Consensus in Bitcoin

Bitcoin Transactions and Consensus

1. Explain two reasons behind transaction fees. Why is the block reward not sufficient?

2. Name two functions that are fulfilled by the *coinbase* transaction, the first transaction in a Bitcoin block.

3. Visit the mempool explorer `https://mempool.space/` and observe the blocks on top of the home page.

   (a) Briefly explain each data field available on the blocks.

   (b) What is the difference between yellow/green and blue blocks?

(c) Observe the first yellow/green block next to the last blue block for a couple of seconds. Why do the data fields on it frequently get updated?

4. What does probabilistic consensus mean? Can a transaction be reverted?

5. The timestamp and difficulty fields are part of the header of a Bitcoin block. How are these values related?

Bitcoin Script

UTXOs are locked using Bitcoin Script, ensuring that only the intended recipient can spend them. The simplest type of script is pay-to-public-key (P2PK). In this case, the receiver must provide the sender with their public key. The successor to P2PK is Pay-to-Public-Key-Hash (P2PKH), where the identity is not a public key but the hash of a public key. To redeem the UTXO, a person must provide a public key that hashes to the P2PKH and a signature that belongs to this public key.

How does the Bitcoin Script work?

- scriptSig of the transaction input is concatenated with scriptPubKey of the transaction output, and then executed.
- The script runs sequentially on a stack machine. There are no registers and no external memory.
- The script is executed and if the result is true, the UTXO can be spent, otherwise not.

The below figure shows how a script is executed. Refer to this introduction to solve the following exercises. For additional information on Opcodes and Bitcoin Script execution, your are kindly referred to the following **Bitcoin wiki**.

Stack	Transaction Script	Explanation
	<signature> <public Key> ScriptSig (Txin) OP_DUP OP_HASH160 <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG ScriptPubKey (Txout)	The script is executed line by line and the stack displays the items remaining after the execution of the line
<signature>	➡ <signature> <public Key> OP_DUP OP_HASH160 <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG	The signature is pushed onto the stack.
<public Key> <signature>	<signature> ➡ <public Key> OP_DUP OP_HASH160 <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG	The public key is pushed onto the stack.
<public Key> <public Key> <signature>	<signature> <public Key> ➡ OP_DUP OP_HASH160 <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG	OP_DUP duplicates the current top item on the stack. Therefore, is placed on top of the stack again.
<public KeyHash> <public Key> <signature>	<signature> <public Key> OP_DUP ➡ OP_HASH160 <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG	The command tells us to pop the top stack value, compute its cryptographic hash and push the result top the top of the stack.
<public KeyHash?> <public KeyHash> <public Key> <signature>	<signature> <public Key> OP_DUP OP_HASH160 ➡ <public KeyHash?> OP_EQUALVERIFY OP_CHECKSIG	The public key specified in the Txout is pushed onto the stack. This was the hash specified by the sender of the transaction.
<public Key> <signature>	<signature> <public Key> OP_DUP OP_HASH160 <public KeyHash?> ➡ OP_EQUALVERIFY OP_CHECKSIG	The OP_EQUALVERIFY command checks whether the two top items on the stack are equal or not. If they are not equal, the program stops, and an error is thrown. Otherwise, both items are consumed. (In this case: Success!)
true	<signature> <public Key> OP_DUP OP_HASH160 <public KeyHash?> OP_EQUALVERIFY ➡ OP_CHECKSIG	The command OP_CHECKSIG pops two items of the stack and does an entire signature verification. If the signature verification is valid, then both items are consumed and true is pushed on the stack. The stack contains true and the transaction valid.

1. The following transaction output is provided:

```
OP_DUP OP_HASH160 8a014218a5a42e2c6fc5d573ab54a91ff555d1de OP_EQUALVERIFY OP_CHECKSIG
```

   (a) Can you tell which entity has created this transaction output?

   (b) Can you tell if this transaction output is spent?

   (c) Can you tell which entity is allowed to spend this transaction output?

   (d) What specific data is required to spend the transaction output?

2. Bitcoin Script allows setting rules for the spending of Bitcoins. The following script represents a standard Pay-to-Public-Key-Hash (P2PKH) script.

OP_DUP
OP_HASH160
PubKeyHash1
OP_EQUALVERIFY
OP_CHECKSIG

The TxOut-script.

As input, you would provide the corresponding signature and public key. Out of simplicity and reduced computational effort, Bob removes the following codes:

`OP_DUP OP_HASH160`

The entity that wants to spend this TxOut-script must provide the hash of the public key in addition to the signature and public key. Explain how you would attack this script and steal the funds.

PubKeyHash1
OP_EQUALVERIFY
OP_CHECKSIG

The TxOut-script.

Public Key Hash
Public Key
Signature

TxIn Script