Stakechain: A Bitcoin-backed Proof-of-Stake

Robin Linus

December 20, 2021

Abstract

We propose an energy-efficient solution to the double-spending problem using a bitcoin-backed proof-of-stake. Stakers vote on sidechain blocks forming a record that cannot be changed without destroying their collateral. Every user can become a staker by locking Bitcoins in the bitcoin blockchain. One-time signatures guarantee that stakers lose their bitcoin stake for publishing conflicting histories. As long as 34% of the stakers are honest the sidechain provides safety, and with a 67%-majority it provides liveness. Overwriting a finalized block costs at least 34% of the total stake. Checkpoints in Bitcoin's blockchain mitigate classical attacks against conventional proof-of-stake algorithms. A stakechain's footprint within the mainchain is minimal. The protocol is a generic consensus mechanism allowing for arbitrary sidechain architectures. Spawning multiple, independent instances scales horizontally to a free market of sidechains which can potentially serve billions of users.

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 because growing the chain faster reduces decentralization significantly. 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 because the required block space grows linearly with the number of users. Adoption is even further constrained by the inbound-capacity of payment channels and the need to lock funds for every new user to receive a payment. 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 consensus mechanisms depend on trusted federations or Bitcoin miners validating sidechain blocks, which limits social scalability, and thus, security. We introduce a novel sidechain consensus mechanism with a permissionless, bitcoin-backed proof-of-stake. This results in a fast, flexible and scalable consensus mechanism, that enables a free market of trustless sidechains.

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]. The security of a consensus mechanism is proportional to the amount of external resources consumed. We anchor our proof-of-stake mechanism into Bitcoin's proof-of-work consensus which is computationally, and therefore thermodynamically, very expensive to change. We leverage the value of bitcoins as an external resource to produce sidechain consensus. One-time signatures ensure stakers lose their bitcoin stake when voting on conflicting sidechain histories. This forms a linear record, which is costly to change without burning significant amounts of resources.

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

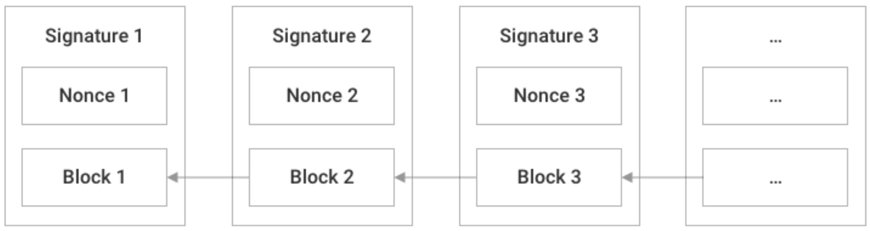

Figure 1: A staker votes for blocks using one-time signatures. A vote for the n-th block is valid only if it is signed with their n-th nonce. Voting twice with the same nonce leaks the staker's private key.

2.2 Staking Contracts

For stakers' 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 a *staking contract* which serves 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 staker 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.

This simple contract is sufficient to make her one-time signatures scarce. If Bob sees two signatures of her with a reused nonce, Alice leaks her key and loses her stake. In Appendix A.1 we summarize different implementations of this staking contract in bitcoin script. We describe the ideal, trustless solution using a *Bitcoin Covenant* [8]. This is possible with one of various future bitcoin features such as either `SIGHASH_NOINPUT` or `OP_CHECKTEMPLATEVERIFY`. Furthermore, we discuss two trust-minimized workarounds that are possible with Bitcoin's current consensus rules. In Appendix A.2 we describe a scheme for stakers to commit to a unique sequence of nonces. The key idea is that each staker forms a chain of nonces by signing their next nonce with their previous signature.

3 Consensus Mechanism

The sidechain's consensus is anchored into Bitcoin's proof-of-work consensus. Stakers are defined by staking 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 staker set without exchanging any messages. All randomness is determined by Bitcoin's

proof-of-work, which is studied well [9].
The staker set is determined only by Bitcoin's blockchain. There are three operations on the staker set: *stake*, *redeem*, and *burn*. All three are executed in the bitcoin blockchain. This 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] [10]. Bitcoin-backed proof-of-stake enables fundamentally more robust mitigations because Bitcoin's blockchain offers a reliable ground truth for the status of the staker set at each point in time.

3.1 Voting for the next Block

The set of stakers 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 of votes such that stakers would have to burn stake to attack it.

The leader is determined from randomness derived from bitcoin's most recent block. The leader proposes a signed sidechain block and all other stakers confirm that block by signing it, too. Once 34% of stakers voted for a block it cannot be reverted without burning stake. A block is considered *final* once a supermajority of 67% signed it. Changing finality costs at least 34% of all stake. As soon as the next leader receives a final block, it starts waiting one minute until it proposes a next block.[a]
If a leader does not propose a block within a given time then all other stakers start broadcasting skip messages for that block. Once 67% of stakers signed a skip message that block is skipped and the next round begins with a block proposed by the next staker.

3.2 Safety and Liveness

By definition, a decentralized system must be susceptible to malicious majority attacks whether by hashrate, stake, or other permissionlessly-acquirable resources.

$$\begin{tikzcd}
 & & B_2' & 67\% \\
B_0 \arrow[r] & B_1 \arrow[r] \arrow[ur] & B_2 & 67\%
\end{tikzcd}$$

Figure 2: Finalizing a block requires a supermajority of 67%. Therefore, the chain can fork only if an attacker controls at least 67% of all stake and at least 34% of all stake is slashed.

In our mechanism, forks require burning 34% of all stake. Forks are impossible if 34% of all stakers are honest or offline. Violations of the 34% assumption can be mitigated to some degree by selecting the chain that requires the most capital to burn to attack its

[a]A staker is incentivized not to wait for too long before broadcasting their block because otherwise the other stakers will skip it. Also they cannot speed up the block time much, as we will see in the next chapter.

finality. That means a block with more votes wins over a block with fewer votes. E.g., in the ideal case a block has 100% of all votes; then 100% of the total stack has to get burned to overwrite that block. Such a supermajority block finalizes all previous blocks, because overwriting any previous block requires to also overwrite that supermajority block when the fork reaches the same height. Waiting for a supermajority block to confirm a transaction has a similar effect as waiting for confirmations in a proof-of-work mechanism, because it increases the cost to attack the chain.

$$\begin{tikzcd}
 & & B_2' & 67\% \\
B_0 \arrow[r] & B_1 \arrow[ur] \arrow[r] & B_2 & 99\%
\end{tikzcd}$$

Figure 3: The fork with the most votes wins because reverting it requires the most stake to burn.

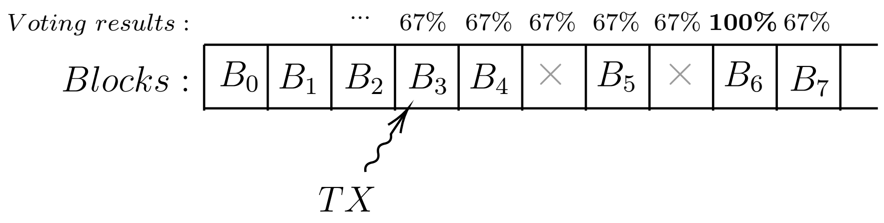

Figure 4: Finality of a transaction depends on the best voting result that comes after it. In this example, the votes for block B_6 guarantee that 100% of all stakes have to get burned to reverse the transaction in block B_3.

To harden voting results, the next leader commits in their block to all votes received for the previous block. At least 67% of all votes are required for a successor block to be valid. Honest stakers try to collect as many votes for the preceding block as possible while waiting to broadcast their current block. Majorities larger than 67% make finality proportionally more costly to attack.

Liveness requires a honest majority of 67%. Thus, an attacker can halt the network as long as 34% are offline or malicious. They can halt the chain until honest stakers stake enough bitcoins such that they can form again a 67% majority to fill up the missing votes to finalize and unstall the current block. In the worst case 100% of all stakers are offline. Then new stakers have to stake two times the total stake to form a new 67% majority.

$$\begin{tikzcd}
 & & & 34\% \\
B_0 \arrow[r] & B_1 \arrow[r] & B_2 & 66\%
\end{tikzcd}$$

Figure 5: Liveness requires a majority of 67%. So, 34% malicous or offline stakers can halt the chain.

However, introducing this recovery mechanism for stalled chains introduces a new attack vector. Now we have to mitigate the case that newer majorities overwrite older majorities' final blocks. We define the *age* of a staker set as the bitcoin block height at which the last staker joined that set. Honest full nodes select a block by an earlier majority over a block by a newer majority. Therefore, a block cannot get overwritten by a majority that formed after that block was finalized. This ensures again that an attacker has to burn at least 34% of all stake to create a fork.

For more strict finality, we enforce sideblocks to commit to the most recent bitcoin block hash.

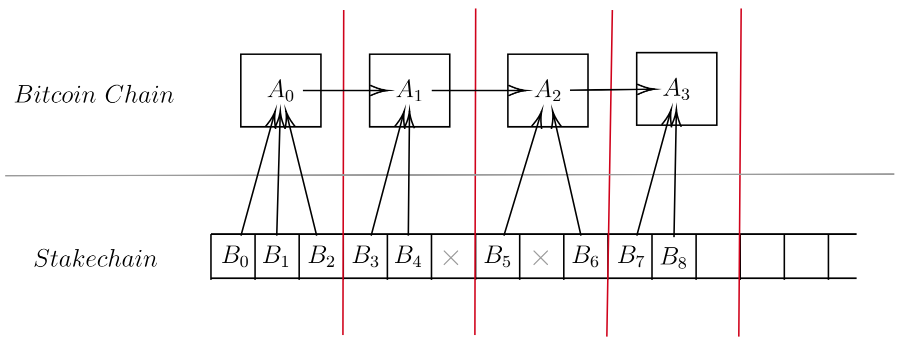

Figure 6: Sidechain blocks have to commit to bitcoin block hashes. The height of the bitcoin block must be at least the same height as in the preceding sideblock. In this example there are at most 3 sideblocks per mainblock. This ensures basic synchronicity.

We allow new stakers to change previous majorities only up to a certain number of bitcoin blocks in the past. E.g. Stakers can vote at most one week (1008 bitcoin blocks) back in the past. This implies, if the chain ever stalls for a week it will stall indefinitely until the current stakers finalize the current block. New stakers cannot overwrite a one week old chain. However, if that happens and there's a malicious 67% majority, then the sidechain's incentive model must be already fundamentally broken in some other way.

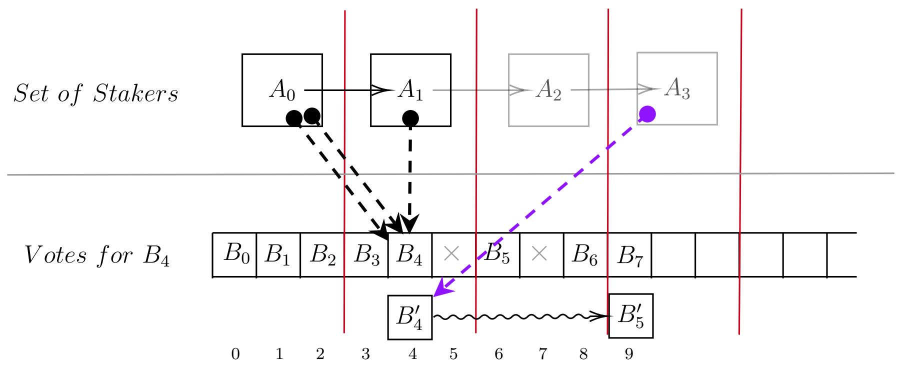

Figure 7: In this example the older stakers voted for B_4 but a newer majority votes for B'_4. This chain split will get resolved at the latest in slot number 9, because then the set of stakers is unambiguous again. Honest nodes don't accept a rewrite by a newer majority, if they already know a block finalized by an older majority. Additionally, the newer majority can vote at most 1 week back in the past to recover a stalled chain.

An even stricter model is to enforce stakers to periodically publish signatures in the bitcoin blockchain to remain within the staker set. However, this costs block space and fees, so we want to minimize the on-chain footprint. The most simple anchor is to let new stakers commit to sidechain blocks within their funding transaction for the staking contract within the bitcoin blockchain. Incentives are aligned if we enforce that new stakers can only vote on the same chain they committed to. Other stakers implicitly sign the on-chain commitment by accepting the new staker's votes and signing it in subsequent blocks.

3.3 Ongoing Cost to Attack the Chain

One might argue that time value of locked bitcoins is too cheap to derive a secure consensus. That is a misunderstanding of the fundamental underlying market mechanism.

In Bitcoin, the proof of work security is not determined by the price of electricity, but by the Dollar value of the block revenue. Miners have an incentive to spend about

$$1 \text{ revenue} = 1 \text{ reward} + \text{fees} \approx 7.25 \text{ BTC} \approx \$250000$$

to produce a block (as of June 2021). The mining difficulty adjustment stabilizes this price of block production. The cost of block production equals the cost of attacking the chain. So, more demand for Bitcoin leads to a higher bitcoin price and thus, a higher value of the block revenue and subsequently, to more security. The exact same market forces align the incentives for our proof of stake. In a free market the total time value of all staked bitcoins converges against 1 sidechain revenue per block. Whenever there is less time value locked, someone will stake their bitcoins to earn the cheap sidechain revenue.

Therefore, the ongoing cost for an attacker to stall the chain is about 1 sidechain revenue per block time. This is independent of the consumed resource. Security depends only on the value of the sidechain's revenue. The sidechain asset price is mostly driven by usage and resulting network effects. Thus, a sidechain provides security proportional to the network effect of its user base. The seemingly low investment of time value of bitcoins is automatically priced in by the competition for the stakechain rewards.

Additionally, it is important to notice that overwriting the chain costs slashing 34% of all stake, which is orders of magnitudes more expensive than a year worth of time value of that stake. For that reason, stakechains have a quick settlement finality because a reorg is significantly more expensive than the cost of block production.

3.4 Limitation: Pegs and the Altcoin Problem

This system requires an independent asset per stakechain. A stakechain's security is limited by its asset's value. Only if a sidechain's revenue is sufficiently valuable, it can motivate many Bitcoin holders to protect it. Therefore, only large sidechains with lots of users and valuable assets can provide security. Small sidechains are insecure.

We do not want to introduce another speculative asset because that creates unnecessary friction to users. Trustless two-way pegs between bitcoin and sidechains are an ongoing research topic. As of today there are at least two workarounds to peg sidechain assets to BTC. For example, the peg of the Liquid Sidechain is a federated 2-way-peg. A trustless alternative is a perpetual one-way peg [11]. It dampens the price fluctuation of a sidechain's asset. Additionally, it is possible to demand from stakers to burn a certain amount of BTC to become a staker. Those burned BTC can be issued in the sidechain as block subsidy. Existing research [12] and future research on *trustless* two-way pegs e.g., based on zero-knowledge proofs [13] can be developed and deployed on top of a stakechain consensus.

4 Conclusion

We have proposed a consensus-mechanism that potentially scales to a global payment system. 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. Stakers 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. Stakers 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. Spawning multiple stakechains scales horizontally to potentially billions of users.

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

[10] 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.

[11] Ruben Somsen. 21 million bitcoins to rule all sidechains: The perpetual one-way peg. *Medium*, 2020.

[12] Jason Teutsch, Michael Straka, and Dan Boneh. Retrofitting a two-way peg between blockchains. *arXiv preprint arXiv:1908.03999*, 2019.

[13] Alberto Garoffolo, Dmytro Kaidalov, and Roman Oliynykov. Zendoo: a zk-snark verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains. *CoRR*, abs/2002.01847, 2020.

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

[15] 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.

[16] 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.

A Appendix

A.1 Staking Contracts in Bitcoin Script

Currently, burning funds is not supported in Bitcoin script. Committing to a certain spending transaction requires a construct called *Covenants*[b]. Yet, upcoming Bitcoin features such as `SIGHASH_NOINPUT` [c] or. `OP_CHECKTEMPLATEVERIFY` [d] or `OP_CAT` allow trustless covenants. For now, we need a workaround to implement the staking contract. In the following we discuss solutions. All of these trust-minimizing workarounds are highly undesirable additional complexity. Fortunately, in the long term, we will certainly have some clean solution for covenants.

A.1.1 Trust-minimized Workaround 1

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 [14].

Containing more than 60 signatures, the punishment transaction becomes large and expensive, but as long as the staker 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

[b]https://medium.com/blockstream/cat-and-schnorr-tricks-i-faf1b59bd298
[c]https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki
[d]https://github.com/bitcoin/bips/blob/master/bip-0119.mediawiki

[15]. When Schnorr signatures become available in Bitcoin, such aggregated signatures will become more simple because of the linearity of Schnorr's scheme [16].

The Bobs could be a trusted federation or a percentage of the current stakers. 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. This way Bob doesn't learn what he signs until Alice publishes her contract.

A.1.2 Trust-minimized Workaround 2

The bitcoin mailing list member, *ZmnSCPxj*, came up with the following trust-minimized covenant construction based on replace-by-fee, which requires no softfork.

We can implement the staking contract with a simple

```
<one year> OP_CHECKSEQUENCEVERIFY OP_DROP <A> OP_CHECKSIG
```

`OP_CHECKSEQUENCEVERIFY` ensures, as a side effect, that the spending transaction opts in to replace-by-fee. Thus, if the pubkey `<A>` is used in a single-sign signature scheme (which reveals the privkey if double-signed), then at the end of the period, anyone who saw the double-signing can claim that fund and thus act as "Bob". Indeed, many "Bob"s will act and claim this fund, increasing the fee each time to try to get their version onchain. Eventually, some "Bob" will just put the entire fund as fee and put a measly `OP_RETURN` as single output. This "burns" the funds by donating it to miners.

From the point of view of Alice this is hardly distinguishable from losing the fund right now, since Alice will have a vanishingly low chance of spending it after the collateral period ends, and Alice still cannot touch the funds now anyway. Alice also cannot plausibly bribe a miner, since the miner could always get more funds by replacing the transaction internally with a spend-everything-on-fees `OP_RETURN` output transaction, and can only persuade the miner not to engage in this behavior by offering more than the collateral is worth (which is always worse than just losing the collateral).

A `OP_CHECKTEMPLATEVERIFY` would work better for this use-case, but even without it you do not need a trusted party to implement the staking contract.

Drawback of this solution is that cheating stakers are not immediately removed from Bitcoin's UTXO set. Yet, this might be a decent tradeoff because we have a succinct proof to exclude a cheating staker: knowledge of his private key.

Another drawback is that it requires trust in miners. If Alice cooperates with a significant share of Bitcoin's hash power, she has proportional chances of mining the transaction herself. Then she would not have any cost of attacking the chain.

A.2 Staker Signatures

In this chapter we discuss details of the stakers signatures. We explain how to pre-commit to a particular nonce per block efficiently. Finally, we explain a scheme for better hot key security.

A.2.1 Compact Nonce Commitments

Constructed naively, stakers would have to pre-commit to millions of nonces within their funding transaction. A more efficient construction is to let stakers subsequently commit to their next nonce by signing it within their previous signature. This amortizes the inclusion proof size to basically zero. Furthermore, each staker has to store only one nonce at each point in time.

Jeremy Rubin contributed this scheme for constant-sized commitments to sequences of nonces. Furthermore, he implemented a staking contract in Sapio[e].

A.2.2 Hot Key Security

Hardware wallets are usually stateless and therefore they're incompatible with nonce commitments. Thus, to protect the stakers' nodes, they can use a *staking 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 staking key are a much less attractive target for attackers.

[e] https://github.com/sapio-lang/sapio/blob/master/sapio-contrib/src/contracts/staked_signer.rs