Aristotle University of Thessaloniki

Data & Web Science MSc Program

Decentralized Technologies Course

Assignment 1, 2023 - Bitcoin

For this assignment you will need to implement two scripts using Python 3 and any additional library of your choice.

The first one will create a P2SH Bitcoin address where all funds sent to it should be locked until a specific time, specified either by block height, or UNIX Epoch time; other than the time locking the redeem script should be equivalent to P2PKH.

The second program will allow someone to spend all funds from this address.

Both programs should:

- use regtest
- assume a local Bitcoin regtest node is running

The first program should:

- accept a public (or optionally a private) key for the P2PKH part of the redeem script
- accept a future time expressed either in block height or in UNIX Epoch time
- display the P2SH address

The second program should:

- accept a future time, expressed either in block height or in UNIX Epoch time, and a private key (to recreate the redeem script as above and also use to unlock the P2PKH part)
- accept a P2SH address to get the funds from (the one created by the first script)
- check if the P2SH address has any UTXOs to get funds from
- accept a P2PKH address to send the funds to
- calculate the appropriate fees with respect to the size of the transaction
- send all funds that the P2SH address received to the P2PKH address provided
- display the raw unsigned transaction
- sign the transaction
- display the raw signed transaction
- display the transaction id
- verify that the transaction is valid and will be accepted by the Bitcoin nodes
- if the transaction is valid, send it to the blockchain

Notes:

- there is some repetition between the 2 programs; this is fine
- you may test your scripts by sending some funds to the P2SH address you created
- you may query the local Bitcoin regtest node using the JSON-RPC interface directly or through a library
- you may query an external API for the currently accepted fees/byte
- the P2SH address might have received funds from multiple transactions. Create an initial version of your script where it handles a single known transaction. Expand it to using multiple unknown transactions later.
- when dealing with multiple inputs, you will need to sign all of them
- you will submit a single compressed file (ZIP or TGZ) that contains the Python source code. It should include a text file with detailed instructions on how to run your programs
- Also include a requirements.txt file that will specify any extra Python libraries you have used. You can easily create such a file using the following command in your Python virtual environment:
  ```
  $ pip freeze > requirements.txt
  ```

- the source code is your main submission and it should contain everything you want to share. It should include detailed comments and everything else you think we should be aware of
- you are expected to manually construct the Bitcoin locking/unlocking script for the timelock transactions, using the appropriate OP_codes. If the programming libraries you are using have functionality to automatically create timelock transactions do not use them (it will be penalized)