Consensus in Bitcoin

Bitcoin Transactions and Consensus

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

> Transaction fees protect the network from spam transactions. Additionally, it enables a market around transactions, meaning that high-priority transactions can pay a higher price than low-priority transactions. Furthermore, transaction fees should incentivize miners to continue working when the block reward gets less and less (because of the halving). Another reason is that there is no incentive to spend energy to include transactions in a block when there is no transaction fee, as empty blocks can be mined.

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

> - It mints new coins providing mining incentive.
> - It can be used for data storage.

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.

> *Top to bottom:*
> i. Median fee rate (represented in satoshis per virtual byte) paid by transactions in the block. You can read more about virtual bytes and weight units in Bitcoin here `https://en.bitcoin.it/wiki/Weight_units`.
> ii. Span of the fee rates
> iii. Block size
> iv. Number of transactions included in the block
> v. When the block is expected to be mined (yellow blocks) or how long has it been since the block was mined (blue blocks)

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

> The yellow blocks represent the unconfirmed, future blocks expected to be mined next. Blue blocks represent already mined/confirmed 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?

> The first yellow block represents the first block a miner is expected to build on top of the latest mined block. As this block is not yet confirmed (or not even proposed in the first place), the set of transactions to be included in the block is not final. Hence, the website can only compute the data fields based on the **set of transactions expected to be included in the block**.
>
> Since new transactions are submitted to the network while mining is ongoing, miners can be incentivized to reorganize the set of transactions included in the block if the fees offered by the latest transactions which joined the miner's mempool are significantly larger than the fees offered by the current transactions included in the block. In such situations (i.e., when more profitable transactions join the mempool), the website updates the transactions expected to be included in the next block and recomputes the data fields displayed.

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

> A probabilistic consensus means that if a network agrees on a particular set of transactions to be executed, it is not 100% certain to be valid in the long term. Bitcoin and other blockchains with a probabilistic consensus have no settlement finality directive. This means that transactions can be reverted, however, with a very low (and decreasing) probability. The reason for such a reversal could be a chain re-organization attack.

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

> The timestamp is one of the values required to govern Bitcoins mining difficulty. Timestamps allow the monitoring of block times, which indicate whether difficulty is currently too low or high.

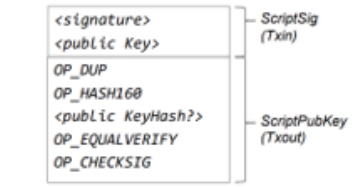
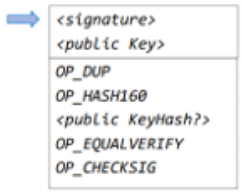
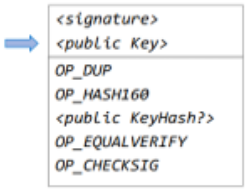
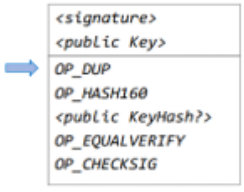
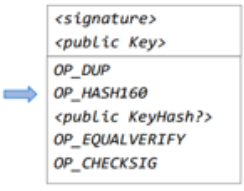
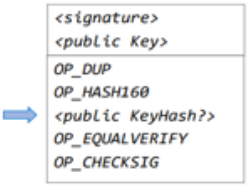
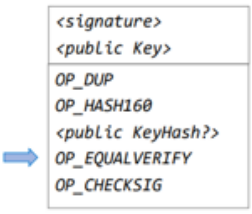
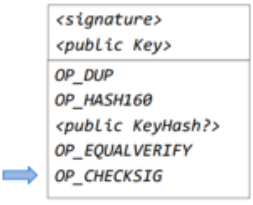
Bitcoin Script

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
		The script is executed line by line and the stack displays the items remaining after the execution of the line
<signature>		The signature is pushed onto the stack.
<public Key> <signature>		The public key is pushed onto the stack.
<public Key> <public Key> <signature>		OP_DUP duplicates the current top item on the stack. Therefore, is placed on top of the stack again.
<public KeyHash> <public Key> <signature>		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>		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>		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		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?

   > No. As we only see the transaction output, we only know the receiver of the transaction.

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

   > No. We do not know if a transaction exists which spends the output.

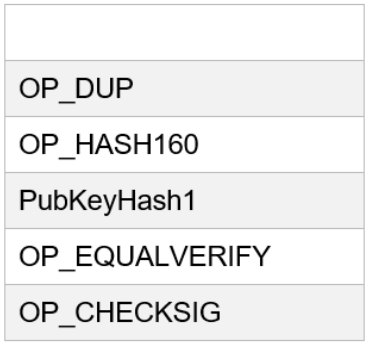
   (c) Can you tell which entity is allowed to spend this transaction output?

   > The owner of the private key which corresponds to the public key which corresponds to the hash 8a014...1de.

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

   > The public key of the hash 8a014...1de and the corresponding signature (therefore the private key).

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



The TxOut-script.

As input, you would provide the corresponding signature and public key. Out 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.

The TxOut-script.

In this case, the provided public key does not have to correspond to the public key hash. Therefore, the attacker could provide the required public key hash (equalverify checks if they are identical) and his own public key, which verifies the signature (generated with the private key matching the attacker's public key), stealing the funds.