The Meteor Feature and Forking the Project

Christopher Harrison

April 2020

Contents

1 Purpose of Document

This document is published as a reassurance that in the highly unlikely event of Electroneum Ltd ceasing to exist, the ETN stored on our distributed blockchain would still be accessible, *providing the owners possess the corresponding spend key(s)*. This would be the case both for an orderly closure, or another unexpected fatality such as a meteor as the title jokingly alludes to. Most cryptocurrencies rely on complete decentralisation to ensure continuity. We have instead chosen to deploy a system of centralised governance in which Electroneum Ltd is the sole central trusted authority (at this time, however that may change again in future releases). We have taken this stance as it allows us to enter into previously out of reach contracts with third parties (such as mobile network operators) and pursue recognition from the UK regulated financial environment. Because PoR is such a radical change and is bound to provoke concern and discussion about whether the system has in-built redundancies, we feel it's important to give our users the reassurance that our blockchain is, and will continue to be, distributed far and wide. We want to make it clear that anybody can run an anonymous node, meaning that the ETN ledger can persist beyond the destruction of Electroneum Ltd. To achieve this we've implemented a very an easy way for all blockchain-stored ETN to quickly and easily rise from the ashes of the theoretical meteorite and we've enabled our users to easily re-animate the Proof of Work protocol and turn ETN back into a fully decentralised cryptocurrency if need be. This document is to give fully testable assurance that whilst we have a centralised authority, we are unlike other centralised fintechs such as banking apps or remittance companies.

2 Governance and Moderation

2.1 A Short Introduction to Electroneum's Blockchain

Electroneum's blockchain is currently both permissioned and public. The permissioned component is the governance mechanism that over-arches the project, wherein a number of entities are sanctioned by Electroneum Ltd to progress the public ELectroneum ledger and fulfil transactions taking place within the network. This is achieved through a system of cryptographic block signatures applied on top of a blockchain/p2p layer which employs the Cryptonote consensus protocol. For their efforts, the block validators are imbursed with the native cryptocurrency (ETN), and the continued issuance of these rewards is contingent upon both their responsible behaviour within the network, and their humanitarian use of ETN. Electroneum Ltd has internal systems in place to monitor both of these things, and there exists blockchain functionality to revoke validator permissions immediately and *without external concensus* if necessary.

2.2 The Validator List

Electroneum's Validator List JSON can be found at at vl.electroneum.com and it contains a base64-encoded JSON object/list consisting of validator keys and mining permissions as well as the list timestamp, which is updated each time the list is updated. Each Validator key is generated by a Elliptic Curve Digital Signature Algorithm known as ED25519. The whole Validator List is also digitally signed three times using this algorithm, once with each of Electroneum's three master keys, in order to assure the information present on the above endpoint does come from Electroneum Ltd and cannot be modified by non-authorized users. The daemon verifies the authenticity of these signatures during startup, or when pulling a new list from the endpoint, before deciding that the list is genuine. The validator list JSON looks like this:


```
{
   "pubkeys":[
      "814A92F191735D989FFD3A2A7B33A2EE3ED6AD746B1530AED8E91E3B259DCD4B",
      "38BBE01388170750FAF8FE9B9C31DF6432987283F49171DA86039566C3288BF9",
      "8BC9D71CE4CD0DE0D50F45C8619257399B108D35C8CBB72FC29B38DFE3847769"
   ],
   "blob":"eyAidmFsaWRhdG9ycyI6WyB7InZhbGlkYXRpb25fcH
   VibGljX2tleSI6IjYxNzk3ODM4NjUxOEZGQ0UzMzMxRTk0RUN
   ....",
   "signatures":[
      "411B0E4A7605AAF0AEDB5D224CCEA7E0755884AE02FAFBD5098992AE7480971
      04268C684F76B7FBE9CBA15E5AC1F048B3950A6301B8E247A4E9C790DD1CD5E09",
      "0DA385DF9DEE9D8380EC95EC5B9597086F0EB2A9CE91C51E2595F23CC35DC3C90
      6F896EE052BF8B6FBAAB2E36DFE319A0B75396D8E9465993DF2DA9B49E86705",
      "F0C5DEA92AAB696143246DA5628C93A7EDF20BB5935B7A7220B6AA0AFD47E6FC5
      7E3847AD68C68ED7E8F39D9CB5C5606677D6959C31C01EC66F0CC4C41935806"
   ],
   "version":1
}
```


and the base64 decoded version of the blob gives the validator list:


```
{
   "validators":[
      {
         "validation_public_key":"617978386518FFCE3331E94ECB0811
         99F9CDF9796274A1CA2EC7E14DF8A1E0C2",
         "valid_from_height":0,
         "valid_to_height":0
      },
      {
         "validation_public_key":"147A3603C27F13E48E7AA3A8ED9A1B
         9B39CEB1C2FF8958554998278E9A87A4F0",
         "valid_from_height":0,
         "valid_to_height":0
```

```
    },
    {
       "validation_public_key":"0B35DA939714F3E697F14FEE11880F
       1BCE80D202220B9FAF990720E0EF9A1288",
       "valid_from_height":0,
       "valid_to_height":0
    },.....
  ],
  "list_timestamp":1559638396
}
```


3 The Meteor Feature

Whilst Electroneum Ltd is operating as a limited company, we do not intend to return to an old school decentralised PoW system. However, we have provided facility for our user base to autonomously revert to such a system should Electroneum Ltd dissolve, be 'hit by a meteorite' or meet some other unexpected and/or unlikely fatality.

3.1 Continuity, Disclaimers and a Few Assurances

3.1.1 The Wallet System

In the extremely unlikely event of a meteor event or the complete destruction or permanent shutdown of all of our servers, the funds within our app based custodial wallet system may not be recoverable. As is the case with virtually all other cryptocurrencies, we recommend keeping the bulk of your ETN outside of any custodial system (our wallet system/exchanges/third party wallets/etc). We provide a cold storage paper wallet system to facilitate this `https://my.electroneum.com/Electroneum_Offline_Wallet.zip` and users may also use command line wallets or hardware wallets. In the unlikely event of an orderly shutdown of our custodial wallet system, we would inform our users of a block height set at a sensibly distant point in the future before which users must withdraw their funds to a paper wallet or other storage location and/or provide direct access to the underlying private keys.

3.1.2 Blockchain

The blockchain will continue with V8 blockchain consensus rules minus the PoR features as we will see later, unless somebody clones the blockchain and codes in other features. To prevent a user building an alternative chain to 51% attack/overwrite wallet system withdrawals, we have coded a feature to allow users to specify a blockchain checkpoint to the wallet and daemon in the event of a shutdown which will ensure that their node only accepts blockchains that contain that block. The hash and height for which should be decided by the community amongst themselves. Users our groups with differing opinions

are thereby enabled to create their own unique forks with their own signature checkpoint should they wish to do so.

3.1.3 Mining

Anybody will be able to mine blocks. The mining algorithm will *not*be ASIC resistant.

3.1.4 Exchanges

In any eventuality, we will make sure that the exchanges are aware of this document, and the likely next steps of the community. We will not however, advise the exchanges to prefer or honour any specific fork(s).

3.2 How to Revert

3.3 Short Term Solution for Rapid Continuity

An Electroneum node is instantiated by running the electroneumd daemon program from the command line and allowing it to connect to the Electroneum network. When starting the program, there are a number of flags that can be passed program to switch on certain settings or preferences. The flag that invokes the meteor feature is –fallback-to-pow. The flags to specify a preferred checkpoint for your new fork are –fallback-to-pow-checkpoint-height and –fallback-to-pow-checkpoint-hash, and these two **must be passed when starting the wallet programs too**. That is, you would simply start the daemon by running

```
./electroneumd --fallback-to-pow --fallback to-pow-checkpoint-height
    <height> --fallback-to-pow-checkpoint-hash <hash>
```

and the wallet like

```
./electroneum-wallet-cli(or rpc) --fallback-to-pow-checkpoint-height
    <height> --fallback-to-pow-checkpoint-hash <hash>
```

These flags are primarily for a 'meteor' scenario, but in the case of an orderly shutdown, we will provide a final set of binaries with our own final checkpoint as an alternative option to using these flags, but users may choose to ignore these binaries and use the flags anyway.

For the non-technically minded, these steps are solely sufficient to fall back to a PoW based model in an emergency scenario. However if you're interested what's going on under the hood, more permanent solutions, or how you could instead duplicate PoR, please read on.

3.4 Under the Hood and Advanced Steps

3.4.1 The Code Behind the Fallback Flag

The daemon flag –fallback-to-pow will trigger the following setting in the daemon code at `src/cryptonote_core` 223:

```
const command_line::arg_descriptor<bool> arg_fallback_to_pow = {
  "fallback-to-pow"
  , "Disables all Validator feature and fallback consensus to standard
      Proof-of-Work (CryptoNote V1)."
    "This argument is a decentralization safety measure in case
        something happens with Electroneum Ltd"
    "so that users can fork the network to Proof of Work. (Anti Meteor
        Feature)."
    "***WARNING: IF YOU USE THIS ARGUMENT AND MINE BLOCKS AND LATER WISH
        TO RETURN TO THE TIP OF THE V8 *MODERATED* BLOCKCHAIN, YOU WILL
        HAVE TO MANUALLY POP BLOCKS BACK USING THE DAEMON (OR IMPORT)
        PROGRAM"
  , false
};
```

To see the effect of this we can examine `src/crypytonote_core/blockchain.cpp::handle_block_to_main_chain` 3723 onwards (and analogously in the function for handling blocks to alternative chains):

```
if(bl.major_version >= 8 && m_nettype != FAKECHAIN) {
  if(!m_fallback_to_pow) {
    if(!m_validators->isEnabled()) {
      m_validators->enable();
    }
    if(!m_validators->isValid()) {
      bvc.m_validator_list_update_failed = true;
      goto leave;
    }
    if(!verify_block_signature(bl) && !m_ignore_bsig) {
      MERROR_VER("Block with id: " << id << std::endl << " has wrong
          digital signature");
      bvc.m_verifivation_failed = true;
      goto leave;
    }
    if(bl.signatory == m_db->get_block(bl.prev_id).signatory &&
        !m_ignore_bsig){
      MERROR_VER("Block with id: " << id << std::endl << " has the same
          signatory as the previous block, which is not allowed");
      bvc.m_verifivation_failed = true;
      bvc.m_sequential_block = true;
      goto leave;
    }
  }
```

```
}
```

We can see that if this flag is set

a) the PoR permissioned validator list is no longer observed or respected

b) blocks are no longer required to have valid signatures to pass concensus

c) block ordering rules no longer need to be obeyed

That is, in using this flag, one can revert back to the original PoW system, where anybody with hashing power can mine blocks and no one entity holds absolute control over forward progression of the network. This new system is progressed by the subset of your peers who also chose to flick the switch.

3.4.2 Permanence of the New PoW Blockchain

If after triggering the PoW fallback, users want to make their new block-chain more permanent, customise their newly established blockchain with new code features, and isolate their chain from the old Electroneum network traffic, they could start by

a) forking or dupicating the blockchain codebase freely and legally at `https://www.github.com/electroneum/electroneum` and making this new version public,

b) changing the network id inside `src/cryptonote_config.h::config` 203:

```
boost::uuids::uuid const NETWORK_ID = { {
    0x04, 0xF8, 0x23, 0xE1, 0x66, 0xC2, 0xE3, 0xA4, 0xEA, 0x5D,
        0xD1, 0x2C, 0x85, 0x8E, 0xC8, 0x39
  } };
```

c) (optional) adding concensus features into blockchain.cpp and elsewhere, which are coded to begin at a new hard fork height defined at `src/cryptonote_core/blockchain.cpp::` 89:

```
static const struct {
  uint8_t version;
  uint64_t height;
  uint8_t threshold;
  time_t time;
} mainnet_hard_forks[] = {
  // version 1 from the start of the blockchain
  { 1, 1, 0, 1341378000 },
  { 6, 307500, 0, 1538815057 }, //1538815057
  { 7, 324500, 0, 1538985600 }, // Estimated July 5th, 8:30AM UTC
  { 8, 589169, 0, 1562547600 },
};
```

d) (optional) adding checkpoints to their code consisting of the hashes for blocks on their new chain, to delineate between both the beginning and the continuation of their personal fork and anyone elses. Checkpoints could, and should be added regularly anyway, for security reasons. This can be done in the code either via DNSSEC, or by hardcoding into the blockchain codebase. Hardcoding for example can be done at src/checkpoints/checkpoints.cpp 161:

```
      bool checkpoints::init_default_checkpoints(network_type
          nettype)
{
  if (nettype == MAINNET)
  {
    ADD_CHECKPOINT(1, "4536e1e23ff7179a126a7e61cd9e89ded
    0e258176f2bc879c999caa155f68cc3");
    ADD_CHECKPOINT(10, "e5aefcb1d575a788ecfb65bb7be3bdd1
    35eb76ccefb38a60d7800e86d25d408e");
    ADD_CHECKPOINT(100, "e3548600cc0e2991af4f36bbf44addd
    95051748fc09e8cac5f8237fd841132c0");
    ............
```

e) Specifying new seed servers(/nodes) at `src/p2p/net_node.inl` 447

```
        std::set<std::string>
            node_server<t_payload_net_handler>::get_seed_nodes
        (cryptonote::network_type nettype) const
  {
  .......
    else
    {
      full_addrs.insert("13.125.37.208:26967");
      full_addrs.insert("13.125.50.165:26967");
      full_addrs.insert("13.124.43.88:26967");
    .......
    }
.......
```

3.4.3 Duplicating PoR Instead

If after, or instead of, falling back to a PoW model, somebody wanted to replicate the v8 system of validators, they would be able to do so by following the aforementioned steps and then modifying the json for the list of allowed validators found at vl.electroneum.com, such that

a) The current Electroneum NGO validators (should you wish to prohibit them from mining) have a mining cut off point of the new blockchain governer's choosing (**do not delete the old miner entries, only update them**),

b) New validator keys are included for each of the future miners alongside a corresponding set of definite/indefinite mining cut off points,

c) The list timestamp is updated (this is a UNIX timestamp)

d) The Electroneum master keys ('pubkeys' in the JSON) are changed to new master keys and the old list signatures are replaced with signatures of the new list ('blob' in the JSON), generated by the new owner(s).

Users can easily generate master and validator keypairs using the single daemon command, `generate_ed25519_keypair`:

```
generate_ed25519_keypair
Private
    Key:E9CF1DBFC4C8B5978B0B3A0341E49CDACD3A1B6294EAFE4892F0C6260D528AC9
Public
    Key:14697CCD280E2A51B93DED4FCB2F62989B9529D33EF01B5B50B3594A5FB42324
```

and sign lists with the daemon command `sign_message`. The address of the validatator JSON host server must be pointed to at in `src/cryptonote_basic/validators/h::Validators` 108:

```
string endpoint_addr = "vl.electroneum.com";
string endpoint_port = "80";
```

and the new master keys must be assigned at `src/cryptonote_basic/validators.cpp` 50:

```
std::vector<std::string> mainnet_vl_publicKeys =
    {"814A92F191735D989FFD3A2A7B33A2EE3ED6AD746B1530AED8E91E3B259DCD4B",
 "38BBE01388170750FAF8FE9B9C31DF6432987283F49171DA86039566C3288BF9",
 "8BC9D71CE4CD0DE0D50F45C8619257399B108D35C8CBB72FC29B38DFE3847769"};
```