SyncChain
Synchronized Sidechains for Improved Security and Usability

DRAFT White paper

Author: Sergio Lerner, IOV Labs Chief Innovation Officer
Revision: 6
Date: March 3, 2020

Abstract

In this article we present a new type of merge-mined sidechain we call SyncChain, which enables fast peg-ins and peg-out, and we focus on how this new technology can be applied to RSK. When applied to RSK, the peg-in time could be as low as 30 minutes and the peg-out time as low as 2 hours. A SyncChain can also be enhanced with anchoring to unconditionally protect the peg-outs from double-spends, at the expense of a higher confirmation time. This is a huge improvement from all existing sidechains as they require hundreds of block confirmations, and base their security on cryptoeconomic assumptions. The SyncChain protocol, which was built over some ideas we published in 2016, is orthogonal to the method used to unlock peg-outs, so a a Federated SyncChain is possible. We also discuss the pros and cons for RSK to migrate from a Bitcoin SPV-sidechain to a Bitcoin SyncChain, and we lay down a tentative plan for a SyncChain network upgrade.

Index

Introduction

When RSK was initially designed, one of the design goals was to keep RSK independent of Bitcoin as much as possible to prevent any failure in Bitcoin from affecting RSK. During the 2015-2016 period Bitcoin was under continuous pressure and threats of contentious hard forks, and transaction fees rose considerably, preventing the use of Bitcoin for financial inclusion, which requires cheap payments. For these reasons RSK adopted a merge-mining consensus over being a counterparty-like overlay protocol. Merge-mining provides complete independence (including from Bitcoin block reversals) while the overlays do not. We used SPV proofs for proving foreign consensus, for exactly the same reason.
The risk of Bitcoin being torn apart has diluted over time. We can therefore re-consider the requirements for the two-way-peg and reexamine the design space searching for better alternatives.

The SPV Bridge Security

One of the properties of the RSK peg is that the reversal of the bitcoin blockchain does not imply the reversal of the RSK blockchain. Therefore, to prevent a double-spend, RSK requires the highest assurance that the Bitcoin blockchain will not revert past the peg-out transaction, such as waiting for 100 Bitcoin block confirmations. Let's assume that the merge-mining engagement is around 50%, and there is one large malicious mining pool we'll call Mallory, that has 51% of the RSK hashrate (25% of Bitcoin's hashrate). Mallory could try to use all her hashrate to create a false header-only blockchain and feed it to the bridge, causing the bridge to accept whatever fake transaction Mallory asserts. This fake header-chain would not be a valid Bitcoin blockchain, and therefore Mallory would not disrupt the Bitcoin blockchain, but she wouldn't be able to collect block rewards during the preparation of the attack. RSK is currently protected from this attack because federation functionaries will not issue a peg-in registration message if their local best chain does not match the Bitcoin best chain loaded in the bridge contract. However, we can think that if the SPV-based peg-in was maximally decentralized removing intervention from the federation, then the attack could succeed. We'll call this a "fake peg-in" attack. The cost to perform a fake peg-in for Mallory in an unprotected variant of RSK is about 10M USD in electricity. Although the fake peg-in attack seems easily attributable to the "missing miner" by looking at the bitcoin coinbases, Mallory could claim her systems were hacked, and shift the blame to an unknown party.
Also there is the peg-in double-spend attack. That attack requires informing an alternate best chain to the bridge contract and at the same time isolating federation functionaries to make them believe that this alternate best chain is the only one in existence. The cost of attack is still 10M USD in electricity, plus hacking the functionaries but the plunder now is not arbitrary bitcoin creation, but just to double the initially invested amount.

The opposite attack is the peg-out double-spend. In this attack Mallory performs a peg-out and then reverts the RSK blockchain to the point where the transaction that commands the

peg-out is located. Again, the attack has a high cost in electricity, but, if the amount of bitcoins to be stolen is high enough, then the risk exists. Note that RSK has a network monitoring tool called Armadillo that enables decentralized alerts to prevent “free” merge-mining attacks, but this attack is still possible.

Another downside of the SPV bridge is that it requires 100 Bitcoin block confirmations (or equivalent RSK blocks for the same cumulative difficulty) for peg-ins and peg-outs. This impairs usability. But we can do better than the SPV Bridge in terms of security and usability.

The Security of Two Way Pegs

To compare two way peg designs we define 8 protections that prevent double-spend attacks or plain theft of bitcoins by the miners. We do not consider attacks from any other groups. These protections apply to any peg system S that accepts cross-chain transfers over two proof-of-work blockchains. The protections will be defined either based on a security assumption or on the cost to attack (an assumption on the budget of the attacker). The protections are ordered from strongest (1) to weakest (8), but a system may offer more than one protection. A system S may not be fully characterized by a combination of the described protections because a system could use different protections for peg-ins and peg-outs. Since peg-out is always the weaker sub-protocol, we’ll focus on peg-out security for all protocols presented in this section, as attackers would choose the weakest spot to attack. Also a system may require different assumptions for soundness and liveness, but in this categorization we focus on soundness. Also for the sake of simplicity we’re ignoring other complementary attacks on the network, such as the feasibility to isolate a node. Note that generally a double-spend attack does not cause a loss of block rewards to the attacker: if an attacker reverts a blockchain to privately create a new best chain, this new chain will pay rewards to himself, so only other miners’ rewards are lost. The only exception to this rule is the protection provided by Armadillo, or some (generally broken) consensus algorithms that score blocks by time of reception [1][2][3].
Since Bitcoin cannot evaluate other sidechain’s cumulative proof of work, we assume the existence of some form of static or dynamic multi-signing group to sign peg-outs and receive peg-ins. However, we ignore any attack that involves malicious holders of the multi-signature that receives the pegged coins. We assume key holders are 100% honest or that they run Hardware Security Modules (HSMs) that prevent them from accessing the private keys, as RSK does.

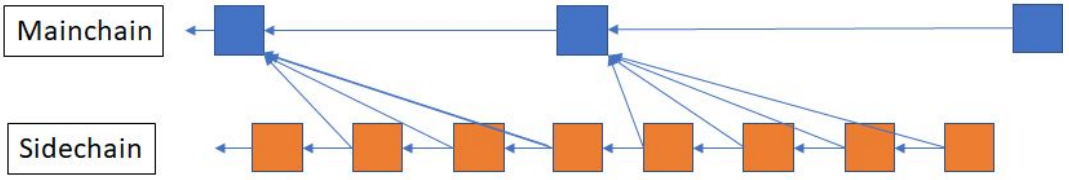
In the following table we compare RSK to other types of two way pegs, but modifying them to connect Bitcoin with a merge-mined sidechain. This helps to achieve a fair comparison of protocols.

System (peg-out security)	unconditional security (1)	Comp. Infeasibility (2)	Bitcoin M.A.D. (3)	Long attack awareness on Bitcoin (4)	Sidechain M.A.D (5)	Long attack awareness on Sidechain (6)	Loss of Bitcoin block rewards (7)	Loss of sidechain block rewards (8)
TS SyncChain with peg-out anchoring	✓	✓	✓	✓	✓	✓	✓	✓
SyncChain with peg-out anchoring		✓	✓	✓	✓	✓	✓	✓
SyncChain without anchoring			✓	✓	✓	✓	✓	✓
RSK with Armadillo				✓	✓		✓	
RSK prior Armadillo					✓			
XCLAIM					✓			
TBTC					✓			
Drivechain			✓	✓	✓			

The first protection “Unconditional security”, which means that under the rules of the protocol for S, double-spends cannot occur. The next is “Computational infeasibility” (2), which represents an impossibility to double-spend on S on the assumption that the attacker cannot perform a difficult computational task. The next protection is “Bitcoin M.A.D” (3). M.A.D. stands for mutual assured destruction. A protocol is protected by Bitcoin M.A.D if to perform a double-spend the attacker is forced to revert the mainchain, which would impact the mainchain native token price, and therefore the long term miner incentives to maintain the mainchain unharmed apply to the peg also. The same concept can be applied to the sidechain as in (5). The protection “Long Attack Awareness” (4) is when the network can detect in advance malicious actions of a colluding group of miners. RSK plus the Armadillo monitoring system has a cryptoeconomic variant of Long Attack Awareness: the attacker either reveals he is preparing an attack or needs to renounce the block rewards for the blocks he produces in private. The next protection is “Loss of Bitcoin block rewards" (7), which represent all cryptoeconomic systems based on SPV proofs where the attacker needs to mine blocks with a fake transaction to fool the system into unlocking coins in a chain that weren’t locked in the opposite chain. The current RSK peg protocol (with the Armadillo monitoring system) has this protection, and also XCLAIM and TBTC do. The other categories self-explain.

Dual-parenting

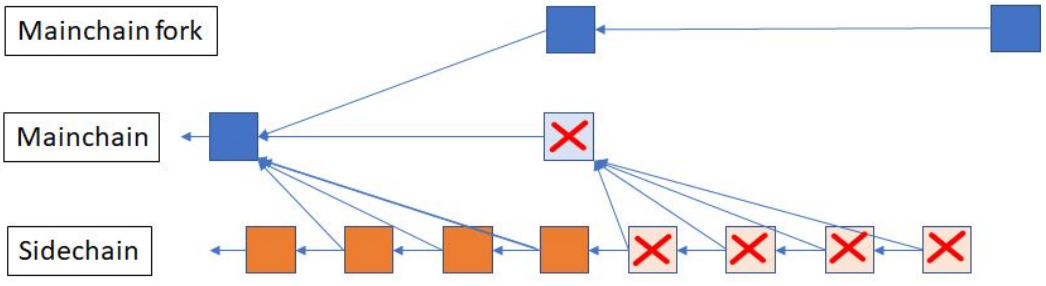
In my article [Drivechains, Sidechains and Hybrid 2-way peg Designs] (2016) I reviewed almost all possible peg designs I could imagine and analyzed the pros and cons of each one. However, the article didn't analyze the possibility of direct blockchain entanglement. Entanglement informally means that the reversal of the main chain forces the reversal of the sidechain at the block level. This is crucial, as the decision to synchronize or not the two blockchains shapes all other design and security considerations.



Dual parenting

Note that I omitted from the definition of entanglement that the mainchain should be reverted if the sidechain reverts: this seems impossible without changing the mainchain consensus. So if a peg-in lock reverses, the peg-in release in the sidechain will also revert, but if a peg-out lock reverses then that doesn't imply the peg-out release does. While the peg-out linking is impossible to do without a consensus change in the mainchain, we can make sure that if the sidechain is reverted prior to the peg-out lock then the sidechain cannot go forward again without repeating the peg-out lock transaction. This could be pictured as in the typical time-travelling film where no matter how the traveller tries to modify the past he cannot change crucial events of the future. Using this allegory, we'll call this property **peg-out determinism**.

A specific design for entanglement was briefly discussed in our blogpost that was published along with the aforementioned long article about sidechain designs. That proposal has two problems: it doesn't provide peg-out determinism and it cannot be used with an increasing sidechain block rate. If you try to increase the sidechain rate, a single block reversal in the mainchain may revert many blocks of the sidechain.



Surprise! reversal in the sidechain

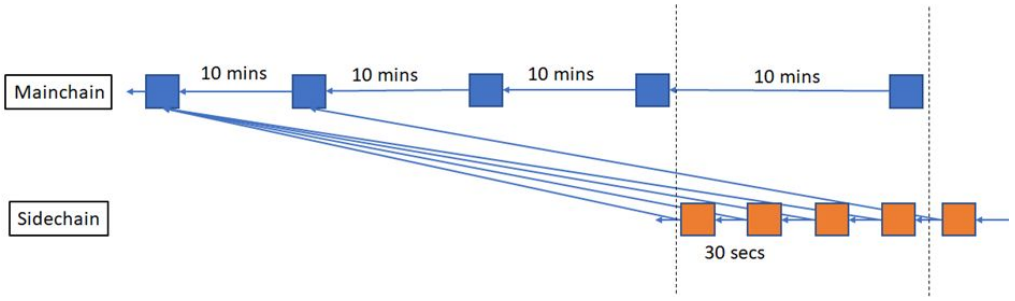
But the blogpost gives no proof that increasing the block rate is impossible, and it turned out that it's not. I came up with a design that does not restrict block rate, and has many side benefits. There are two known patterns to entangle a sidechain with a mainchain that I proposed in 2016: anchoring and dual-parenting. Anchoring is when the mainchain includes, from time to time, an unconstrained hash of a sidechain block, which the sidechain cannot revert back, as a kind of checkpoint. This solution does not work for higher block rates, because the inclusion of an unexpected hash indicating a different checkpoint may automatically and unexpectedly revert many sidechain blocks. Dual-parenting is when each sidechain block has a block parent in the sidechain and also a parent in the mainchain. The same restrictions apply to both parents: the inherited state prior to the processing of the sidechain block corresponds to a state after both parents have been processed, and the reversal of one parent causes the reversal of the sidechain child block. But dual parenting as proposed in the blogpost has the same drawback as anchoring: a reversal of the mainchain block may cause a reversal of many sidechain blocks.
A new form of entanglement is needed that supports higher block rates but does not suffer from long reversals.

The SyncChain Design

If we assume Bitcoin has a honest supermajority of miners, then we could argue that a SPV-based bridge that has reached 90% of merge-miner engagement would be secure enough, as there would be a large honest majority doing merge-mining. Bitcoin history doesn't show any episode of large mining pool collusion. However, this reasoning excludes the fact that merge-miners are under much higher temptation to collude than mainchain miners, because if they collude they double-spend coins from a decentralized system, which does not have any KYC protection, while in mainchain they generally can only double-spend coins from a centralized exchange. Therefore Bitcoin mining history does not provide a clear positive answer to the security of an SPV peg. Any SPV bridge that connects Bitcoin to a merge-mined sidechain that expects a fixed number of confirmations for peg-outs and does not rate limit peg-outs will tempt the miners to corrupt if the value of the coins transferred is much higher than the direct or indirect benefit that the sidechain provides to mainchain miners in immediate rewards or in a long-term value proposition. The Mutual Assured Destruction game that has been protecting Bitcoin from malicious mining pools does not fully apply to a non-rate limited SPV sidechain, leaving the sidechain in greater danger than the mainchain.
RSK is rate limited, and RSK provides a cryptoeconomic security guarantee, but yet we'll show reasons to improve it. In this article we present a new sidechain design we call SyncChain, one that provides a security guarantee that is much stronger than that provided by cryptoeconomics. It provides unconditional security to peg-ins and peg-outs. With a SyncChain we can eliminate the double-spend risk, and therefore do not overstress the merge-mining network. We present 3 variants of the SyncChain design. All variants are based on three components: delayed dual-parenting, peg transaction linking and coinbase anchoring. Each variant then requires some additional change to the consensus rules.

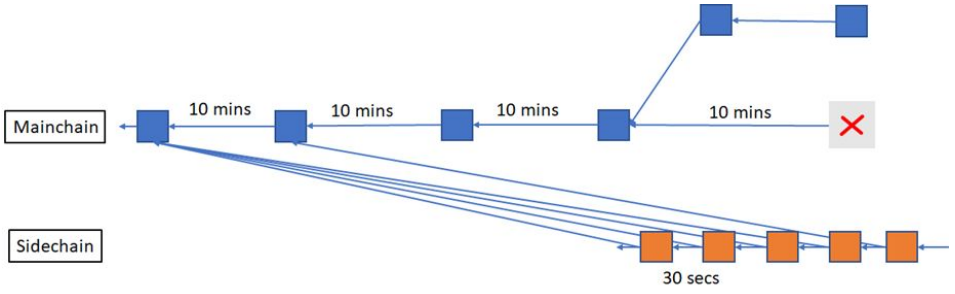
Delayed Dual-Parenting

Dual-parenting is the name we gave to a simple technique of entangling the mainchain and the sidechain. The technique is based on having each sidechain block have two parents, one in the sidechain and the other on the mainchain. But we don't use this technique as is, we use a variation of dual parenting that we call Delayed Dual-Parenting (DDP). First, to simplify our terminology, we'll call the mainchain parent a "checkpoint" but the reader shouldn't confuse a synchcain checkpoint with other checkpointing systems based on authorities [4][5]. With DDP, the checkpoint is set to lag by a number of blocks (K blocks on average) based both on timestamps and mainchain block confirmation count. For example, a realistic value for K is 3, forcing checkpoints to lag by an average of 30 minutes. The following diagram shows an example for K=3.



Delayed dual-parenting

As previously described, the problem with immediate entanglement is that the reversal of a Bitcoin block automatically causes the reversal of about 20 RSK blocks, which is not tolerable from a UX perspective, but more importantly it is a transaction settlement security risk. With the delayed checkpoint, the RSK blockchain is only reverted if more than K mainchain blocks are reverted. To simplify the explanations, we'll assume the RSK average block interval is 30 seconds and Bitcoin average block interval is 10 minutes. Then if the Bitcoin blockchain is reverted R blocks where R>K, then the RSK blockchain will be reverted (R-K)*20 blocks. In the following diagram we can see that there are no surprise removals in RSK if only a single Bitcoin block is reversed.



Reversal of a single Mainchain block does not cause reversal on the SyncChain

The minimum value for K is 1, which means that a block can only be checkpointed if it has one additional block for confirmation.

Dual Nodes

A SyncChain requires that each sidechain client runs both an instance of the mainchain node and the sidechain-specific node. The main benefit of the SyncChain over a SPV-sidechain is that it prevents an invalid Bitcoin blockchain branch to be presented to a SPV-based bridge contract as a valid SPV (header-only) chain. In a SPV-sidechain, a SPV proof can be hidden from the honest network, and that reduces M.A.D. tension. We want transparency to judge and eventually punish miners as soon as they attempt to attack, and before the attack finishes. Therefore we must assure that whatever blocks the bridge accepts to validate a peg-in or peg-out, those blocks are visible and can be part of the Bitcoin blockchain as defined by Bitcoin Core reference software. If there is any difference in consensus validation between what the bridge accepts and what Bitcoin Core accepts, then that can be used to attack the bridge by the same vector that exists for a SPV sidechain design. Since rskj already includes bitcoinj code, it is tempting to to include a bitcoin full node in rskj, based on bitcoinj, but this will not solve the problem entirely, as bitcoinj does not validate the same acceptance rules as bitcoin core, and modifying bitcoinj to do so is not trivial. Therefore we recommend that a SyncChain node should run the mainchain reference node, together with the sidechain-only node. In other words, a synchain node runs an instance of bitcoind, and an instance of rskj.

A user is still able to run only the sidechain-only node (rskj), but he will be unable to validate peg-ins and peg-outs. In that sense, it automatically becomes an SPV “lightweight” node, with the same security model as SPV clients get, as soon as there is a peg-in or peg-out transaction.

Improving Bitcoin Block timestamps

To create a delayed dual-parent we need a **Checkpoint Selection Algorithm (CSA)**. A CSA is a consensus algorithm that selects the checkpoint and validates if a checkpoint is correctly selected. A CSA must essentially base its decisions on the block timestamps. But block timestamps in Bitcoin do not reflect wall clock times and even less a global clock. A timestamp of a Bitcoin block can sometimes be lower than its parent block timestamp. Therefore the CSA requires a **Reference Time Algorithm (RTA)**. The RTA needs to correct timestamp anomalies and provide a robust time source for linking. We present two simple RTA algorithms and one CSA algorithm. The first RTA algorithm, called **MedianTime11**, requires the existence of 5 bitcoin blocks following the block to checkpoint. MedianTime11 establishes the reference time of a Bitcoin block B to be the median of 11 blocks centered on B. MedianTime11 is non-decreasing. This property is given “for-free” by Bitcoin, because Bitcoin consensus establishes this exact rule for timestamps.

Another proposed RTA algorithm is the **AdjustedTime**. This algorithm sets the reference time of a Bitcoin block B to be the more recent of the block timestamp and previous block reference time. This simple change makes the timestamp always non-descending.

MedianTime11 seems at first glance more secure than AdjustedTime, because it doesn't let a minority of miners (with some chance) skew Bitcoin timestamps to deviate from honest timestamping. However, as we'll see later, any attack based on forged timestamps can easily be detected and deterred automatically.

Now that we have a non-descending reference time we can define a CSA algorithm.

The Simple Checkpoint Selection Algorithm (SICSA)

We say that a mainchain block B **is compatible** with a sidechain block S if B's reference time is lower than S's timestamp, and not closer than M minutes, and B's child reference time is not compatible with S. This means that we need to evaluate at least one child of B to checkpoint B. Compatibility is a rule that can be verified in consensus.
The **SICSA(K,M)** is a CSA that simply chooses the Bitcoin block B that is compatible with S and has at least K confirmations, and a specified forced delay M in minutes. It must be noted that having K confirmations is not something that can be verified in consensus, but it's a local policy that nodes implement to protect from double-spends. When using the AdjustedTime, we can use SICSA(1,60) and expect an average linking delay of 60 minutes. The forced delay prevents the sidechain from frequently stopping to wait for the checkpoint confirmation blocks to occur. With the MedianTime11, we need K to be at least 5, because those blocks are inspected by the MediantTime11. The farther the blocks are from B, the less they should affect consensus over B's reference time. Therefore we only ask for two additional confirmations for MedianTime11, so we suggest using SICSA(5,110) for this RTA.
To protect from miners setting their timestamps too far in the future (up to 2 hours are allowed by Bitcoin Core software) the consensus needs to perform some additional checks when peg-outs are involved.
Also from the definition of compatibility, we notice that if the reference times of two consecutive blocks are equal, the algorithm selects the most recent, and therefore checkpointed block numbers are not allowed to decrease.
SICSA has the advantage that it's security doesn't rely on the honest majority of miners: it mandates exactly which block must be checkpointed. The downside is that it forces delays that may be unnecessarily long only to account for rare cases where mainchain blocks are delayed.
We now present another CSA we call Honest Minority CSA (HOMCSA).

Honest Minority Checkpoint Selection Algorithm (HOMCSA)

We'll now define a new type of CSA we call HOMCSA. For HOMCSA we always use the AdjustedTime as reference. By consensus, the mainchain block chosen as checkpoint must be one whose reference time is not older than 2 hours and has 1 confirmation. Also the checkpointed block number is always non-descending. Honest miners will always choose the latest block that verifies the time restrictions, but malicious miners may not.

Now, as long as miners do not censor other miners' blocks, the checkpointed block will generally be the most recent compatible one. When we study the peg-out process, we'll see how an honest minority prevents attacks by delaying the checkpoint.

Implementation

In the sidechain block header the checkpoint is defined by a mainchain block number (**checkPointBN**) and a mainchain block hash (**checkPointHash**). It's possible to use a reduced-size header of the hash digest to save space.

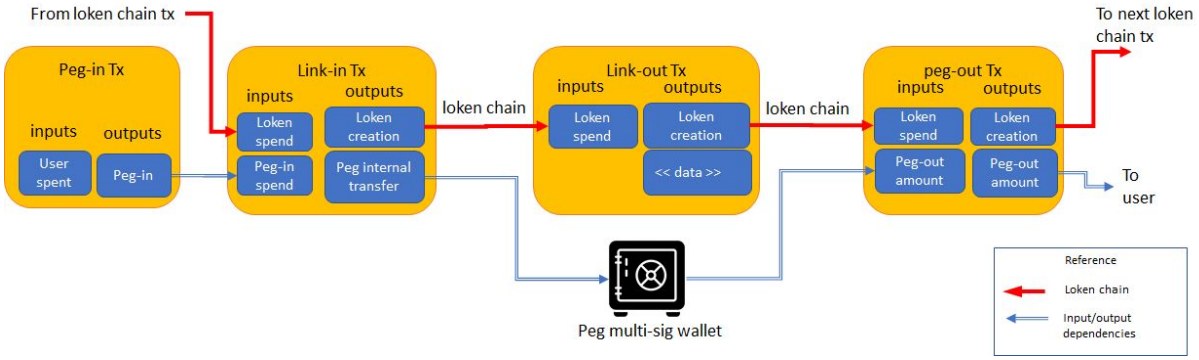
Checkpoints and Block Processing

Every sidechain node must run a mainchain node (bitcoind in case of RSK) to monitor the correctness of the checkpoints in RSK block headers. If the RSK checkpoint hash does not match with the Bitcoin block hash referenced by block number, then the RSK block is "temporarily invalid". The RSK block could be reconsidered later if the Bitcoin best chain reorganizes.
Not every Bitcoin block will be a checkpoint: the checkPointBN values do not cover all blocks, and there will be gaps. When a block checkPointBN refers to a peg-in transaction in a Bitcoin block, or jumps over a peg-in transaction by skipping the block, then the RSK consensus dictates that the related RSK block MUST include a transaction sending a peg-in message to the bridge contract to release the associated coins pegged-in immediately. The consensus also mandates that the peg-in transactions take priority over the others, meaning that they precede them in the block. Some precautions must be taken, as an abrupt jump over many Bitcoin blocks by the checkpoint may force the inclusion of many peg-ins transactions in RSK blocks, which may not fit in a single RSK block and may require spanning multiple blocks, taking priority over other transactions. It's also possible to perform the RSK state changes related to transferring the coins pegged-in without really including any associated transaction in RSK, because those actions are implicit. However we think that by making them explicit we enable light clients that do not run bitcoind to keep an updated state of the sidechain by processing sidechain blocks only.

Peg Transaction Linking

All Bitcoin peg-ins and peg-out transactions are linked together. This is done for several reasons. One is to avoid attacks where the attacker reorganizes the Bitcoin and RSK blockchains to double-spend funds from the peg multi-sig, where the attacker himself has pegged-in or out. Linking greatly reduces the attack surface. But a second reason relates to how peg-outs are secured, and enables us to prove the infeasibility of peg-out double-spends. The following diagram depicts a peg-in and a peg-out for a federated sidechain such as RSK. The peg funds are protected by a federated multi-sig, and the parties that have the private keys of this multisig are called functionaries. In the chain shown,

there are two additional internal transactions called link-in and link-out that we'll explain later. The red line corresponds to a chain of references using "dummy" inputs/outputs. All transactions signed by functionaries are tied to this chain. Because each transaction consumes one dummy input and creates one dummy output, there is only one unspent transaction output at all times. We call this special UTXO the **loken** (short of **link token**). We use "loken chain" to refer to the chain of transactions that consume and create the loken. The value of the loken will be some small value above the Bitcoin dust limit.

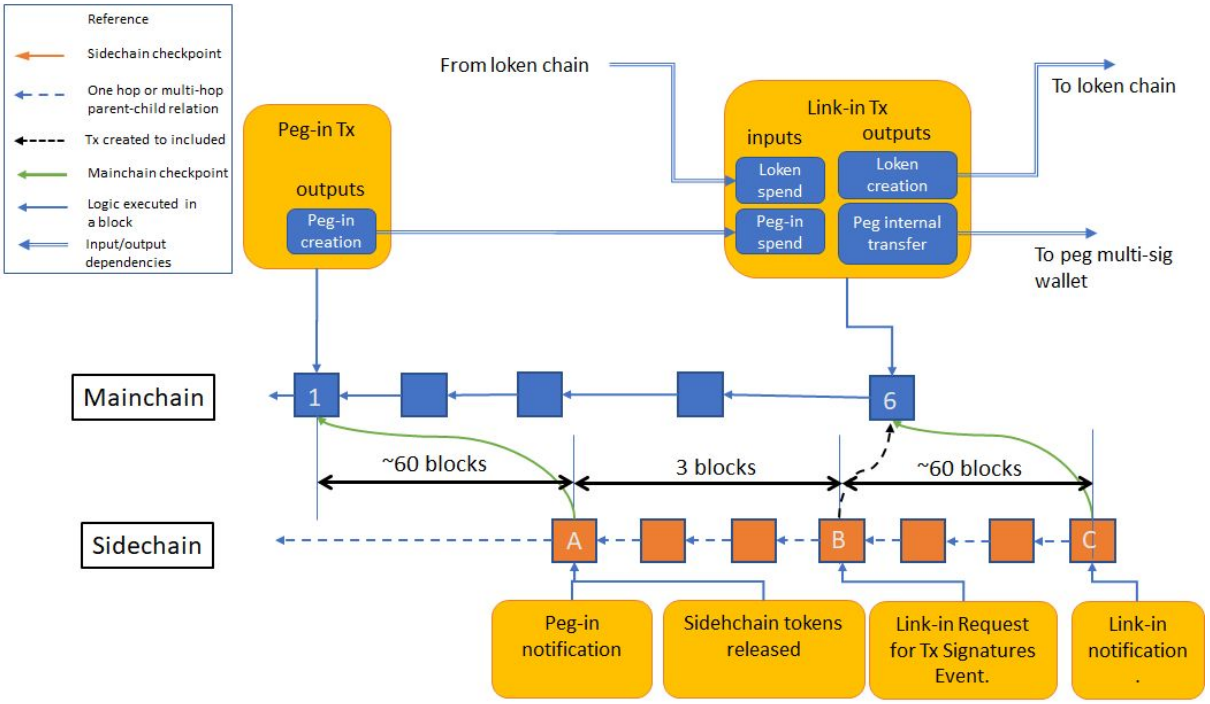


The loken chain

Care must be taken when upgrading the federation functionaries by adding or removing members. This triggers, in RSK, a lengthy and forcibly delayed process where funds are automatically moved from an old federation multisig to a new one. We haven't yet proposed a migration process adapted to a SyncChain, but we believe this can be done securely.

Peg-ins

The bridge smart contract, which controls all the peg-in and peg-out processes, automatically commands functionaries for the forwarding of the coins received in the peg-in transaction to a different UTXO, whose address can be the same or different from the one in the peg-in, but it is equally controlled by the same federation. This initial forwarding of coins received is performed in a Bitcoin transaction we call the **link-in**. The link-in transaction consumes the loken and creates a new loken. The following example diagram shows how a user-initiated transaction ("Peg-in Tx" in the diagram) in mainchain block 1 triggers actions from the bridge when the block 1 is checkpointed by sidechain block A. The first action is an initial wait period of 3 sidechain blocks, and afterward, in block B, the bridge commands the federation functionaries to sign and broadcast the Link-in transaction, which immediately consumes the peg-in funds, and move them to the final multi-signature peg address. Note that annotations regarding the number of blocks correspond to using the AdjustedTime. Using the MedianTime11 generates slightly larger delays.



The peg-in process

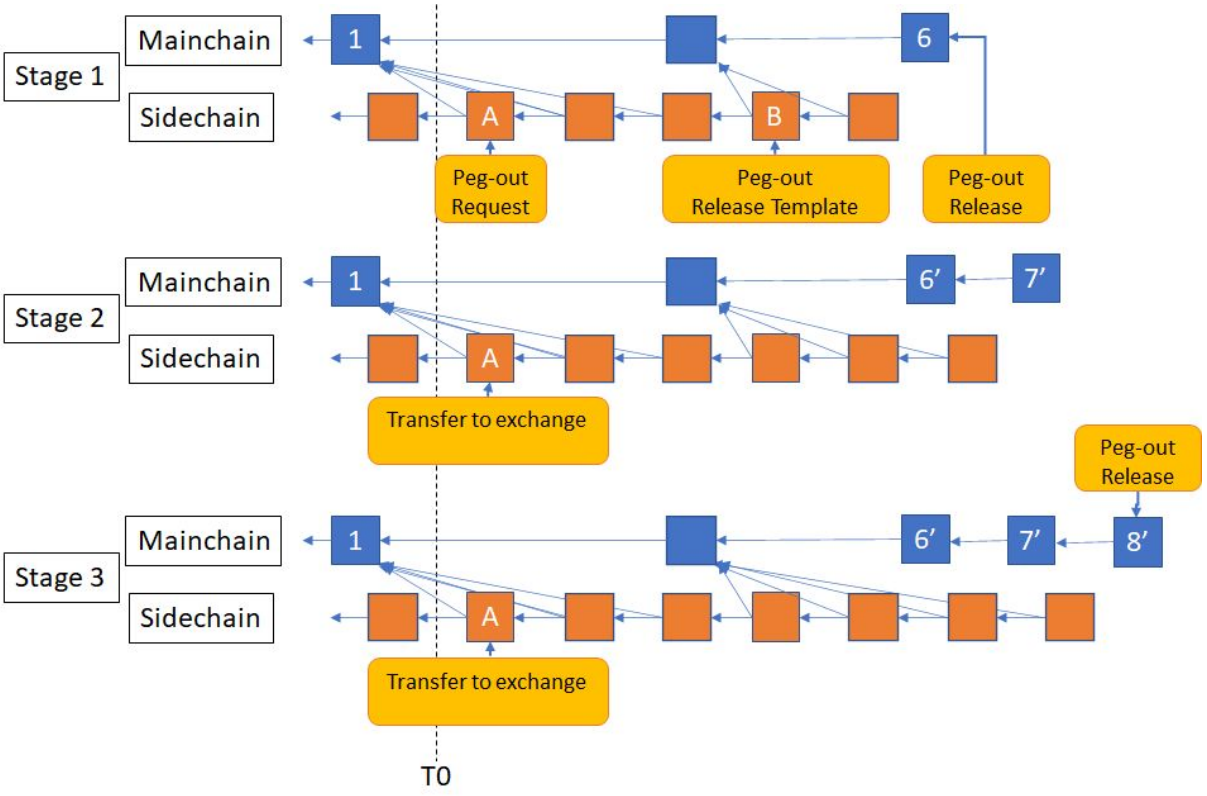
The sidechain coins are released immediately in block A, but no peg out can occur until the link-in transaction is included in the mainchain.

Peg-outs

To understand the difficulty of synchronizing peg-outs let's first imagine there exists a system that guarantees that the peg-out request in the sidechain and the peg-out release transaction in the mainchain are atomic (either both occur, or none) after the number of block confirmations surpasses a predefined large number. We can prove that if the mainchain is Bitcoin, and the peg-out transactions are solely generated by an independent federation with no help of the miners, then no such system can exists.

Let's present the following attack by Mallory, who controls a majority of the hashrate. In stage 1, Mallory has some tokens in the sidechain at time T0, which she pegs-out, and once they are in the mainchain she exchanges them for cash. In stage 2 of the attack, she reverts both the sidechain to the time T0, and reverts the mainchain to a point prior to the peg-out release, then she extends both blockchains again to outpace the previous honest chains. In the new sidechain blockchain branch, she transfers the original sidechain tokens to an online exchange, where she sells them and cashes-out. In the mainchain, she just fills with seemingly innocent blocks, but without the peg-out release transaction. Finally, at stage 3, she adds to the mainchain the peg-out release transaction, but in a much later block. If the system is sound, then the sidechain needs to halt as soon it detects that the mainchain peg-out release transaction does not have a corresponding peg-out request, and that's the end of the sidechain. Mallory has succeeded in performing the double-spend and as a

collateral damage she blocked the sidechain until a hard-fork unblocks it again (she could also make some extra money by shorting all tokens that exist in the sidechain). Therefore we proved that knowing only the state of both best chains is not enough to deter a double-spend attack on the peg if the peg-out release transaction can be moved to the future. In Bitcoin, a transaction can always be moved into the future, so the attack exists. The following diagram shows the stages of the attack:



An attack on any peg-out system without anchoring

Peg-Out Anchoring

If a peg-out release transaction could be built such that it becomes invalid if the block timestamp is higher than a value embedded in the transaction, then we could prevent the attack. This feature exists in smart-contract platforms but does not exist in Bitcoin. However, it can be emulated: first we note that if the payment is taken right from a coinbase transaction, then the peg-out release would be tied to a specific block number and block hash, and we would achieve unconditional double-spend security. We could build such a system, but it would add the complexity of paying the miners back atomically, without assuming a trust relationship. Also peg-outs would be limited to block rewards. There are several workarounds to this:

a) split the peg-out into two transactions: one that publishes the sidechain checkpoint information (called link-out), and a second that actually performs the payment, but the

second also must consume an output of a coinbase transaction included in a block after the link-out was included. This mechanism will be discussed later.

b) Include in any part of a coinbase transaction a message containing the information about the expected peg-out, signed by the federation functionaries, and also include in this transaction a dummy output that only the federation can spend. Then in a following block include the peg-out transaction consuming the dummy output. Interestingly, the message to be signed and embedded could be the peg-out transaction itself.

While option (b) requires less resources from the network, we'll see that for some variants of the SyncChain option (a) enable mainchain M.A.D. security without anchoring.

For peg synchronicity, we must assume nodes act only on information that is in the best chain. Other sidechain peg designs, such as Liquid, which is base on a PBFT-type consensus that provides settlement finality, are incompatible with sidechain-mainchain synchronization and double-parenting.

Peg-out Protocols

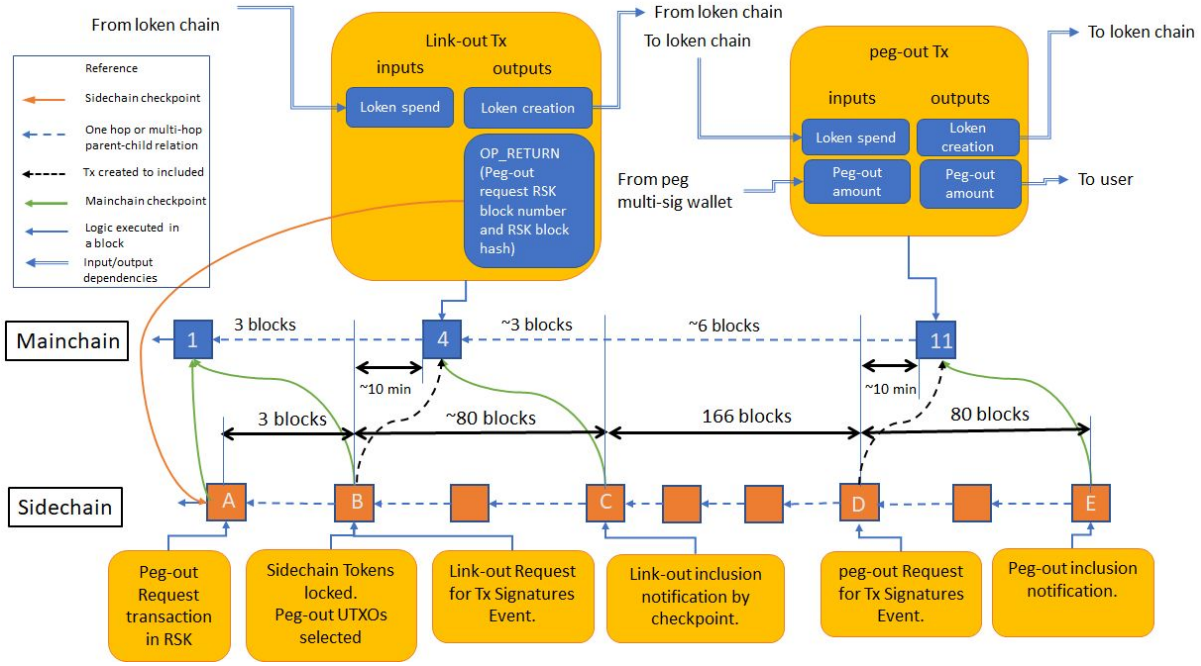
We found three different protocols to achieve a M.A.D. and unconditional guarantees, based on different assumptions to add to standard Nakamoto consensus. In the following sections we briefly describe each one.

- Peg-out for a More-Populous-Chain-Wins (MPCW) SyncChain
- Peg-out for T-Synchronized (TS) SyncChain
- Peg-out for a GHOST-CSC (not T-synchronized) SyncChain

Peg-out for a More-Populous-Chain-Wins (MPCW) SyncChain

This method is based on changing the standard Nakamoto Consensus, also known as More-Cumulative-Difficulty-Wins rule (MCDW), to the Longest Chain Wins rule (LCW), and then further adapting LCW to a sidechain that supports uncle references. First we note that if the difficulty adjustment algorithm corrects linearly with the block rate, then the two consensus methods will only differ by a small constant amount of work (if compared to the highest difficulty reached). The Longest-chain-wins rule may incentivize the manipulation of the block timestamp to decrease the difficulty for mining, but this is a minor problem easily solvable. In a blockchain with uncles, the concept of longest chain becomes the "more populous chain". In other words, the chain that has the highest quantity of blocks, either blocks in the main chain sequence or sibling blocks referenced by the main chain. Assuming a sidechain using this rule, we'll now describe the actual mechanism of peg-out.
To move tokens back from an account in RSK to Bitcoin, an RSK transaction called a **Peg-out Request** is created which transfers the RBTC to the bridge smart-contract. In the following figure, this transaction has been included in the block labeled A. After a short number of RSK block confirmations (such as 3), the bridge creates a **Link-out transaction**

Template containing placeholders for federation functionaries to insert their signatures. This template can use the Partially Signed Bitcoin Transaction (PSBT) standard. Then the bridge requests federation functionaries to sign it and this event is called a **Link-out Request**. This occurs in the RSK block B in the figure. The Link-out transaction must contain a special data output that comprises an OP_RETURN, the block hash for block A, and the block number of A. We call this data payload the **sidechain checkpoint slot**. This slot may contain one or more actual sidechain checkpoints. We note that sidechain checkpoints together with mainchain checkpoints produce *reciprocal checkpointing references*.



Peg-out process

Once a majority of functionary signatures are collected for the link-out transaction, the transaction is broadcast and is expected to be included in the Bitcoin blockchain soon. In the figure, this occurs in Bitcoin block 4. The link-out transaction must be part of the loken chain (it must consume the loken, and create a new loken). After the link-out transaction is included in a Bitcoin block 4, block 4 should be referenced by the checkpoint of an RSK block (or skipped by a checkpoint, which causes the same effect). The checkpoint to block 4 will usually be delayed approximately 30 minutes (for AdjustedTime) or 140 minutes (for MedianTime11). In the figure, block 4 is referenced in RSK block C. Afterward the bridge will wait for V blocks (V=166 in the figure, between block C and D). After the waiting period is over, the bridge will create the **Peg-out Transaction Template** and ask federation functionaries to sign it. The peg-out transaction is the final transaction that actually pays back to the Bitcoin user the required amount. As all the rest peg and link transactions signed by the federation, it links into the loken chain. The peg-out transaction will be included in a Bitcoin block, and this occurs in block 11 of the figure. After approximately 30 minutes (for AdjustedTime), it will be referenced by an RSK checkpoint, and the loken will be available for other link-in or link-out transactions.

The whole peg-out process when using the AdjustedTime takes on average 2 hours 10 minutes, requires two rounds of federation signatures and produces two Bitcoin transactions. We note that the number of blocks chosen for confirmation (166) is not hard-coded, and we use that number in these examples to show how the algorithm works based on average block times. When implementing a production ready system, one has to tune these parameters to take into account the random variations in block creation times, and how they affect security.
If a peg-in transaction is confirmed while a transaction that consumes a loken is waiting to be included in the mainchain, the peg-in will be queued. In fact, many link-in and peg-out transactions could be joined in a batch, having a single loken creation and destruction input/outputs.

To prove the security of the peg-outs, we'll first discuss the sidechain difficulty adjustment algorithm required for a SyncChain.

Fast Upward Difficulty Adjustment Algorithm

To use the more-populous-chain wins rule in the sidechain, we need a sidechain difficulty adjustment algorithm that can increase the difficulty very rapidly in the sidechain. However, we cannot make huge upward adjustments by looking at single blocks (as Ethereum does) because mining is a randomized process where timestamps are distributed approximately according to a poisson probability (some other factors, such as block timestamp update frequency affect the real distribution). Therefore we need to average the changes of an interval of blocks (or apply a suitable low-pass filter). The Bitcoin difficulty algorithm matches our expectations and it's a good starting point. We only have to reduce the retarget interval to react faster and correct its known vulnerabilities. For RSK, we'll use the GHOST best chain weighting algorithm and we'll pick a retarget interval of 20 blocks, and a maximum upward correction of 4. We will also adopt the current RSK policy of considering uncles as part of the proof of work, and target a specific block density rather than a specific number of mainchain blocks per time period. This prevents an attack on our SyncChain where proof of work is shifted to uncles to prevent difficulty adjustments.
Because the difficulty adjustment algorithm is a negative feedback control system, and because uncle inclusion adds a delay component into the system (that can range between 1 and 6 blocks), care must be taken to avoid over-correcting, leading to system instability through oscillations. This is accomplished by considering uncle real position as sibling, and not uncle inclusion position, when computing the difficulty adjustment.

Peg-out Security

The following sections analyze different theoretical attacks and we show how the SyncChain resists them. The most important attack to any cryptocurrency accepting system is the double-spend. To attempt a double-spend, the attacker could try to revert RSK, revert Bitcoin or revert both. Currently RSK is merge-mined by between 35% and 50% of the Bitcoin miners, but also by some Bitcoin Cash miners. It is possible, although unlikely, that RSK achieves a higher hashrate than Bitcoin by adding up all available SHA256D hashrate.

In this article we assume that RSK hashrate is lower than Bitcoin, and therefore if an attacker reverts Bitcoin, because of merged-mining, he also gets the chance to revert both chains.

Reverting only RSK

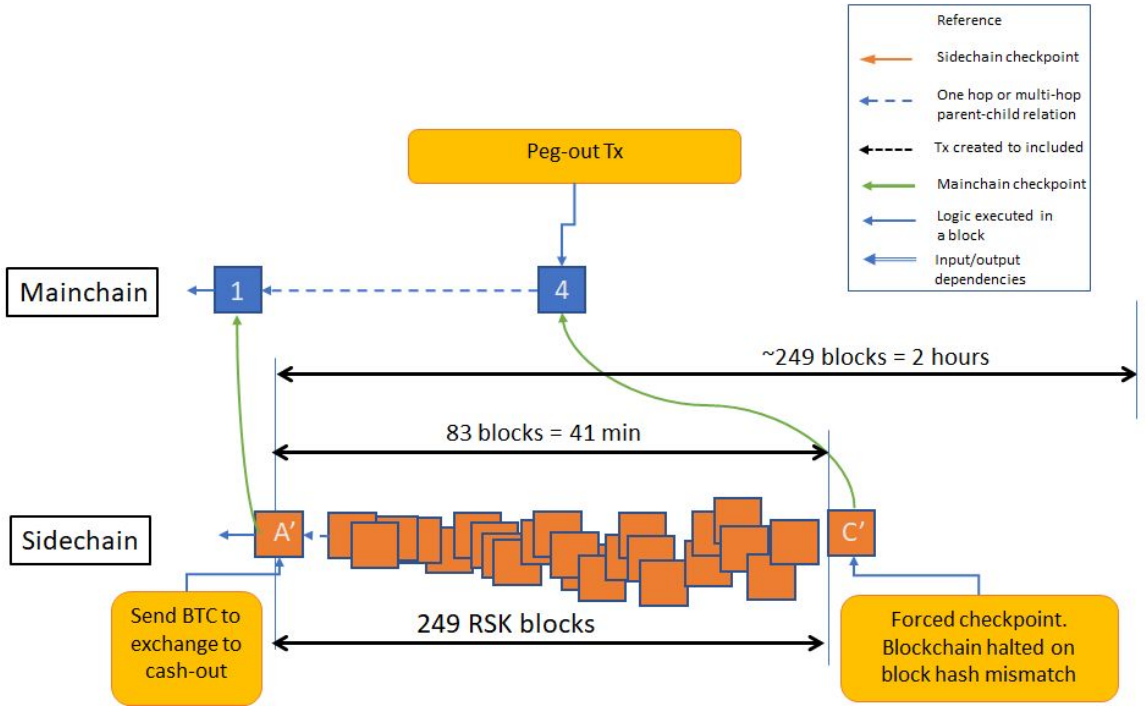
Reverting RSK seems to be the easiest path to double-spend, as we assume RSK's hashrate is lower than Bitcoin's. We've seen how peg-in transactions are synchronized, because RSK blocks depend on Bitcoin blocks. However, in the other direction, synchronization is not guaranteed. Therefore, for peg-outs we force a link-out transaction to occur on Bitcoin. The timestamp of the link-out transaction establishes a horizon that RSK cannot surpass without performing the corresponding peg-out request.

We present an example of the best possible attempt to double-spend a peg-out, and how the system reacts in this case, deterring the attack.

The attack starts with the attacker having RIF (or any other token on RSK) for an amount equivalent to 100 RBTC. The attacker buys 100 RBTC in a decentralized exchange by swapping RIF for RBTC. This occurs at a block Y. Then the attacker initiates a normal peg-out of the 100 RBTC using the process discussed before. At the end the attacker owns the released BTC. Then the attacker tries to revert the RSK blockchain to a point prior to Y. He creates a block X (prior to Y) that transfers the RIF to a different RSK account he controls. The attacker then tries to mine the blocks following X to outperform the current RSK best chain with a higher number of blocks (because of the MPCW rule). His aim is to convince all network nodes to accept (at least temporarily) the attacker's chain. In this supposedly new best chain the attacker has both the RIF and the BTC in Bitcoin and he leaves no record in the blockchain of a decentralized exchange nor a peg-out.

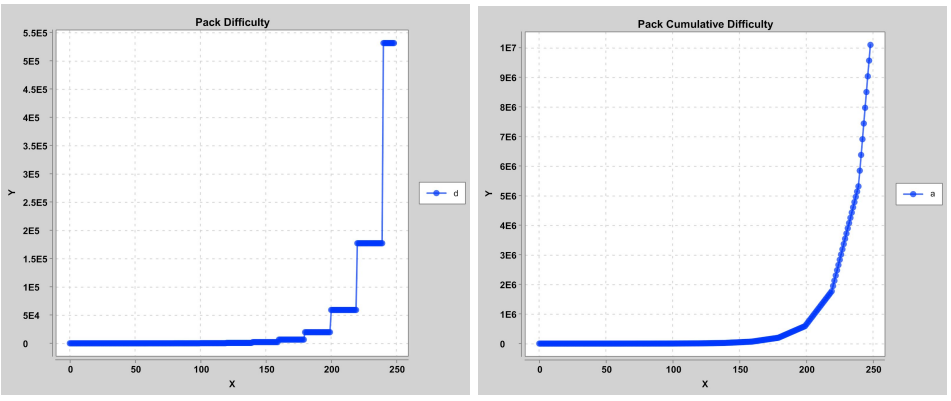
If we look again at the peg-out diagram, It's important to note that all of the attacker's blocks must have checkpoints in the Bitcoin blockchain prior to the block B. If any of the attacker's blocks referenced B or a block past B, then full nodes would find the link-out transaction and see it doesn't match with the contents of block A. We can assume that most full nodes are fully synchronized, so the link-out transaction is part of the best chain at most full nodes.
An attack is prevented because the attacker cannot outperform the sidechain best chain. It must revert the Bitcoin blockchain to remove the link-out transaction. The attacker cannot mine a blockchain with more blocks than the honest chain without reaching a block that forces a checkpoint past the link-out transaction in a Bitcoin block 4. The only way to outperform the current best chain is by "packing many blocks in a short time" between block A and block C. We'll now show that this alone would require the equivalent energy to mine Bitcoin for several years.
To illustrate, let's assume for a moment that the attacker's chain contains uncles and blocks that are spread by the attacker uniformly over the time interval between A and C. Assuming RSK has an average block interval of 30 seconds, this corresponds to 83 RSK blocks. But to revert from the tip of the best chain to A, the attacker's needs to mine blocks 249 blocks (to be more precise, he needs to present a block branch with an equivalent of 249 units of proof

of work between executed blocks and uncles). After the first 20 blocks have been packed into the available time, the difficulty is multiplied by 3 (249/83). The same happens after the second 20 blocks have been spread over the available interval. After 249 blocks, the difficulty has grown to more than a million times and the cumulative difficulty reaches 3506 days of mining! That's 40551 times higher than the cumulative difficulty of the honest chain! This is of course impossible at the current RSK merge-mining engagement.



An attacker trying to pack 249 blocks in a period of 41 minutes

The following figures show how the difficulty and the cumulative difficulty grow while the attacker tries to create the highly dense block set.



But we can do even better. First, we note that MedianTime11 requires 7 confirmations for the block checkpointed, and AdjustedTime requires 1. This means that once a mainchain block is checkpointed, the consensus has already evaluated some following Bitcoin blocks. If one of these blocks contains a link-out transaction, the system can react before the block

with the link-out transaction is checkpointed. You may ask why it's ok to bypass a delay that was specifically designed to prevent sidechain reorganizations as a consequence of peeking into the "future" of the mainchain. The answer is that it's ok if the system prevents a sidechain branch to be accepted due to "future" events even if later the system accepts the previously discarded branch due to a change in those "future" events. But the opposite is not ok! If the system first accepts the branch, and then invalidates it, this can lead to double-spends or loss of funds. Therefore we have a free mainchain look-ahead to invalidate sidechain blocks: K=1 block for AdjustedTime and K=7 blocks for MedianTime11. Assuming a look-ahead of 3 blocks (K=3), the final cumulative difficulty cost of this attack is 76 bitcoin years! This is more than 76 times harder than without the look-ahead.

In our paper, which is in preparation, we study all attacks involving faking timestamps and modifying block rates before and during the reversal, and conclude that the system can be protected from all attack variants, either by reacting to attacks or by resisting them initially.

Reverting Bitcoin and RSK (but replaying later the peg-out tx)

One possible way to perform a double-spend attack is to try to revert the Bitcoin blockchain prior to the link-out transaction, and at the same time revert the RSK blockchain prior to the peg-out-request, creating a new Bitcoin branch that has both the link-out and peg-out transactions, but at a much later block. For example, the attacker reverts 10 Bitcoin blocks, and mines another 10 blocks without peg transactions, and finally adds an 11th block containing both the link-out and the peg-out. Assuming dishonest rational miners, a 10 block reversal is approximately costs 1.5M USD[1], and therefore that would be maximum amount the peg-out transaction can transfer. But if we want not unconditional security, then we must do better with anchoring. With anchoring we can prevent the link-out transaction to move be moved into future blocks.

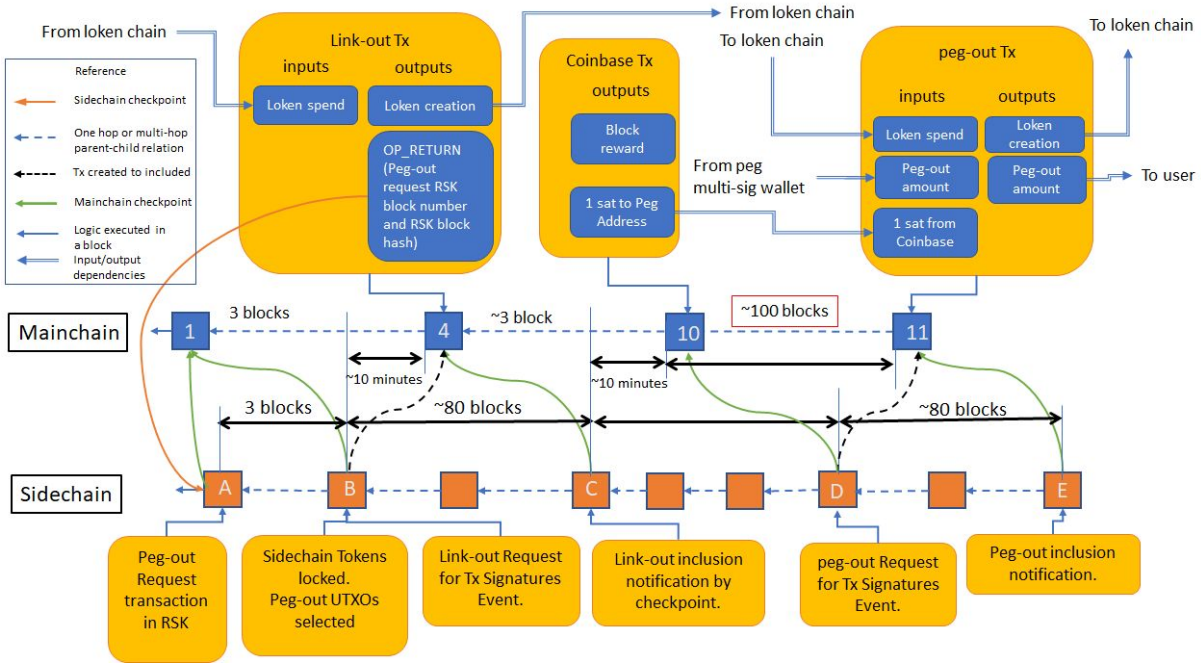
The SyncChain would be secured for peg-outs with low confirmations if Bitcoin could allow transactions to be anchored to specific blocks. This would allow binding the link-out or peg-out transaction to a specific blockhash and therefore the attack wouldn't work. We call this technique anchoring, and we show several ways to implement it.

Anchoring

The easiest way to anchor a transaction to a specific block would be by adding to Bitcoin a new opcode OP_CHECK_INPUT_BLOCK_HASH, which receives a block hash as argument in the stack and invalidates the block if the block hash given does not match the hash of the block where the input being spent was created. We don't believe this new opcode would be accepted by the Bitcoin community because Bitcoin was specifically designed to allow transactions to be poured from one branch into another if there is a chain reorganization up to a depth of 100 blocks. This opcode would restrict moving transactions, and therefore would create bitcoins that are less fungible than others. Another opcode that can do the

[1] according to crypto51.app website

same trick is OP_CHECK_INPUT_BLOCK_TIME. This opcode would invalidate the transaction if the block corresponding to the input being spent has a timestamp higher than the opcode argument. In contrast, this opcode interferes much less with fungibility, as transactions can generally be rearranged to respect timestamps in a new best chain.
However, without any new opcode, there is still a way to achieve the same result and that is by consuming in the peg-out transactions an output from a coinbase transaction that exists in a block B between the link-out and the peg-out transactions blocks. The peg-out transaction would be invalid if B is reverted, and the only way to remove the link-out would be to revert B. The downside is that because the coinbase transaction outputs have a 100-block maturity period, this binding introduces a 100-block delay for the peg-out. While the peg-out transaction won't be able to be included before the 100-block period, the peg-out transaction can be signed and published much earlier and therefore the user has a very strong guarantee that the peg-out transaction will occur. To bind the peg-out to a coinbase, RSK could request that RSK merge-miners include an extra output paying an amount 1 satoshi[2] to a specific federation address, and that satoshi is consumed in the peg-out transaction.



Peg-out process with coinbase anchoring

Peg-out for a T-Synchronized (TS) SyncChain

One possibility to build a secure system that may not be based on GHOST consensus is to add a new rule on how transactions are settled to the blockchain protocol. Nakamoto consensus establishes clear rules for every aspect of blockchain handling except for when a transaction is considered settled. This is left to each user, and the only guidance that is

[2] There is no need to overpass the "dust" limit because coinbase transactions are not forwarded over the network prior inclusion in a block

given is that the more block confirmations the less the probability of reversal. To guarantee the infeasibility of peg-out double-spends we must add to the Nakamoto Consensus in the sidechain a condition that must be met to accept transactions. First, we introduce a definition: A network node is **T-synchronized** if its local best chain is up T seconds behind the current local time.

To build a SyncChain we need that the sidechain meets the following new condition: nodes only accept a transaction as settled if they are T-synchronized. We could also accept the transaction W if it is confirmed by a huge amount of cumulative difficulty, but we'll use the simplest possible definition now.

Now the new protocol also have a link-out, a coinbase and a peg-out transaction, but the information stored in the link-out's sidechain checkpoint slot is the block hash of A, the block number of A, and the timestamp of A. Another difference is that we don't need any wait between block B and C.

Peg-out Security

Because of the T-synchronization property, we can see that any attack that reverts the sidechain must add at least one sidechain block with timestamp past the timestamp of block C, where block 4 is checkpointed. Therefore, as long as the time between block 4 and block C is greater than T (which can be assured by the RTA), the attacker chain (on the sidechain) can never confirm transactions. This is assuming the node local time can only have advanced since the attack began.

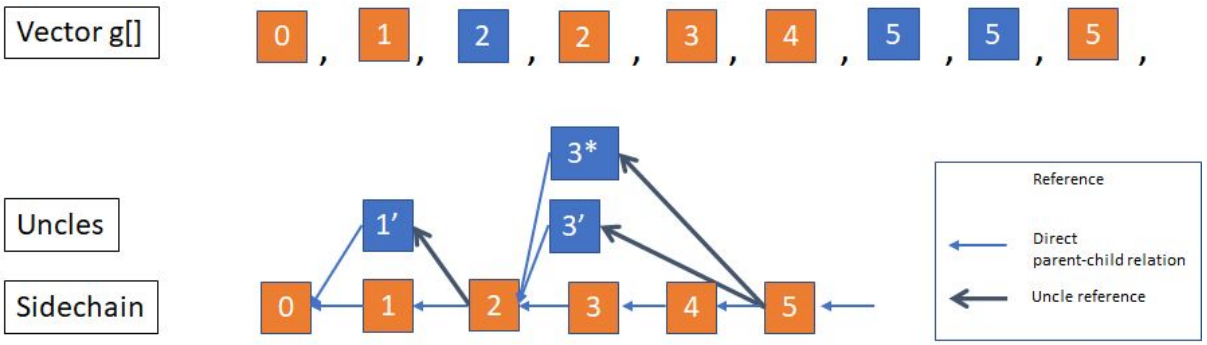
For low value payments, the whole peg-out process when using the AdjustedTime could take only 12 minutes, while high-value payments would require coinbase anchoring, and therefore would take on average 16 hours (100 bitcoin blocks).

Peg-out for a GHOST-CSC (GHOST-CSC) SyncChain

It's also possible to achieve the same result without a new consensus rule on transaction settlement and yet maintain the GHOST algorithm to select the best chain. The idea is to include in the mainchain multiple sidechain checkpoints so that attempts to rewrite the sidechain always hit one checkpoint before the sidechain is reorganized. To achieve this, we store different information in the sidechain checkpoint slot (the payload in the OP_RETURN in the link-out transaction). We won't store full information of block A, but only reduced information, in a field we call **Commit to Sidechain Checkpoints (CSC).** This field is a vector and has similarities with the Commits to Parents Vector (CPV) used in Armadillo. It describes in a succinct way a series of sidechain checkpoints parents starting from block A, and going backwards, by using short hash digest prefixes. Although these succinct checkpoints are not secure under standard cryptographic assumptions, they provide security under proof-of-work computational assumptions. We note again that we use the term

“sidechain checkpoints” pointer to blocks in the sidechain and “mainchain checkpoints” pointer to blocks in the mainchain. Let’s define **WorkWord(B)** to be the 2 least significant bytes of the hash of the Bitcoin block associated with the sidechain block B by merge-mining. Let A_n be the sidechain block at height n. The CSC is a zero-started uint16 array of length L, where CSC[i] is a tuple (n,w) where n is a block number and w is WorkWord(A_n). The CSC is serialized in a way to allow an unique deserialization. The CSC[i].n values are chosen according to a function f(i), which we’ll now describe. This description does not lead to an efficient implementation, but is simple to understand. All past blocks numbers starting from A_{BN} and going backwards in block number are stored in a vector g[j]. Every block number must appear in g[j] as many times as block uncles it references. The first element is g[0]=w, where w is the height of block A. For example, if w=1000 and block 1000 references no uncles, and block 999 references 3 uncles, then g starts with <1000,999,999,999,998,...>. We set f(i)=g[k_i], where k_0 = 0 and k_i is first index in g such that k_i-k_{i-1} = i*U. In other words, the distance from f(i) to w in g is U*i*(i+1)/2. U is chosen to be equal to the minimum amount of blocks that can pack D units of proof of work by continuously increasing its difficulty.

For example, if a block number 5 contains 3 uncle references, then the block consumes 3 slots in the linearized checkpoint space. The following figure shows how linearization works.



Generating the vector g[]

To prove security, we need to use in the sidechain a difficulty adjustment algorithm that adjusts down and up slowly. For example, the algorithm can establish a maximum reduction of 1/400 of the difficulty per block, as RSK currently does.

With D=60 and the +-1/400 difficulty adjustment factor per block, we obtain U=56. This means that 56 blocks can pack the same difficulty of 60 with maximum upward adjustment, Assuming D=60, U=56, w=1000, and no uncles in the best chain, the function f(i) starts with the sequence (1000, 944, 832, 664, 440, 160). Assuming 1 uncle per block, the sequence results in (1000, 972, 916, 832, 720, 580, 412, 216).

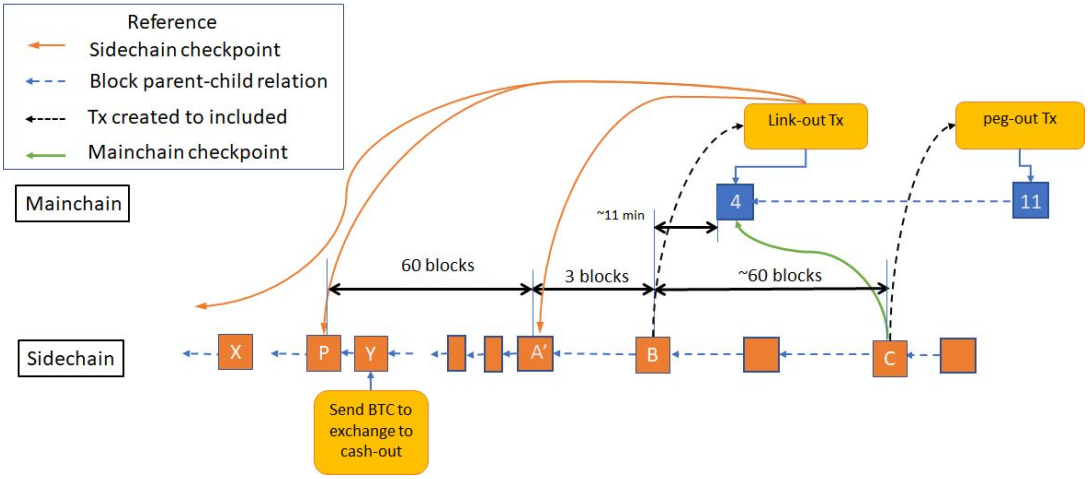
We can make L long enough that it overlaps with the block number where the previous link-out transaction is checkpointed, and therefore the CSC vector will cover the whole blockchain. However, we can also set L to a high value such that reverting L checkpoints is consider computationally infeasible.

We again analyze reverting only RSK, while we won't analyze reverting RSK and Bitcoin because this analysis does not differ from the previous cases.

Peg-out Security

The CSC of the peg-out transaction establishes checkpoints back in the sidechain that the sidechain must match. When an attacker tries to create a competing block of a sidechain checkpoint, the probability to match the block checkpoint (the WorkWord) is 2^{-16}. In other words, if sidechain difficulty is not decreased, the attacker must perform 2^{16} times the average amount of work per block to produce a different best chain that matches the WorkWord established by the CSC.

We present an example of the best possible attempt to double-spend a peg-out by reverting only RSK, and we show how the system reacts in this case, deterring the attack. The following figure shows an example of an attack.



An attack attempt to a GHOST-CSC SyncChain

The attack preparation phase starts with the attacker having RBTC in block X. The attacker initiates a normal peg-out of the RBTC in the block A, by sending the RBTC to the bridge contract, in a similar process that was described before. When this process finishes, the attacker owns the released BTC. Now the attack phase starts. The attacker tries to revert the RSK blockchain to a point prior to A. Immediately after some CSC-checkpointed block P, he creates a block Y that transfers the RBTC to an online crypto-exchange, in order to cash-out as soon as possible. The best attack is to choose Y to be between blocks f(1) and f(0). The attacker then tries to mine the blocks following Y to outperform the current RSK best chain with more cumulative difficulty. He aims to convince all network nodes to accept (at least temporarily) the attacker's chain. In this supposedly new best chain the attacker has cashed-out the RBTC but has also received the BTC in Bitcoin. In the figure, the orange lines show which blocks the CSC is pointing to (X, P and A).

All of the attacker's blocks must fit between two sidechain checkpoints. If they can't fit, then the average effort to mine the attacker's chain will go up by 2^{16} units of the PoW at the difficulty of the sidechain checkpoint that is crossed. In this example, the attacker's chain

needs to contain at least 63 units of proof of work to outperform the best chain. The attacker could try to pack as many uncles as he can in less than 56 blocks to increase the cumulative difficulty, but because of how the function g() was constructed, he can't. Therefore the best he can do is try to lower the difficulty during the first 55 blocks to reach the checkpoint with the lowest possible block difficulty, so that the 2^{16} multiplier applies to that lowered difficulty. With the aforementioned difficulty adjustment algorithm, the difficulty can decrease only 14% in 56 blocks, and therefore the average cumulative difficulty of the attacker's branch required to beat the best chain is at least 940 times higher than the cumulative difficulty of the blocks reversed. Another strategy would be to raise the block difficulty in order to avoid reaching the the sidechain checkpoint, but because U is exactly the minimum amount of blocks that can hold the cumulative difficulty of 60 blocks with uniform difficulty, this can't be done. Note that for simplicity we haven't taken into account a state where the difficulty decreases after block A. In this case, the number of confirmation blocks to wait can be dynamically adjusted (it would increase to 65).

Migrating RSK to a SyncChain

Although SyncChain provides many benefits over an SPV Sidechain, it is too early to say when and how RSK could transition to become a SyncChain. There are benefits but also there are migration risks. Coding, testing, simulations and security audits must validate each design decision. However having such a clean design that provides so many benefits both from usability and security is comforting. As soon as all the R&D stages have been concluded, we'll open the discussion for a migration to a SyncChain, and hopefully, with the feedback of the RSK community, we may see the migration in 2020 or 2021.

Summary

In this article we presented the SyncChain, a new type of sidechain that reduces the peg-in time. In the case of RSK, this could go from 16 hours to only 30 minutes. The time for peg-out can be reduced also from 16 hours to about 1.6 hours, if RSK keeps a rate limiter so that no more than 38 BTCs are transferred per hour. We also presented a variant of the SynChain that uses coinbase anchoring to provide unconditional peg-out security, achieving a peg-out time of 16 hours. This is less than what RSK currently requires, while RSK only provides cryptoeconomic security. We showed 3 protocol variants that achieve the same security guarantees under different consensus assumptions (GHOST-CSC, TS, and MPCW). We also proposed a new opcode OP_CHECK_INPUT_BLOCK_HASH for Bitcoin that enables peg-outs with unconditional security in a few hours, and without requiring a rate limiter.

The synchain also provides other secondary benefits for RSK such as :

1. It reduces the amount of code RSK runs in consensus in rskj: some of that code functionality is now provided directly by bitcoind.

2. Depending on the protocol selected, it provides unconditional security or it forces an attack on RSK to become an attack on Bitcoin: the M.A.D. property in game theory.
3. It allows cross-address peg-ins, such as investing in a crowd-fund directly from Bitcoin

The SyncChain also has some minor downsides. For example, SyncChain cannot provide short settlement finality. Considering pros and cons, SyncChain is certainly a superior protocol for Bitcoin sidechains.

References

[1] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In Cryptographers' Track at the RSA Conference, pages 277–292. Springer, 2017

[2] A Proposal to Modify Satoshi Consensus to Enhance Protection Against 51% Attacks. A Penalty system for Delayed Block Submission.
https://www.horizen.global/assets/files/A-Penalty-System-for-Delayed-Block-Submission-by-Horizen.pdf

[3] Delaying chain reorgs.
https://bitslog.com/2013/06/26/the-bitcoin-eternal-choice-for-the-dark-side-attack-ecdsa/

[4] Transaction Finality through Ledger Checkpoints
(https://ieeexplore.ieee.org/document/8975825 or
https://github.com/MuhammadNurYanhaona/checkpoint-paper/blob/master/checkpoint-paper-reviewed.pdf)

[5] Securing Proof-of-Work Ledgers via Checkpointing (https://eprint.iacr.org/2020/173.pdf)