Using Superblocks to Bridge Dogecoin to Ethereum

July 30th, 2018
Revision: 2

Ismael Bejarano ismael.bejarano@coinfabrik.com
Oscar Guindzberg oscar.guindzberg@gmail.com

Abstract

The Doge->Eth bridge requires over 20k USD in gas (at 900 USD per ether) per day to keep up with the Doge blockchain. We propose a system where no Doge block headers are submitted to ethereum. Instead, all the headers from Doge blocks mined within the last hour are used to build a Merkle tree whose root is submitted to the Superblocks contract. Verifiers can start a challenge/response process to prove this root wrong.

Motivation

Our first approach to build a Dogecoin-Ethereum bridge was to port BTCRelay to Solidity. BTCRelay requires every new Bitcoin block header to be submitted to the contract in order to validate transactions belonging to a particular block.

For BTCRelay, storing an 80-byte block header costs around 200K gas. It needs to store 144 block headers per day. At a gas price of 10 gwei, this requires spending 0.288 ether per day in storage costs.

The success of Ethereum caused the costs to increase notably: at a valuation of 900 USD per ether, running BTCRelay for one day costs 259.2 USD. We could use a lower gas price, but the fees are higher with increased network usage and our transactions may take a long time to be mined.

Dogecoin generates one block per minute; even assuming the Doge->Eth bridge uses a challenge/response system for scrypt hash verification, this makes the cost of the Doge->Eth bridge ten times that of BTCRelay at the same block header size. In addition to this, Dogecoin is merge mined: block headers are larger, with an average size of 700 bytes, and this data requires extra validations, thus increasing the costs and making them prohibitive.

Superblocks

Inspired by the article Efficiently Bridging EVM Blockchains, one cost-saving alternative is to store not all block headers, but the root of the Merkle tree of several blocks.

We propose to just send and store a “superblock” that will represent a range of blocks. It will contain:

- The root hash of a Merkle tree built from the block hashes.
- The blocks’ accumulated difficulty.
- The hash of the last block of the superblock.
- The timestamp of the last block of the superblock.
- The hash of the parent superblock data.

Grouping several blocks into a superblock will minimize the costs. Instead of sending, validating and storing 60 700-byte blocks in one hour, we only need to store one superblock which will be less than 200 bytes.

There is a tradeoff -- more blocks per superblock means lower costs, but also longer times for relaying a transaction. The one hour per superblock is an arbitrary decision that will be reviewed when more data is available.

Almost no validations will be done on-chain on superblocks. Superblocks will be validated using a challenge/response system.

The hash of the previous superblock data is included in order to link superblocks; they can be considered a chain of superblocks. This facilitates confirming previous superblocks and ignoring superblocks from small forks that are considered attacks.

Relaying transactions

In order to receive Doge tokens, a user will not only need to send the Superblocks contract an SPV proof of the transaction that she sent to the lock address, but also an SPV proof that the block this transaction belongs to is part of a certain superblock.

Security Assumptions

We assume the basic components on which this solution is built (Dogecoin blockchain, Ethereum blockchain, Challenge/Response mechanisms and BtcRelay) are safe to use.

- There is an honest majority mining Dogecoin blockchain: no single miner or group of miners has enough hashing power to create an arbitrarily long fork of the main

Dogecoin blockchain. This discourages long range attacks, as any possible fork will have less accumulated difficulty than the main chain.
- There is an honest majority mining Ethereum blockchain.
- There are no huge chain reorganizations (e.g. over 100 blocks).
- No participant wants to lose money: the objective of an attack is to make a profit or to cause damage at a very low cost. No one is going to cause a small denial of service or another type of network disruption just to lose a lot of money.
- We will not try to solve attacks that are inherent to any blockchain, for example eclipse attacks.
- We assume challenge/response mechanisms work in general, e.g. we assume there will be always at least 1 honest validator online.
- Dogecoin will neither change its PoW algorithm nor migrate to PoS.
- Dogecoin will neither change the header format nor the transaction format.

Validation

We will submit 1 superblock per hour (estimated to represent 60 Dogecoin blocks). This is an arbitrary decision subject to further tests.

To avoid dealing with chain reorganizations, the superblocks won't be accepted by the contract until 3 hours have passed since their last block was mined; for example, a superblock sent at 6pm will contain blocks from the [2pm-3pm] interval.

The 3-hour delay to accept a superblock is to minimize the probability a submitter sending a small fork by mistake; it is an arbitrary value subject to more evaluations.

We will treat superblocks not containing exactly the main chain blocks for that period as an attack, since the probability of someone submitting small forks by mistake after 3 hours is negligible.

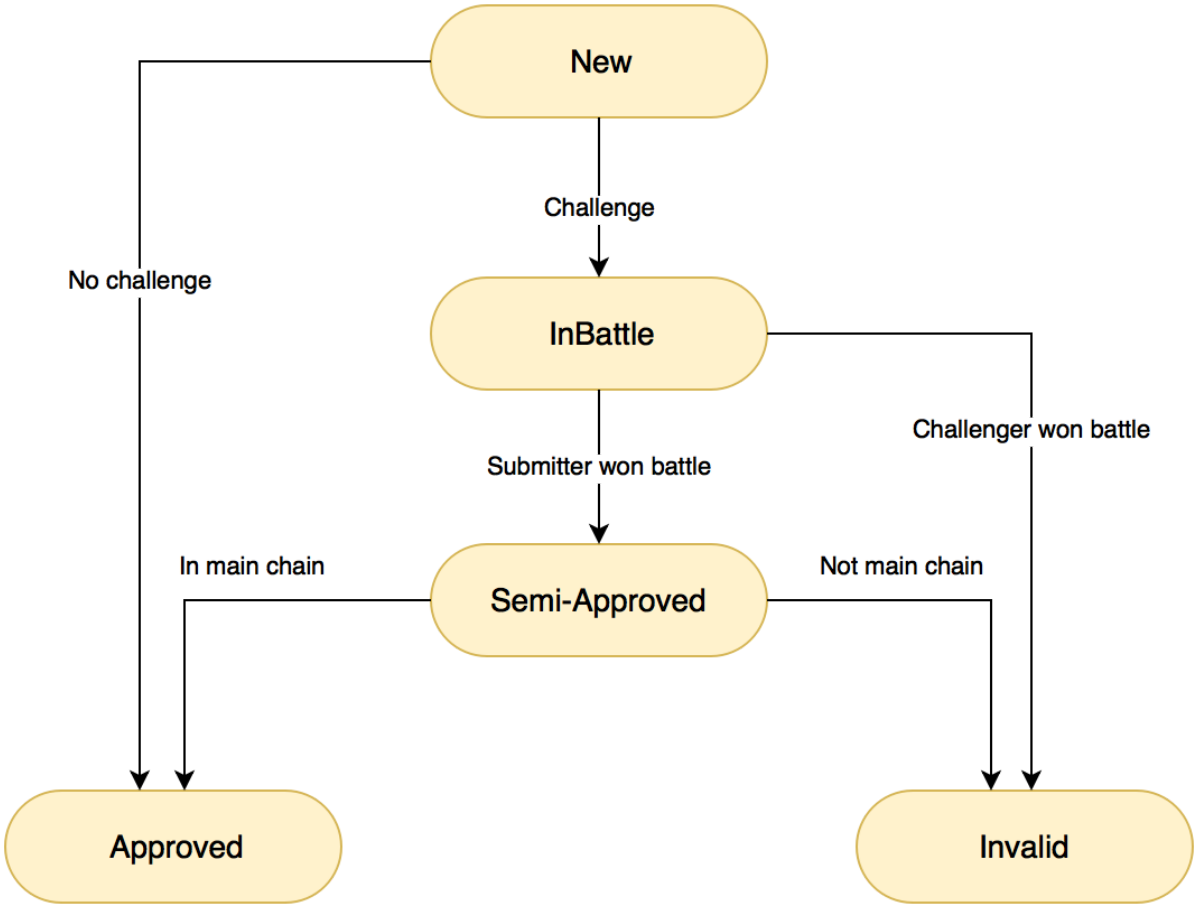
The validation of the superblocks will be done using a challenge-response protocol. Both submitter and challenger will have to make a deposit in ether so as to disincentivize fake submissions. The winner of the challenge will get the loser's deposit.

When a submitter wins the challenge, she will not recover the deposits immediately but only after the superblock is confirmed by enough future superblocks. (See "Superblock with blocks not in the main chain" attack below).

Superblock states

- **New**: the superblock has just arrived, verifiers can challenge it.
- **InBattle**: a verifier has done a challenge on it, the battle has not finished yet.
- **Semi-Approved**: the submitter won the battle, but we are waiting for another superblock to confirm it.
- **Approved** (Final state): the superblock is valid.

- **Invalid** (Final state): the submitter lost the battle.



Superblock verification battle

The submitter and the challenger will take turns to send messages to the contracts to advance the battle. A failure to reply in a timely manner will be considered as the forfeiture of the battle and the counterpart will be declared as the winner.

1. Submitter: makes a deposit.
2. Submitter: sends a superblock, deposit is locked. Superblock status is **New**.
3. Challenger: makes a deposit.
4. Challenger: sends a challenge, deposit is locked. Superblock status is **InBattle**.
5. Submitter: sends array with all the block hashes in the superblock (that are used to form the Merkle tree that relates them to superblock root hash). The block hashes, Merkle tree and root hash combination is validated on-chain.
6. Challenger: requests a block header.
7. Submitter: sends the requested block header and its scrypt hash. The block header is verified on-chain.

8. TrueBit interactive scrypt hash verification off-chain.
9. Steps 6, 7 and 8 can be executed again until all blocks have been sent.
10. If all blocks have been sent:
    a. verify that the blocks' on-chain accumulated difficulty matches the superblock's accumulated difficulty.
    b. verify on-chain that the superblock was in fact submitted after 3 hours
11. If all steps were performed successfully or all challengers dropped from the battle the superblock is marked as **Semi-Approved**.
12. If the submitter failed, the superblock is marked as **Invalid**.

If the submitter loses the battle, then the superblock will be considered invalid immediately and the challenger will be awarded the submitter's deposit.

If the challenger abandons, a new challenger can continue the challenge after making a deposit of her own.

If no challenger won the battle the superblock will be considered semi-approved, but the deposits will be locked until it is confirmed by the following superblocks to be in the main chain.

A semi-approved superblock is considered to be in the main chain if new superblocks arrive that have the superblock as ancestor and they have enough accumulated proof of work to beat other proposed superblocks.

If the superblock is not in the main chain the submitter's deposit will be paid to the challenger. In the case of multiple challengers it will be split between all the challengers proportionally to their deposit.

Possible attacks

Superblock submitted before 3 hours have passed

We validate using the timestamp from the last block of the superblock. One of the reasons is to minimize the possibility of a small fork causes a honest submitter to send an invalid superblock.

This requirements also prevent when an attacker creates an arbitrarily long fork mining blocks with a timestamp in the future.

Attacker submits superblocks while all verifiers are offline

If all verifiers are offline for a short period of time, an attacker might submit several fake superblocks during that period.

In order to reduce the odds of success of this attack, no more than one valid superblock will be accepted every 30 minutes. This means that at least 30 minutes must have passed between a parent and a child.

This minimum time between superblocks should also discourage long range attacks. Someone can mine an arbitrarily long chain from a block in the past, but submitting will take a long time and the main chain should also grow at the same time.

Superblock with blocks not in the main chain

One possible attack is to send a superblock that is built in such a way that all the blocks are valid Dogecoin blocks but some of them are not in the current main chain; for example, the last block of the superblock could be an orphaned block mined by the attacker.

If the superblock contains blocks from a temporary fork, it does not seem possible to challenge it successfully, because all the data will be a valid part of the Doge blockchain (regardless of whether it is part of the Doge main chain).

The attacker might even keep sending “fake” superblocks on top of the attack superblock to keep her “fake” chain growing. Each fake superblock may contain just 1 orphaned Doge block mined by the attacker. The cost of mining 1 orphan Doge block per hour is relatively low.

If a superblock is challenged and the submitter (i.e. the attacker) wins the battle, the superblock will be considered “semi-approved”. Deposits will be held and it will not yet be possible to use the superblock to relay transactions, but superblocks on top of it will be accepted.

Assuming the legit “superblock” chain keeps growing, after 24 superblocks (i.e. after 24 hours), the challenger can request to get her deposit back and the “semi-approved” superblock to be considered “invalid” because it is not part of the superblock mainchain. On the other hand, the submitter can request to get her deposit back and the “semi-approved” superblock to be considered “approved” if it IS part of the superblock mainchain.

Questions and Answers

- How are the deadlines set for each stage?

  The deadlines for each stage are not defined yet. There's a trade-off -- with higher timeouts it’s easier to cause a denial of service by always replying in the last second, a lower timeout can cause an honest participant to miss replying in time.

- Do the deadlines increase on every battle turn?

Yes, the deadline is reset for every battle turn.

- Are the deadlines fixed?

  There's no max time for the battle to resolve.

- What about new superblocks that arrive when a battle is being held?

  If a superblock arrives and its parent is in battle, then the new superblock is rejected. Otherwise it is accepted and marked as "New".

  For example, an attacker can challenge an honest superblock and send a competing fake superblock at the same time. It is supposed an honest challenger to challenge the fake superblock. The assumption is that the fake superblock will eventually lose the battle and the honest superblock will win.

Fine tuning

Dealing with block timestamps going backwards

A Doge block’s timestamp is not necessarily higher than its parent block’s timestamp (it just has to be higher than the median timestamp of recent blocks).
Therefore, two valid competing superblocks could potentially coexist, leading to two valid competing superblock chains.
Consider this Doge blockchain: B1 (16:59:58), B2 (17:00:01), B3 (16:59:59), B4 (17:00:02), B5 (18:00:01).
Then:
Superblock chain A: S1 (B1), S2 (B2, B3, B4), S3(B5)
Superblock chain B: S1 (B1, B2, B3), S2 (B4), S3(B5)
Both superblock chains have the same accumulated PoW, so they will compete forever.
To avoid that problem, as benevolent dictators, we establish the rule that superblock chain A is valid and chain B is invalid.
For example, if we are processing the superblock for blocks between 4pm and 5pm, no blocks are allowed with a timestamp after 5pm.
Note: The superblock might contain (most likely in the first couple of blocks) blocks whose timestamp is before 4pm.

Dealing with long periods without blocks

In the unlikely event that no blocks are produced during an entire hour, the superblock for that hour will be skipped.
Consider this Doge blockchain:
B1 (16:30:00), B2 (16:45:00), B3 (19:20:00).

The Superblock chain should be:
S1 (B1, B2), S2 (B3).

Conclusion and next steps

The presented solution appears to be a viable option to minimize gas usage when building a Doge to Ethereum peg. Our next steps are building a prototype and investigating further possible attacks.

Acknowledgements

Many thanks to Jason Teutsch, Sina Habibian and Sergio Lerner for providing feedback. And Catalina Juarros for proofreading and editing.

References

- “Efficiently Bridging EVM Blockchains”, https://blog.gridplus.io/efficiently-bridging-evm-blockchains-8421504e9ced
- “A bridge between the Bitcoin blockchain & Ethereum smart contracts”, http://btcrelay.org/
- “Reduce gas usage, in particular when adding doge block headers”, https://github.com/dogethereum/dogethereum-contracts/issues/16
- Scrypt interactive verification source code https://github.com/TrueBitFoundation/scrypt-interactive
- Doge<->Eth bridge documentation https://github.com/dogethereum/docs