : d

public key : T, P

현존하는 public키 암호 알고리즘

- 타원곡선 암호
  - 티원곡선 상에서 역원 계산이 “어려움”
  - 반대로 곱셈은 쉬움
- 소인수 분해 (RSA)
  - 합성수를 두 개의 소수의 곱으로 분해라는 방법이 “어려움“
  - 반대로 두 개의 소수의 곱은 쉬움
- 이산로그 (DH)
  - a^x = b mod p 에서 x를 찾는 방법이 “어려움“
  - e.g. 3^k = 5 (mod 7)에서 k의 값은?
  - 반대로 거듭제곱(a^x)은 쉬움