

Claims

1. A method for verifying that a transaction is included in a blockchain, the method comprising the steps of:
 - identifying a transaction T to be verified;
 - obtaining a certificate C associated with the transaction T, wherein the certificate includes a block identifier for a given block and an inclusion proof linking the transaction to the given block in the blockchain;
 - determining a longest chain of valid blocks in the blockchain; and
 - verifying that the given block is included in the longest chain.
2. The method as claimed in claim 1 wherein the certificate C is obtained from a local storage associated with a client.
3. The method as claimed in claim 1 wherein the certificate C is obtained from a storage associated with a verifier entity.
4. The method as claimed in claim 1 wherein the certificate C is obtained from a storage associated with a platform.
5. The method as claimed in any preceding claim including the step of sourcing the longest blockchain using a headers client that is configured to store block headers associated with the transaction T.
6. The method as claimed in any preceding claims further including the steps of :
 - calculating a Merkle root R' from the proof of inclusion connecting the transaction T to the Merkle root R associated with the given block;
 - responsive to $R = R'$, the method includes the steps of :
 - determining that R' is included in the given block; and
 - determining that the given block is included in the longest chain.
7. The method as claimed in claim 6 wherein the method further comprises,
 - based on a determination that R does not match R', generating an error message;
 - and /or

based on a determination R' is not included in the given block, generating an error message; and/or

based on a determination that the given block is not included in the longest chain, generating an error message.

8. The method as claimed in any one of claims 1 to 7 further comprising the steps of :
obtaining data D associated with a client;
based on data D , determining a value of data committed to the blockchain d ; and
extracting or identifying the transaction T associated with the committed value d .
9. The method as claimed in claim 8 wherein the committed value d is based on a salt value S .
10. The method as claimed in claim 9 wherein the committed value d is a hash of the client data D and salt S .
11. The method as claimed in any one of claims 8 to 10 comprising:
verifying creation of an event stream $ES_{n=0 \text{ to } N}$ by verifying the inclusion of a first transaction T_0 associated with ES_0 using the method of any one of claims 1 to 7, where n is an integer from 0 to N , n representing the length of the event stream, where 0 is a first or create event and N is a final or terminate event;
determining that the first input to first transaction T_0 is not dust;
determining that the first output of T_0 is dust;
for each n^{th} data entry D_n for an event associated with a client for the event stream $ES_{n=0 \text{ to } N}$, performing the method as claimed in any one of claims 8 to 10; and
verifying that an input corresponding to an n^{th} transaction T_n in the event stream ES_n spends an output associated with a previous transaction T_{n-1} , when $n > 0$.
12. The method as claimed in claim 11 wherein,
if the first input to first transaction T_0 is dust, then generate an error message; and
/or
if the first output of T_0 is not dust , then generate an error message.
13. The method as claimed in claims 11 or 12 further comprising:
verifying closure of an event stream ES_N by verifying the inclusion of a final transaction T_N associated with ES_N using the method of any one of claims 1 to 7;

determining that the first input to first transaction T_N is dust;
determining that the first output of T_0 is not dust; and
verify that an input corresponding to the final N^{th} transaction T_N in the event stream ES_N spends an output associated with a previous transaction T_{N-1} .

14. The method as claimed in claim 14 wherein,
if the first input to first transaction T_N is not dust, then generate an error message;
and /or
if the first output of T_N is dust , then generate an error message.
15. The method as claimed in any one of the preceding claims implemented by one or more processors associated with the client.
16. The method as claimed in any one of claims 1 to 15 implemented by one or more processors associated with a platform.
17. The method as claimed in any one of claims 1 to 15 implemented by one or more processors associated with verifier entity.
18. The method as claimed in claim 17 wherein the verifier entity is independent from the client and/or the platform.
19. A computer implemented method for implementing a channel service for one or more clients, the method implemented by a channel processor and comprising the steps of :
receiving a request from a given client among the one or more clients, the request pertaining to creation of a channel;
providing the given client access to one or more functions that enable direct communication between the given client and another entity via the channel, wherein said one or more functions include:
channel functions or procedures pertaining to the channel for transmission of data;
and/or
message functions or procedures pertaining to the data being transmitted using the channel;
issuing one or more access tokens for the channel, said one or more access tokens being configured for secure communication with another entity via the channel; and

storing and/or providing one or more notifications associated with the channel for the given client.

20. A computer implemented method for processing transactions associated with a blockchain, the method implemented by one or more processors associated with a client and comprising the steps of:

obtaining from a channel service access to one or more functions that enable direct communication between the given client and another entity, said one or more functions including:

channel functions or procedures pertaining to a channel for transmission of data; and/or

message functions or procedures pertaining to the data being transmitted using a channel;

obtaining from the channel service one or more access tokens, said access tokens enabling secure communication with the other entity;

responsive to obtaining or identifying a transaction identifier (TxID) for a given transaction associated with the client;

using one or more channel functions received from the channel processor, creating a given channel for communication with another entity;

sending the one or more access tokens associated with the given channel to the other entity;

receiving a notification associated with the given channel, the notification pertaining to data in the given for verification that the given transaction is included in the blockchain.