RSK: A Bitcoin sidechain with stateful smart-contracts

Sergio Demian Lerner, Javier Álvarez Cid-Fuentes, Julian Len, Ramsès Fernàndez-València, Patricio Gallardo, Nicolás Vescovo, Raúl Laprida, Shreemoy Mishra, Federico Jinich, Diego Masini

IOV Labs Ltd.

Abstract

In recent years, Bitcoin and Ethereum have emerged as the two largest and most popular blockchain networks. While Bitcoin provides the most secure digital asset, Ethereum provides the smart contract execution platform with the richest application ecosystem. In this paper, we present RSK, a sidechain that extends Bitcoin with Ethereum-compatible and stateful smart contract functionality. RSK's goal is to bring Ethereum's advantages to Bitcoin, allowing Bitcoin users to fully benefit from decentralized finance without having to exchange their bitcoin for other assets or having to renounce Bitcoin's consensus security. As a sidechain, RSK does not define a currency of its own, and thus, RSK does not compete with Bitcoin as store of value. Instead, RSK extends Bitcoin with new capabilities without taking resources from the Bitcoin network. Among other features, RSK provides a highly secure mechanism to transfer bitcoins from Bitcoin and back, and implements a merged mining protocol that provides Bitcoin miners with additional fees without adding significant overhead.

1. Introduction

Several years after Satoshi Nakamoto proposed his peer-to-peer electronic cash system [1], Bitcoin has become the greatest blockchain network in terms of size, volume, popularity, and market capitalization. Meanwhile, Ethereum [2] has emerged as the reference smart contract platform for decentralized finance. Bitcoin and Ethereum serve two distinct purposes, however, they compete in human, computational and monetary resources against each other. Although there are some mechanisms to transfer assets between the two platforms, there is not yet a satisfactory solution that brings the functionality of both blockchains together.

In this paper, we present Rootstock (RSK), a *sidechain* that extends Bitcoin with the ability to run Ethereum-compatible and stateful smart contracts. Although there are numerous ways to define a sidechain [3, 4, 5, 6, 7, 8], in this paper, we propose a more restrictive definition that better captures the general view of sidechains. The widely accepted view of a sidechain is a blockchain that enhances a *parent* blockchain in some way, such as improving its performance or providing a new functionality [9]. To achieve this, the sidechain requires a mechanism to transfer assets from the parent chain and back. Thus, we propose to characterize sidechains using the following definition:

Definition 1 (Sidechain). *A sidechain is a blockchain that satisfies three conditions: i) it has a native token pegged to a parent chain token in a way that permits secure transfers between both chains; ii) it does not require a money token different from the main token of the parent chain to function; and iii) it achieves transaction finality independently from the parent chain.*

Our definition puts emphasis on the two-way transfer of assets and on aligning the incentives of the sidechain and the parent chain. The two-way transfer of assets is an essential property that is included in previous definitions of sidechain. Without a two-way peg, a sidechain would be just an alternative blockchain that utilizes parts of another blockchain. The novelty of our definition lies in the second condition, which aligns the incentives of the sidechain with the parent chain. We believe that this condition is crucial for a sidechain because a blockchain that takes advantage of parts of another blockchain but uses a different money token competes as store of value, and thus it is not simply providing improvements or additional features. Our definition captures the idea that a sidechain extends the functionality of the parent chain without being an alternative blockchain that can function independently. The third condition establishes exactly the degree of independence required, which is that even if the main chain is down, slow or congested, the sidechain can keep confirming transactions (with the exception of cross-chain transfers). This property is essential to scalability and sustainability of the sidechain, as it isolates the cost of transaction settlement in the sidechain from the cost of transaction execution or data publication in the main chain. Second layer solutions [10] are known to depend on the main chain for transaction settlement, and therefore are not commonly considered sidechains.

The fact that RSK satisfies our sidechain definition means that, unlike Ethereum or other blockchains, RSK does not compete for resources with Bitcoin. Instead, RSK shares Bitcoin's goals and incentives, and strengthens Bitcoin by bringing decentralized finance to its participants in the most efficient manner. RSK is the first Bitcoin sidechain with a Turing-complete and stateful smart contract execution engine that uses bitcoin as native currency. Since RSK can be secured using the same computational power that secures the Bitcoin network, RSK has the potential to be the most secure smart contract platform that exists. In addition to this, RSK provides decentralized and highly secure bitcoin transfers from the Bitcoin network and back, and can be used to achieve fast and inexpensive bitcoin transfers between users. In the rest of this paper, we refer to the process of transferring value from the parent chain to a sidechain and back as *peg-in* and *peg-out* respectively.

The remainder of this paper is organized as follows: Section 2 reviews existing blockchain networks that can be considered sidechains, Section 3 describes the RSK blockchain in detail including consensus and mining, Section 4 presents the bridging protocol that RSK uses to communicate with Bitcoin, and Section 5 concludes the paper.

2. Related work

In this section, we review the various existing solutions and proposals that are related or close to the definition of sidechain even if they do not fit completely with our Definition 1.

Dilley et al. [5] propose the notion of *strong federations* and present Liquid, a Bitcoin sidechain based on this concept. A strong federation is a two-way pegged sidechain where a number of federated entities called *functionaries* are in charge of the sidechain consensus and the transfer of assets between the sidechain and the parent chain. In Liquid, members of the federation take turns to propose new block additions to the blockchain, which are verified by the other members of the federation. In this way, Liquid achieves consensus in a byzantine fault-tolerant manner. In order to transfer value from Bitcoin to Liquid (i.e., peg-in), users must send bitcoin to the federation and then create a Liquid transaction that references these bitcoin. In order to transfer value from Liquid to Bitcoin (i.e., peg-out), users need to create a especial transaction in Liquid that destroys the assets and references a Bitcoin address. The federation continuously monitors the Liquid blockchain and creates a Bitcoin transaction when it finds a peg-out transaction in Liquid. Liquid mainly serves as a network of international exchanges that provides asset issuance, and more privacy and higher transaction throughput than Bitcoin.

The POA Network [6] is an Ethereum-compatible blockchain that provides higher transaction throughput and lower fees than Ethereum by using a *Proof-of-Authority (PoA)* consensus algorithm. The POA algorithm is similar to strong federations in that a set of functionaries take turns in producing block candidates to be included in the blockchain. In POA, the functionaries consist of a fixed number of public U.S. notaries. The POA Network then provides bridging capabilities to transfer POA tokens from Ethereum to the POA Network and vice versa. POSDAO [11] is a blockchain similar to the POA Network but one that uses a *Proof-of-Stake (PoS)* consensus algorithm.

Kiayias et al. [4] propose a construction for trustless *Proof-of-Work (PoW)* sidechains based on *Non-Interactive Proofs of Proof-of-Work (NIPoPoWs)*, which allow building succinct proofs of the occurrence of events in PoW blockchains. With this construction, users can transfer assets between two blockchains that support smart contracts. To transfer an asset, the user makes a deposit in one blockchain to a smart contract that generates a specific event. The user then builds and submits a NIPoPoWs of this event in the other blockchain to a smart contract that verifies the proof and credits the user's account with the appropriate balance. Kiayias et al. illustrate this procedure by transferring ETH from Ethereum, acting as parent chain, to Ethereum Classic [12], acting as sidechain. In this scenario, the transferred ETH becomes an ERC20 token named ETH20 in Ethereum Classic.

Binance Smart Chain (BSC) [13] is a sidechain of the Binance Chain (BC) blockchain that provides Ethereum-compatible smart contract functionality. Asset transfers from BC to BSC are achieved via a light client smart contract on BSC. Validating BC data on BSC is possible using a single block header because BC implements Delegated Proof-of-Stake (DPoS) consensus with instant finality. Asset transfers from BSC to BC are achieved using a set of oracles that rely and validate BSC data on BC. This set of oracles is equivalent to the set of validators used in the DPoS consensus algorithm.

Gaži et al. [14] provide a formal cryptographic definition of the notion of sidechains with emphasis on security. Gaži et al. then provide a construction of a PoS sidechain based on the Ouroboros algorithm [15].

Garoffolo and Viglione [7] propose a sidechain construction compatible with Horizen [16], which is a Bitcoin-like PoW blockchain. This construction allows the attachment of multiple sidechains to Horizen through a protocol called the Cross-Chain Transfer Protocol (CCTP). The different sidechains are free to implement their own consensus algorithm, and use the Horizen parent chain as a provider of a universal token of value called ZEN. CCTP then defines how ZEN can be transferred from Horizen to the sidechains and vice versa.

Zendoo [17] is an improvement for CCTP that introduces zk-SNARKS [18] to simplify the process of transferring assets from sidechains to Horizen.

Among these existing approaches, only Liquid can be considered a sidechain under our Definition 1 because it is the only blockchain that does not define a money token different from the main token of the parent chain. To the best of our knowledge, RSK is the first Bitcoin sidechain that fits under our Definition 1, and one that allows the execution of Turing complete smart contracts using bitcoins.

3. The RSK blockchain

RSK is a sidechain that extends the Bitcoin blockchain with stateful smart contract functionality. To achieve this in a seamless manner, RSK adheres to our sidechain definition in Definition 1. On the one hand, RSK does not provide a different currency from bitcoin and does not compete with Bitcoin as a store of value. On the other hand, RSK provides a secure system to transfer bitcoin from the Bitcoin network and back. Although RSK does not provide a currency different from bitcoin, we refer to bitcoins in RSK as RBTC to differentiate them from bitcoins in the Bitcoin blockchain (i.e., BTC). In this Section, we describe the RSK blockchain in detail, while in Section 4, we present the bridging system between RSK and Bitcoin called the Powpeg.

RSK's blockchain and consensus are based on Ethereum [2]. Like in Ethereum, the RSK network maintains a global state that can be modified using transactions. This global state is defined by data stored in accounts, which can be of two types: Externally Owned Accounts (EOA) and smart contract accounts. All accounts store nonce and balance, while smart contract accounts also contain program code and persistent storage. Transactions modify RSK's global state by transferring value between accounts or by executing smart contracts that modify their storage. Accounts in RSK are identified by 20-byte addresses.

RSK's smart contract execution engine, or the RSK Virtual Machine (RVM), is also based on the Ethereum Virtual Machine (EVM). The RVM is Turing-complete, uses *gas* as the unit of computation, and implements the same operations as the EVM. This means that Ethereum smart contracts can be deployed and executed on RSK with minimal or no changes at all.

Block headers in RSK are similar to Ethereum block headers with some differences. RSK block headers contain the following fields as defined in Ethereum: *parentHash, unclesHash, coinbase, stateRoot, txTrieRoot, receiptTrieRoot, logsBloom, difficulty, number, gasLimit, gasUsed, timestamp, extraData*, plus the following additional fields:

paidFees: a scalar value equal to the total paid fees in transactions in the block.

minimumGasPrice: a scalar value representing the minimum gas price for a transaction to be included in the block.

uncleCount: an integer equal to the number of uncles of this block.

bitcoinMergedMiningHeader: 80-byte Bitcoin block header for merged mining.

bitcoinMergedMiningMerkleProof: the Bitcoin Merkle tree proof for merged mining.

bitcoinMergedMiningCoinbaseTransaction: part of the Bitcoin merged mining transaction for merged mining.

The field *minimumGasPrice* is set by miners and can only change a 1% in value from one block to the next. The field serves two main purposes. On the one hand, it helps users to estimate the gas price for their transactions. On the other hand, it prevents gas price from dropping below a certain value during early stages of deployment due to a lack of network activity. This protects RSK from denial-of-service attacks consisting in flooding the network with low gas price transactions.

RSK uses the fields *paidFees* and *uncleCount* during fee distribution, which is explained in Section 3.4. The rest of the RSK-specific header fields are used during the merged mining process, which is explained in Section 3.5.

3.1. The RSK Virtual Machine

The RVM is based on the EVM with some key differences. On the one hand, the RVM implements the same instruction set as the EVM but with different gas costs. In addition, RSK uses a gas estimation algorithm that allows estimating the gas limit of a transaction with a single execution instead of the 24 executions required by Ethereum on average. On the other hand, the RVM provides the `OP_HEADER` opcode at the beginning of the smart contract code to specify the runtime environment in which the code must be executed. This allows for future changes in the RVM, instruction set, and binary object format. Finally, the `BLOCKHASH` opcode in RSK does not include the proof-of-work due to the way in which RSK blocks are produced (see Section 3.5). This means that `BLOCKHASH` in RSK should not be used as a source of cryptoeconomic randomness. Instead, RSK provides more reliable sources of entropy via precompiled smart contracts.

3.2. Precompiled Contracts

RSK supports all of Ethereum active precompiles except BLAKE2 [1]. In addition, RSK provides the following precompiled contracts not available in Ethereum:

- `BRIDGE (0x0...1000006)`: Handles the process of transferring bitcoin from and to the Bitcoin blockchain (see Section 4) and acts as a Bitcoin oracle.
- `REMASC (0x0...1000008)`: Distributes rewards to miners. Called once every block (see Section 3.5).
- `HD_WALLET_UTILS (0x0...1000009)`: Provides utilities to develop a Bitcoin Token Offering.
- `BLOCK_HEADER (0x0...1000010)`: Provides access to the past 4,000 RSK block headers. This is useful for proving transaction execution or blockchain congestion [19].

[1] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-152.md

3.3. The Unitrie

As mentioned before, RSK maintains a global state based on two types of accounts: EOAs and contracts. EOAs are controlled with an Elliptic Curve Digital Signature Algorithm (ECDSA) private key and store nonce and balance in RBTC. Contract accounts store nonce, balance in RBTC, and program code and storage cells. The contents of all accounts at a given time defines the global state of the RSK blockchain. Inspired by Ethereum, RSK uses a trie structure, called the Unitrie, to store this global state. In contrast to Ethereum's radix 16 trie, the Unitrie is a binary radix tree that combines accounts and storage in a single data structure. Figure 1 shows the main structure of the Unitrie.

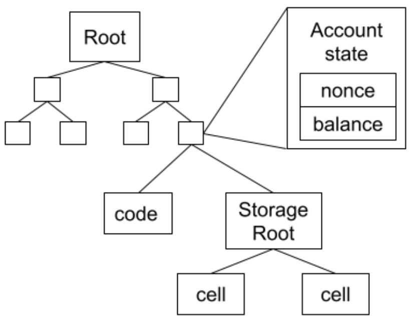

Figure 1: Unitrie organization of data.

The Unitrie works externally as a key-value database. Internally, each non-leaf node defines a key prefix that is shared among all of its child nodes. The Unitrie uniformly distributes data nodes across the tree structure to minimize node access times. This is achieved by combining account and storage cell addresses in a single key path using 10-byte prefixes as shown in Figure 2.

In addition to the trie unification, the Unitrie is designed to minimize the amount

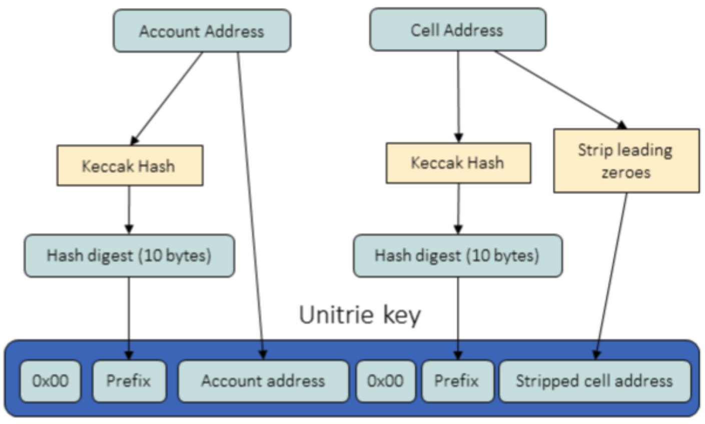

Figure 2: Unitrie key derivation from account and cell addresses.

of space that EOAs and storage cells consume in their encoded form. On the one hand, EOAs in the Unitrie do not contain a storage tree root or code hash as in Ethereum, which reduces their size by 64 bytes. This results in most EOA nodes in RSK taking less than 42 bytes in their encoded form. On the other hand, the Unitrie embeds small leaf nodes (44 bytes or less) into their parent nodes, which removes the need for an additional 32-byte hash digest reference and for storing the indexing keys of embedded nodes in caches and key-value databases. These optimizations result in EOAs taking approximately 20% of the space consumed by an EOA in the Ethereum's trie. This compression also allows the Unitrie to store more accounts in memory than Ethereum, which increases the cache hit ratio, reduces the number of I/O operations, and allows for higher transaction throughput.

3.4. The DECOR consensus algorithm

RSK implements a modified version of Nakamoto consensus [1]. In Nakamoto consensus, miners choose the tip of the blockchain by committing their computational power to a particular block. Forks can occur when different miners commit their computational power to different blocks that have been mined at the same time. The network solves the conflict when miners progressively switch to the branch of the blockchain with the highest amount of accumulated proof-of-work. RSK's consensus algorithm, called Deterministic Conflict Resolution (DECOR) [20], is based on GHOST [21] and addresses this type of conflicts in a more efficient manner that is deterministic (assuming that all parties have access to the same blockchain state information), maximizes miners' revenue, and takes negligible execution time.

DECOR is mainly based on a pair of functions: the selection function S and the reward function R. The S function enables fast and accurate consensus between miners when dealing with conflicting blocks, and prevents the consensus mechanism from manipulation. The S function must be deterministic, meaning that the miners must agree on the same block; and non-forward settable, meaning that miners should not be able to influence the block selection criteria in their favor. The reward function R enables the design of incentive mechanisms for miners to obtain specific behaviors. The R function can update the rewards for the conflicting blocks, and can be modified to consider other factors besides the mining difficulty or the transaction fees. A non-forward settable S function combined with punishment fees in R provides the blockchain with selfish-mining resistance.

RSK implements DECOR in the precompiled Reward Manager Smart Contract (REMASC). Newly mined RSK blocks include a especial transaction that pays all transaction fees to the REMASC contract instead of the miner. The REMASC contract then distributes the fees among various participating parties after a number of blocks. REMASC stores rewards in an internal account named Reward Balance, and distributes the rewards for block number N in block number $N + 4,000$.

In the following, we refer to blocks that have a common parent as *siblings*, and we refer to the sibling of the parent of a block as an *uncle* block. Miners receive additional rewards for including references to uncle blocks in their block headers. We refer to these miners as *publishers* of the uncle blocks.

The S function in REMASC requires the network to forward conflicting blocks up to ten steps back from the tip of the blockchain, and solves conflicts by choosing the following blocks (in order of preference):

1. The block with the higher number of uncles
2. The block with the higher reward (only if the reward of one of the blocks is more than double the reward of the other block)
3. The block with the lowest hash

The reward function R in REMASC uses 10% of Reward Balance to reward the participating parties at block $N + 4000$. This amount is called the Full Block Reward or ρ_{block}, and it is distributed as follows: past collaborators and core developers receive

$$\rho_{contrib} = \frac{1}{5} \cdot \rho_{block} \quad ,$$

the Pegnatories (see Section 4) receive

$$\rho_{peg} = \frac{\rho_{block} - \rho_{contrib}}{100} \quad ,$$

and the miners receive

$$\rho_{miner} = \rho_{block} - \rho_{contrib} - \rho_{peg} \quad .$$

If the block has no siblings at height N, the miner receives ρ_{miner} if he respected the selection function S, and $\rho'_{miner} = \frac{9}{10} \cdot \rho_{miner}$ otherwise. In the case the block has siblings, REMASC distributes ρ_{miner} among miners and publishers. Taking into account that each sibling has a miner and a publisher, we define the total publishers reward as

$$\rho_{pub} = \frac{\rho_{miner}}{10} \quad ,$$

and the total miners reward as

$$\rho_{miner} = \rho_{miner} - \rho_{pub} \quad .$$

Each publisher then receives

$$\frac{\rho_{pub}}{N_s} \quad ,$$

where N_s is the total number of siblings. Each miner receives

$$\rho_{ind\,mining} = \frac{\rho_{miner}}{N_s + 1}$$

if S was respected or

$$\rho'_{ind\,mining} = \rho_{ind\,mining} - \frac{\rho_{ind\,mining}}{10}$$

otherwise. In addition, REMASC reduces the individual miner reward of sibling blocks by

$$\frac{(D-1)\rho_{ind\,mining}}{20} \quad ,$$

where D is the distance between the sibling block and the block that includes a reference to the sibling in its block header. This penalizes the late publication of siblings on the blockchain.

3.5. Merged mining

RSK uses the same proof-of-work (PoW) algorithm as Bitcoin, which consists of generating block headers in iteration until the block header SHA-256d hash is lower than a predefined target. The lower the target, the more difficult it is to find a header that produces a valid hash. RSK defines a higher target than Bitcoin, which means that miners can produce RSK blocks at a higher rate than in Bitcoin.

RSK blocks are merged mined together with Bitcoin blocks. Merged mining is a mechanism to utilize the available computational power to mine blocks in multiple blockchains at the same time. This allows miners to claim fees in all of the involved blockchains. Merge mining RSK with Bitcoin means that RSK does not compete for hashing power with the Bitcoin network, and further aligns RSK with our sidechain definition in Definition 1.

Blocks in RSK must include a Bitcoin block header in their *bitcoinMergedMining-Header* field. This means that publishing a block in the RSK blockchain requires producing valid Bitcoin block headers by design. The SHA-256 hash of the Bitcoin header included in the RSK block must be lower than the RSK target to be accepted in the RSK blockchain. Miners are faced with three possible scenarios every time a Bitcoin header is generated as part of the mining process:

1. The block's header hash is lower than the RSK target: the miner includes the Bitcoin block header in an RSK block and publishes on the RSK blockchain.
2. The block's header hash is lower than the Bitcoin target: the miner publishes the block on the Bitcoin blockchain and, since RSK defines a higher target than Bitcoin, the miner also uses the Bitcoin header to generate and publish a block in RSK.
3. The block's header hash is greater than both targets: the miner discards the header and the miner generates a new one.

This merged mining scheme allows miners to further capitalize their hashing power by publishing and collecting fees in RSK with minimal overhead. The process also aligns the Bitcoin network with RSK, and allows RSK to be secured with the same amount of computational power as Bitcoin.

Miners need to insert some additional information in their blocks to guarantee the security of the merged mining process. On the one hand, miners need to include a reference to RSK, called the RSK tag, in the coinbase transaction of their Bitcoin blocks. On the other hand, miners must prove the correct placement of the RSK tag by filling the *bitcoinMergedMiningCoinbaseTransaction* and *bitcoinMergedMiningMerkleProof* fields in the RSK block header. The *bitcoinMergedMiningCoinbaseTransaction* field contains part of the Bitcoin transaction with the RSK tag, and the *bitcoinMergedMiningMerkleProof* field contains the Merkle tree path that proves the inclusion of the Bitcoin transaction in the *bitcoinMergedMiningHeader* header.

The RSK tag prevents miners from using the same Bitcoin header in multiple RSK blocks, and consists of the ASCII identifier "RSKBLOCK:" concatenated with the Keccak-256 hash of the referenced RSK block header (without the merged mining fields). Miners can insert the RSK tag in the coinbase input data or in a transaction output after the `OP_RETURN` code. The number of bytes immediately after the RSK block header hash up to the end of the coinbase transaction must be lower than or equal to 128. This means that the RSK tag will be generally included in the coinbase data or in the last four

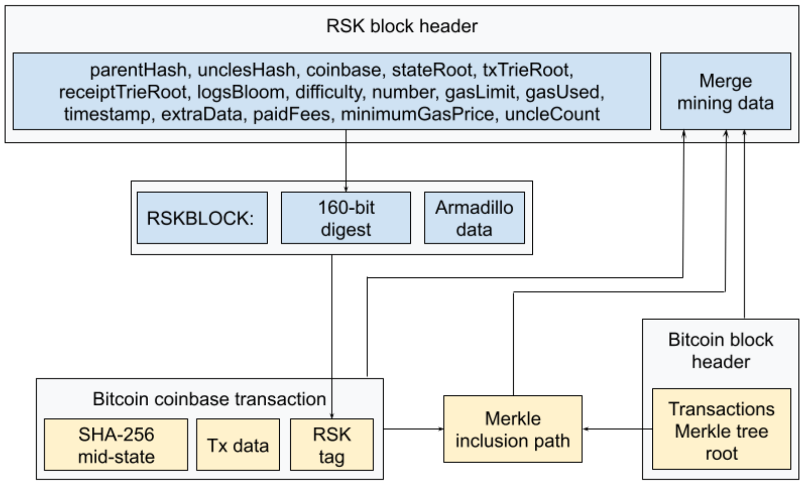
RSK block header
parentHash, unclesHash, coinbase, stateRoot, txTrieRoot, receiptTrieRoot, logsBloom, difficulty, number, gasLimit, gasUsed, timestamp, extraData, paidFees, minimumGasPrice, uncleCount
Merge mining data
RSKBLOCK:
160-bit digest
Armadillo data
Bitcoin block header
Transactions Merkle tree root
Bitcoin coinbase transaction
SHA-256 mid-state
Tx data
RSK tag
Merkle inclusion path

Figure 3: Placement of merged mining data in RSK and Bitcoin blocks.

outputs of the coinbase transaction. The RSK tag together with the proof of inclusion in the Bitcoin header establish a mapping from the Bitcoin header to the RSK header. To guarantee soundness of the construction, we assume it is unfeasible to produce a different mapping from the same Bitcoin header to a different RSK block performing less computation (or at a lower monetary cost) than the amount required to produce the associated proof-of-work. Figure 3 illustrates the relationship between Bitcoin and RSK blocks using merged mining data. Note that to optimize the size of the merged mining data, the transaction with the RSK tag is stored partially hashed using a 32-byte SHA-256 mid-state and a 64-byte aligned chunk of the transaction. We detail the contents of the Armadillo data in Section 4.

4. The Powpeg

The Powpeg is the two-way protocol designed to transfer bitcoin from the Bitcoin network to the RSK network and vice versa. RSK is a sidechain that does not provide a currency of its own, as stated in Definition 1. This makes the Powpeg one of the most critical parts of the RSK network. Contrary to other bridge solutions that rely on a single complex cryptographic protocol, such as tBTC [22] and RenVM [23], the Powpeg is based on a defense-in-depth layered security model, where each layer can be easily analyzed and tested.

The transfer of funds between the Bitcoin network and RSK is controlled by a vault and a smart contract respectively. Transferring bitcoins from the Bitcoin network to RSK (i.e., peg-in) requires the user to lock an amount of BTC on a vault on Bitcoin, which enables the unlocking of the corresponding amount on RBTC on RSK. Transferring bitcoins from RSK to the Bitcoin network (i.e., peg-out) requires the user to send an amount of RBTC to a smart contract on RSK, unlocking the corresponding amount of BTC from the vault in Bitcoin. In order to achieve these two processes with complete security guarantees, the Powpeg protocol consists of three main components: the Bridge

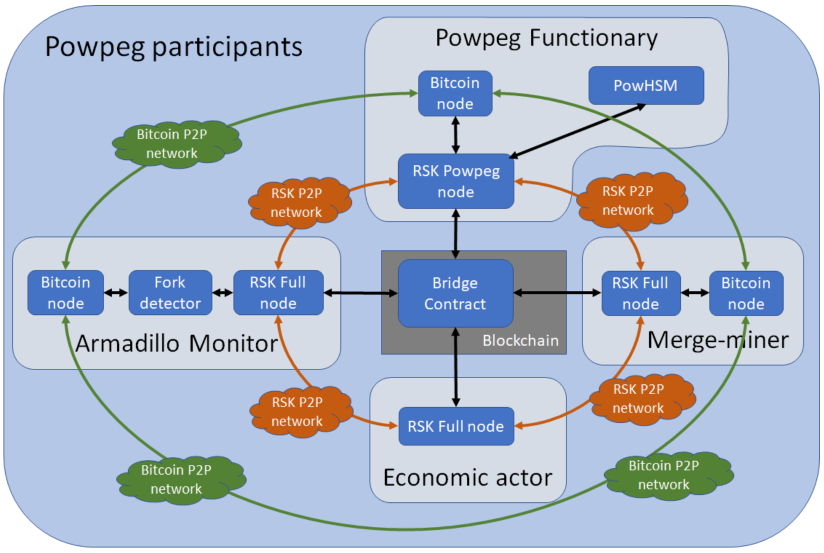

Figure 4: Architecture of the Powpeg

smart contract, the *Pegnatories*, and the *Armadillo* monitoring system. Figure 4 shows the different components of the Powpeg architecture.

4.1. The Bridge

The Bridge smart contract is a precompiled contract on RSK that provides consensus security guarantees to both peg-ins and peg-outs. The Bridge contract holds the total supply of RBTC and controls the locking and unlocking of RBTC in RSK. The Bridge contract also crafts Bitcoin transactions to unlock BTC from the vault when required. The Bridge contract acts as a Bitcoin light client that stores and validates Bitcoin block headers. Any user can submit Bitcoin block headers to the Bridge; however, this task is carried out usually by the Pegnatories.

4.2. The Pegnatories

The Pegnatories are a set of federated functionaries that oversee the correct operation of the Powpeg. Pegnatories indirectly control the vault on the Bitcoin network, which consists of a multi-signature pay-to-script-hash (P2SH) address. Each Pegnatory runs a Bitcoin node, a modified RSK node (the Powpeg node), and controls a tamper-proof Hardware Security Module that we call the PowHSM.

Each PowHSM controls a private key that can be used to unlock BTC from the vault (i.e., the multi-signature address). Although Pegnatories control the PowHSMs, they do not have direct access to the private keys, and they cannot force the PowHSMs to sign invalid transactions. The PowHSMs verify PoW consensus in SPV mode and require a minimum amount of cumulative PoW in order to sign Bitcoin unlocking transactions. This means that the PowHSMs can verify PoW consensus without knowing the identity of the block producers or the Pegnatories. Thus, the Pegnatories mainly serve as data relays between Bitcoin, RSK, and the PowHSMs. On the one hand, the Pegnatories inform the Bridge contract with Bitcoin block headers so that the Bridge light client remains updated. On the other hand, the Pegnatories transmit Bitcoin transactions from the Bridge contract to the PowHSM for validation and signing, and from the Bridge contract to the Bitcoin network for unlocking funds from the multi-signature address.

4.3. Armadillo

Armadillo is a monitoring system that enhances the security of the Powpeg and any other external service that is vulnerable to double-spending attacks. Armadillo detects the presence of malicious forks and alerts the RSK network, which can then transition to safe mode. External services, such as exchanges, can also react to Armadillo alerts to avoid double-spending attacks.

Anyone can run an Armadillo node, which consists of a Bitcoin node, an RSK full node, and the Armadillo monitoring process. Armadillo uses some data included in the RSK tag embedded in Bitcoin blocks to detect the presence of hidden RSK blocks. More precisely, Armadillo uses three fields in the RSK tag: the Commit to Parent Vector (CPV), the number of uncles, and the RSK block number. The CPV is a vector of 7 bytes that contains the least significant byte of the double SHA-256 hash digest of the *bitcoinMergedMiningHeader* field of the 7 RSK blocks positioned at predefined checkpoint positions in the RSK blockchain. In this way, the CPV links a Bitcoin block to a specific RSK branch.

A malicious miner building a hidden RSK blockchain will have to include references to the hidden fork in the CPV field of his Bitcoin blocks. Armadillo monitors the Bitcoin network and identifies Bitcoin blocks with a CPV that does not match the current main RSK branch. A Bitcoin block with more than one mismatch on the CPV with respect to the main RSK branch indicates the presence of a hidden RSK fork. Moreover, the number of mismatching blocks in the CPV reveals the approximate length of the hidden fork, while the frequency of the tags reveal the approximate mining capacity of the attacker. In this way, Armadillo and the CPV force malicious miners to give up on Bitcoin block rewards to keep their forks hidden, making malicious mining much less profitable.

Armadillo is designed to broadcast a Malicious Fork Proof (MFP) across the RSK network when detecting a hidden fork from an attacker with more mining capacity than the current RSK network. MFPs consist of various interconnected Bitcoin block headers with their RSK tag and Merkle inclusion proofs (recall that the RSK tag is embedded in a transaction). These headers serve as PoW and set a minimum cost for creating false alerts. Armadillo will keep distributing MFPs as long as the hidden fork persists, that is, until the honest RSK branch reaches the block numbers published in the MFP, or until the hidden fork is revealed and a re-organization occurs.

4.4. From Bitcoin to RSK

The process of transferring bitcoins from the Bitcoin network to RSK is called peg-in, and consists of the following steps:

1. The user sends N BTC to the multi-signature address controlled by the PowHSMs. This is the deposit transaction.
2. The user or the Pegnatories send the deposit transaction to the Bridge smart contract on RSK.

3. The Bridge contract verifies the number of confirmations and cumulative PoW of the deposit transaction. This is possible because the Pegnatories periodically relay Bitcoin block headers to the Bridge.
4. The Bridge contract transfers N RBTC to the user in RSK.

The Pegnatories periodically inform the Bridge contract about new block headers and peg-in transactions. However, users can also update the Bridge if the Pegnatories are offline.

The Powpeg currently supports two ways of specifying the address that receives the funds in RSK. The first mechanism consists of deriving the destination RSK address from the deposit transaction. More precisely, the RSK address is derived from the Keccak-256 hash of the user's public key, which is obtained from the *scriptSig* field of the first input in the deposit transaction. This derivation method is compatible with address formats P2PKH, P2SH-P2WSH and P2SH-P2WPKH, and ensures that the destination RSK address is controlled with the same private key that controls the deposit input. However, some hardware wallets do not allow using a single private key for multiple blockchains. To address this issue, the Powpeg also allows users to specify the destination RSK address in a separate `OP_RETURN` output of the deposit transaction.

Finally, the Bridge contract specifies a limit on the amount of BTC that can be locked on the multi-signature address. If a user sends BTC to the multi-signature address and this limit is exceeded, the Bridge contract refunds the user back through a peg-out process (see below) instead of unlocking RBTC on RSK.

4.5. From RSK to Bitcoin

The process of transferring bitcoins from RSK to the Bitcoin network is called peg-out and is more complex than the peg-in process due to the fact that the locked funds in Bitcoin cannot be controlled by a smart contract. The peg-out process consists of the following steps:

1. The user sends N RBTC to the Bridge contract on RSK. These funds are locked.
2. The Bridge contract builds a Bitcoin transaction that transfers N BTC from the multi-signature address to the user. This Bitcoin transaction is generated in RSK and thus it is embedded in an RSK transaction.
3. After a number of predefined block confirmations, the Pegnatories send the RSK transaction that generated the Bitcoin transaction to their PowHSMs.
4. The PowHSM verifies that the Bitcoin transaction has been generated by the Bridge contract and that the embedding RSK transaction carries a certain amount of cumulative work. The PowHSM signs the Bitcoin transaction if these conditions are met.
5. The Pegnatories send the PowHSM signature to the Bridge contract.
6. Upon receiving the required number of signatures, the Bridge contract assembles the signed Bitcoin transaction.
7. The Pegnatories send the signed Bitcoin transaction to the Bitcoin network, unlocking N BTC for the user.

The Bridge contract processes peg-out requests one at a time on a first come, first served basis. To build the unlocking Bitcoin transaction, the Bridge contract selects one or more UTXOs from deposit transactions to be spent. If the total amount of bitcoins locked in these UTXOs is greater than the user's peg-out, the Bridge contract creates a change output in the Bitcoin transaction that sends the remaining amount to the multi-signature address.

5. Conclusions

In this paper, we present RSK, a sidechain that provides Ethereum-compatible smart contract functionality on top of Bitcoin. Unlike other existing blockchains, RSK adheres to our strict definition of sidechain as it does not require a currency different from bitcoin to function. This means that RSK does not compete with Bitcoin in resources. Instead, RSK enhances the Bitcoin network and allows Bitcoin miners to claim extra fees by merge mining RSK blocks with no additional cost.

RSK allows bitcoin transfers from and to Bitcoin through a highly secure bridge called the Powpeg. The Powpeg is mainly controlled by a smart contract on RSK, and is overseen by a set of Pegnatories that act as information relays. The integrity of the Powpeg is guaranteed at all times by the Bridge smart contract and a set of tamper-proof security devices that control the locked funds.

Future research directions include faster transfers between RSK and Bitcoin, scalability and privacy improvements in RSK, and the deployment of second layer protocols on RSK among others.

References

[1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, Decentralized Business Review (2008).

[2] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, Tech. rep., Ethereum (2014).

[3] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling Blockchain Innovations with Pegged Sidechains, Tech. rep. (2014).

[4] A. Kiayias, D. Zindros, Proof-of-Work Sidechains, in: Proceedings of the International Conference on Financial Cryptography and Data Security, 2019, pp. 21–34.

[5] J. Dilley, A. Poelstra, J. Wilkins, M. Piekarska, B. Gorlick, M. Friedenbach, Strong federations: An interoperable blockchain solution to centralized third-party risks, arXiv preprint arXiv:1612.05491 (2016).

[6] I. Barinov, V. Baranov, P. Khahulin, POA Network, Tech. rep. (2017).

[7] A. Garoffolo, R. Viglione, Sidechains: Decoupled Consensus Between Chains (2018). arXiv:1812.05441.

[8] M. Ali, Stacks 2.0: Apps and Smart Contracts for Bitcoin, Tech. rep., Stacks (2020).

[9] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, K.-K. R. Choo, Sidechain technologies in blockchain networks: An examination and state-of-the-art review, Journal of Network and Computer Applications 149 (2020) 102471.

[10] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, SoK: Off The Chain Transactions, Cryptology ePrint Archive, Report 2019/360 (2019).

[11] I. Barinov, V. Arasev, A. Fackler, V. Komendantskiy, A. Gross, A. Kolotov, D. Isakova, POSDAO: Proof of Stake Decentralized Autonomous Organization, SSRN (2019).

[12] Ethereum Classic.
URL `https://ethereumclassic.org/`

[13] Binance Smart Chain.
URL `https://github.com/bnb-chain/whitepaper/blob/master/WHITEPAPER.md`

[14] P. Gaži, A. Kiayias, D. Zindros, Proof-of-Stake Sidechains (2019).

[15] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol, in: Proceedings of the International Cryptology Conference, 2017, pp. 357–388.

[16] Horizen - Cryptocurrency and sidechain platform.
URL `https://www.horizen.io/es/`

[17] A. Garoffolo, D. Kaidalov, R. Oliynykov, Zendoo: a zk-SNARK Verifiable Cross-Chain Transfer Protocol Enabling Decoupled and Decentralized Sidechains (2020). arXiv:2002.01847.

[18] E. Ben-Sasson, A. Chiesa, E. Tromer, M. Virza, Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture, in: Proceedings of the 23rd USENIX Security Symposium, 2014, pp. 781–796.

[19] R. Khalil, N. Dulay, Adaptive layer-two dispute periods in blockchains, Cryptology ePrint Archive, Report 2020/1601 (2020).

[20] P. Camacho, S. D. Lerner, DECOR+LAMI: A Scalable Blockchain Protocol, in: Scaling Bitcoin Workshops, 2016.

[21] Y. Sompolinsky, A. Zohar, Secure High-Rate Transaction Processing in Bitcoin, in: Financial Cryptography and Data Security, 2015, pp. 507–527.

[22] tBTC A Decentralized Redeemable BTC-backed ERC-20 Token, Tech. rep., Keep Network.

[23] RenVM.
URL `https://github.com/renproject/ren/wiki#how-it-works`