Coins: A Billion Bitcoin Users

Robin Linus

January 12, 2020

Abstract

Coins is a Bitcoin extension designed for payments at scale. We propose an efficient solution to the double-spending problem using a bitcoin-backed proof-of-stake. Validators vote on sidechain blocks with one-time signatures, forming a record that cannot be changed without destroying their collateral. Every user can become a validator by locking bitcoins. One-time signatures guarantee that validators lose their stake for publishing conflicting histories. Checkpoints can be additionally secured with a bitcoin-backed proof-of-burn. Assuming a rational majority of validators the sidechain provides safety and liveness. The sidechain's footprint within bitcoin's blockchain is minimal. The protocol is a generic consensus mechanism allowing for arbitrary sidechain assets. Spawning multiple, independent instances scales horizontally.

1 Introduction

Bitcoin is a revolutionary alternative to the traditional, government-approved banking system [1]. However when Satoshi Nakamoto introduced it in 2008 the world's first response was: *"We very, very much need such a system, but the way I understand your proposal, it does not seem to scale to the required size"* [2]. Ever since numerous scalability solutions have been proposed. Still, as of today, the Bitcoin network processes less than seven transactions per second. In contrast, centralized payment services such as PayPal or Visa serve billions of users at up to 50,000 TX/s.

Currently, off-chain payments via the Lightning Network are the most promising approach to scale Bitcoin [3]. They allow a much higher throughput, yet they hardly scale to billions of users. They still require too many on-chain transactions to open and close payment channels. Adoption is even further constrained by the inbound-capacity of payment channels and the need to lock funds for every new user. These constraints lead to many layers of complexity and a tendency towards centralized and custodial solutions which contrast Bitcoin's purpose of being permissionless, trustless and censorship-resistant.

Sidechains have been proposed as an alternative solution for scalability [4]. They introduce parallel blockchains enabling payments within a simplistic system similar to Bitcoin. Yet, their designs depend on federations or miners validating sidechain blocks which limits security, scalability and flexibility. We introduce a new sidechain consensus mechanism with a permissionless, bitcoin-backed proof-of-stake.

2 Bitcoin Stake

It is impossible to produce distributed consensus except by consuming an external resource. This is because if block production has no ongoing costs, neither does attacking the chain [5] [6]. We anchor our proof-of-stake mechanism into Bitcoin's proof-of-work consensus which is computationally, and therefore thermodynamically, very expensive. We use bitcoins as an external resource to produce sidechain consensus.

2.1 One-time Signatures

A characteristic of Bitcoin's digital signature algorithm is that it needs to produce, for each signature generation, a fresh random value (hereafter designated as *nonce*). Reusing the nonce value on two signatures of different messages allows attackers to recover the private key algebraically.

The nonce reuse vulnerability can be used to discourage validators from participating in double-spending attacks [7] [8]. Each validator pre-commits to his sequence of nonces, such that the system can constrain a validator's signature for the n-th block to be valid only if he signed it using his n-th nonce. This guarantees validators can not create valid signatures for conflicting blocks without leaking their private key and losing their collateral.

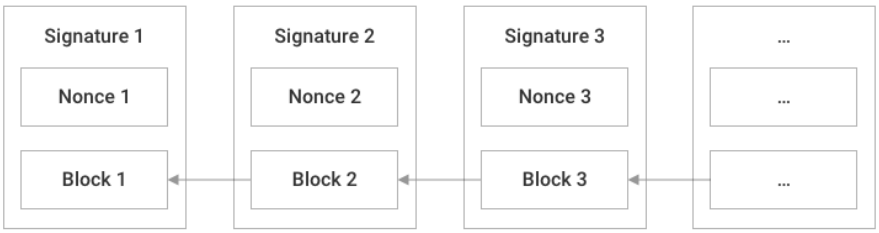

Figure 1: A validator votes for blocks with one-time signatures.

2.2 Collateral Contracts

For validators' one-time signatures to be scarce, each of them has to lock a collateral such that the bitcoin network can penalize malicious actors for signing conflicting histories. In the following we discuss *collateral contracts* which serve as an adaptor to produce a sidechain consensus from Bitcoin's consensus. Our contract expresses: *If Alice leaks her key she loses her bitcoins.* So for Alice to become a validator she locks bitcoins in an output such that:

- Option A: One year later she gets her money back.
- Option B: She can destroy her money right now.

2.3 Collateral Contracts in Bitcoin Script

Currently, burning funds is not supported in Bitcoin script. Yet, future Bitcoin features such as `OP_SECURETHEBAG` or `SIGHASH_NOINPUT` allow trustless covenants.

2.3.1 Trust-minimized Workaround

For now, we need a workaround to implement the collateral contract. A simple solution is to introduce a trusted party, say, Bob:

- Alice creates a funding transaction with the following output:
  - Alice can spend her money in one year.
  - Alice and Bob can always spend her money collaboratively.
- Bob pre-signs and publishes a punishment transaction:
  - burn her collateral now to address 0x000...000 (if Alice agrees).
- Bob immediately deletes his secret key (in case his machine gets compromised).

There is no counter-party risk here for Alice, because she executes her funding transaction only after she received Bob's signature for the punishment transaction and completed her setup.

For the system to minimize trust in Bob, multiple parties can participate and if only one is honest and deletes their key, then this scheme is secure. Bitcoin script currently supports more than 60 participants per multi-signature transaction [9].

Containing more than 60 signatures, the punishment transaction becomes large and expensive, but as long as the validator is honest, the transaction doesn't need to be included in Bitcoin's blockchain. To incentivize Bitcoin miners to execute the punishment transaction quickly in case of misbehavior, it pays a high miners fee.

It is possible to further minimize trust in single parties by using ECDSA signature aggregation to allow for thousands of Bobs participating in a single combined signature [10]. When Schnorr signatures become available in Bitcoin, such aggregated signatures will become more simple because of the linearity of Schnorr's scheme [11].

The Bobs could be a trusted federation or a percentage of the current validators. With aggregated signatures they could also be a significant percentage of current coin holders. The trusted parties sign blindly to reduce the risk of censorship. A simple scheme exploits that Bitcoin transactions use double SHA256. Thus, Alice can ask Bob to sign the single-round SHA256 hash of her transaction. The second round is Bob's hashing function for his signature algorithm.

All of these trust-minimizing workarounds are highly undesirable additional complexity. Fortunately, in the long term, we will have a clean and trustless solution with upcomming features such as `SIGHASH_NOINPUT`.

3 Consensus Mechanism

The sidechain's consensus is anchored into Bitcoin's proof-of-work consensus. Validators are defined by collateral contracts included in Bitcoin's UTXO set. Assuming all sidechain nodes are running a Bitcoin full node, they implicitly are in consensus about the exact validator set at every point in time without exchanging any messages. All randomness is determined by Bitcoin's proof-of-work which is studied well [12]. Validators regularily commit their votes into Bitcoin's blockchain. This approach prevents many classical problems of pure proof-of-stake protocols such as the nothing-at-stake problem, long-range attacks, stake grinding and costless simulation [5] [13]. Still, a malicious majority can halt the system.

3.1 Voting for the next Block

The set of validators is known and so is the number of possible votes per sidechain block. An election is economically final as soon as there is a majority such that validators would have to burn deposits to change it. Votes "wasted" for conflicting blocks help finalize the result, too.

An *epoch* starts when a validator receives a Bitcoin block. He calculates for every validator V a score $score_V = Hash(bitcoinheader||publickey_V)$ and then he sorts the validators by score. The resulting indexes are the validator's ranks, $rank_V$. He calculates his own delay: $delay = rank_{self} * 15sec$. He waits up to $delay$ seconds for some other validator with a shorter delay to broadcast a next sidechain block. If he receives a valid

block he adds his signature and broadcasts it. Otherwise, if his delay times out, he creates, signs and broadcasts his own block. Validators' block signatures are broadcasted on a best-effort basis. Nodes relay blocks only if its delay is appropriate.

In case there are many validators and a high percentage of attackers the delay function is actually not linear, but exponential for higher ranks. This allows collisions and favors a chaotic election over perfect order, but guarantees that an honest majority can always finalize a next block.

3.2 Timestamping Elections

Every block references the election of the previous block. Elections are also frequently timestamped within the bitcoin blockchain. The validators have to commit their vote on every 43,800th sidechain block (once a month) into Bitcoin's blockchain. If a validator does not commit his signature within 2 days (288 Bitcoin blocks) he is removed from the validator set. This mechanism detects offline validators. Publicly enforced votes also act as historical checkpoints to ensure validators can not create conflicting histories after they received their deposits back.

3.3 Block Rewards

Similar to Bitcoin's miner's reward via coinbase transactions, the epoch leader creates a reward for himself in his block. The rewards for the members of the current delegation are also such a coinbase transaction.

Validators receive their rewards only if they voted for the winning block. Block rewards are locked similar to collateral contracts such that a validator signing conflicting histories destroys both his bitcoin collateral and his recent sidechain block rewards.

4 Design Details

4.1 Compact Nonce Commitments

Every validator pre-commits to his set of nonces in his funding transaction. The commitment is the root of a Merkle tree of a sequence of nonces. A proof for a nonce's index is a merkle path. Constructed naively, validators would have to pre-commit to millions of nonces. To make this construction efficient, validators commit only to fixed-sized sequences of nonces and before these are all used up, they create a next commitment and sign it with the last nonce of their previous sequence.
Validators need to keep track of their nonces such that they do not reuse them. A solution is to derive nonces from their private key using hierarchical deterministic hardened keys [14]. This prevents unintentional nonce reuse efficiently. For any sequence of nonces the

validator needs to keep track only of the index of the most recent nonce. The index can be safely recovered when lost with the following scheme: The index currently required for voting always equals the number of sidechain blocks, which equals the number of Bitcoin blocks since sidechain genesis.

4.2 ECDSA vs Schnorr Signatures

Currently Bitcoin supports only ECDSA over curve SECP256k1 though we can simply reuse ECDSA key pairs with the Schnorr scheme over the same curve for all signatures in the sidechain. This is more efficient and enables sophisticated features such as aggregation, batching, or compact multi-signature schemes like MuSig [11].

4.3 Delayed Commitments

Bitcoin re-organizes a couple of blocks from time to time. However, the likelihood of bitcoin re-organizing six blocks is neglectable [1]. So to avoid any re-branching in the sidechain, it derives its randomness from the sixth most recent Bitcoin block. This mechanism is vulnerable to stake grinding attacks. However, an attacker would need to create a new Bitcoin block for every query, which is very expensive. To further reduce the risk, verifiable delay functions have been popular recently as a strategy for getting entropy for validator and committee selection in PoS networks [15].

4.4 Scalability and Sharding

Coins can scale horizontally almost indefinitely by spawning multiple fully independent instances in parallel. All instances are naturally synchronized by Bitcoin's blockchain. Each instance manages a separate currency, but in a standardized format to facilitate cross-chain transactions via atomic swaps. Cross-chain communication requires no cross-chain validation. A swap is validated only between the two swapping clients.

This scales well. Assuming each instance grows approximately like Bitcoin's blockchain. Assuming further, optimizations such as Schnorr Signatures make the sidechain about three times more compact than Bitcoin, such that it can process about 20 transactions per second at the same network load. Then to achieve transaction rates as high as centralized services with billions of users we need 2,500 independent instances. For 200 validators per instance we need about 500,000 on-chain transactions per month, which the current Bitcoin network could process in less than a day.

5 Limitations

5.1 Safety and Liveness

By definition, a decentralized cryptocurrency must be susceptible to 51% attacks whether by hashrate, stake, or other permissionlessly-acquirable resources. Our system always provides safety, but an attacker can halt the network as long as he controls a majority of all bitcoins at stake. Halting the system costs at least the time value of locking sufficient amounts of bitcoins.

5.2 Ongoing Cost to Attack the Chain

Time value of locked bitcoins might be too cheap to protect the chain. We can introduce an additional cost and let validators burn bitcoins for every on-chain vote. This is much more robust because there is an ongoing cost for halting the system. Proof-of-burn has recently been formally analysed [16].
The economic implications of burning significant amounts of Bitcoin are questionable. A level of security comparable to Bitcoin requires the system's BTC burn rate to be equal to Bitcoin's infaltion rate.

5.3 Partitioning Attacks

We assume an attacker cannot split the network into two or more disjoint groups, such that significant percentage of validators cannot communicate. That would allow an attacker with less than 51% of all stakes to overwrite the chain.

5.4 Hot Key Security

Hardware wallets are stateless and therefore they're incompatible with nonce commitments. To be compatible they would need to keep track of every nonce they have signed. Otherwise an attacker can extract the private key by requesting two different signatures with the same nonce.
Therefore, to protect the validators' nodes, they can use a *validation key* to sign sidechain blocks and a seperate *redeem key* to spend the bitcoin deposit once it is unlocked. Until then, the redeem key remains in a cold wallet. Nodes having access only to the validation key are a much less attractive target for attackers.

5.5 Overhead of Sharding

Transactions within an instance are simple, but cross-shard communication is more complex and expensive. Shards might be off-sync because our clock is Bitcoin's consensus

which has a high latency. This is a difficulty for cross-chain atomic swaps. Also every shard manages a separate currency which implies varying exchange rates, but also creates a market for balancing shards and flexibility to tolerate local failures. Varying exchange rates introduce the free option problem [17] which might become practical to exploit in situations when there is a high price volatility. The overhead of naive cross-chain communication scales quadratically because each shard needs a channel into each other shard. HTLCs allow for multi-hop communication such that shards can share their channels. It is efficient to centralize the communication around Bitcoin as a settlement layer. This allows payments between arbitrary shards within 2 hops. Theoretically every shard requires only one payment channel – the one with bitcoin. With the same argument each currency requires only one exchange rate. This enhances usability because wallets can abstract away all underlying currencies and display to users only a single currency: BTC.

6 Conclusion

We have proposed a Bitcoin extension, potentially scalable beyond 50,000 transactions per second. We started with the usual framework of a sidechain, which provides strong control of ownership, but is incomplete without a trust-minimized way to prevent double-spending. To solve this, we proposed a bitcoin-backed proof-of-stake consensus mechanism that quickly becomes economically impractical for an attacker to change if honest nodes control a majority of stakes. The network becomes more robust with an additional proof-of-burn. Validators and leaders are elected via Bitcoin's consensus with little coordination. Nodes can leave and rejoin the network at will, accepting checkpoints in the bitcoin blockchain as proof of what happened while they were gone. Validators can not create conflicting histories without losing their Bitcoin collateral. They vote with their signatures, expressing their acceptance of valid blocks by signing them and rejecting invalid blocks by refusing to sign them. Any needed rules and incentives can be enforced with this consensus mechanism.

References

[1] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] James A. Donald. The cryptography mailing list - bitcoin p2p e-cash paper. Bitcoin Stack Exchange.

[3] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[4] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, page 72, 2014.

[5] Andrew Poelstra et al. Distributed consensus from proof of stake is impossible. 2014.

[6] Cristina Pérez-Solà, Sergi Delgado-Segura, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. Double-spending prevention for bitcoin zero-confirmation transactions. *International Journal of Information Security*, 18(4):451–463, 2019.

[7] Tim Ruffing, Aniket Kate, and Dominique Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 219–230. ACM, 2015.

[8] Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In *International Conference on Financial Cryptography and Data Security*, pages 126–141. Springer, 2016.

[9] Pieter Wuille. Will segwit allow for m of n multisig with very large n and m? Bitcoin Stack Exchange.

[10] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194. ACM, 2018.

[11] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[12] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. *IACR Cryptology ePrint Archive*, 2015:1015, 2015.

[13] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473. ACM, 2019.

[14] Pieter Wuille. Hierarchical deterministic wallets. Bitcoin Improvement Proposal.

[15] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual International Cryptology Conference*, pages 757–788. Springer, 2018.

[16] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn.

[17] ZmnSCPxj. [lightning-dev] an argument for single-asset lightning network. Lightning Developers Mailing List, 2018.