Contents

This directory contains tools related to Signet, both for running a Signet yourself and for using one.

getcoins.py

A script to call a faucet to get Signet coins.

Syntax: `getcoins.py [-h|--help] [-c|--cmd=<bitcoin-cli path>] [-f|--faucet=<faucet URL>] [-a|--addr=<signet bech32 address>] [-p|--password=<faucet password>] [--] [<bitcoin-cli args>]`

- `--cmd` lets you customize the bitcoin-cli path. By default it will look for it in the PATH
- `--faucet` lets you specify which faucet to use; the faucet is assumed to be compatible with https://github.com/kallewoof/bitcoin-faucet
- `--addr` lets you specify a Signet address; by default, the address must be a bech32 address. This and `--cmd` above complement each other (i.e. you do not need `bitcoin-cli` if you use `--addr`)
- `--password` lets you specify a faucet password; this is handy if you are in a classroom and set up your own faucet for your students; (above faucet does not limit by IP when password is enabled)

If using the default network, invoking the script with no arguments should be sufficient under normal circumstances, but if multiple people are behind the same IP address, the faucet will by default only accept one claim per day. See `--password` above.

miner

You will first need to pick a difficulty target. Since signet chains are primarily protected by a signature rather than proof of work, there is no need to spend as much energy as possible mining, however you may wish to choose to spend more time than the absolute minimum. The calibrate subcommand can be used to pick a target appropriate for your hardware, eg:

```
cd src/
MINER="../contrib/signet/miner"
GRIND="./bitcoin-util grind"
$MINER calibrate --grind-cmd="$GRIND"
nbits=1e00f403 for 25s average mining time
```

It defaults to estimating an nbits value resulting in 25s average time to find a block, but the –seconds parameter can be used to pick a different target, or the –nbits parameter can be used to estimate how long it will take for a given difficulty.

To mine the first block in your custom chain, you can run:

```
CLI="./bitcoin-cli -conf=mysignet.conf"
ADDR=$($CLI -signet getnewaddress)
NBITS=1e00f403
$MINER --cli="$CLI" generate --grind-cmd="$GRIND" --address="$ADDR" --nbits=$NBITS
```

This will mine a single block with a backdated timestamp designed to allow 100 blocks to be mined as quickly as possible, so that it is possible to do transactions.

Adding the –ongoing parameter will then cause the signet miner to create blocks indefinitely. It will pick the time between blocks so that difficulty is adjusted to match the provided –nbits value.

```
$MINER --cli="$CLI" generate --grind-cmd="$GRIND" --address="$ADDR" --nbits=$NBITS --ongoing
```

Other options

The –debug and –quiet options are available to control how noisy the signet miner's output is. Note that the –debug, –quiet and –cli parameters must all appear before the subcommand (generate, calibrate, etc) if used.

Instead of specifying –ongoing, you can specify –max-blocks=N to mine N blocks and stop.

The –set-block-time option is available to manually move timestamps forward or backward (subject to the rules that blocktime must be greater than mediantime, and dates can't be more than two hours in the future). It can only be used when mining a single block (ie, not when using –ongoing or –max-blocks greater than 1).

Instead of using a single address, a ranged descriptor may be provided via the –descriptor parameter, with the reward for the block at height H being sent to the H'th address generated from the descriptor.

Instead of calculating a specific nbits value, –min-nbits can be specified instead, in which case the minimum signet difficulty will be targeted. Signet's minimum difficulty corresponds to –nbits=1e0377ae.

By default, the signet miner mines blocks at fixed intervals with minimal variation. If you want blocks to appear more randomly, as they do in mainnet, specify the –poisson option.

Using the –multiminer parameter allows mining to be distributed amongst multiple miners. For example, if you have 3 miners and want to share blocks between them, specify –multiminer=1/3 on one, –multiminer=2/3 on another, and –multiminer=3/3 on the last one. If you want one to do 10% of blocks and two others to do 45% each, –multiminer=1-10/100 on the first, and –multiminer=11-55 and –multiminer=56-100 on the others. Note that which miner mines which block is determined by the previous block hash, so occasional runs of one miner doing many blocks in a row is to be expected.

When –multiminer is used, if a miner is down and does not mine a block within five minutes of when it is due, the other miners will automatically act as redundant backups ensuring the chain does not halt. The –backup-delay parameter can be used to change how long a given miner waits, allowing one to be the primary backup (after five minutes) and another to be the secondary backup (after six minutes, eg).

The –standby-delay parameter can be used to make a backup miner that only mines if a block doesn't arrive on time. This can be combined with –multiminer if desired. Setting –standby-delay also prevents the first block from being mined immediately.

Advanced usage

The process generate follows internally is to get a block template, convert that into a PSBT, sign the PSBT, move the signature from the signed PSBT into the block template's coinbase, grind proof of work for the block, and then submit the block to the network.

These steps can instead be done explicitly:

```
$CLI -signet getblocktemplate '{"rules": ["signet","segwit"]}' |
  $MINER --cli="$CLI" genpsbt --address="$ADDR" |
  $CLI -signet -stdin walletprocesspsbt |
  jq -r .psbt |
  $MINER --cli="$CLI" solvepsbt --grind-cmd="$GRIND" |
  $CLI -signet -stdin submitblock
```

This is intended to allow you to replace part of the pipeline for further experimentation (eg, to sign the block with a hardware wallet).