

Back-Step Blockchain for blockchains containing backward chained elements in Fintech applications

Background of the Application

Field of the Application

The application belongs to the field of electronic secure ledger technology especially for use in Fintech applications. Throughout this application BSB stands for Back-Step- Blockchain.

Discussion of Related Art

Any discussion of the prior art throughout the specification should in no way be considered as an admission that such prior art is widely known or forms part of common general knowledge in the field.

Blockchains are linked-list type data structures called blocks, where each block depends on the result of a cryptographic operation, normally a hash function, of the previous block. This results in a chain of blocks, where each block depends on the past blocks, providing protection against manipulation of past records. The block may contain a hash, a hash tree or other payload information that are intended to be secured by the Blockchain. Manipulating an earlier block in the chain means that following blocks need to be recalculated. Many Blockchains require the distribution of the block to multiple nodes in a network. In a dispute it is assumed that the majority of the nodes will have the correct chain. This means consensus of at least 51% of the nodes. To make the block calculation harder a so called “proof of work” is included in most blockchains. This is a computation intensive procedure that takes time and electricity. If the calculation of a block is distributed to many nodes working in parallel, one node will have the result of the “proof of work” first and informs all other nodes. Now, creating a manipulated Blockchain would require recalculating all previous blocks including all the “proof of work”. To manipulate a distributed Blockchain, more than half of the nodes need to cooperate and take over the new chain and claim that this is the correct chain. Distributing a faked chain may also be achieved using malicious code like viruses.

Other mechanisms have been developed, that replace the “proof of work” with less costly calculations like “proof of stake” as described in Ethereum.

Objects of the Application

The application provides an enhanced secure ledger technology, especially for fintech applications.

A further feature is to provide a secure blockchain without the need for “Proof of Work”.

A further feature is the elimination of the need for distributed nodes.

A further feature is the ability to provide physically small single chip blockchain solutions, including the ability to provide mobile blockchain solutions.

A further feature is to provide a secure blockchain, which can have a predefined maximum number of blocks.

A further feature is to provide a blockchain that can create a block within a defined number of operations, resulting in a defined response time for a block request.

A further feature is to provide a high speed blockchain capable to be implemented in a single electronic chip.

A further feature is to provide a blockchain, which can be integrated into other systems, sensors or other equipment close to or at the source of the payload data.

A further feature of the chain is to provide easy detection of forked chains

A further feature is to provide a blockchain that allows to detect manipulation by the owner of the chain.

Brief Description of the Drawings

Field that may contain a value or contain other fields



Function that may apply modification to a value hash or ☐

Object that comprises multiple fields, functions or other objects
An object may be a container that is detailed in a separate drawing



Paths or signals that appear on multiple drawings use the numbering from the drawing where they originated or being driven.

Figure 1 BSB Sample Application

The diagram shows one embodiment of the application as it may be used by financial institutions.

Figure 1.1 Function Flow Overview

This diagram shows the functional blocks and their basic relations. It is used to explain the basic flow of operations.

Figure 2 BSB Principle

The diagram shows the different elements of a block and the principle flow of information.

Figure 3 Back-Step Generator

The generation of the backstep values and the seal values are shown

Figure 4 Back-Step Generator - 2

. The figure shows how intermediate back-step values can be permanent or temporary stored to achieve the required performance.

Figure 5 Seal Value Delivery

The figure shows the delivery of seal values.

Brief description

The application does not require a “Proof of Work” or distributed networks and is therefore fast and can be implemented in small silicon chips in smart phones of bank customers. The application allows a finite number of steps in a chain. The number of steps may be astronomical large and can be adjusted to the application requirements. The key element of the application is the use of additional input values to the hash of the block. These values run from the **future to the past**. These values are called back-step value and back-step-seal value. The back-step value and the seal have a one-way property. Calculating from the current value to any value used in a previous step is easy, while calculating the next value to be used in a future block is cryptographically hard, without knowledge of a secret. Two secret must be kept confidential in the chain system. Verifying the chain is easy and requires two steps. First a backward calculation from the current block to the previous blocks and then a traditional forward hash verification. Both calculations are operational cheap and fast. In addition, the chain provides a seal that prevents against manipulation by the chain owner/operator. The chain provides an optional function, which allows authorities to verify the correct operation of the chain by the owner. This function is provided for government blockchain applications.

The BSB provides through its architecture, enhancements in speed, size and universal embodiment. Speed enhancements are essential for all applications, where a high number of blocks per time unit are required. The BSB uses a backwards chaining based on cryptographic functions to eliminate the “Proof of Work”, required in a traditional blockchain. Without “Proof of Work” a new block can be calculated in Microseconds. This is required by financial institutions where the number of transactions per second can be very high. An example implementation of the BSB method in an apparatus as small as a USB token can generate >1000 blocks in a second, at an example block size of 1024 byte. A further embodiment of the method is in a small silicon chip, which allows the use of a secure blockchain in small devices like smart phones. This enables bank customers to have their account data in a personalized blockchain and enables trusted payment applications. A third embodiment of the BSB is the integration of the method into sensors, which can seal the data at the source. This includes the integration into Black Boxes, like flight recorders. The number of embodiments is only limited by the imagination.

Specific description

The BSB supports a finite number of blocks. The maximum number of blocks may be defined as design parameter or as a configuration parameter, which is applied when the chain is initialized. The number of blocks is also called the steps of the chain. A BSB can have trillions of steps.

The BSB requires a secure area to keep secrets private, which are required for the trusted operation. The BSB method may be implemented in Software or Hardware or in a combination of both. It is the responsibility of the actual implementation to provide the required security. The implementation in Hardware implies the advantage that the required security for the secrets is easier to provide.

Beside the forward chaining of the blocks using a hash value, as known from a standard blockchain, the BSB comprises one or multiple chained field that propagate their values from the last block in a chain to the first block in a chain. These fields in a chain run backwards from the future to the past. The calculation of these values is performed in a secure area. Future values must only be calculated in the secured area. For longer chains and this is by far the regular case it may not be possible to store all the values in the secure area. For that reason, the BB may contain a logic to calculate and provide the values when they are needed. Between two adjacent blocks holds the relation, that backward chained fields of a block are calculated from the values of the next block in time, by applying a cryptographic function with “one way” properties. Between a current block and all previous blocks, the back-step values in the chain can be easily calculated and verified, but the reverse operation is cryptographically hard. The type of one-way function used is design dependent. It is required that brute force or other attacks are not feasible to succeed in the available time. Traditional forward chaining and backward chaining are combined in each block and provide the security of the chain and eliminates the need for proof of work known from state of the art blockchains.

An embodiment for securing transactions, including financial transactions is shown in Figure 1. At a financial institution 10 a BSB is used for bank specific applications to secure the history of transactions. A data base system 11 stores the transaction data, which are to be inserted as payload into a block. The data base system 11 sends the payload to the BSB device 12, which has the BSB method implemented. After a new block is generated by 12, it is returned to the requesting system 11 and stored in the blockchain database file. The ongoing chain of new blocks may be distributed via a computer network to other nodes 13, which may store copies of the blockchain on their own computer database file 14. These distributed computers may at any time verify the consistency of the chain but are not needed for the block generation. In this embodiment the block is totally calculated and secured inside the hardware device 12.

Functional Flow Overview

The BSB requires a set of basic functions to be operational. These functions are shown in Figure 1.1. The BSB provides connections to the outside environment. These are external information processing units like, but not limited to, servers, computers, bank terminals, public terminals, personal electronic devices, smartphones, machinery and others. The blocks are stores at this outside environment, not inside the BSB. Therefore, the BSB

comprises an input interfaces 101 and an output interface 106, where the calculated blocks are returned.

Processing a block needs the input to be sent to the block generation 105 data via path 108. Input data may be payload data, metadata, hash from the previous block if this is the first block after power-on and others dependent on the actual use case and the implementation. Following the first block, continues operation may store the hash of the previous block internally, so it does not to be input. A non-volatile storage may be implemented to keep the previous hash over power off cycles. This is an implementation dependent option. A start command may tell the control unit 102 to initiate the calculation of a new block.

The control unit signals the block request to the back-step-generator via path 109. The back-step-generator 103 communicates with the checkpoint and cache logic to present required back-step-values to the block-generation 105.

At that point the block is ready to be hashed. When the hash operation is completed the newly calculated block is passed to the output interface via path 112 and eventually stored in an optional non-volatile storage. Storing of the blocks in a database is the responsibility of the outside systems and is out of the scope of this document.

Block generation

The block generation using the BSB is shown in Figure 2. Therefore, two blocks 200 and 201 out of a longer chain are presented. Each block 200, 201 is a logical field and comprised of multiple detailed fields. A BB may contain among other application specific fields, a payload field 202, a metadata field 203, a previous hash value field 204 providing the forward chaining of the blocks, a seal field 205 and at least one back-step-value field 206. The forward chaining of the blocks is provided by the hash function 207. The forward chain can be verified by calculating the hash value 208 over the complete fields of a block. The hash value 208 calculated of block 200 becomes the input value to the next block 201. The backstep value 420 can be calculated from an initial value inside the back-step generator 210 all the way down to the current block, by repeated application of a one-way function 209. A Back-Step-Seal value, temporarily stored in field 213 is used to create the seal value 421, which may be used to provide additional security of the chain against intentional manipulation. Therefore, the back-step-seal value 418 of the current step is applied to a XOR function 411, together with the hash value 208 of the previous step and results in the value 419. This intermediate value is the input to the one-way function 412, which creates the seal value 421, which is placed into the block field 205. In cases where the block length of the forward hash is chosen to be larger than the block length of the one way function a subset of the hash result can be used as input to the function 411.

All data fields comprising the same block 200, 201 are logically adjacent but may or may not be physically adjacent, dependent on the implementation. The fields and functions shown in the containers 210 are explained in detail in the back-step generator section 300 and 400.

Everything required to build or provide the values 420 and 421 should be performed in a trusted area, where it is hard for unauthorized processes or people to capture or otherwise spy out any future value of the chain. Figure 3 and figure 4 comprise the trusted area.

Protection Against Attacks

The mechanism of the BSB that provides security against forked chains is explained now. A forked chain has a manipulated content at one or multiple steps of the chain and is from that step on different to the original chain. Whenever a third party presents a chain and claims that its chain is the correct one, such a manipulated chain cannot calculate the next step, because the next back-step value can only be calculated by the original chain, which owns the backstep generator. By this mechanism the need for the “proof of work” or other time or computing intensive measures is eliminated.

In addition, the protection provided by the seal value ensures that the owner/operator of the chain itself, which has access to the backstep generator, cannot manipulate the chain. Trusted authorities, which may receive the back-step-seal value of a specific block can verify the correctness of the chain backwards. Authorities will not have any information to be able to manipulate future blocks at all. Such an authority-based verification requires a secure channel to present the required back-step seal value to the authorities and only to the authorities.

Backstep Generator

The detailed description of the generation and the provision of the Back-Step values follows here with reference to Figure 3 and 4.

The BSB can only function in real life and contribute its benefits, when the time from a block request to the block result is reasonably fast. Required speed for today's Fintech applications is defined by the transactions per second during peak periods. Current transaction rates may reach up to 5000 transactions per second. This is beyond the speed of existing block chains. The BSB can provide such and even higher speeds. This requires that the Back-Step value and the Back-Step-Seal value are precalculated and already available at the time of a block request. A pre-calculation and cache system is required to sustain full speed.

The calculation of the back-step values is shown in Figure 3 and the caching system is shown in Figure 4.

Backstep Generator Initial Operation

The generation of back-step values and back-step-seal values comprises a repeated application of a function with one-way property as shown in figure 3. Both values use the same logical flow illustrated in figure 3 but must contain different initial values. Prior to the first use of the BSB the initial back-step-value 301 and the initial back-step-seal-value must be loaded into the corresponding non-volatile storage 303 and 304. Both values must be kept secret. The implementation of the BSB must ensure that both values do not have any algorithmic relation to each other, they must be independent random.

Processing the first block of a chain requires the back-step generator to cycle through all values backwards until the values for the first step are reached. This finite number of blocks/steps is a configuration parameter the controller 308 will need.

The number of steps 305 the chain shall provide may be loaded into a non-volatile storage 306. This number can be loaded via an interface from the outside of the system or it can be a predefined value as part of the actual implementation. The controller 308 receives the output of storage 306 as value 307. For the first one-way calculation, the control logic selects inputs C via signal 313 and 314 of the input selector 311 and 312 respectively. The selected signals 315 and 316 are the input to the one-way function 317 and 138. The controller using path 319 and 320 starts and stops the operation of the one-way logic. The output of the one-way logic, path 321 and 322 are routed to output selector 323 and 324.

Both selectors are switched to output A using the control path 325 and 326 from the control unit 308. The selectors output values 333 and 334 are connected to the input selectors port A. The controller selects input A for the following rounds of calculations. Now these operations cycles for a number of steps.

Checkpoint Values

The intermediate values generated by the one-way loops may be by far too many to store them all in a storage of the system. Instead, after a predefined number of cycles, called steps-size, a value can be stored in a non-volatile storage 401, shown on Figure 4, as so-called check-point value 342 and 343. For that purpose, the controller 308 enables at the output selectors 323 and 324 exit C329 and 332 for path 342 and 342 in addition to exit A. Because the time required to store the values in a non-volatile storage may be larger than the one-way cycle time the exits C port 329 and 332 may be implemented as registers. This avoids stopping the one-way cycling for some time. Actual implementations depend on the timing behaviour of the used components, architecture and may vary. Therefore, this implementation is showing the technical principle.

A system may be switched off, or loose for other reason power. This may happen at initialization time or afterwards at normal operation. After restart the calculation of one-way values can start from the nearest checkpoint values and not necessarily from the initial values. That will save calculation time. A larger number of checkpoint values provides faster response but requires more storage. This is a trade-off between storage and time and is a design parameter for the BSB. During power on, the system must calculate from the nearest checkpoint to the next step that will be requested. This calculation takes time during each power on. The checkpoint-storage requires a checkpoint address logic 337, which manages the addresses for storing checkpoint values during initialization and for fetching values during normal operation. The checkpoint address logic 337 receives instructions and starting address values from the controller 308 and generates the write, read, addresses and all other required control signals on path 339 for the checkpoint-storage. The checkpoint address logic further manages the functions of the cache-controller 408 using signals on path 340.

Details of Checkpoint Storage

The checkpoint storage 401 is logically organized as a matrix of 2 columns 402, 403 and m rows. The bit width of a column must be large enough to hold a back-step values 335. The same applies for the column holding the back-step-seal values 336. The actual

implementation may make these columns wider, if implementation specific control bits are required.

The checkpoint storage 401 needs the required number of rows to store all needed checkpoint values. The number m is defined by the finite number of blocks divided by the checkpoint step-size, which is defined as the number of steps between two adjacent checkpoints.

Checkpoint values to be stored are passed through the selectors 323 and 324, using exit C 329 and 332 on path 342 and 343 to the input of the checkpoint storage 401. The control signals to read, write and to define the actual address are originated at the checkpoint address logic using path 339.

When blocks are processed during normal blockchain operation the checkpoint values are essential to calculate the back-step values and back-step-seal values. To select the checkpoint values needed, the current step count number may be used to calculate the nearest checkpoint. The back-step value 404 and the back-step-seal value 405 on Figure 4 are then loaded via the selector 311 and 312 as input 315 and 316 into the one-way functions 317 and 318. The generator cycles now in the already describe way, generating back-step values and back-step-seal values.

Backstep Cache

In real applications, especially in the Fintech field where fast blockchain generation is needed, additional cache logic is required to provide the expected speed.

The checkpoint storage provides most time back-step data that are required for a more future block. Starting from that future values the data for the next blocks must then be calculated backwards. The results are used to constantly fill two alternating cache storages, which then deliver the actual values to the block calculation. The caches can be implemented in multiple levels. If the generation of back-step data is faster than the block generation a multi-level cache may save storage size requirements. Figure 4 shows a single cache implementation.

The generator shown at Figure 4 is used to fill two cache storages named Cache A and Cache B, 406 and 407. Each one of the cache storages has fields for the back-step value and the back-step-seal value. Output values using path 335 and 336 from the output selectors 323 and 324 of the generator 300 are connected to both caches. The cache used is selected by the cache controller 408. The caches are organized as First-In Last-Out storage. The corresponding addresses and the read/write control signals are generated by the cache

controller 408 using path 409. The cache controller generates also the input signals for the output multiplexers 410, 412 and 413.

At the end of the initialization the checkpoint storage is loaded with its values. During operation the two cache storages perform alternating operations, while cache A provides values to the block calculation logic, cache B is loaded with the next values. When the last value from cache A is consumed the caches switch their role, Cache B provides values while cache A is loaded. This supports seamless block generation in high speed operations. In the rare case of slow speed operation, the caches can be bypassed using the multiplexors 413 and 410. During high speed operation these multiplexers alternate between cache A and cache B.

The multiplexer 413 provides the back-step value 420 to the block generation and uses during cache operation the input path 416 and 417 alternatively. The multiplexer 410 is responsible for selecting the back-step-seal value and uses during cache operation the input path 414 and 415 alternatively. In the cases where no cache is required multiplexer 413 selects path 335 for the back-step value and multiplexer 410 selects path 336 for the back-step-seal value.

The back-step-seal value selected by multiplexer 410 on path 418 is the input to the seal generation. The seal generation uses the XOR function 411 to combine a seal value on path 418 with a hash value from the previous block on path 208. The XOR result 419 is the input to another one-way function 412. This one-way function produces the seal on path 421 for the current block being processed.

Authority Verification

BSB provides two methods for verification. First, one can easily follow the chain backwards by calculating the one-way function over the back-step value from the current record to the first record in the chain. Then the hash from the first record to the current record can be calculated. If both, the backward and the forward calculation match, then the chain is not manipulated by third parties. To ensure that the owner/operator of the BSB is not able to modify the chain the seal is implemented. The seal is a combination of forward and backward chaining, is inserted in the block, but can even not be verified by the owner / operator of the chain.

For seal verifying of the chain by third parties, like government authorities or courts, the release of the back-step seal value for the block in question is required. The BSB provides the value 418 of the required step in protected (encrypted) form to the requestor via the interface.

A set of functions are provided to deliver a requested seal value when an authorized request is received. These functions may or may not be enabled during initial configuration. This is shown in Figure 5. During initial configuration the non-volatile storage 502 is loaded with one or more authentication secrets 508, while the non-volatile storage 506 is loaded with one or more protection secrets 509. A request 510 for a seal value of a defined block is provided from an external interface to function 501. The authentication and verification process 501 is used to decide if the request must be granted or not. The authentication process may use permanently stored authentication secrets 508. Existing challenge response or other authentication protocols may be used. The actual protocol is implementation dependent and outside of the scope of this document.

If the external request is granted an internal request using path 341 is provided to the control logic 308. The control logic 308 initiates the calculation of the requested back-step-seal value. It may use a checkpoint value or start the calculation from the initial value. As soon as the requested value is calculated path 343 is used to provide the value to the seal protection process 504. This process may encrypt the seal value and may need a protection secret from storage 506. Such a secret is initially transported through path 509 to the protection unit. The details of the protection and the eventual used encryption are implementation dependent. After the seal value is protected, the seal value is provided at an interface using path 505. External verification processes, which know the two involved one-way functions 317 and 318 can now verify the whole chain back to the first block and detect any modification applied.

Claims

Claim 1

A method to enhance blockchains, especially in Fintech applications comprising:
one or more values (420, 213) chained from the future towards the past;
a cryptographic one-way function (209) applied between adjacent blocks, such that the value of an earlier block can be easily calculated from the value of the next later block while calculating the value of the next later block from an earlier block is cryptographically hard; and
a storage (303, 304) to hold one or more secrets.

Claim 2

A method according to claim 1, characterized in that, the blockchain does not need any “proof of work” function.

Claim 3

A method according to claim 1, characterized in that, the blockchain does not need to be distributed over multiple systems to provide trusted blocks.

Claim 4

A method according to claim 1, comprising: a back-step-seal value (213); a xor function, such that the seal value is xored (421) with the previous block hash (208); a one-way-function (216), whereby the xor result is protected against reverse calculation; and a seal value to protect the blockchain against manipulation.

Claim 5

A method according to claim 1, characterized in that, a defined fixed number of blocks can be calculated, such that the chain has a predefined end.

Claim 6

A method according to claim 1, comprising: one or more one-way functions (317, 318); a set of multiplexers (311,312,323,324); and a control logic (308), such that back chained values can be pre-calculated.

Claim 7

A method according to claim 1, comprising: a non-volatile storage holding intermediate checkpoint values, such that, future calculation of back-chained values can start at values that are close to the required step; control logic to store and fetch these values;

Claim 8

A method according to claim 7, comprising: one or more volatile cache storage layers; a cache control logic; a set of multiplexers, characterized in that, intermediate chain values can be precalculated and stored for fast use.

Claim 9

A method according to claim 8, wherein a second set of cache storage is provided, such that one cache is filled while the other cache is emptied.

Claim 10

A method according to claim 4, comprising:
an authentication function;
one or multiple authentication secrets;
a data protection function;
data protection secrets, such that,
an authenticated person or process can request and receive back-step-seal values related to a requested block.

Claim 11

A method according to claim 1, characterized in that, the number of operations to generate a block is fixed resulting in a fixed response time for real time applications.

Claim 12

A method according to claim 1, characterized in that, all functions can be implemented into a single integrated circuit.

Claim 13

A method according to claim 12, characterized in that, implementation into one or more high speed chips allows integration into systems generating high data rates of block input.

Claim 14

A method according to claim 12, characterized in that, implementation into small smart-card like chip allows blockchain applications in personal devices and sensors.

Claim 15

A method according to claim 1, characterized in that, integration into other systems or chips as Intellectual Property is possible.