Bitcoin 2: Freedom of Transaction

Sid Angeles and Eric Gonzalez

sid.angeles@cryptogroup.net
AF76 05E5 EB15 2B00 AB18 204C 11C9 1C8B EA6C 7389

eric.gonzalez@cryptogroup.net
3A85 7ABC 9453 96E4 B9DB 2A11 7370 E15E 8F60 2788

Donations: **1btctwo**jvohHSLaXaJAHWebZF2RS8n1NB

Revision 2

July 23, 2013

Abstract

We propose a set of changes to the original Bitcoin protocol (called Bitcoin 2) that allows Bitcoin to evolve into a system that is future-proof against developing threats to its original vision – an alternative, decentralized payment system which allows censorship-resistant, irreversible transactions.

Bitcoin 2 strives to be a minimal set that lays the foundations for a long-lived system capable of delivering the original Bitcoin vision, with enough room for growth to layer additional improvements on top of it. These changes include a sliding blockchain with fixed block sizes, the redistribution of dead coins with an unforgeable lottery, enforced mixing, and miner ostracism.

The proposed changes require a proactive fork of the original Bitcoin block chain, but they allow final transfers of existing coins into Bitcoin 2 and a reuse of the existing Bitcoin infrastructure.

1 Introduction

The Bitcoin peer-to-peer electronic cash system [11] was introduced in 2009 and it proved to be a remarkable piece of work which found widespread adoption. The system was started with the explicit goal of providing a completely alternative payment system without a single point of failure that allows anonymous, but not untraceable, transactions.

However, certain shortcomings in the original protocol and unforeseen developments inside and outside of the growing Bitcoin ecosystem threaten these very goals and there is good reason to believe that Bitcoin could develop into a system that is a complete perversion of the original vision – a completely transparent payment system with very few points of control which has been totally absorbed by the established financial and regulatory environment. In such a scenario, the development of alternative digital currencies would be postponed for at least several years and such alternatives would have to compete with a dominating player in this previously open space.

Therefore, we propose Bitcoin 2, an evolution of the original Bitcoin protocol which would allow to deliver the promises of the original vision: a payment system with the necessary economic incentives for sufficient decentralization, protection against external attempts to trace all transactions, and enough room for further innovation on top of it.

Bitcoin 2 would fork the original blockchain, but it would allow further use of the developed Bitcoin 1 infrastructure with minimal changes. Existing coins from Bitcoin 1 could be transferred to Bitcoin 2 piecemeal, allowing a gradual switch with minor disruptions of the Bitcoin economy.

2 Current Threats to Bitcoin

Bitcoin Payment Messages & Transaction Metadata

In practically all jurisdictions the tax man requires merchants to issue receipts for and register cash transactions to make sure that he gets his cut. In the current Bitcoin protocol it is not possible to issue such receipts.

The proposed Bitcoin Payment Messages [1], which are scheduled to appear in the 0.9 release of Bitcoin-Qt [7], would allow to issue so-called payment requests (which are digitally signed by the merchant with a X.509 certificate, a public-key infrastructure with *central* certificate authorities). If a customer pays such a request the resulting transaction would commit the payment request to the blockchain (for this reason Bitcoin Transaction Metadata has been proposed [2]).

That means that with the introduction of Bitcoin Payment Messages it becomes possible to issue receipts for Bitcoin transactions and therefore merchants in most countries are *required* to do so, even with *current* regulations.

Of course, the reason for the introduction of Bitcoin Payment Messages and Transaction Metadata given by Gavin Andresen is a different one.

Know-Your-Customer (KYC) Scoring

The widespread use of Bitcoin Payment Messages would identify all fiat money flowing into Bitcoin (a customer buys Bitcoin at an exchange) and out of Bitcoin (a merchant receives Bitcoin for a payment request). The Bitcoin transactions that reach merchants can usually be traced back to the exchanger where they have been bought and the exchanger has to know the customer by name (to comply with KYC regulations). Software to analyze financial flows for money laundering patterns in the traditional banking system is widely available and it could be adapted to Bitcoin easily, which would make it harder to conceal ones identity. If this identification of customers becomes too cumbersome it could be made even easier, either by:

- Making payment requests mandatory for exchange customers. That is, customers would need a *registered* address to buy Bitcoin.
- Or, more likely, KYC scoring.

KYC scoring is the adaption of credit card fraud detection algorithms to customer identification. That is, a merchant would be forced by regulations to score all his incoming transactions for KYC compliance. If the score is high enough (i.e., the customer can be identified with a high enough probability) the merchant can process the transaction. If the score is too low (i.e., the customer cannot be identified with a high enough probability) the merchant is required to decline the transaction. KYC

scoring would make Bitcoin mixing services (e.g., [3, 4]) practically useless, because after using such a service you could not spend your Bitcoin with most merchants anymore.

Whichever route is taken, the end result is the same: The end of anonymous payments with Bitcoin.

Repudiable Transactions

One of the main features of Bitcoin are non-repudiable transactions: Once a transaction has been confirmed in the blockchain there is no way to reverse it. A feature that significantly reduces the risk of fraud for merchants, but that is contrary to the way wire transfers and credit cards work today (both allow reversals or chargebacks). Some people believe that repudiablity is necessary to get regulatory compliance for Bitcoin.

Repudiablity could be build directly into the Bitcoin protocol. Or the Transaction Metadata [2] (discussed above) could be used to add third parties to Bitcoin payments (in addition to the merchant and the customer) that would allow to repudiate a payment. Of course, these third parties would require their customers to identify themselves.

In any case, repudiablity would push more towards identified *customers*, because to protect himself against chargebacks the merchants have to identify their customers, and the practical anonymity of Bitcoin would erode further.

3 The Philosophy and Goals of Bitcoin 2

Bitcoin 2 aims for the following goals:

- censorship resistance (resilience)
- continuous decentralization
- better security
- better economic incentives

Our design attempts to keep the changes to the Bitcoin protocol as minimal as possible *while* meeting the stated goals. Although the proposed changes make the protocol quite complex we remain strong opponents of feature creep. Additional features should be implemented on top of the proposed protocol and not as extensions to it.

Therefore, Bitcoin 2 does **not** intend to provide:

- micropayments
- instant confirmations
- untraceable or offline digital cash

In our opinion these very important objectives are best implemented in overlay currencies (digital payment systems implemented *on top* of Bitcoin 2, for example Open Transactions [6] or Voucher Safe [9]).

We propose the following changes to the original Bitcoin protocol (justifications and technical details given below):

- Sliding blockchain (with **fixed** block size) with auctioning of inclusion enabled by "maxblock" (include transactions before block y).
- Distribution of dead coins with pseudo-random lottery.
- Forced mixing of coins with Zerocoin [10] every n blocks.
- Miner ostracism.
- Final transfer of coins from Bitcoin 1 to Bitcoin 2 with devalidating transactions (a time-limited offer).

4 Economics

The current Bitcoin protocol has some economic properties which are problematic for the stated philosophical goals.

The infinitely growing blockchain makes full clients more and more infeasible, leading to centralization. Suggestions like the Simplified Payment Verification from the original Bitcoin paper [11] do not remedy this problem, because they lead to *thin* clients – clients that rely on full network nodes for their verification.

Furthermore, the current mining incentives cause an increasing specialization and growth of mining pools, thereby introducing single points of control.

Another property is that coins which have not been moved for a long time can either be dead (the corresponding private key has been lost) or just appear to be so (representing long-term savings).

Although the deflation caused by dead coins is not problematic from the perspective of economics, they lead to an unknown size of the monetary base (from the outside it cannot be determined which coins are truly dead and which are mere savings). The resulting unknown size of the monetary base leads to economic insecurities and poses the risk of disruptions of the Bitcoin economy by large dumps of apparently dead coins.

Sticking to a fixed block size **forever** incentivizes overlay currencies, because with a growing number of transactions it makes economic sense to move smaller (and therefore relatively expensive) transactions off-chain into overlay currencies.

5 Technical Goals

(P1: Necessary, P2: Preferable, P3: Convenient)

Goal 1: Security

- P1: Keep it simple, stupid. Battle feature creep.
- P1: Do **not** add new features unless they are critical for the *existence* of the system.
- P1: Have a good algorithm migration implementation in protocol.
- P2: Embrace a good code base: btcd [5]. The btcd code base is much easier to review for security flaws and much more accessible to new developers than bitcoind.

Goal 2: Keep the blockchain small

Helps decentralization and reduces money supply uncertainties. Suggests use of overlays/off-chain systems.

- P1: Fixed, small block size limit (150KB size):
  - P2: Limit the number of transactions: Transactions sorted by fee with provision to allow timely auctioning rounds by terminating transactions early.
  - P2: Allow for random selection as well, inclusion by fee is not mandatory.
- P1: Expire unspent transactions after a set number of blocks (see implementation note 1).
  - P2: Reflow money via lottery (see implementation note 2).

Goal 3: Make Bitcoin resilient technically

Resilience against suppression and censorship: Make Bitcoin resistant against regulatory attack.

- P1: Support for nodes and complete network on I2P, Tor, sneakernet [8].
  - Incoming: Communicate peers a user-configurable address (i2p/tor/ipv4/ipv6).
  - Outgoing: Keep current SOCKS5 support.
- P1: Forced Zerocoin mixing to prevent scoring attacks (see implementation note 3).

Goal 4: Enforce decentralization of mining and code

- P1: Add a decentralization incentive to the miner community (see implementation note 4)
- P2: Write and publish a good standards paper (an RFC).
- P2: Miner ostracism: Allow clients to exclude miners from adding the client's transaction (see implementation note 4).
- P3: Include versioning in blocks and transactions, with forward and backward compatibility. After network has 80% "new protocol" ability (as shown in blocks and transactions) select new code. Implement updates as parallel implementation to prevent unpredicted dependencies.

6 Implementation notes

Note 1: Sliding blockchain

Blocks over a certain age (preceding N other blocks) become invalid for transactions. Only the block header remains and is used for long term blockchain security. No lookups into these old blocks are performed. Unspent transactions in these blocks become unspendable. The sum of unspent transaction amounts of the oldest block is reclaimed by distributing the coins with a build-in lottery (see note 2). To allow "pre-renewal" of transactions to stay in the sliding window: All transactions **must** include the setting "last-block" and may include "locktime" and "max-block". Last-block is the hash of the last block known to the client. Locktime is the minimum number of blocks after last-block that must pass

before the transaction may be included into a new block. It is already implemented in the current code. Last-block makes every transaction sign the blockchain. Min-block allows pre-created transactions to escape the sliding window.

Max-block is the maximum number of blocks after last-block that may pass before the transaction must be included into a new block or dropped. This primarily serves as way to speed up block-inclusion auctioning.

To prevent border-cases in validation, nodes must differentiate between

- Spendable window from which transactions can be *used* as inputs for new transactions (transactions can be spend). We suggest 1 year of blocks (≈ 52600).
- Validating window which serves to *verify* transaction inputs. This we suggest to be 1 year and 1 month of blocks (≈ 56920).

Using the extended validating window prevents double spends of old transactions in case of blockchain forks.

Note 2: Lottery

- Latest devalidated block: $B0$
- Sum of coins from unspent outputs from $B0$: A
- Newest created block: Bn
- Take a hash of all block headers $B1 - B(n-1) \rightarrow$ Seed
- Create a list of all addresses used in *spent* transactions in blocks $B144 - B(n-2)$ and order them ascending $\rightarrow$ Pool
- Create SEED_PRNG(Seed)

Algorithm 1 Coin distribution in lottery

$R := A$
for $i = 2$ to 6 **do**
  Transfer R/i Bitcoin to address at Pool[PRNG(1,length(Pool))]
  $R = R - R/i$
end for
Transfer R Bitcoin to address at Pool[PRNG(1,length(Pool))]

A simple and sufficiently secure implementation of this would be to use SHA256 as a PRNG by calculating:

$PRNG(1, length(Pool)) := (SHA256(i|Seed) \bmod length(Pool))$

Note 3: Forced Zerocoin mixing

Zerocoin mixing provides only a small anonymity set. But it can be used to prevent scoring attacks (KYC for addresses). This however works **only** if *all* coins in the blockchain are forcibly mixed - otherwise mixed coins could be discriminated against. Zerocoin is furthermore computationally and

storage expensive. The Zerocoin anonymity set is as big as the number of transactions added to the accumulator at the same time (in one Zerocoin block). For implementation, the scoring risk of the coins must be taken into account.

For each transaction, calculate the new scoring risks for the outgoing parts as follows:

Algorithm 2 Calculate scoring risks

```
Score = 1
Total = 0 (for newly mined coins, total := 1)
for all incoming transaction do
    Score += TransactionScore * Amount
    Total += Amount
end for
Score = Score / Total (integer divide aka floor(score/amount) )
```

This way the bigger amounts have bigger affect on the total score of a transaction. Set this score for each outgoing transaction and then add it to the blockchain. This way all transactions have an easy to calculate and lookup score. If the score of an outgoing transaction > 100 (protocol defined), then then coins can only be claimed by either:

1. Transaction to the *same* Bitcoin address (coins cannot be used in payment anymore but allow forward transaction to escape the sliding window, implement by comparing scripts).

2. In a transaction to Zerocoin mixing.

Zerocoin mixing happens only infrequently (once every 12h for example) to increase the anonymity set. Coins leaving the Zerocoin mix have an initial score of 1 again (this leaves some attacks on the anonymity set open, but not enough to justify successful scoring). Withdrawal from the Zerocoin chain, of course, only happens within the Zerocoin chain (every 12h hours). This results in a minimum wait time of 24h for de-scored coins (longer if the client prefers that), and an anonymity set of all Zerocoin transactions within 12h (that are included in the block).

This does **not** provide untraceable or anonymous money, it only prevents censorship of transactions (via KYC scoring etc.).

Since Zerocoin transactions do not allow the easy enumeration of included coins (that is one of its purposes after all), this conflicts with the sliding window concept. This could be solved by constructing a new accumulator key (and thus a fresh Zerocoin chain) frequently and to redistribute the remaining coins (tracked by summation) via the lottery. To make this feasible, accumulator designs without the need for trusted generation are available and should be investigated.

Note 4: Miner ostracism

Without question, mining is the crown of the Bitcoin design. It serves as an elegant and functional means to:

1. Control the creation of new money

2. Synchronize the database of valid transactions between many nodes

3. Distribute the inclusion decision for transactions into blocks and the verification of validity so that Bitcoin does not depend on single points of failure

4. Define the order of transactions to prevent double spending

Any change to the central design described above could fundamentally change the nature of Bitcoin, and should require strong justifications.

However, from the view of our design goals the function of mining as distribution of the inclusion decision needs to be evaluated.

It is important to note that miners not only have the power to exclude invalid transactions from the block, but also have considerable leverage in defining validity. While the restrictions of the original design and implementation define validity, miners can redefine validity as a subset of the original design, but not as a superset. This can easily lead to client transactions being ”nudged” towards a much stricter definition of validity by increasing the time until such a transaction is included into a block, if ever.

This would not pose a problem if mining were controlled by a large number of individuals with varying motivation, and who do not differ widely in access to processing power. However, the link between monetary incentives (block and fee profit) and the decision over which transactions are declared valid and included in a block is prone to lead to conflicting motivations between keeping Bitcoin as profitable as possible in the light of outside regulatory attempts and keeping it as open and censorship-resistant as possible. Numerous discussions about the exclusion of SatoshiDICE transactions by miners speak as evidence to this, as does the possible application of FinCEN definitions for Money Service Businesses to miners. We must therefor consider that regulatory compliance may very well become a decisive factor for the profitability of mining operators.

There is an immediate response to this is: That enough miners that do not apply overly restrictive validity definitions to transactions will remain, and that the risk of transactions being effectively discriminated against is small. This is not an argument but an expression of hope. Nothing in the protocol design of Bitcoin prevents miners or groups of miners from gaining a decisive position in relation to the sum of all processing power on the network. Especially so if mining pool operators assume the inclusion decisions for transactions, while leaving the number-crunching to mining rig owners that are not even connected to the Bitcoin network. Furthermore, since the success of a mining operation rests solely in the capital investment decisions of the miners competing with each other and **not** in the fulfillment of Bitcoin user needs, no self-regulatory model for the relationship between the mining community and all other Bitcoin users exists in the current implementation. To make matters worse, holders of Bitcoin cannot easily leave Bitcoin without the risk of seriously depressing the price of their assets. This is a classical lock-in situation that easily leads to an oligarchy. After that point, the only questions remaining are whether this oligarchy is benevolent or not, and whether Lord Acton was right that power corrupts.

We have not chosen to put our hopes into the continued goodwill of the majority of miners. The recent lobbying campaigns and additions to the protocol simply speak too much of creating a distributed PayPal instead of a censorship resistant currency; one that remains judgement-free as to how people use their money. Instead, we present an interlocking set of ideas that ties the economic success of a miner to the wishes of Bitcoin users.

As mentioned above, we are well aware of the substantial changes and added complexity that these ideas entail. We are also under no illusion that these ideas are sufficient to protect Bitcoin 2 in the long term. The problems of validity definition and mining centralization are part of Bitcoin’s nature - we have to deal with them.

We define the following goals to resist mining-centralization and increase the link between miner-success and users’ interests:

1. The use of *expensive* miner pseudonyms to link miners to their negative reputation.

2. Economically disincentivize the centralization of mining-power.
3. The shifting of inclusion decisions closer to the person and/or hardware that does the number crunching.
4. Enable users to personally ostracize miners that behave contrary to their wishes.

To reach these goals, changes to the design are necessary because anything that is not demanded by the protocol is ultimately without force.

Expensive miner pseudonyms

Currently there exists no way to reliably identify miners and bind them to their positive or negative reputation. Miners self-identify only through the script of the coinbase transaction, making their Bitcoin address a pseudonym. However, this pseudonym is inconsequential for reputation tracking, because it is not linked to the actual processing work of the miner. (It only defines who gets the profit, not who mined a block.) It is also very cheap to change. Thus, the current design only allows for a "cheap" pseudonym. Since our goal is to link miners to their negative reputation, a sufficiently expensive pseudonym must be created and linked to the miner's operations.

For this reason we introduce a new concept into the design of Bitcoin 2, the *miner key* (short: *mkey*).

Miner keys are hard to produce and are linked to various computational operations that are involved in mining a block. To produce them, we suggest that miner keys are created in this way:

- Create a new ECDSA key pair: $pkey$, $skey$
- Find a $nonce$ so that $(target << P >= SHA256(SHA256(pkey|nonce)))$
- Create $S := ECDSA\_Sign(skey, pkey|nonce)$
- Include $pkey$, S and $nonce$ in the blockchain, using a new script operator. $pkey$ is thereby created as a new mining key.

P is an easing factor that we suggest to be 1 (halving the difficulty of the operation). $target$ is the target calculated for the block in which the key is to be announced. "|" denotes concatenation.

This uses current primitives of mining and should thus be easily ported to existing mining hardware. Finding such a key does indeed take substantial time but is well in reach of most existing miners.

Mining keys created this way are visible to all nodes in the network since the operation is included in the blockchain. Only blocks signed with one of these keys (see below) may be included in the blockchain. In addition, each mining key is only valid for a limited number of block signatures after which the mining key becomes invalid and cannot be used in mining operations any longer but is not restricted in any other way. We suggest a mining key is good for 144 block signatures.

To incentivize linking of multiple generations of a miner's mining keys, subsequent generations of keys referencing each other is proposed. For this, the new mkey is published with a cross-signature of the old mkey (old mkey signs new mkey, new mkey signs old mkey). Common ownership is thus established. Furthermore, P is increased by 1 for this operation, halving the effort to create a new mkey. This linking operation is only valid for mkeys that have less then 100 signed blocks. Afterward, no linking of keys is allowed. For verification, nodes need to track the creation and the use of these keys on the blockchain.

Miner keys represent a potential future value because it allows participation in the block mining operation, while at the same time it is a monetary expense due to the fact that processing resources are spent on finding miner keys that could be used to find a new block. Miners therefor have a certain, though limited, incentive to allow their reputation be linked by means of the miner key instead of frequently changing their pseudonym. Miner keys are thus more *expensive* pseudonyms.

Alternative methods of creating expensive pseudonyms exist, such as webs of trust or smartcards. These should be discussed and evaluated elsewhere.

Economically disincentivize the centralization of mining-power and shift inclusion decisions closer to the number crunching

To decentralize the inclusion decision, from mining pool operators to miners, we propose two further changes to the mining process: Nonce-signatures and block-signatures.

Nonce-Signatures

Found blocks should be tied to the miner that finds it. For this we use a nonce signature created as follows:

$nonce\_sig = ECDSA(mkey, nonce >> Y)$

The nonce_signature is included in the block and requires the miner to create an ECDSA signature with his miner key every 2^Y tries to find a nonce that wins the block creation challenge. When Y is selected well this incentivizes the signature to be created close to the machine that does the number crunching and therefor supports blocks to be calculated by individual miners (as in: Persons). Furthermore we require that the miner key is necessary to transfer the block profit recorded in the coinbase transaction. However, to still support economic cooperation in mining pools without decision centralization, the coinbase transaction can be a n-of-n multi-signature transaction that requires mkey (a n-of-m transaction could circumvent the requirement that the mkey is required to transfer out of the coinbase transaction).

Requiring a signature only for every 2^Y nonces that are tested, still utilizes existing mining hardware. We suggest Y to be 9.

Block-Signatures

The nonce signature alone however does not enforce that the inclusion decision is done by the individual miner. For this, we propose that the transaction selection in the block is signed by the same miner key as well. This signature is created as follows:

Algorithm 3 Block signature

$h := previous\_block$
for all transactions in block **do**
  $h := SHA256(nonce\_sig|transaction|h)$
end for
$block\_signature := ECDSA(mkey, h)$

Notice that the $block\_signature$ depends on the transaction order in the block, the nonce and the link to previous blocks in the blockchain. Also notice that the operation requires the full transaction content and not just a transaction hash - using a large-block signature scheme could further

enforce that. Reordering of transactions in the block will devalidate the $block\_signature$. The $block\_signature$ is added to the block and has to be verified by other nodes.

Given the above changes, blocks and the inclusion decision are tightly linked to whoever has knowledge of the private miner key, as long as Y is selected to be small enough to make network communication for the central creation of signatures infeasible. Thereby, we propose a *proof-of-presence* to keep processing power close to miner-key knowledge. Another method to implement *proof-of-presence* would be the use of certain smartcard methods, which should be discussed elsewhere.

Enable users to personally ostracize miners that behave contrary to their wishes

The previous changes to the mining process are necessary conditions for supporting miner ostracism. Miner ostracism has the intention to allow individual users to punish miners which they see to harm the freedom of the Bitcoin 2 network. This ostracism gains relevance to the miner with every user agreeing to it and can lead, in the most extreme case, to a miner not being able to profitably mine new blocks with his expensive pseudonym (miner key).

Miner ostracism introduces a new field for every transaction that defines one or more miners identified by miner key which **may not** include the transaction into any block they create and thus may not reap the fees from this transaction. Since the block reward decreases over time, and fees in a small, fixed-size block increase, the collection of fees becomes increasingly important to miner profits.

However, miner ostracism requires some additional conditions for block validity which, we will discuss before we present our idea on how to implement miner ostracism itself.

As mentioned above, miners may not include transactions into a block that ostracize that miner, therefor all nodes have to verify that a block does not include such a transaction for the miner. Miners can however ignore most transactions and still generate a block with the remaining transactions that do not ostracize against them. To make this more difficult, the following additional conditions need to be met for a valid block:

Each miner (and ideally every node) maintains a memory pool that includes transactions that have not been included in the blockchain so far. On receiving a fresh block from another miner, each miner (and ideally every node) needs to check:

1. If $size(transactions\_in\_mempool)/3 > max\_block\_size$,
   then $max\_block\_size/size(new\_block) < 2$.

2. Count all transactions that are both in the new block and in the memory pool $\Rightarrow K$. If $count(transactions\_in\_new\_block)/K < Q$, then the block is invalid. Q must be picked wisely to not produce network splits constantly. We propose 3.

These conditions make it harder for an ostracized miner to fill his block with random transactions, to stay in the game even if a majority of the network prefers that he stops mining. However, these conditions also increase the risk of dropped blocks and splits in the blockchain for miners that are not well integrated and do not see many transactions. This, however, is not a bug but a feature.

With this in mind, we propose miner ostracism to be implemented as follows:

Each node creates a list of miners by creating a list of miner keys and the number of blocks signed by each from the current block backwards to the beginning of the spendable window. The list is sorted descending by number of blocks signed by the mkey and then miner key ascending. The list

is then converted into a balanced binary tree, with the miner keys as leafs. We call it the *miner-tree*. The miner-tree is stored for the last 144 blocks individually. (Please note that the miner-tree can be constructed more efficiently, we only describe it this way for sake of clarity.)

Transactions already include the "last_block" field which is set to the last block known to the client. This field also determines the miner-tree that applies to the transaction – if the transaction includes a miner_ostracize field. Please note that this will only allow ostracism by transactions that are included into the blockchain within approximately 1 day because miner-trees are not retained any longer. We suggest that ostracizing transactions be included without applying the ostracizing method after that time or by enforcing a low-enough "last_block" setting in all transactions. Longer miner-tree retentions should be considered as well.

The miner_ostracize field defines the path through the miner_tree: The depths of the path to the miner or group of miners that should be ostracized against. With roughly 52600 blocks in the spendable window, the miner-tree will include a maximum of 52600 miner keys (if each miner only found one block) and have a depth of $\log_2(\text{leafs}) == 16$ bits maximum. This way we can encode a single branch or leaf in the tree within 24 bits, allowing up to 4 paths per transaction. Encoding both path and depth in the miner_ostracize field also allows users to ostracize against (for example) the top $1/(2^4)$ miners with a single path. While generating the miner_tree for the first time over an existing blockchain requires some computation, updating the tree for subsequential blocks is computationally cheap. Traversing the tree is highly efficient as well. Miners can easily verify that a transaction ostracizes them by simply comparing it only to their own bit-encoded path.

Combining these changes to the mining process, the incentives for miners change and become a tradeoff between:

1. Cost of mining a block
2. Cost of communication between mining hardware
3. Cost of mining a miner-key
4. Cost of choosing a more restrictive validity definition as given in the protocol
5. Block reward
6. Fees through user-friendly inclusion decisions

instead of only the cost of mining a block and total profit per block (including fees).

This will not prevent any specialization or centralization, but it will buy a distributed, censorship resisting miner-community some additional time.

Note 5: Devalidating transactions

To move coins from Bitcoin 1 to Bitcoin 2, devalidating transactions are an option:

1. Create a 2 out of 3 multi-signature transaction in Bitcoin 1 so that:
   (a) Address 1 is the true Bitcoin address of the client (it knows the private key).
   (b) Address 2 and 3 are created this way:
      i. Create a nonce (random, one-use value) N.
      ii. Calculate $SHA512(N)$.

iii. Split $SHA512$ in half: $Addr2 == bits0 - 255$, $Addr3 = bits256 - 511$

2. Create a "claim" Transaction in Bitcoin 2, giving:
   (a) The transaction and block within Bitcoin 1.
   (b) Nonce N.
   (c) Signed by client's private key.

The claim transaction to Bitcoin 2 is valid if:

1. It is included into a block with number < 420000 (see note 6 for details about block numbering in Bitcoin 2). From block 420000 on (when the next reward halving will happen) new claim transactions are **not allowed** anymore.
2. The Bitcoin 1 transaction is at least 120 blocks old (to escape the forking window).
3. The Nonce produces Addr 2 **and** Addr 3.
4. The transaction is signed by the client's private key.
5. The total number of coins in Bitcoin 2 (including the claim transaction) does **not exceed** the number of coins which would exist if the Bitcoin 2 blockchain began at block 0 (i.e., the number of coins that can be transferred to Bitcoin 2 is limited to the number of coins mined in Bitcoin 1 before the fork).

The total number of coins in Bitcoin 2 can be calculated easily by looking at the current blockchain window. The maximal possible number of coins can be computed just from the block number.

Note 6: Blockchain bootstrapping

The first block (the "genesis" block) in the Bitcoin 2 blockchain must reference the latest block (with number b) in the Bitcoin 1 blockchain and be numbered itself as $b + 1$. Block b is the block in Bitcoin 1 from which the Bitcoin 2 blockchain is "forked".

Reward halving works exactly the same as in Bitcoin 1 (i.e., Bitcoin 2 starts with a reward of 25 Bitcoin pro block, because b is already larger than 210.000 at the time of this writing).

7 Conclusion

We presented a set of changes to the original Bitcoin protocol that would lead to much better security, resilience and decentralization: A sliding blockchain with lottery for dead coins, forced Zerocoin mixing, and miner ostracism.

These changes would require a *proactive fork* of the blockchain, but the presented *devalidating transactions* are a practical way to gradually shift the Bitcoin economy over to a better system. A shift that is absolutely necessary, if we want to keep a truly alternative payment system.

We want to close with a nugget from the early days of Bitcoin. In 2008 somebody challenged the Bitcoin idea with the following statement:

> *You will not find a solution to political problems in cryptography.*

Satoshi Nakamoto's answer was:

Yes, but we can win a major battle in the arms race and gain a new territory of freedom for several years.

This statement remains true and the time has come to rearm for the Freedom of Transaction!

References

[1] Bitcoin Payment Messages. https://gist.github.com/gavinandresen/4120476. Accessed: 2013-07-09.

[2] Bitcoin Transaction Metadata Design. https://gist.github.com/gavinandresen/4073937. Accessed: 2013-07-09.

[3] BitLaundry - For all your Bitcoin washing needs. http://app.bitlaundry.com/. Accessed: 2013-07-11.

[4] Blockchain.info - Send Shared. http://blockchain.info/de/wallet/send-shared. Accessed: 2013-07-11.

[5] btcd. https://opensource.conformal.com/wiki/btcd. Accessed: 2013-07-06.

[6] Open Transactions. https://github.com/FellowTraveler/Open-Transactions. Accessed: 2013-07-06.

[7] Significant Merchant Improvements Planned for Bitcoin v0.9. http://thegenesisblock.com/significant-merchant-improvements-planned-for-bitcoin-v0-9/. Accessed: 2013-07-11.

[8] Sneakernet. http://en.wikipedia.org/wiki/Sneakernet. Accessed: 2013-07-11.

[9] Voucher Safe. http://www.voucher-safe.org. Accessed: 2013-07-06.

[10] I. Miers, C. Garman, M. Green, and A.D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. http://zerocoin.org/media/pdf/ZerocoinOakland.pdf, 2013. Accessed: 2013-07-06.

[11] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2013-07-06.