BUIP135: Use OKCoin Donation to Fund Doublespend Proofs

Submitted by: Andrew Stone
Date: 15 Oct 2019

Recently Bitcoin Unlimited received approximately 9000USD from OKcoin to encourage BCH technological development. We want to put this generous funding to work straight away to benefit BCH and all its implementations. Voting YES for this BUIP will devote this money towards producing an implementation of DS (doublespend) proofs that:

1. Is enabled on Bitcoin Unlimited
2. Is MIT licensed and therefore available for adoption by any other full node implementation

In alignment with the BU goal to recognize and foster independent development, we propose to first offer this money to and work with the Flowee project. Flowee has implemented a GPLv3 licensed implementation of doublespend proofs, and may be willing to extract this portion of the Flowee code, offer it with an MIT license, and make some small proposed modifications (see technical details below).

If an agreement with Flowee cannot be reached, the Developer will evaluate proposals from any other applicants to find an alternative solution to providing this functionality.

Doublespend Proof Technical Details

There are currently two specifications on double spend proofs and one implementation within the larger BCH ecosystem.

Imaginary Username:
https://github.com/imaginaryusername/specs_n_stuff/blob/master/dsproof/dsproof.md

Flowee The Hub:
https://gitlab.com/snippets/1883331

Initial Considerations

One key difference in the specifications is in how the DS proofs are announced. Flowee specifies the DS proof identifier to be the double SHA256 of the entire doublespend proof. Imaginary Username specifies the DS proof identifier to be the SHA256 of the actual double spent outpoint.

Flowee's DS proof identifier is a probabilistically unique identifier, similar to block and transaction identifiers. Imaginary Username's identifier has an additional property, which is that different DS proofs that prove the same thing have the same hash. This is a very useful property for DS proofs to have, since nodes are only interested in what they prove, not the specifics of how (apart from verification). Imaginary Username's formulation allows nodes to ignore DS proof messages based on the "INV" content, without actually downloading the proof.

However, we can do slightly better. The purpose of the hashing both specs propose is to create a probabilistically unique and short identifier. However, the data being hashed -- the outpoint -- consists of the double SHA256 hash of the prior input transaction and a 32bit index. This data is already probabilistically unique, so there's no need to hash it again. Instead, let's use the outpoint itself as the DS proof identifier. This allows more efficient handling of DS proof and DS proof INV messages (no hashing).

But probably most importantly, it allows recipients to easily determine whether the DS proof is relevant to any transactions that the recipient is interested in, when the INV is received. The recipient simply compares the announced data to the inputs of all its transactions looking for a match. If there is no match, a wallet does not need to request the full DS proof.

Secondary Considerations

Both specifications place DS proof announcements into INV messages along with any other object announcements. However, this means that the DS proof will be queued with the same priority as other transactions -- potentially waiting for the processing of the very transactions it’s trying to warn against.

Instead DS proof announcement handling should happen at higher priority than transaction announcements. While it would be possible to scan INV messages for DS proof announcements and extract them, this is inefficient and inconvenient.

A better solution is to place DS proof announcements in a separate message. These messages can then be isolated to a high priority handling queue immediately upon receipt, analogous to block and block headers message handling today.

Making a new message type is not hard. And the DS specification already requires new messages to handle transmission of DS proofs, so including another message type is a difference of degree, not kind.