[이론] 블록체인 핵심기술

→ 거래 검증 기술

블록체인 (Blockchain)

06. 01 코인 거래 (Transaction) 기능 구현 (이론)

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@cju.ac.kr)

Orientation for Current Status

강의는 주제별로 구분하여 github.com 에 올려 놓았습니다.

- URL: https://github.com/kafa46/cju_coin

순번	내용	동영상	슬라이드	소스코드
01	오리엔테이션	click	확인	없음
02	이론 1. 블록체인 역사, 개념/논문 소개	click	확인	없음
03	이론 2. 핵심기술, 해시, 머클 트리	click	확인	없음
04	03-01. 개발환경 세팅	click	확인	이동
05	03-02. 플라스크 설정 및 데이터베이스 설정	click	확인	이동
06	03-03. Blockchain 클래스 구현 및 작동	click	확인	이동
07	04-01. Blockchain 거래구현을 위한 사전설명(Bootstrap, jQuery, Ajax)	click	확인	이동
08	04-02. Blockchain transfer 클래스 코딩	click	확인	이동
09	04-03. Blockchain 콘트롤러 main_views.py 코딩	click	확인	이동
10	04-04. UI (index.html) 및 jQuery/Ajax (transfer.js) 코딩	click	확인	이동
11	05-01. Wallet 서버 UI 설명(이론)	click	확인	이동
12	05-02. Wallet 서버 뼈대 만들기	click	확인	이동
13	05-03. login_manager 구현 및 서버 forms.py 작성	click	확인	이동
14	05-04. Wallet 서버 회원가입 템플릿 상속 및 UI 코딩	click	확인	이동
15	05-05. 회원가입 controller auth_views.py 코딩	click	확인	이동
16	05-06. 로그인 페이지 login.html 코딩	click	확인	이동
17	05-07. 코인 이체 페이지 User Interface 구축	click	확인	이동
18	05-08. UI 구축을 마무리하며 (프런트(UI) 코딩에 힘들어하는 분들을 위하여)	click	없음	없음

- 가능하면 프런트 기술을 틈틈이 익히는 것이 중요
- 시간이 없다면 Lecture 05 일단 패스!

Contents

- Transaction 검증의 필요성
- Private / Public Key 생성
- Blockchain Address (지갑 주소) 생성
- Signature 생성
- Blockchain 네트워크에서 Transaction 검증

[이론] 블록체인 핵심기술

→ 거래 검증 이론 (필요성)

Transaction 검증 필요성

거래 기능을 구현해야 하는 상황?

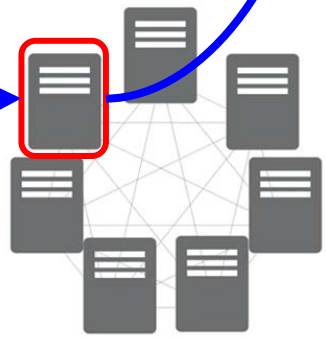
- **교수님 ㅠㅠ**
  - 거래 기능 구현해야 하나요?
  - 우리는 이미 Lecture 04 에서 했잖아요???? ㅠㅠ

- **그래도 필요합니다.**

이미지 출처:
https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=brocolicheeze&logNo=221654382864

Wallet Server
(웹 서버)

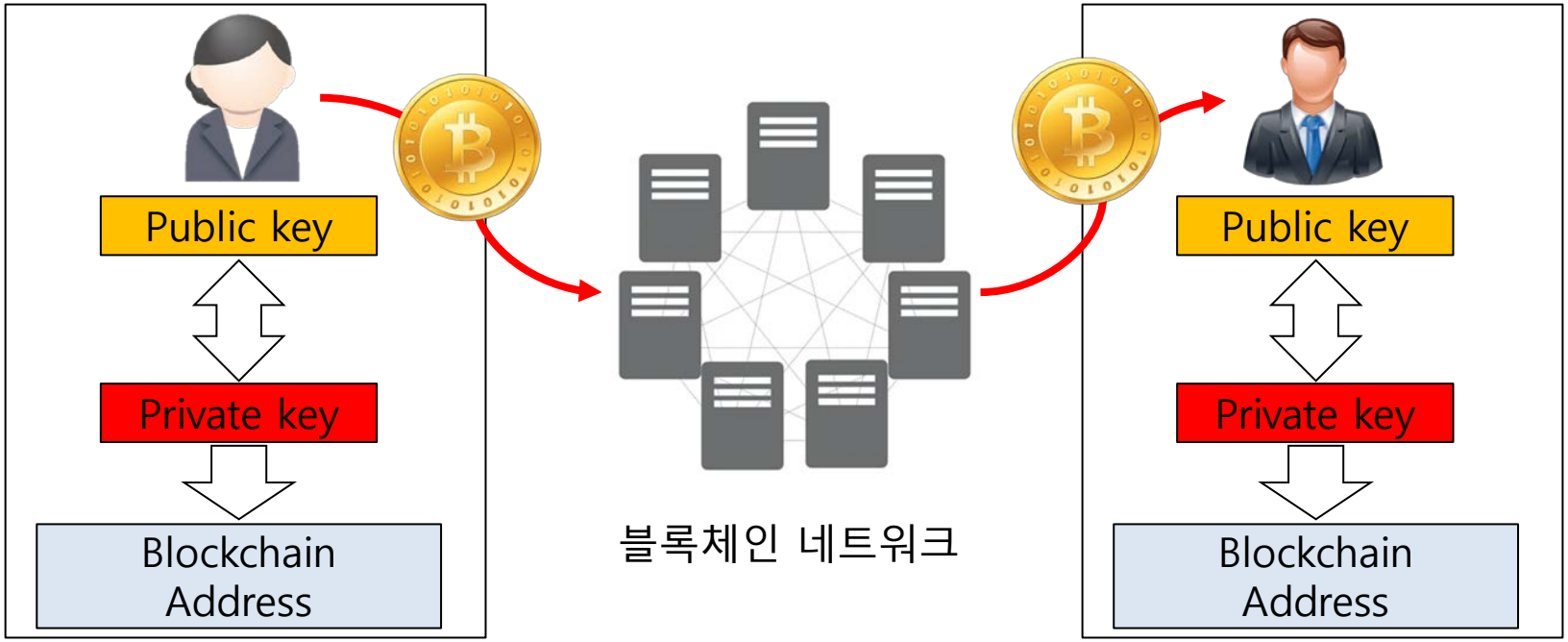
이미지 출처:
https://www.flaticon.com/kr/free-icon/hacker_6463383



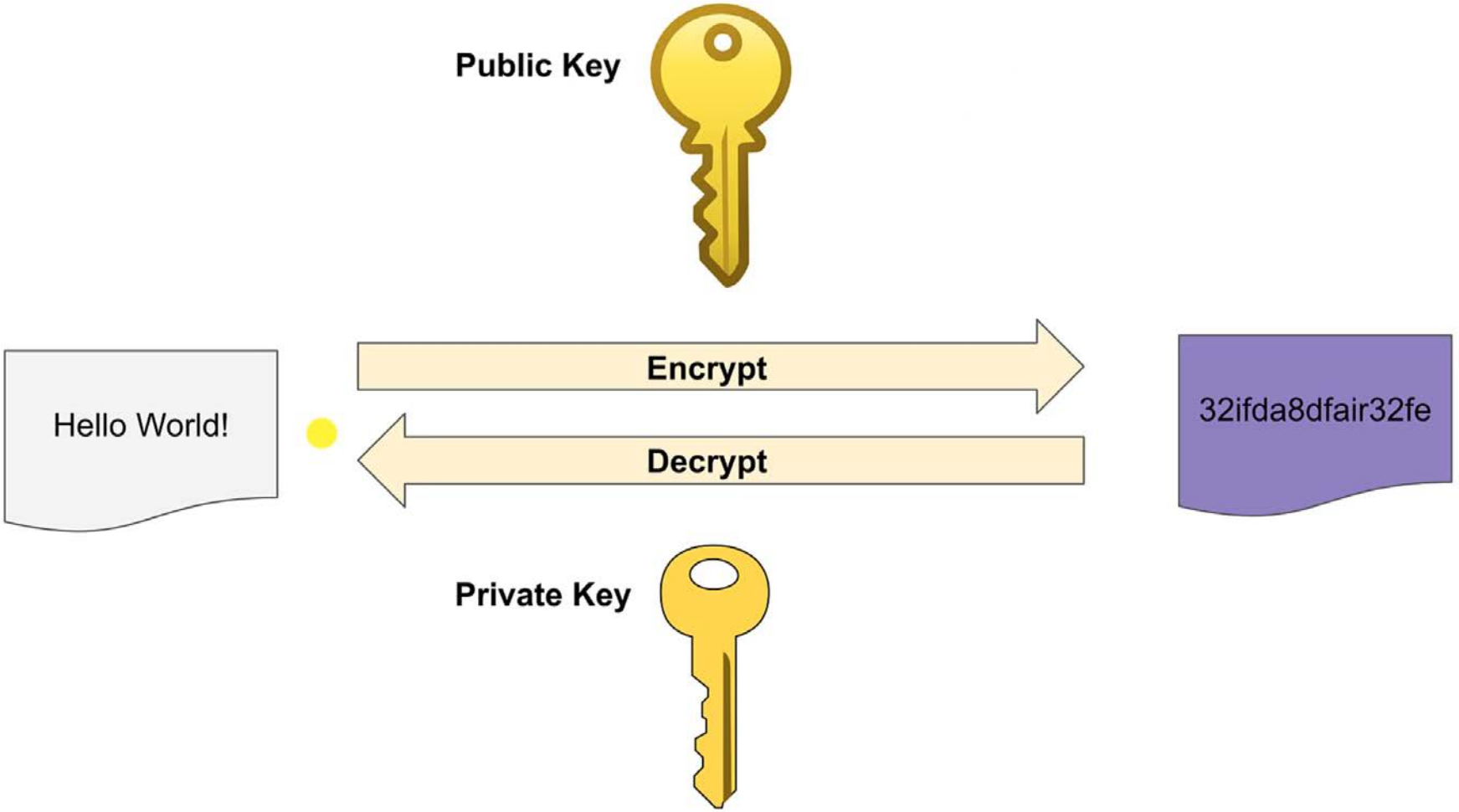
이미지 출처:
https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=brocolicheeze&logNo=221654382864

블록체인을 이용한 안전한 거래를 하려면?

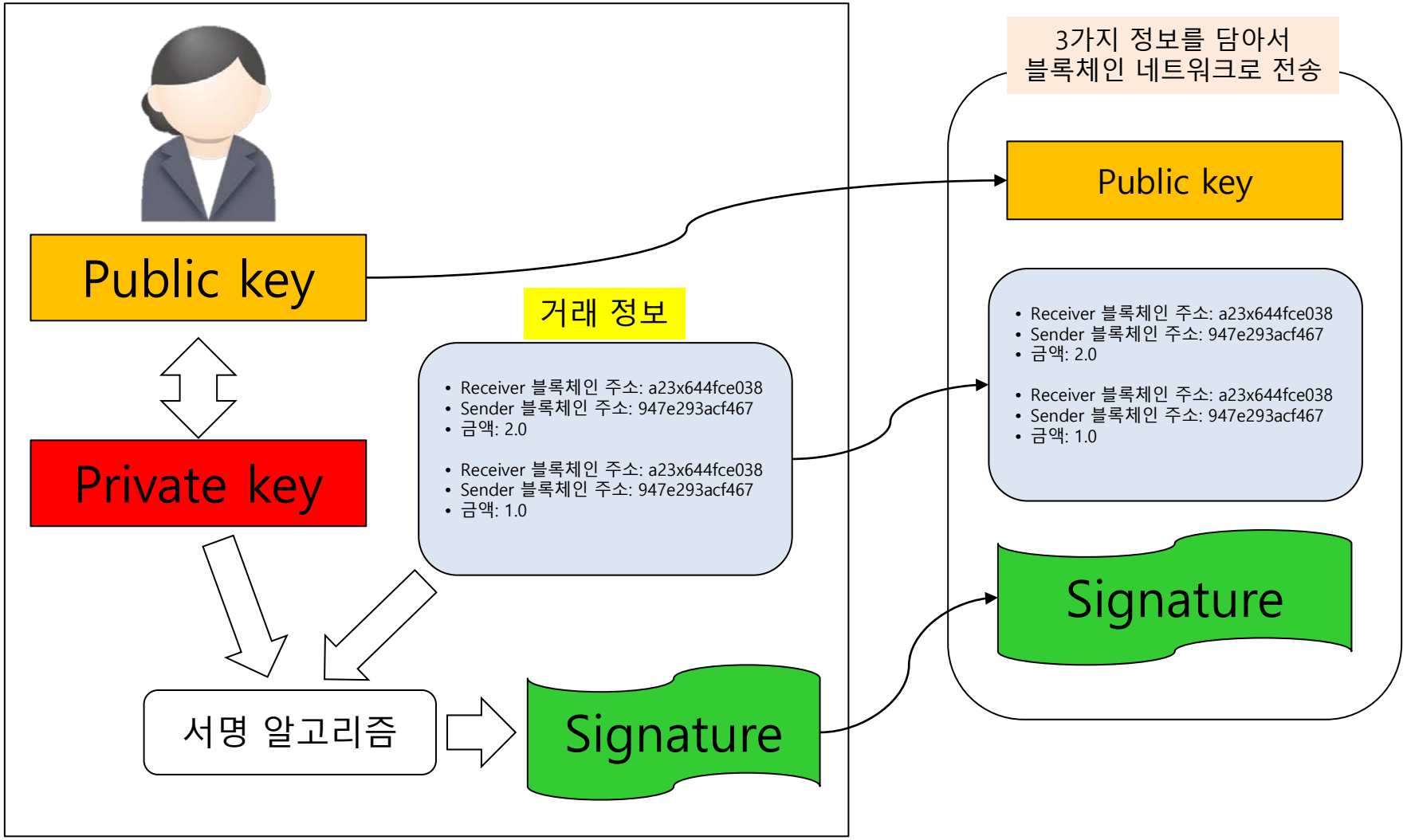
- **신뢰할 수 있는 거래를 하려면 Wallet (지갑) 필요**
- **지갑을 구현하려면?**
  - Private Key, Public Key ➔ 올바른 거래인지 확인하기 위해 사용
  - Blockchain Address ➔ 누구 지갑인지 확인하기 위해 사용



Private Key, Public Key – Review



안전하게 Transaction 추가를 하려면?



[이론] 블록체인 핵심기술

→ 비밀키 생성 기술

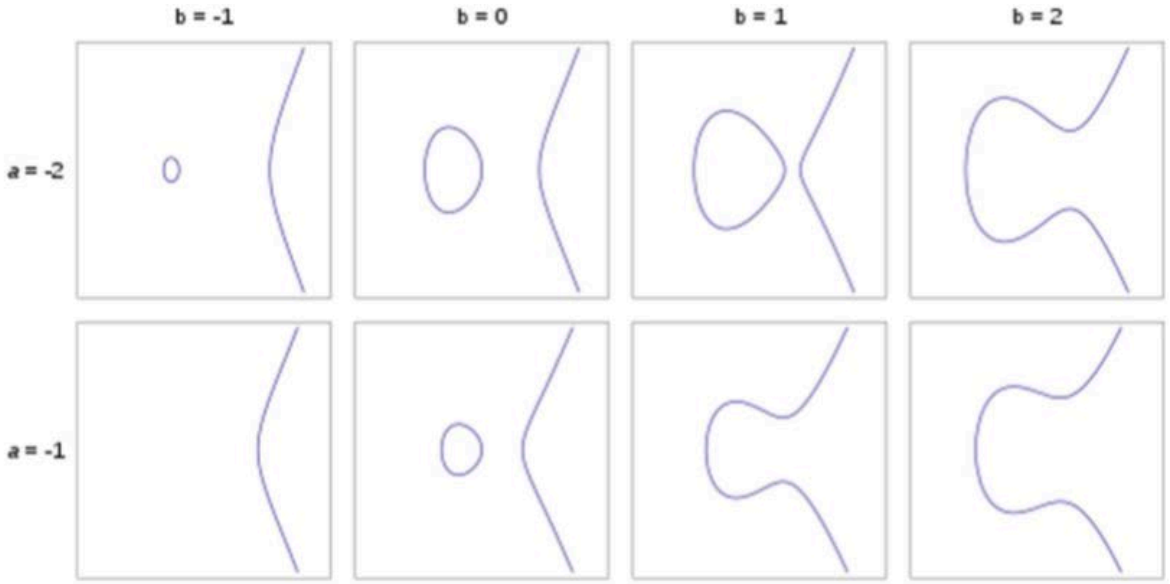
Private / Public Key 생성

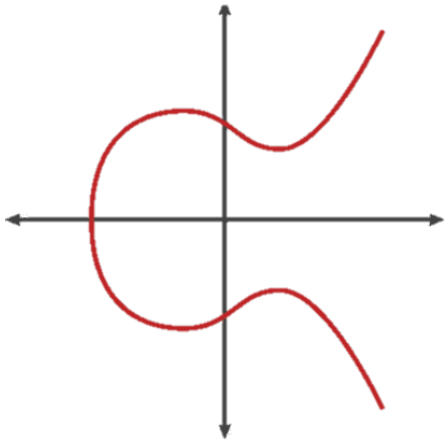
Private/Public Key 생성 – ECDSA 알고리즘 1

■ **ECDSA 알고리즘** ➔ NIST256P (곡선 종류 중 하나) 적용

- Elliptic Curve Digital Signature Algorithm ➔ 복잡한 타원을 이용하자!
- 복잡한 타원? 수식으로 표현 가능
  - $y^2 = x^3 + ax + b$
  - 타원 곡선의 모양? (x축에 대칭인 곡선)
    - 파라미터 a, b 에 따라 결정 (딥러닝 학습 params와 같은 개념)

타원암호(160비트)의 특징
1. RSA(1,024비트)보다 짧다.
2. 그래서 빠르다.
3. 성능은 RSA와 동일한 성능이다.
4. 이동통신 암호화에 주로 사용

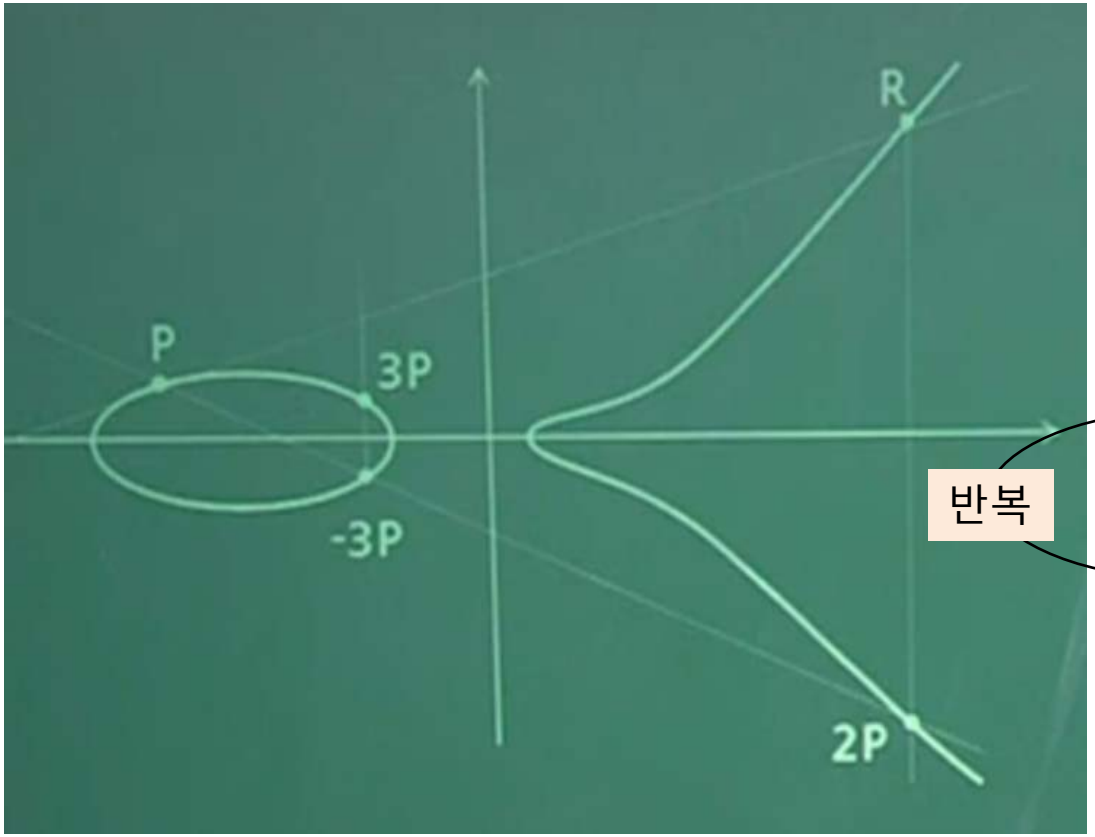




Private/Public Key 생성 – ECDSA 알고리즘 2

■ **ECDSA 알고리즘** ➔ NIST256P (곡선 종류 중 하나) 적용

- Elliptic Curve Digital Signature Algorithm ➔ 복잡한 타원을 이용하자!



https://youtu.be/7jscDr000DI

P + Q + R = 0 되도록
P, Q, R 선택 (P, Q는 중근)

시작점은 P=Q 되도록 선택
2P + R = 0
➔ R = -2P

X 축에 대칭인 점을 찾는다
y값의 위치는 ➔ +2P

반복

+2P에서 시작점으로 직선 긋는다.
P + R + 2P = 0
➔ R = -3P

⋮

R = kP

k → Private Key
kp → Public Key

비밀키, 공개키 생성 – Overview

- wallet.py 파일 생성
- Private/public key 생성

```python
from ecdsa import NIST256p, SigningKey

class Wallet:
    '''비트코인 전자지갑'''
    def __init__(self) -> None:
        self._private_key = SigningKey.generate(curve=NIST256p)
        self._public_key = self._private_key.get_verifying_key()

    @property
    def private_key(self) -> str:
        '''private key를 문자열로 변환'''
        return self._private_key.to_string().hex()

    @property
    def public_key(self) -> str:
        '''public key를 문자열로 변환'''
        return self._public_key.to_string().hex()
```

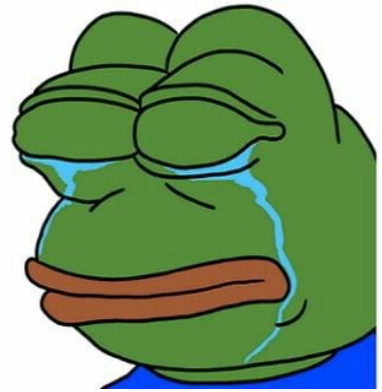
[이론] 블록체인 핵심기술

→ Wallet (지갑) 주소 생성 기술

Blockchain Address (지갑주소) 생성

누군가 만들어 놓은 건 없나요?

- **누군가 잘 정리해 놓은 설명 (tutorial) 없나요?**
  - 있어요 ~~
  - 참고 블로그 → 아래 블로그를 참고하세요^^
    - https://www.freecodecamp.org/news/how-to-create-a-bitcoin-wallet-address-from-a-private-key-eca3ddd9c05f/



- **하지만… ㅠ 저는 여전히 헷갈리네요 ㅠ**
- **교수님이 정리해 주시면 안될까요?**

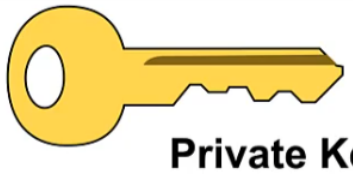
다음 슬라이드로 ^^

- **소스코드는 없나요?**
  - 있어요 ~~
  - Github 주소를 확인하세요 ^^.
    - https://github.com/Destiner/blocksmith
    - 비트코인 주소 생성 코드
      - https://github.com/Destiner/blocksmith/blob/master/blocksmith/bitcoin.py
    - 이더리움 주소 생성 코드
      - https://github.com/Destiner/blocksmith/blob/master/blocksmith/ethereum.py

짧고 고유한 지갑 주소 (blockchain address) 만드는 방법

짧고 고유한 public key 생성

1. ECDSA 이용하여 public key 생성
2. Public key 에 SHA-256 수행
3. SHA-256 결과에 Ripemd160 수행 (추가 암호화)
4. Network Byte 추가
5. SHA-256 2 회 (더블 해시) 수행
6. Checksum 구하기
7. Public key와 checksum 더하기
8. 더한 키를 Base58로 인코딩 하기



전자지갑 주소로 활용

1c246641ee61f81532b6e07086349c7d6e1a71e6373d99245956ccec8b502cf0b
9b973567b67b9800be6dad681a456cb3e41af924a8378d7ab0112dcee2fd5f7

너무 긴 지갑 주소

고유하면서도 짧은 주소

bSLz5q9YT4kwhzRvExPPYwpPtfP41Gq29FoPwqmrBGuDWur2cui54RRsyTRdTfqK

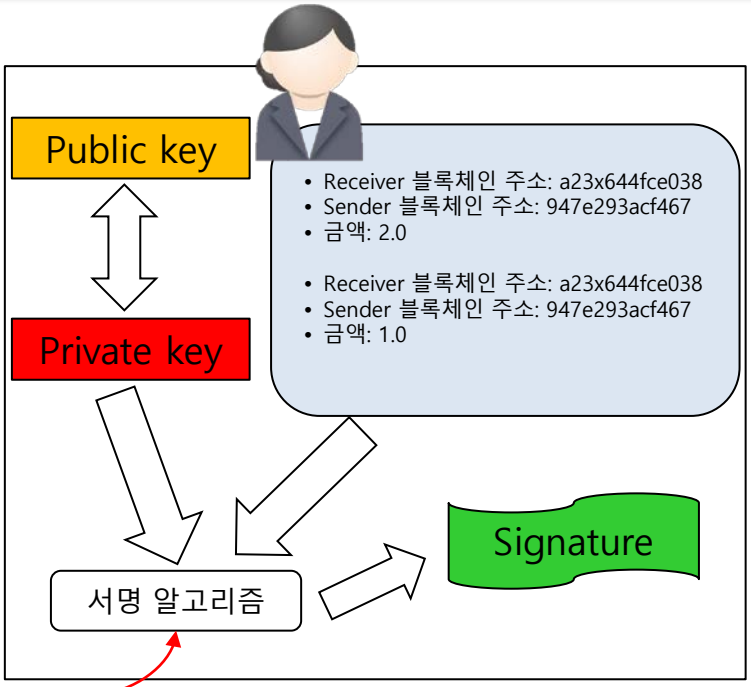
비트코인 방식 → 깊게 생각 X
(블록체인 구현에 따라 다른 알고리즘 적용 가능)

[이론] 블록체인 핵심기술

→ Signature 생성 기술

Signature 생성

Signature 생성 알고리즘




```python
from ecdsa import NIST256p, SigningKey

class Wallet:
    '''비트코인 전자지갑'''
    def __init__(self) -> None:
        self._private_key = SigningKey.generate(curve=NIST256p)
        self._public_key = self._private_key.get_verifying_key()

    @property
    def private_
        '''priv
        return

    @property
    def public_
        '''publ
        return
```

```python
def generate_signature(
    send_blockchain_addr: str,
    recv_blockchain_addr: str,
    send_private_key: str,
    amount: float
) -> str:
    '''거래에 필요한 signature 생성'''
    sha256 = hashlib.sha256()
    transaction = dict_utils.sorted_dict_by_key(
        {
            'send_blockchain_addr': send_blockchain_addr,
            'recv_blockchain_addr': recv_blockchain_addr,
            'amount': float(amount),
        }
    )
    sha256.update(str(transaction).encode('utf-8'))
    message = sha256.digest()
    private_key = SigningKey.from_string(
        bytes().fromhex(send_private_key),
        curve=NIST256p
    )
    # Private Key로 서명하기
    private_key_sign = private_key.sign(message)
    signature = private_key_sign.hex()
    return signature
```

Private / Public Key,
거래 내역을 알고 있으니
서명 알고리즘만 필요한 상황!

그나마 다행입니다.
이미 다 구현되어 있습니다. ^^

[이론] 블록체인 핵심기술

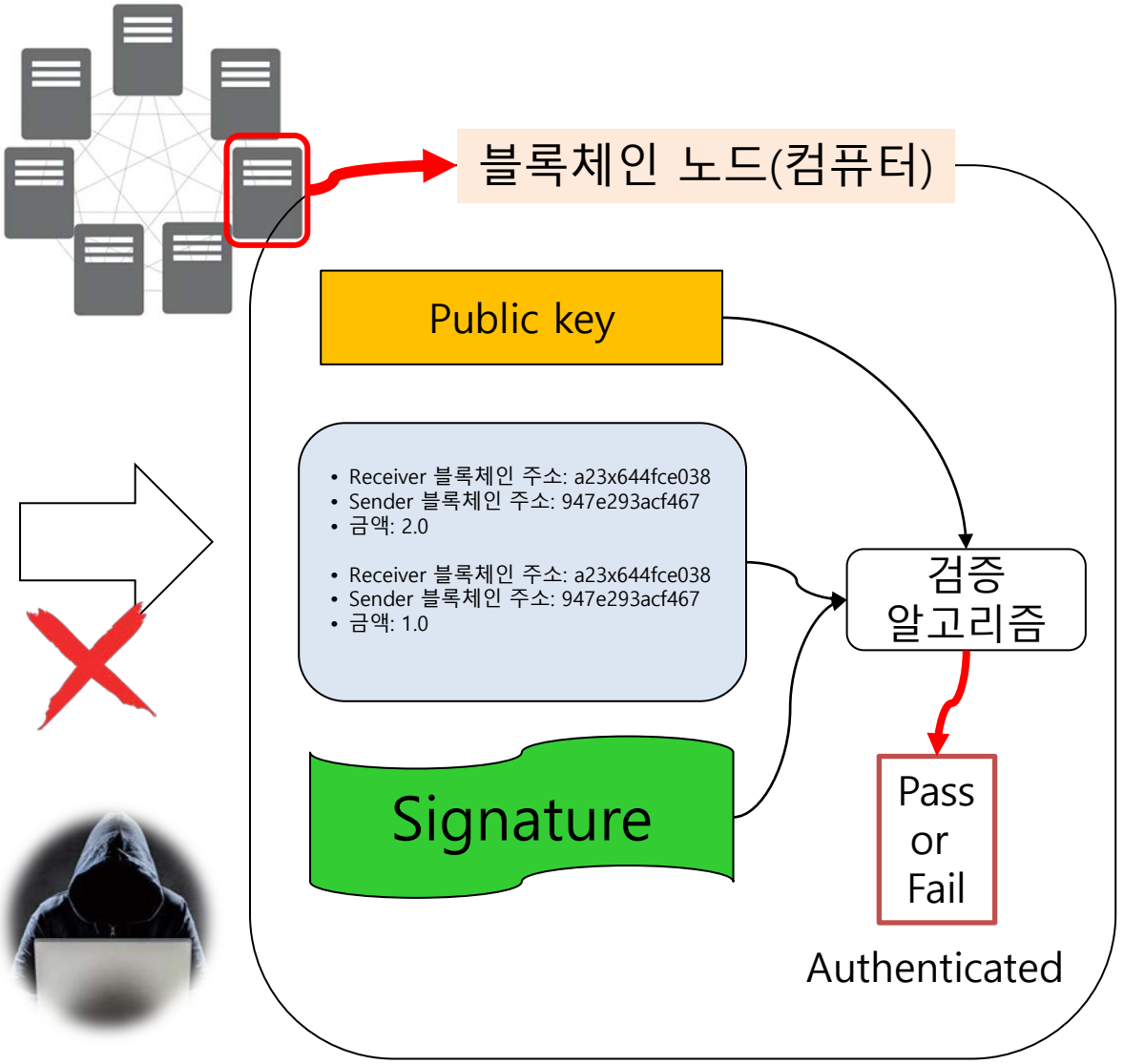
→ Transaction 검증 기술

블록체인 네트워크에서 Transaction 검증

블록체인(채굴 컴퓨터)에서 Transaction 검증 방법

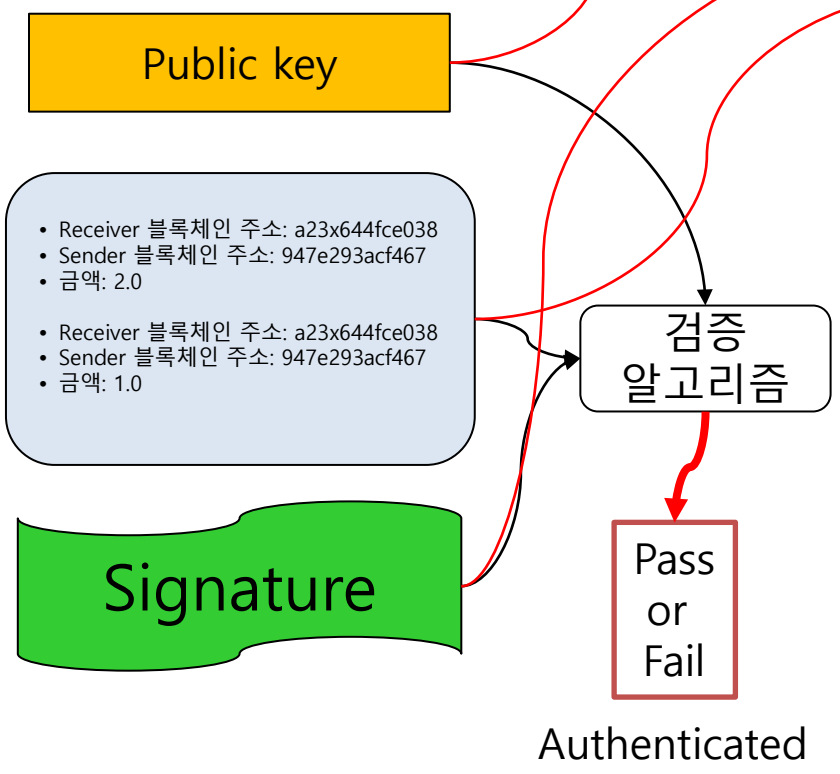
검증 알고리즘이 필요한 상황

넘겨 받은
정보가
올바른 지
검증하는 과정



어떻게 검증 알고리즘을 만들지?

- **머리 아프게 생각하지 않습니다.**
  - VerifyingKey 에서
    검증 알고리즘 제공




```python
# 메서드 추가 (signature 검증 알고리즘)
def verify_transaction_signature(
    self,
    send_public_key: str,
    singature: str,
    transaction: dict
) -> bool:
    sha256 = hashlib.sha256()
    sha256.update(str(transaction).encode('utf-8'))
    message = sha256.digest()
    singature_bytes = bytes().fromhex(singature)
    verifying_key = VerifyingKey.from_string(
        bytes().fromhex(send_public_key), curve=NIST256p
    )
    is_verified = verifying_key.verify(
        signature=singature_bytes,
        data=message,
    )
    # return: True if the verification was successful
    return is_verified
```

다음 강의
이제부터는
실습 해야죠 ~ ~ ~

수고하셨습니다 ..^^..