

Blockchain Summative

Bradley Mackey

for 22nd March 2019

Please be aware some of the hashes character wrap in strange ways—do not miss the odd wrapped character on the next line.

Task 1 - Mining Puzzles

1. *User ID*: wbbz74
2. *Block hash target*: 000003e7fc18000
000
3. *Valid nonce*: 3856645
4. *Number of double hashes*: 3856645
Time taken: 103.41s

5. *Time to mine at initial difficulty of 1*
 Difficulty (D) 0.001 takes 103.41s
 $D = 1 \implies 103.41 \times \frac{1}{0.001} = 103,410$ seconds
 $\implies \frac{103410}{60} = 1,723.5$ minutes
 $\implies \frac{1723.5}{60} = 28.72$ hours.

Time to mine at peak 2018 difficulty of 7,454,968,648,263

$$\begin{aligned} D &= 7454968648263 \implies 103.41 \times \frac{7454968648263}{0.001} = 7.70918 \times 10^{14} \text{ seconds} \\ &\implies \frac{7.70918 \times 10^{14}}{60 \times 60 \times 24} = 8.92266 \times 10^9 \text{ days} \\ &\implies \frac{8.92266 \times 10^9}{365.25} = 24,428,927.04 \text{ years} \\ &\implies \frac{24428927.04}{1000} = 24,428.9 \text{ millenia.} \end{aligned}$$

6. *ECDSA Public Key:* 14afbb92502c9294f19be099ac3fe51f8ea1c943e36a06c43b096864d887145b55e87f1a01b1b9275bcc9d528a2829a774ec6de06dfaed72933ced851105f3ba
7. *Hello World Signature:* acd855318df6ebb70e4c956caad1c7df1a3395c2ead557e6ec304ced9038037aa83e79ab1bb80ca3b912ea2806c67cc387301f1530e730834bb3213cf55b70d6

8. *Signing Key (SK)*: f3fdb06bc3e08e4d97849c7a599d78d5991a629cd446ecef25f8ec7a80adc657
9. ***Hit Value***
 Effective balance (E) = 74
 Base target (T_B) = 1229782938247303 (from prev. block)
 Time since last block (t) = 60 (Nxt block time is 1 min)
 New Target = $E \times T_B \times t = 684ebf8cccccc24519a66e$
 Previous generation signature (G) = 9737957703d4eb54efdff91e15343266123c5f15aaf033292c9903015af817f1
 Signed $G = SK(G) = aa084cddc3d64a4425af1c1b6e4c41c0b9dd60176e41b7134bc3eb87de25f9411f83eddd7031f7048a47c5d0bfc4fdf268d6c7fd4eb41f72e65933ba8c453008$
 Hashed = $SHA256(SK(G)) = de9734e60820253cc47281d56b3e9c20d749c34f353e497000e8238eaa45cd55$
Hit Value = de9734e60820253c
10. *Time to forge new block*

Task 2 - Transactions & bitcoin-testnet

1. User ID: wbbz74
2. (a) <https://www.blockchain.com/btc/tx/cfe6cc5158f435f59c4daa24f66378ff56baf2980d04c92612e2adf222bb19b8>
 (b) <https://www.blockchain.com/btc/tx/348e8846eccc909c67eade94b3df0c84ab07133159b25759f4b3cac303904ec>
 (c) <https://www.blockchain.com/btc/tx/f3d7d00d0534fd7d59fb1cb4311dad4e42fef1b6174321342a9ed2af21d9bd25>
3. (a) This is a transaction with 2 inputs and 3 outputs. One of the outputs is a basic zero-value data transaction, making use of the `OP_RETURN` word to ensure the output can never be redeemed, placing some arbitrary data into the blockchain. The 2 inputs, as well as the 2 remaining outputs, use a Pay-to-Public-Key Hash (P2PKH) scheme for transferring coins—this can be identified as all the addresses begin with the number 1. The inputs prove to the blockchain they are in control of the private keys associated with the previous transaction that sent them the coins by providing a signature—derived from their private key—and their public key. At the end of execution of the script, if the signatures are valid, the stack terminates with `TRUE`. This allows the inputs to be sent successfully. The recipients, also using P2PKH, provide a script which will allow them to later redeem coins in a transaction block, given they are in possession of the private key for the recipient address. This script `PubKey` (the script used to lock the Bitcoins) is of the form: `OP_DUP OP_HASH160 hashedPublicKey OP_EQUALVERIFY OP_CHECKSIG`.

- (b) This is a simple transaction with 1 input and 2 outputs. The input uses a P2PKH scheme similar to the previous transaction. One of our outputs also uses P2PKH, but the other uses Pay-to-Script-Hash (P2SH)—which can be identified as the address begins with a 3. This differs from most other standard scripts available on the Bitcoin network as it allows for any arbitrarily complex script to take constant space on the blockchain, taking only 23 bytes. As the Bitcoin is being sent to a P2SH scheme, little effort is required, the sender only has to check the hash of what is otherwise a very long script (for example, this could be a large multi-sig transaction, but we are unaware of this). P2SH's scriptPubKey is of the (much shorter) form: `OP_HASH160 hashedScript OP_EQUAL`.
- (c) This is a transaction with 2 inputs and 3 outputs. One of the outputs is a zero-value data transaction. One of the inputs and outputs use P2PKH. This transaction also includes an input and output using Pay-to-Multisig (P2MS). These have the same requirement of requiring only 1 possible signature for 3 possible public keys, enabling 3 keyholders to exist, but only 1 of these is needed to authorise the sending of coins. Focusing on the input, we see the scriptPubKey is of the form: `OP_1 pubKey1 pubKey2 pubKey3 OP_3 OP_CHECKMULTISIG`. The scriptSig (script used to unlock the coins, to be sent in a transaction) is of the form: `OP_0 signature`. It is within this script we see the single signature required in order to send the previously locked Bitcoins. The reason for the `OP_0` before the signature is due to the well known bug in `OP_CHECKMULTISIG`, where the signature extraction variable (`OP_1` in our case) will consume 1 more input than stated. Therefore, we pad the stack with a dummy '`OP_0`' to prevent this function consuming data it is not supposed to.

4. *Bitcoin Testnet Address*: mjLjznCbyKuGJ5xuz7Wo1Es3qXHoxoDXgo

5. ***100 Satoshi Transaction***

TX ID: 74b5486e061ac680cde0f132b0dec6c5010d2dee8da3a2856d680fcf5bf41c37

Link: <https://chain.so/tx/BTCTEST/74b5486e061ac680cde0f132b0dec6c5010d2dee8da3a2856d680fcf5bf41c37>

6. ***Student ID Proof-of-Burn Transaction***

TX ID: bd1c2552fc0effda71e4e09137d8106aa6c67239dfba1e760040d1c78b66e0ac

Link: <https://chain.so/tx/BTCTEST/bd1c2552fc0effda71e4e09137d8106aa6c67239dfba1e760040d1c78b66e0ac>

7. ***Student ID Proof-of-Burn Script***

Script Hex: 6a067762627a3734

We can add data to the blockchain by immediately invalidating the script, allowing the remainder of the script to be interpreted as pure data. The first byte, `6a`, is the `OP_RETURN` word. This invalidates the script, such that any attempt to redeem any Bitcoins contained in this transaction would instantly fail, as per the semantics of Bitcoin Script (therefore this is a very bad script to use if actually sending bitcoins!). The next byte, `06`, is the number

of bytes that we will push onto the stack next—“wbbz74” is 6 characters long (6 bytes when ASCII encoded), so this is just 6. The remainder of the script, `7762627a3734`, is the ASCII encoded “wbbz74”, which will be interpreted on the blockchain as pure data.

Task 3 - A Wise Investment?

400-500 WORDS ABOUT WHICH IS THE MOST WISE INVESTMENT? Easy stuff.