Mainstay: enabling trustless immutability in federated sidechains

Tom Trevethan, Nicholas Gregory and Nikos Kostoulas
CommerceBlock
tom@commerceblock.com

Abstract

We present *Mainstay*: a system and method that enables sidechains - employing federated consensus algorithms - to cryptographically bind to an immutable proof-of-work mainchain in a way that it is impossible for the sidechain to fork without simultaneously forking the mainchain. The underlying construction of this method requires no changes to the existing proof-of-work system (e.g. Bitcoin) and is space-efficient and censorship resistant due to the use of homomorphic commitments based on the pay-to-contract protocol. In addition, the method is implemented using a multisignature scheme to maintain consistency with the security guarantees and Byzantine fault tolerance of federated consensus. These protocols enable the realisation of highly efficient, scalable and interoperable sidechain systems that can incorporate advanced smart-contracting and privacy features while simultaneously exploiting the full security and immutability of the Bitcoin blockchain secured via proof-of-work.

Introduction

The invention of Bitcoin solved the issue of double-spending in a fully decentralised digital payments system by ensuring that there is a single, replicated, global ledger that all participants can agree represents the valid ordering of transactions: the blockchain. Reaching consensus on the state of this global ledger is achieved using proof-of-work: adding to the blockchain requires expensive, but easily verifiable, computations that are rewarded with tokens derived from transaction fees (and block rewards). The blockchain with the most accumulated work is considered the only valid history, and all participants are incentivised to contribute their computational work to extending it [1].

The use of proof-of-work [2], which requires the consumption of real world resources (i.e. energy), as the consensus mechanism means that Bitcoin is completely *permissionless*: no permission is required in order to add to the blockchain - only computational work. The work required to extend the blockchain also leads to immutability: any attempt to modify the time-order of transactions in the blockchain requires more computational power than the rest of the network combined [3]. This leads to the Bitcoin blockchain being a unique global system of consensus on the ordering of time-stamped events without the need for any trusted authority.

Of all the cryptocurrency projects that have been launched since, Bitcoin remains by far the most secure, with the most accumulated work. The Bitcoin network has operated persistently for over 9 years, holding hundreds of billions of dollars in value and has resisted constant attack. However, these properties come at the cost of both scalability and upgradability. In order to maintain the decentralisation, security and censorship resistance of the network, block sizes must remain relatively small which limits the transaction capacity [4]: Bitcoin can process only 3 to 6 transactions per second leading to unpredictable transaction fees and confirmation times [5]. In addition, for the very same reasons Bitcoin is so secure, it is also very difficult to change the protocol: adding new features requires the consent of all network participants and must be done extremely conservatively so as to not risk the integrity of the system.

Alternative consensus mechanisms on separate blockchains can be used to improve scalability and build in more advanced features at the protocol level [6]. Sidechains to Bitcoin secured by federated consensus rules enable significantly better scalability, and much faster and more regular block times [7]. In addition, these systems can incorporate more protocol-level functionality, including native smart contracts, asset issuance and cryptographic privacy and anonymity features not possible on Bitcoin [8]. However, such systems are not able to achieve the trustless immutability of Bitcoin with permissionless proof-of-work. Blockchains that are run by a static federated consensus mechanism require collective trust in the federation members: if the federation members collude or leak a threshold of secret keys, conflicting forks of the blockchain can be created at no cost and double-spend attacks launched against token holders [8].

To provide federated *sidechains* with the same level of trustless immutability as Bitcoin, we describe a method that involves cryptographically binding these sidechains to the Bitcoin mainchain in such a way that the sidechain cannot be forked without also simultaneously forking the Bitcoin mainchain. This means that for a fixed set of federated block signers, users of a sidechain do not need to trust the federation to protect them from a double-spend attack: consensus on a single unforked version of the federated sidechain is enforced by Bitcoin's proof-of-work.

This protocol - *MainStay* - employs the underlying concept of a *staychain* of linked transactions within the Bitcoin mainchain, where all transactions in the staychain are enforced to conform to a single output, preventing branching and any possibility of alternate staychain histories. By anchoring the staychain transaction ID into the genesis block of the sidechain, and then committing the state of the sidechain at regular intervals into the staychain, it becomes impossible to roll back or re-write the state of the sidechain without also rolling back the staychain, which is effectively impossible due to the might of Bitcoin's global proof-of-work. Sidechain nodes can validate these commitments and the resulting immutability of the staychain using only lightweight SPV[1] proofs from full Bitcoin nodes. When a sidechain block has been committed to a Bitcoin staychain, we say that block is *reinforced* and is as immutable as a Bitcoin block of the same depth.

To minimise the encumbrance of the mainstay on the Bitcoin blockchain, and to prevent any potential miner censorship of transactions containing OP_RETURN outputs, we employ a homomorphic commitment scheme based on the 'pay-to-contract' (BIP175) protocol [9]. In this approach, commitments from the sidechain are embedded in a single transaction output address, and the staychain is indistinguishable from normal Bitcoin payment transactions. We have designed the scheme so that it is compatible with both multisig (P2SH[2]) and single public key (P2PKH) addresses, and that no additional hosted data is required in order to verify the validity of the mainstay. The remainder of this paper is organized as follows: the next section describes the current state-of-the-art in relation to blockchain attestation and timestamping. The sections following this then describes the proposed scheme, starting with the core principles and then specifying the details of the implementation for a federated consensus protocol.

Attestation and time-stamping

It was recognised early in Bitcoin's history that the blockchain could be utilised to timestamp arbitrary data in a completely trustless and decentralised way [10]. By embedding a cryptographic *commitment* to a piece of data into a valid transaction, which was then mined into the blockchain, it was possible to prove that the data existed at a particular time [11]. In the simplest implementation of this idea, the commitment is a cryptographic hash of the data which is used directly as a payment address (which would be un-spendable). This approach however led to these unspendable addresses remaining in the UTXO set indefinitely, unnecessarily burdening fully validating nodes. To accomodate time-stamping (and other meta-protocols) in a more efficient way, a new prunable transaction output type was introduced via a new OP code: OP_RETURN [12]. This allowed up to 40/80 bytes to be included in an output which was not treated as a spendable output in the UTXO[3] set. The use of OP_RETURN however has significant downsides: it bloats transactions (resulting in higher transaction fees), it offers no privacy (data is included in plain text directly into the transaction) and transactions including them are often rejected (censored) by some mining pools.

There are many services that employ OP_RETURN outputs to time-stamp data onto the Bitcoin blockchain, such as EternityWall [13], Proof of Existence [14] and BlockNotary [15]. Beyond this, there are protocols that can include a much more extensive set of data into a single commitment, such as OpenTimestamps which collects submitted commitments via a

[1] Simplified payment verification: a lightweight transaction confirmation protocol only requiring block headers backed by proof-of-work.

[2] Pay-to-script-hash

[3] Unspent Transaction Output

calendar server and compresses them into Merkle Tree, and then time-stamps the Merkle Root in a transaction. The Factom protocol also time-stamps the state of the Factom system (including Merkle roots of transaction and other data) into the Bitcoin blockchain at regular intervals as part of their value proposition [17].

This type of time-stamping is however fundamentally limited in the type of immutability it can provide. A timestamp can only prove that a particular piece of information existed at a certain point in time, not that the information has any other validity or provenance. A timestamp by itself cannot prove that a commitment to conflicting data has not also been simultaneously timestamped. This is a critical concept in relation to immutability: any proof-of-existence does not act as a proof that anything else (e.g. an alternative ordering of transactions) does not also exist.

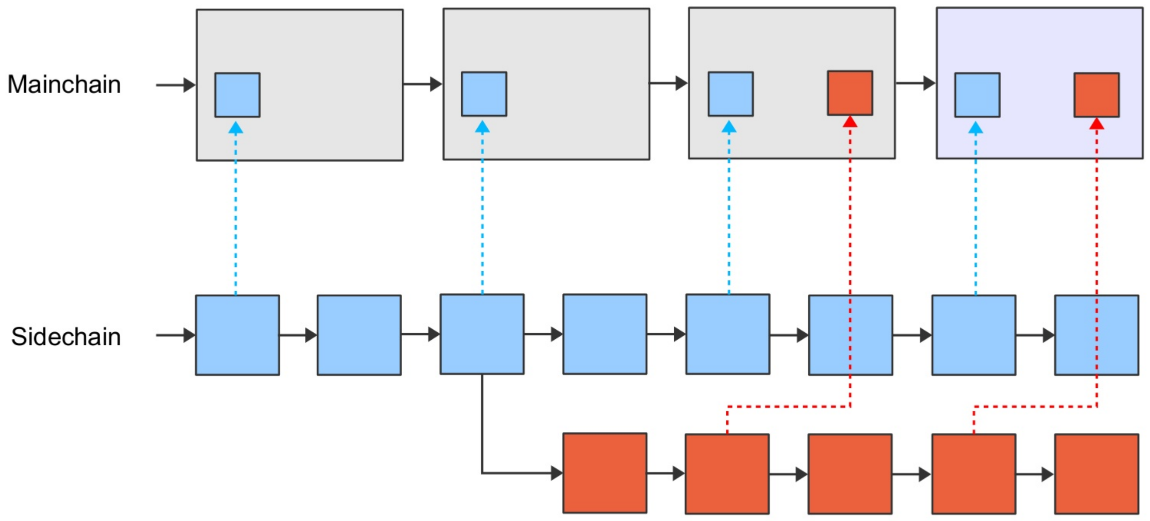


Figure 1. Illustration of conflicting (forked) sidechain blocks simultaneously attesting to a mainchain.

To illustrate this point, we consider a sidechain or alt-chain whose state (i.e. the chain tip block header) is periodically time-stamped into the Bitcoin blockchain. This does not lead to immutability of the sidechain, since alternative conflicting states (i.e. forks) can also be time-stamped simultaneously (see Fig. 1). Any property of immutability then must ensure that the sidechain state is linked to a specific commitment in the Bitcoin mainchain via some trusted mechanism - some authority (who may be effectively operating the sidechain) is responsible for defining the sequence of timestamps that correspond to the un-forked sidechain. This then relies on the integrity of the commitment mechanism: multiple versions of a sidechain can be created with multiple simultaneous timestamped commitments into Bitcoin. This could be used to execute a double spend attack by collusion of a block signing federation with the commitment authority.

The MainStay protocol is designed to eliminate the requirement for any type of trusted commitment mechanism and to provide cryptographic proof of sidechain immutability by initiating a *fan-in-only* transaction *staychain* within the Bitcoin blockchain that is uniquely committed to the genesis block of the sidechain, as described in the next section. The protocol does not employ OP_RETURN outputs, providing additional privacy, censorship resistance and efficiency.

The Mainstay protocol

The aim of the MainStay protocol is to restrict the sequence of periodic commitments of the state of a sidechain to an un-forkable *staychain* of transactions in Bitcoin, and to uniquely identify this staychain by linking to it directly from the sidechain itself. We define a staychain as a sequence of linked transactions where each one has only a *single* output - transactions can have more than one input (fan-in), but maintaining single outputs means only one sequence of commitments is possible from a given initial transaction (Fig. 2). Each unique transaction output then represents a *single use seal* [18].

If the security proposition of a sidechain depends on the integrity of the mainstay then the mechanism of propagating the staychain must be robust and immune from attack: if the staychain fails to propagate or is corrupted (e.g. having multiple outputs) then the sidechain will lose the guarantee of immutability. We describe mechanisms to ensure this integrity in the case of a federated sidechain consensus model in section 3.1, however in the following general description of the protocol, we assume a single mainstay key and signing entity. The protocol is also presented in relation to Bitcoin as the proof-of-work mainchain, but it is in principle compatible with any PoW blockchain.

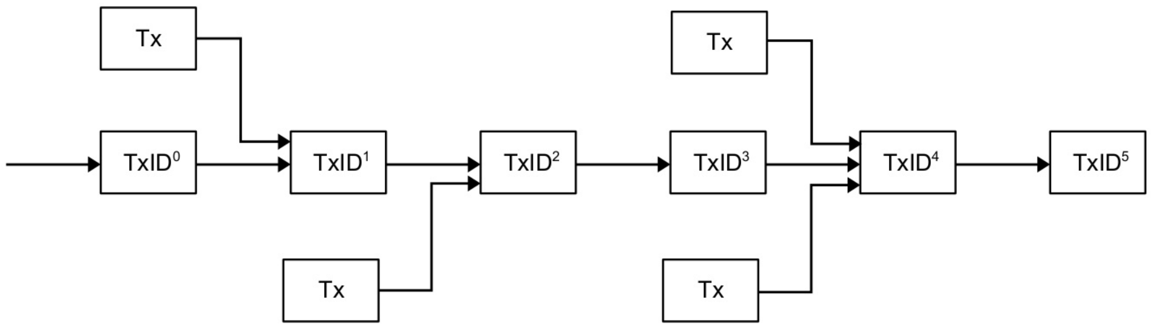


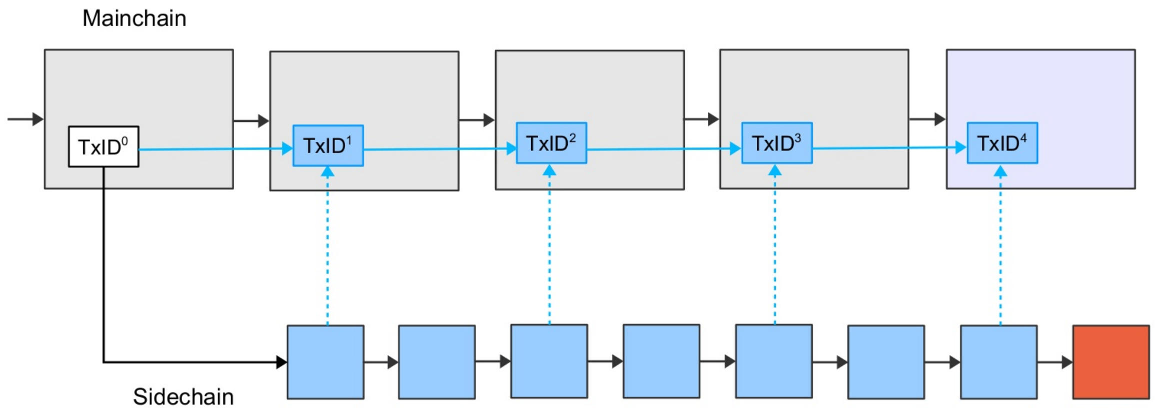
Figure 2. A schematic of a *fan-in-only* chain of linked transactions. By enforcing single outputs only one possible sequence of transactions is possible.

Initialisation

The initial step in the protocol is the creation of the *base* transaction in Bitcoin, which is performed before the initialisation of the sidechain.

1) The signing entity E generates a secret key sk_0 , and corresponding *base public key* $pk_0 = sk_0 \times G$ ($\times G$ denotes multiplication of the generator point on the secp256k1 elliptic curve [19])[4]. The public key is then used to create the *base address*: $Addr_0$
2) Funds are paid (using P2PKH[5]) to the base address (either by entity E or a related party/wallet) on the Bitcoin blockchain (in 1 or more transactions) to at least cover the initial transaction fees.
3) Entity E then creates a transaction (the *base transaction*) paying these funds (potentially consisting of more than one output) again to the same $Addr_0$ in a single output (P2PKH).
4) This transaction is broadcast to the network: once it is confirmed[6] in the Bitcoin blockchain it acquires a unique transaction ID that is a pointer to the start of the staychain: $TxID^0$. This transaction scriptSig also now contains the base public key pk_0 .

At this point, the sidechain can be initialised and linked to the Bitcoin staychain. The pointer $TxID^0$ is embedded directly in the genesis block of the sidechain in a defined location.



[4] Private keys in Bitcoin are 256-bit integers modulo the order the secp256k1 elliptic curve, and public keys are (x,y) coordinates on the curve (512-bit numbers).

[5] Pay-to-Public-Key-Hash: a Bitcoin output type where payment is made to a 160-bit hash of a public key (address).

[6] The transaction ID can be modified by a malicious third party without affecting its validity (transaction malleability). Therefore several confirmations (blocks) should be required before the TxID can be considered final.

Figure 3. Schematic of the mainstay protocol. Dashed lines represent homomorphic commitments.

Attestation

The frequency of Bitcoin attestations is determined by the entity E : the sidechain may generate blocks more frequently but can only attest once per Bitcoin block (average every 10 minutes). The process of attestation will occur as follows:

1) At each interval j (initially $j = 1$), the E will retrieve the sidechain best block hash h_B^j .
2) The base public key pk_0 is modified (via a homomorphic commitment)[7] with the elliptic curve point corresponding to h_B^j :

   $pk_0^j = pk_0 + h_B^j \times G$
3) Entity E then creates a Bitcoin transaction with an input spending the single output of $TxID^{j-1}$ (initially the base transaction $TxID^0$ when $j = 1$) and paying to a single P2PKH output with an address derived from pk_0^j : $Addr_j$
4) The transaction is then signed using the corresponding private key for the $TxID^{j-1}$ output address: $Addr_{j-1}$ (of public key pk_0^{j-1}) which is determined from integer addition modulo the order of the secp256k1 curve.

   $sk_0^{j-1} = sk_0 + h_B^{j-1}$

   (for $j = 1$ $sk_0^0 = sk_0$)

   The private key validity is a result of the homomorphic properties of elliptic curve point multiplication [20]:

   $pk_0^j = pk_0 + h_B^j \times G = sk_0 \times G + h_B^j \times G = \left(sk_0 + h_B^j\right) \times G$
5) The valid transaction is then broadcast to the Bitcoin network. Once it is confirmed in a block, it is referenced by transaction ID: $TxID^j$

Mainstay confirmation

A block generated on a sidechain that has a mainstay commitment is known as *reinforced* and has the same immutability guarantees as a confirmed Bitcoin block. For any client or user to confirm the status of a sidechain block only requires lightweight SPV proofs from

[7] Addition of elliptic curve points

both Bitcoin and sidechain full nodes. No additional information, beyond what is included in the sidechain and Bitcoin blockchains, is required to validate mainstay reinforcements[8].

This confirmation functions as follows:

1) The base transaction ID ($TxID^0$) is retrieved from the sidechain genesis block
2) $TxID^0$ is located in the mainchain (Bitcoin) blockchain and the base public key pk_0 retrieved from its scriptSig.
3) The staychain is tracked[9] until the unspent tip ($TxID^t$) is located, confirming each component transaction consists of only a single output[10]:

   $TxID^0 \rightarrow TxID^1 \rightarrow TxID^2 \rightarrow TxID^3 \rightarrow ... \rightarrow TxID^t$
4) The single output P2PKH address of $TxID^t$ is retrieved: $Addr_t$
5) Starting at the tip (most recent confirmed block) of the sidechain (block height w) with block hash h_B^w , $Addr_t$ is checked to determine if it incorporates the homomorphic commitment to h_B^w :

   $Addr\,(pk_0 + h_B^w \times G) =?Addr_t$
6) If not true, the sidechain block height is decremented: $w \leftarrow w - 1$ and the check repeated.
7) When evaluated as true, block w on the sidechain (and all below it) are confirmed as reinforced.

The above protocol would only need to be followed for the initial sync of a mainstay connected node: once the staychain tip transaction ($TxID^t$) has been located, additional attestations can be confirmed by monitoring when $TxID^t$ is removed from the Bitcoin UTXO set. The new tip $TxID^{t+1}$ will then be included in the most recent Bitcoin block.

Staychain fees

To maintain the persistent operation of a staychain, it must be continually funded to pay for mainchain (Bitcoin) mining fees. The staychain can always be funded with a substantial

[8] However, if the transaction ID of the staychain tip is provided to the client can substantially reduce the time required to verify the staychain.

[9] This requires the use of a searching algorithm that determines the spending TxID of a given output. If the staychain tip TxID is cached or retrieved from a third party, the staychain can be verified from the tip down which is much faster.

[10] These confirmations for each transaction can come from SPV proofs in the case of a lightweight client.

amount of Bitcoin at the beginning (i.e. at the base transaction stage) however it may be required to 'top-up' the funding at a later stage. This is possible without breaking the immutability of the staychain: the only required condition for immutability is that there is always only one output of any transaction in the chain - and that the staychain cannot bifurcate. Inputs however can be added by *anyone*: additional funding can be added with SIGHASH_ANYONECANPAY inputs. The base transaction will always define the commitment sequence through to the tip.

Federated sidechains

An important property of the mainstay protocol is that it does not require trust in any party, including the entity holding the staychain base private key (sk_0) to confirm that a given sidechain state is immutable. However trust *is* required in this entity to ensure that the mainstay is persistent, and that the system continues to operate (i.e. commitments continue to be generated). If the key was stolen then an attacker could steal the Bitcoin in the staychain tip output and prevent further confirmations. To remedy this, the sidechain would need to be hard-forked to reset the mainstay (i.e. commit a new base transaction into the sidechain).

Sidechains can be operated using a federated consensus protocol, where a fixed federation of separate entities are required to cooperate to generate a new block to add to the blockchain [7]. This is typically implemented with m distinct entities, where a threshold of n are required to add their signature to generate a new valid block. This has the advantage of being very scalable and efficient, and also retains some level of decentralisation, not requiring trust in any single entity. In the case of a federated sidechain employing a mainstay to Bitcoin, the operation of the mainstay can achieve the same security properties and guarantees as the federated block signing protocol. In this case, the staychain would be controlled with an n of m multisignature script: n signers are required to cooperate to operate the mainstay. $m - n$ keys can be lost or compromised and the mainstay will still function.

This requires some modifications to the protocol described above, as follows.

Initialisation

1) Each signing node i ($i = 1, ..., m$) generates a secret key sk_i , and corresponding public key $pk_i = sk_i \times G$

2) Each signing node i publishes their public mainstay key pk_i and funds (to cover initial mining fees) are paid to each of them (by some related entity/wallet) via P2PKH.

3) The signing nodes then cooperate to create an n of m multisig *redeem script* (where

m is the total number of signing nodes and n is the number of signatures required)[11] containing each of the m base public keys (pk_i). The redeem script is then hashed[12] to create a P2SH address.

4) A transaction is then created (by all signing nodes) with the P2SH address as a single output and funded with with the m inputs spending from the P2PKH outputs created in step 2. This transaction is signed by each signing node in turn with the corresponding sk_i in turn (using SIGHASH_ALL), and then broadcast to the Bitcoin network.

5) Once confirmed, each of the m public keys pk_i are revealed on the Bitcoin blockchain as each of the (scriptSig) transaction inputs. It is now publicly verifiable that the redeem script hash corresponds to the published n , m and all the pk_i ($i = 1, ..., m$).

6) The TxID of the transaction ($TxID^0$) is retrieved and committed into the genesis block of the sidechain.

Mainstay attestation

1) At each attestation interval j (initially $j = 1$), each of the mainstay signing nodes will retrieve the sidechain tip block hash h_B^j .

2) Each of the m public keys pk_i is tweaked with h_B^j : $pk_i^j = pk_i + h_B^j \times G$ ($i = 1, ..., m$) by each signing node.

3) n of m signing nodes will then create a transaction spending the single output of $TxID^{j-1}$ and paying to a single P2SH output with an n of m multisig redeem script formed from all m tweaked public keys pk_i^j ($i = 1, ..., m$) in order.

4) Each n of m signing nodes then verify that the redeem script hash consists of n , m and pk_i^j ($i = 1, ..., m$) in order.

5) The transaction is then signed by each of n (any subset of m) signing nodes in turn using the private key (sk_i^{j-1}) corresponding to the tweaked public key (pk_i^{j-1}) used in the redeem script for $TxID^{j-1}$

   $sk_i^{j-1} = sk_i + h_B^{j-1}$ for ($i = 1, ..., n$)

   (for $j = 1$ $sk_i^0 = sk_i$)

6) The transaction is then broadcast to the Bitcoin network, validated and then mined into a block, generating $TxID^j$

[11] The redeem script takes the format: OP_n <pk1> <pk2> . . . <pkm> OP_m OP_CHECKMULTISIG

[12] HASH160

Mainstay confirmation

1) The base transaction ID ($TxID^0$) is retrieved from the sidechain genesis block.

2) $TxID^0$ is located in the mainchain (Bitcoin) blockchain and the m base public keys pk_i ($i = 1, ..., m$) determined from the base redeem script.

3) The staychain is tracked until the unspent tip ($TxID^t$) is located, confirming it consists of only single output transactions:

   $TxID^0 \rightarrow TxID^1 \rightarrow TxID^2 \rightarrow TxID^3 \rightarrow ... \rightarrow TxID^t$

4) The $TxID^t$ UTXO redeem script hash is determined: h_R^t

5) Starting at the tip (best block) of the sidechain (block height w) with block hash h_B^w , h_R^t is checked to determine if it is generated from the multisig ordered list of base public keys (the redeem script), each with homomorphic commitments to h_B^w :

   $Hash_{160}\left(n|pk_1 + h_B^w \times G|...|pk_m + h_B^w \times G|m\right) =?h_R^t$

6) If not true, the the sidechain block height is decremented: $w \leftarrow w - 1$ and the check repeated.

7) When evaluated as true, block w on the sidechain (and all below it) are confirmed as reinforced.

Conclusion

We have described a system that enables a blockchain (or sidechain) secured by a federated consensus protocol to gain the same level of trustless and decentralised immutability as Bitcoin, but without requiring an independent proof-of-work. This leads to several advances: federated sidechains can provide much higher transactional throughput at lower latency than Bitcoin as well as more advanced tokenization and privacy features. By employing a mainstay, a sidechain can retain these properties and also gain the unique immutability that only a restricted (in terms of block size and time) decentralised Bitcoin can achieve - all while only placing minimal burden on the Bitcoin blockchain. A federated blockchain with a mainstay to Bitcoin will remain more centralised and lack the censorship resistance of the Bitcoin blockchain, however this can be an advantage in many situations, such as when a sidechain is being used for a particular purpose, such as issuing tokenized assets, and control over transaction permissions is desirable.

[1] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).

[2] A. Back, "Hashcash - a denial of service counter-measure" (2002)

[3] Rosenfeld, Meni. "Analysis of hashrate-based double spending." arXiv preprint arXiv:1402.2009 (2014).

[4] Decker, Christian, and Roger Wattenhofer. "Information propagation in the bitcoin network." Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on. IEEE, 2013.

[5] Croman, Kyle, et al. "On scaling decentralized blockchains." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2016.

[6] Back, Adam, et al. "Enabling blockchain innovations with pegged sidechains." URL: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains (2014).

[7] Dilley, Johnny, et al. "Strong Federations: An Interoperable Blockchain Solution to Centralized Third Party Risks." arXiv preprint arXiv:1612.05491 (2016).

[8] Poelstra, Andrew, et al. "Confidential assets." Financial Cryptography Bitcoin Workshop. https://blockstream. com/bitcoin17-final41.pdf 2017.

[9] Shibli, Omar and Nicholas Gregory. “BIP 175: Pay to Contract Protocol” https://github.com/bitcoin/bi
0175.mediawiki 2017

[10] Gipp, Bela, Norman Meuschke, and André Gernandt. "Decentralized trusted timestamping using the crypto currency bitcoin." arXiv preprint arXiv:1502.04015 (2015).

[11] Clark, Jeremy, and Aleksander Essex. "Commitcoin: Carbon dating commitments with bitcoin." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2012.

[12] Sward, Andrew, Ivy Vecna, and Forrest Stonedahl. "Data Insertion in Bitcoin’s Blockchain." Ledger 3 (2018).

[13] https://eternitywall.it

[14] https://proofofexistence.com

[15] https://www.blocknotary.com

[16] https://github.com/opentimestamps

[17] Snow, Paul, et al. "Factom Business Processes Secured by Immutable Audit Trails on the Blockchain." Whitepaper, Factom, November (2014).

[18] Todd, Peter https://petertodd.org/2016/commitments-and-single-use-seals

[19] SEC, SECG. "2: Recommended elliptic curve domain parameters." Standards for Efficient Cryptography Group, Certicom Corp (2000).

[20] Comandini, Leo “sign-to-contract: how to achieve digital notarization with zero marginal

cost” https://github.com/LeoComandini/Thesis