SpreadCoin

October 5, 2014

Introduction

In proof-of-work cryptocurrencies new coins are generated by the network through the process of mining. One of the purposes of mining is to protect network from double spending attacks and history rewriting. Miners generate new blocks and check contents of the blocks generated by other peers for conformation to the network rules. However, many miners now delegate all the checking work crucial to cryptocurrency security to pools. This means that pool operators do not have any large hashing power but have control over generation of new blocks. This brings unnecessary centralization to otherwise decentralized system. Controlling more than 50% of mining power allows to perform double-spending attacks with 100% chance of success but even with less than 50% control it is possible to perform attacks which have chances to succeed[1]. The core idea of SpreadCoin is to prevent creation of pools and thus make mining more decentralized and the whole system more secure.

Pool Prevention

In pooled mining miners perform only the work which is necessary to fulfill the proof-of-work requirements and pools take care of block generation and broadcasting and distribute reward among miners according to the shares they submit.

In this scheme miner has two alternatives:

1. Solo mining. In this case miner cannot send shares to the pool because they will not be accepted.
2. Pooled mining. Miner's shares will be accepted by the pool but in the case miner will actually generate a new block its reward will go to the pool which will redistribute it to all miners.

This allows organization of pools because miners has no way to cheat and steal generated money. To prevent creation of pools we must remove this possibility so that if pool will be created than miner can mine in a pool, submit shares as usual and get reward for them but in the case of actually finding a block miner can send it directly to the network instead of the pool and get full reward for it.

In SpreadCoin mining is organized in such way that miner must know the following things:

1. Private key corresponding to the coinbase transaction.
2. Whole block, not only its header.

This ensures that miner can broadcast mined block and spend coins generated in that block. It may seem that it is necessary to know only the private key to spend coinbase transaction. If two conflicting transactions will appear on the network then the one that was broadcasted first will have much higher probability to be included in a block because each peer remembers and retransmits only the first one of the conflicting transactions. If both miner and pool know private key but only pool knows the content of the block than pool can generate and broadcast spending transaction earlier than miner. If both miner

[1] Double-spending. *Bitcoin Wiki.* https://en.bitcoin.it/wiki/Double-spending

and pool know content of the block than miner will be the first one who can broadcast block and spending transaction.

To prove knowledge of the private key and whole block there are two new fields in the block header: `MinerSignature` and `hashWholeBlock`.

`MinerSignature` is a digital signature of all fields of the block header except for the `hashWholeBlock`. Changing any information in the block requires regeneration of this signature which means that it is necessary to recalculate it during each iteration of the mining process. This implies that miner must be able to sign any arbitrary data.

`hashWholeBlock` is a SHA-256 hash of the block data arranged as follows:

Padding ensures that there is no incentive to mine empty blocks without transactions. Padding values are computed using simple algorithm which initializes last 32 bytes (8 `uint32`) with `hashPrevBlock` and then goes backward and computes remaining `uint32` values using the following recursive formula: $I_i = I_{i+3} \cdot I_{i+7}$. This algorithm ensures that there is no efficient way to compute padding values on the fly during hash computation which otherwise could potentially give some advantage to mine empty blocks in certain computing environments.

It is important that block is hashed twice. If it was hashed only once then pool could hash the beginning of the block and send resulting hash state to the miners. Each miner would then modify some information in the end of the block and recalculate the hash based on the known state without actual knowledge about what is contained in the beginning of the block. Appending block data to itself make it necessary to know the whole block to recalculate `hashWholeBlock`.

Pool may detect and ban cheating miners. However, many miners may still prefer to cheat so that pool will be completely unusable for honest miners.

Miners that have low probability of finding a block will get more profit by stealing reward for accidentally found block even if pool will ban them thereafter.

Miners that have enough mining power to find blocks consistently can still connect to a pool and submit shares for some time but steal the first found block. This way they can get both reward for their shares and the actual mined block.

Given all this it is expected that no one will create a pool. But even if someone will than it can be countered by releasing stealing miner software which many miners will switch to.

Compact Transactions

SpreadCoin as well as Bitcoin uses ECDSA signatures. Each address in Bitcoin is a hash of an ECDSA public key. To spend coins sent to an address it is necessary to provide public key matching to that hash and a signature. This results in 139 or 107 bytes for each transaction input script (`scriptSig`) depending on whether compact public key is used. However, it is possible to recover public key from the signature[2] which means that it is not necessary to provide it in transaction input. Together with using compact representation of the signature[3] it allows to reduce size of transaction input script from 139 or 107 bytes in Bitcoin to 67 bytes in SpreadCoin. Recovering public key has almost no extra CPU cost compared to the usual signature verification process used in Bitcoin. This is important because the CPU cost of ECDSA signature verification is a bottleneck for Bitcoin transaction processing.

Usual output script (`scriptPubKey`) in Bitcoin looks as follows:

```
OP_DUP OP_HASH160 5bd18804e4bb43a4bb8b6bc88408970bafaf4a38 OP_EQUALVERIFY OP_CHECKSIG
```

In SpreadCoin the semantics of the `OP_CHECKSIG` instruction was changed to checking signature by hash of the public key (it recovers public key and compares its hash with the provided one). This results in a much simpler script in SpreadCoin:

```
5bd18804e4bb43a4bb8b6bc88408970bafaf4a38 OP_CHECKSIG
```

This results in additional minor space saving because this script is 3 bytes smaller.

Smooth Supply

Block reward in Bitcoin is computed using the following formula:

$$R_h = R_0 \cdot 2^{-\left\lfloor \frac{h}{p} \right\rfloor},$$

where

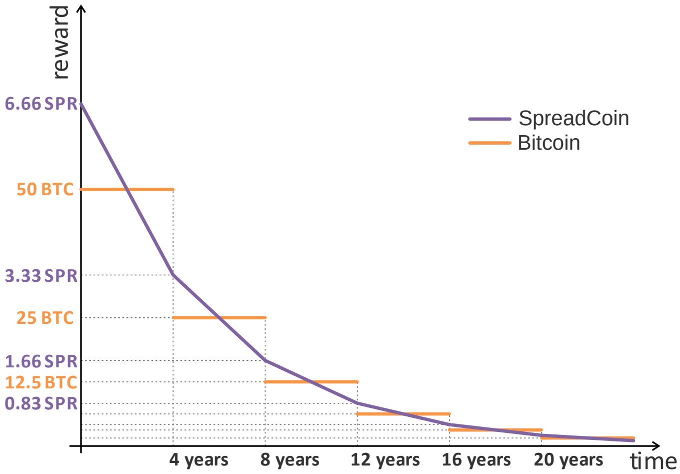
h – block height,

p – reward halving period,

R_0 – initial reward,

R_h – reward for block h,

$\lfloor \quad \rfloor$ – floor function.



This method results in abrupt reward changes near halving points. SpreadCoin uses simple linear interpolation between halving points to make reward decrease much smother. This is achieved by modifying reward using the following formula:

$$R'_h = \frac{4}{3}\left(R_h - R_h \cdot \frac{h \bmod p}{2p}\right).$$

SpreadCoin uses $p = 2 \cdot 10^6$ as its reward halving period.

[2] ECDSA Signatures allow recovery of the public key. *Bitcoin Forum.* https://bitcointalk.org/?topic=6430.0%29%3F

[3] Why the signature is always 65 (1+32+32) bytes long? *Bitcoin Stack Exchange.* https://bitcoin.stackexchange.com/questions/12554/why-the-signature-is-always-65-13232-bytes-long