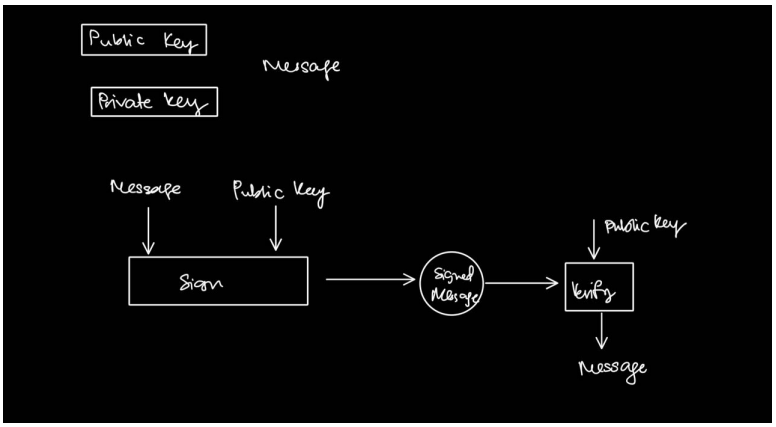
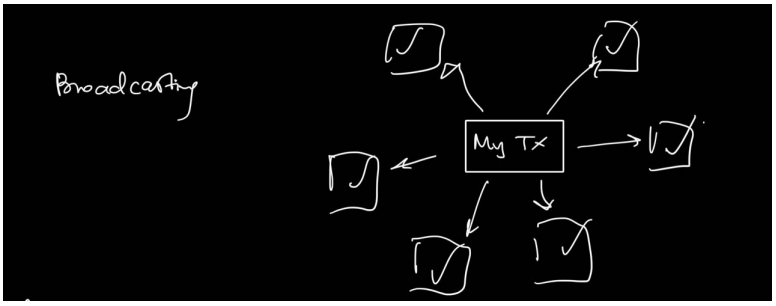
Problem 5 - Switcheo Code Challenge

The act of sending and receiving a transaction on the EVM consists of 3 intermediary steps:

1. Signing:
    - The transaction needs to be signed using a private key
    - The resulting hash is the signed transaction
    - This signed transaction will later be decoded using the public key for verification



2. Broadcasting:
    - The signed transaction is broadcasted to other nodes on the Ethereum chain.
    - It gets stored on a dedicated memory space on each node called a Mempool.
    - Global mempool comprises of all these dedicated memory spaces
    - This is where an unconfirmed transactions reside before being added to a block



3. Confirmation:
    - Miners group together transactions from the mempool, prioritizing those with higher transaction fees.
    - Blocks are then mined and miners are rewarded.

Broadcasting service

A broadcasting service should carry out the first two steps of the transaction sending process.

- Internal endpoint

```
POST /broadcast_transaction

//json payload
{"message_type": "add_weight(address _addr, uint256 _weight)",
 "data": "0xd713632800000000000000000000000005eb715d601c2f27f83cb554b6b36e047822fb70a0000000000000000000000000000000000000000000000000000000000000000
```

- The post endpoint handler should handle the JSON payload and send to to a transaction signing service.
- We can use the Web3.py library in Python for this

```
signed_txn = w3.eth.sign_transaction(transactionData, privateKey)
```

- signed_txn is returned back to the POST API caller.

Global array of map of transaction hashes, response codes and signed transaction hashes

- Each map in the array should have three keys indicating transaction_hash, response_code and signed_transaction.
- When a transaction is signed, a map should be appended to the array with an empty transaction hash, the signed_tx and response_code as “PENDING”
- This is so that if the service unexpectedly restarts, “PENDING” transactions are re-sent to the broadcasting function

Function handling the broadcasting using async programming

- We then used the signed transaction and asynchronously broadcast it to an EVM chain

```
tx_hash = await w3.eth.send_raw_transaction(signed_txn.rawTransaction)
```

- The transaction broadcasting should retry automatically once upon failure
- If the broadcast reponds with a success code, we update the global array of map: {transaction_hash: tx_hash, response_code: <success_code>, signed_transaction: signed_tx}.
- If the broadcast returns failure code the second time, we should update the global array of maps: {transaction_hash: tx_hash, response_code: <failure_code>, signed_transaction: signed_tx}.
- If the server unexpectedly restarts, we should first check the global array of maps and re-broadcast “PENDING” transactions.