Bioblockchain

Release 2022.1.0

Marek Ziska

Jul 28, 2022

CONTENTS:

1 bioblockchain **3**
- 1.1 Submodules . . . 3
- 1.2 bioblockchain.bioblockchain module . . . 3
- 1.3 bioblockchain.block module . . . 4
- 1.4 bioblockchain.blockchain module . . . 6
- 1.5 bioblockchain.config module . . . 8
- 1.6 bioblockchain.main module . . . 8
- 1.7 bioblockchain.message module . . . 8
- 1.8 bioblockchain.message_log module . . . 9
- 1.9 bioblockchain.node module . . . 9
- 1.10 bioblockchain.parser module . . . 15
- 1.11 bioblockchain.pbft module . . . 15
- 1.12 bioblockchain.chain_utils module . . . 17
- 1.13 bioblockchain.transaction module . . . 18
- 1.14 bioblockchain.time_utils module . . . 19
- 1.15 bioblockchain.wallet module . . . 19
- 1.16 Module contents . . . 20

2 Indices and tables **21**

Python Module Index **23**

Index **25**

This is a documentation webpage for the Bioblockchain demonstrator that showcases the integration of blockchain in biometric systems with the goal of making the system more secure. The demonstrator is a terminal application implemented in Python language, visualizing the communication in the distributed network of nodes. These nodes are participating in a byzantine consensus that validates executed processes in a single node among more nodes thus resulting in the decentralization of the classical biometric system. Each operation outcome is stored in a blockchain data structure that extends the system's security. Blockchain characteristic immutability, transparency, and verifiability allow the nodes to verify previous transactions/operations, thus the proposed system can mitigate potential security weak points(such as overriding of the components, replay of old/artifical data, etc..).

CHAPTER

ONE

BIOBLOCKCHAIN

1.1 Submodules

1.2 bioblockchain.bioblockchain module

class bioblockchain.bioblockchain.**BioBlockchain**(*verbosity*)

Bases: object

BioBlockchain is a main class containing scenarios showcasing integration of blockchain and biometric system processes

get_random_node()

get_random_node random node to simulate undeterministic choice of biometric terminal/node

Returns
node selected from the network

Return type
Node

async run_authentication(*process*, *claimed_identity=None*, *unknown_biometrics=True*, *unknown_user=False*)

run_authentication showcases identification or verification scenario in the biometric system

Parameters

- **process** (*str*) – type of authentication/recognition process
- **claimed_identity** (*str, optional*) – in case of verification, claimed identity is needed. Defaults to None.
- **unknown_biometrics** (*bool, optional*) – new biometrics. Defaults to True.
- **unknown_user** (*bool, optional*) – _description_. Defaults to False.
- **compromised_matcher** (*bool, optional*) – _description_. Defaults to False.

async run_authentication_no_feature_extraction(*process*)

run_authentication_no_feature_extraction showcases malicious identification or verification scenario in the biometric system with replayed data

Parameters

- **process** (*str*) – type of authentication/recognition process
- **claimed_identity** (*str, optional*) – in case of verification, claimed identity is needed. Defaults to None.

- **unknown_biometrics** (*bool, optional*) – new biometrics. Defaults to True.
- **unknown_user** (*bool, optional*) – _description_. Defaults to False.
- **compromised_matcher** (*bool, optional*) – _description_. Defaults to False.

async run_enrollment()

run_enrollment showcases enrollment scenario of a new user to the biometric system

Parameters

compromised_feature_extractor (*bool, optional*) – True when the proposing extractor is compromised. Defaults to False.

1.3 bioblockchain.block module

class bioblockchain.block.**Block**(*timestamp*, *previous_hash*, *current_hash*, *data*, *proposer*, *signature*, *seq_number*)

Bases: object

class representing one block in the blockchain

static block_content_to_json(*timestamp*, *previous_hash*, *data*)

block_content_to_json constructs json/dict from block's content

Parameters

- **timestamp** (*String*) – date
- **previous_hash** (*Hash object*) – hash pointing at the last block
- **data** (*[Transaction objects list]*) – transactions

Returns

dict/json containing info about block

Return type

dict

static block_hash(*block*)

block_hash calculate hash of Block object

Parameters

block (Block) – block object to calculate hash on

Returns

hash of the object

Return type

Hash_ object

static create_block(*previous_block*, *data*, *proposer_wallet*)

create_block is a static method that instantiates Block class

Parameters

- **previous_block** (*Block object*) – previous block needed for needed attachment in the new block
- **data** (*[Transaction object list]*) – list of new transactions

- **proposer_wallet** (*Wallet object*) – voted proposer to append new block to blockchain

Returns
new Block instance

Return type
Block object

static genesis()

genesis creates genesis block of bioblockchain

Returns
genesis block

Return type
Block object

property hash

hash is a property representing hash value of given block

Returns
hash object from hashlib library

Return type
Hash

property hash_hexdigest

hash_hexdigest is a property representing string hash value of given block

Returns
string hash value

Return type
str

property previous_hash

previous_hash is a property representing hash value of previous block

Returns
hash object from hashlib library

Return type
_Hash

property previous_hash_hexdigest

previous_hash_hexdigest is a property representing string hash value of previous block

Returns
string hash value of previous block

Return type
str

search_by_process(*process_id*)

search_by_process looks for transaction containing given process_id

Parameters
process_id (*Str*) – string representation of the searched process_id

Returns
found Transaction or None

Return type
Transaction

seq_number

Payload

static sign_block_hash(*hash*, *wallet*)

sign_block_hash signs passed hash value with given wallet

Parameters

- **hash** (Hash_ object) – passed hash value
- **wallet** (*Wallet object*) – wallet object signing the hash value

Returns
signature

Return type
signature

verify_block()

verifies block

Returns
True if valid block, else False

Return type
bool

verify_proposer(*wallet:* Wallet)

verify_proposer verifies if the block has been proposed by a given wallet

Parameters
wallet (Wallet) – wallet being verified

Returns
True if valid proposer, else False

Return type
bool

1.4 bioblockchain.blockchain module

class bioblockchain.blockchain.**Blockchain**

Bases: object

blockchain class containing list(chain) of blocks

add_block(*block:* Block)

add_block appends block to the blockchain

Parameters
block (*Block object*) – block to be appended

Returns
appended block

Return type
Block object

create_block(*transactions*, *wallet*)

create_block wrapper function for Block instantiation

Parameters

- **transactions** (*[Transaction object]*) – list of transactions that will be stored in newly created block
- **wallet** (*Wallet object*) – proposer of the block

Returns
instance of Block object

Return type
Block object

display_chain()

prints out the blockchain content

is_valid_block(*block:* Block)

check for validity of block

Parameters
block (*Block object*) – Block object to be validated

Returns
True if valid, else False

Return type
bool

property last_block

last_block property containing last block in the blockchain

Returns
latest block object appended to the blockchain

Return type
Block object

search_by_process(*process_id*)

search_by_process looks for block containing transaction with given process_id

Parameters
process_id (*Str*) – string representation of the searched process_id

Returns
found Block or None

Return type
Block

1.5 bioblockchain.config module

bioblockchain.config.**MAX_FAULT = 1**

maximum number of faulty nodes

bioblockchain.config.**MIN_WEIGHT_COMMIT = 0.6666666666666666**

minimum weight of commits needed to enter reply phase

bioblockchain.config.**MIN_WEIGHT_PREPARE = 0.6666666666666666**

minimum weight of prepares needed to enter commit phase

bioblockchain.config.**MIN_WEIGHT_REPLY = 0.6666666666666666**

minimum weight of replies needed to respond to the request

bioblockchain.config.**NUM_PARTICIPATING_NODES = 5**

total count of replicas participating in the PBFT network

1.6 bioblockchain.main module

bioblockchain.main.**query_yes_no**(*question*, *default='yes'*)

query_yes_no Ask a yes/no question via and returns the answer.

Parameters

- **question** (*Str*) – question asked to the user
- **default** (*Str, optional*) – Default answear when the entered answear is blank. Defaults to "yes".

Returns

Depending on the answer returns True for "yes" or False for "no".

Return type

Bool

bioblockchain.main.**run**()

run is a main script running the bioblockchain demonstrator

1.7 bioblockchain.message module

class bioblockchain.message.**Message**(*ttype=None*, *sender=None*, *content=None*)

Bases: object

Object passed between nodes in peer-to-peer communication

toJSON()

toJSON handles custom object to json conversion

Returns

json/dict object

Return type

dict

verify()

verifies message

Returns
True if valid, else False

Return type
Bool

class bioblockchain.message.**PBFT_Message**(*value*)

Bases: str, Enum

PBFT message types

Parameters

- **str** (*str*) – string representation
- **Enum** (*Enum*) – Enum inheritance

COMMIT = 'COMMIT'

PREPARE = 'PREPARE'

PRE_PREPARE = 'PRE_PREPARE'

REPLY = 'REPLY'

REQUEST = 'REQUEST'

ROUND_CHANGE = 'ROUND_CHANGE'

1.8 bioblockchain.message_log module

class bioblockchain.message_log.**MessageLogged**(*message*, *weight*)

Bases: object

Single log stored in a node for given message

update_nums(*view_num*, *seq_num*)

update_nums updates view and sequence number of PBFT protocol

Parameters

- **view_num** (*Int*) – current view number
- **seq_num** (*Int*) – current sequence number

1.9 bioblockchain.node module

class bioblockchain.node.**Biometric_Processes**(*value*)

Bases: str, Enum

processes in biometric systems

Parameters

- **str** (*str*) – string representation

- **()** (*Enum*) – Enum inheritance

ENROLLMENT = 'enrollment'

IDENTIFICATION = 'identification'

VERIFICATION = 'verification'

class bioblockchain.node.**Node**(*node_num*, *storage*, *users*, *blockchain*, *weight*)

Bases: object

single node participating in the peer-2-peer network

add_block_to_blockchain(*block*)

add_block_to_blockchain given node adds block to blockchain

Parameters
block (Block) – block to be added to blockchain

add_request_to_log(*message*)

store a newly received request

Parameters
message (*Message Object*) – request message object

Returns
hash representation of found message

Return type
Str

add_transaction(*transaction*)

add transaction to the queue waiting for block proposal

Parameters
transaction (Transaction) – processed transaction ready to be stored in blockchain

claimed_identity_in_database(*features*, *claimed_identity*)

claimed_identity_in_database method validates if given feature is stored in the template database in form of claimed identity(1:1 search)

Parameters

- **features** (*Str*) – string representation of acquired features
- **claimed_identity** (*str*) – string representation of the claimed identity

Returns
string representation of found user's key/identifier or None

Return type
Str

compare_features(*features1*, *features2*)

naive comparison of features

Parameters

- **features1** (*str*) – first features
- **features2** (*str*) – second features

Returns
True if they are similar

Return type
Bool

`corresponding_view_seq`(*view*, *seq*)

corresponding_view_seq check for valid sequence and view number

Parameters

- **view** (*int*) – current view number of pbft
- **seq** (*int*) – current sequence number of pbft

Returns
true

Return type
Bool

`create_block`(*transaction*)

create_block creation of a new block with given transaction proposed by this node

Parameters
transaction (*Transaction object*) – transaction to be added to the block

Returns
created block object

Return type
Block

`execute_operation`(*log*)

execute_operation after receiving enough replies the primary node performs this operation in this method according to the vote result

Parameters
log (*MessageLogged object*) – message log for given request for which the primary received enough replies

`extract`(*data*, *compromised=False*)

naive way of extracting features from biometric data

Parameters
data (*Str*) – biometric data

Returns
string representation of extracted features

Return type
Str

`feature_extractor`(*data_sensory*, *process_type*, *process_id*)

simulation of feature extraction component

Parameters

- **data_sensory** (*Str*) – biometric data acquired from sensor
- **process_type** (Biometric_Processes) – type of operation executed by the extractor(enrollment/authentication)

Raises
Exception – repetitive enrollment()

Returns
json/dict data containing acquired information with timestamps

Return type
Dict

`features_in_database`(*features*)

features_in_database method searches for a given feature in the whole database of templates(1:M search)

Parameters
features (*str*) – string representation of feature

Returns
found key/identifier for given features/user or None

Return type
str

`get_extraction_compromised`()

get_extraction_compromised after calling this function the nodes feature extraction module gets compromised

`get_matching_compromised`()

get_extraction_compromised after calling this function the nodes feature extraction module gets compromised

`get_other_nodes`(*nodes*)

get other nodes from the network

Parameters
nodes (*List(Node)*) – list of nodes in the network

Returns
list of nodes containing all other nodes

Return type
List(*Node*)

`get_sensor_data`(*string*)

naive way of acquiring sensor biometric data

Returns
representation of acquired biometric data

Return type
Str

`matcher`(*features*, *mode*, *claimed_identity*, *process_id*)

matcher operates as the matcher component that is proposing the operation to the network

Parameters

- **features** (*Str*) – representation of extracted features
- **mode** (*Str*) – mode of operation(either identification or verification)
- **claimed_identity** (*Str*) – identifier of the claimed identity
- **process_id** (*Str*) – process identifier for given operation
- **compromised** (*Bool*) – flag used for compromised node simulation (proposing wrong match result)

Returns
dictionary containing all the operational data and outcome of matching

Return type
Dict

`async received_commit(msg_hash)`

update the count of received commits

Parameters
msg_hash (*Str*) – hash message representation

Returns
log of message that received prepare message

Return type
MessageLog

`async received_prepare(msg_hash)`

update the count of received prepares

Parameters
msg_hash (*Str*) – hash message representation

Returns
log of message that received prepare message

Return type
MessageLog

`async received_reply(msg_hash, result)`

update the count of received replies

Parameters
msg_hash (*Str*) – hash message representation

Returns
log of message that received prepare message

Return type
MessageLog

`search_log_msg(message)`

search for a given message in node's backlog

Parameters
message (Message) – searched message

Returns
Message if succesfull search, else None

Return type
Message/None

`search_log_msghash(message_hash)`

search message in log of request messages

Parameters
message_hash (*bytes*) – hash representation of searched message

Returns
message on succesful search of message, else None

Return type
Message object

`search_user_in_database`(*user*)

searches for identifier of user in database

Parameters
user (*Str*) – representation of users identification in database

Returns
return found user or None

Return type
Str

`set_seq_number`(*message_hash*)

set_seq_number assign next available sequence number to the request

Parameters
message_hash (*str*) – hash of the request message

Returns
sequence and view number that was assigned to the request

Return type
Tuple

`store_features`(*features*, *new_user*)

store features in template storage

Parameters
features (*Str*) – string features to be stored

`verify_block`(*block*)

verify_block node verifies validity of proposen block

Parameters
block (Block) – proposed block

Returns
is the block valid or not

Return type
Bool

`verify_decision`(*transaction*, *biometrics*)

verifies the transaction

Parameters
transaction (Transaction) – verifies the operation outcome of given transaction

Returns
If the result of the operation is validated returns None, otherwise returns the different found result

Return type
Str or None

1.10 bioblockchain.parser module

class bioblockchain.parser.**MyParser**

Bases: object

argument parser

1.11 bioblockchain.pbft module

class bioblockchain.pbft.**PBFT**(*nodes*, *verbosity*)

Bases: object

PBFT protocol communication

async async_range(*count*)

async broadcast_commit(*content=None*, *from_node=None*)

brodcasts commit to all nodes in the network

Parameters

- **content** (*Dict, optional*) – json data neccesary for commit message(msg_hash/seq_no/view). Defaults to None.
- **from_node** (Node, *optional*) – node object sending the commit message. Defaults to None.

async broadcast_pre_prepare(*content=None*, *from_node=None*)

brodcasts pre_prepare to all nodes in the network

Parameters

- **content** (*Dict, optional*) – json data neccesary for pre_prepare message(msg_hash/seq_no/view). Defaults to None.
- **from_node** (Node, *optional*) – node object sending the pre_prepare message. Defaults to None.

async broadcast_prepare(*content=None*, *from_node=None*)

brodcasts prepare to all nodes in the network

Parameters

- **content** (*Dict, optional*) – json data neccesary for prepare message(msg_hash/seq_no/view). Defaults to None.
- **from_node** (Node, *optional*) – node object sending the prepare message. Defaults to None.

async broadcast_request(*content*, *from_node*)

brodcasts new request to all nodes in the network

Parameters

- **content** (Transaction, Block) – data sent to network in request
- **user** (Node) – Node who sent the message

broadcast_round_change()

`get_leader()`

returns current leader

Returns
Node object with current leader

Return type
Node

`is_leader(node)`

is_leader checks if given node is current leader

Parameters
node (*Node object*) – tested node

Returns
True when is primary, otherwise False

Return type
Bool

`is_verified_node(node)`

is_verified_node verifies if given Node is verified

Parameters
node (*Node object*) – tested node

Returns
True when is verified, otherwise False

Return type
Bool

`async message_handler(message, recipient)`

message execution manager function

Parameters

- **message** (*Message object*) – message object being handled by the PBFT protocol
- **recipient** (*Node object*) – adressee of the message being handled

`async validate_block(block, user)`

interface of the pbft protocol where the Block is proposed to be validated in PBFT

Parameters

- **block** (Block) – proposed Block
- **user** (Node) – entry Node proposing the block

`async validate_decision(transaction_data, biometric_data, user)`

interface of the pbft protocol where the Transaction is made for given operation and broadcasts request to validate decision/operation to PBFT

Parameters

- **data** (*Dict*) – json/dict data of executed operation to get validated
- **user** (Node) – entry Node executing the operation

Returns
result of the validation

Return type
Bool

1.12 bioblockchain.chain_utils module

class bioblockchain.chain_utils.**ChainUtils**

Bases: `object`

ChainUtils contains static methods for cryptographic operations and operations with blockchain

Raises

- **ValueError** – Raised when passed arguments to verify_signature are of wrong type
- **BadSignatureError** – Raised in verify_signature when the signature is wrong

static generate_key_from_seed(*seed*)

generate_key_from_seed generates private key from secret phrase

Parameters
seed (*string*) – secret phrase

Returns
private key generated from *seed*

Return type
SigningKey

static hash(*data_to_hash: str*)

hash calculates hash from string with sha256 hashing function

Parameters
string_to_hash (*str | dict*) – string/dict to be hashed

Returns
object representing calculated hash from given *string_to_hash*

Return type
_Hash object

static id()

generates UUID(universally unique identifier)

Returns
generated UUID

Return type
String

static string_from_verifkey(*verifying_key*)

string_from_verifkey turns bytes like public key, to its string representation

Parameters
verifying_key (*[Bytes]*) – [public key]

Returns
public key's string representation

Return type
String

static transactionlist_to_json(*tx_list*)

transactionlist_to_json converts Transaction objects to list of dictionaries(json)

Parameters
tx_list (*List or Transaction*) – list of transactions or single Transaction

Returns
list of dictionaries

Return type
List

static verify_signature(*public_key*, *signature*, *data*)

verify_signature verifies validity of signature

Parameters

- **public_key** (*VerifyingKey*) – public key associated with *signature*
- **signature** (*Bytes*) – encoded signature over *data*
- **data** (*String*) – data which was used for signature calculation

Returns
True when the signature is valid, False otherwise

Return type
Bool

1.13 bioblockchain.transaction module

class bioblockchain.transaction.**Transaction**(*data*, *wallet*)

Bases: object

Transaction class is holding certain data with accompanying info needed for it's verification

get_data()

get_data returns data contained in Transaction

Returns
dictionary with data

Return type
Dict

property hash_hexdigest

hash_hexdigest is a property value of Transaction's hash in it's string representation

Returns
Hash representation of string

Return type
Str

toJSON()

transforms class variables into json/dict

Returns
dictionary from this transaction

Return type
dict

`update_transaction`(*data*, *node*)

`verify_transaction`()

verify_transaction verifies wether the transaction is valid

Parameters
transaction (*[type]*) – [description]

Returns
[description]

Return type
[type]

1.14 bioblockchain.time_utils module

`class` bioblockchain.time_utils.**TimeUtils**

Bases: `object`

TimeUtils contains static methods for time operations

`static my_date`()

my_date returns current date in given format

Returns
description

Return type
type

1.15 bioblockchain.wallet module

`class` bioblockchain.wallet.**Wallet**(*secret*)

Bases: `object`

The wallet holds the public key and key pair. It is also responsible for signing data hashes and creating signed transactions, messages…

`create_transaction`(*data*)

creates transaction

Parameters
data (*Dict*) – transaction payload

Returns
[object representing transaction]

Return type
(Transaction object)

property private_key

private_key property contains private or SigningKey property

Returns
private key of given wallet

Return type
SigningKey

sign(*data*)

calculates signature over given data

Parameters
data (*Dict|String|Bytes*) – [data to be signed]

Raises
ValueError – Passed unexpected argument type

Returns
encoded signature over *data*

Return type
(String)

property verif_key

verif_key property contains public or VerifyingKey property

Returns
public key of given wallet

Return type
VerifyingKey

1.16 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

bioblockchain, 20
bioblockchain.bioblockchain, 3
bioblockchain.block, 4
bioblockchain.blockchain, 6
bioblockchain.chain_utils, 17
bioblockchain.config, 8
bioblockchain.main, 8
bioblockchain.message, 8
bioblockchain.message_log, 9
bioblockchain.node, 9
bioblockchain.parser, 15
bioblockchain.pbft, 15
bioblockchain.time_utils, 19
bioblockchain.transaction, 18
bioblockchain.wallet, 19

INDEX

A

add_block() (*bioblockchain.blockchain.Blockchain method*), 6
add_block_to_blockchain() (*bioblockchain.node.Node method*), 10
add_request_to_log() (*bioblockchain.node.Node method*), 10
add_transaction() (*bioblockchain.node.Node method*), 10
async_range() (*bioblockchain.pbft.PBFT method*), 15

B

bioblockchain
    module, 20
BioBlockchain (*class in bioblockchain.bioblockchain*), 3
bioblockchain.bioblockchain
    module, 3
bioblockchain.block
    module, 4
bioblockchain.blockchain
    module, 6
bioblockchain.chain_utils
    module, 17
bioblockchain.config
    module, 8
bioblockchain.main
    module, 8
bioblockchain.message
    module, 8
bioblockchain.message_log
    module, 9
bioblockchain.node
    module, 9
bioblockchain.parser
    module, 15
bioblockchain.pbft
    module, 15
bioblockchain.time_utils
    module, 19
bioblockchain.transaction
    module, 18
bioblockchain.wallet
    module, 19
Biometric_Processes (*class in bioblockchain.node*), 9
Block (*class in bioblockchain.block*), 4
block_content_to_json() (*bioblockchain.block.Block static method*), 4
block_hash() (*bioblockchain.block.Block static method*), 4
Blockchain (*class in bioblockchain.blockchain*), 6
broadcast_commit() (*bioblockchain.pbft.PBFT method*), 15
broadcast_pre_prepare() (*bioblockchain.pbft.PBFT method*), 15
broadcast_prepare() (*bioblockchain.pbft.PBFT method*), 15
broadcast_request() (*bioblockchain.pbft.PBFT method*), 15
broadcast_round_change() (*bioblockchain.pbft.PBFT method*), 15

C

ChainUtils (*class in bioblockchain.chain_utils*), 17
claimed_identity_in_database() (*bioblockchain.node.Node method*), 10
COMMIT (*bioblockchain.message.PBFT_Message attribute*), 9
compare_features() (*bioblockchain.node.Node method*), 10
corresponding_view_seq() (*bioblockchain.node.Node method*), 11
create_block() (*bioblockchain.block.Block static method*), 4
create_block() (*bioblockchain.blockchain.Blockchain method*), 6
create_block() (*bioblockchain.node.Node method*), 11
create_transaction() (*bioblockchain.wallet.Wallet method*), 19

D

display_chain() (*bioblockchain.blockchain.Blockchain method*), 7

E

ENROLLMENT (*bioblockchain.node.Biometric_Processes attribute*), 10
execute_operation() (*bioblockchain.node.Node method*), 11
extract() (*bioblockchain.node.Node method*), 11

F

feature_extractor() (*bioblockchain.node.Node method*), 11
features_in_database() (*bioblockchain.node.Node method*), 12

G

generate_key_from_seed() (*bioblockchain.chain_utils.ChainUtils static method*), 17
genesis() (*bioblockchain.block.Block static method*), 5
get_data() (*bioblockchain.transaction.Transaction method*), 18
get_extraction_compromised() (*bioblockchain.node.Node method*), 12
get_leader() (*bioblockchain.pbft.PBFT method*), 15
get_matching_compromised() (*bioblockchain.node.Node method*), 12
get_other_nodes() (*bioblockchain.node.Node method*), 12
get_random_node() (*bioblockchain.bioblockchain.BioBlockchain method*), 3
get_sensor_data() (*bioblockchain.node.Node method*), 12

H

hash (*bioblockchain.block.Block property*), 5
hash() (*bioblockchain.chain_utils.ChainUtils static method*), 17
hash_hexdigest (*bioblockchain.block.Block property*), 5
hash_hexdigest (*bioblockchain.transaction.Transaction property*), 18

I

id() (*bioblockchain.chain_utils.ChainUtils static method*), 17
IDENTIFICATION (*bioblockchain.node.Biometric_Processes attribute*), 10
is_leader() (*bioblockchain.pbft.PBFT method*), 16
is_valid_block() (*bioblockchain.blockchain.Blockchain method*), 7
is_verified_node() (*bioblockchain.pbft.PBFT method*), 16

L

last_block (*bioblockchain.blockchain.Blockchain property*), 7

M

matcher() (*bioblockchain.node.Node method*), 12
MAX_FAULT (*in module bioblockchain.config*), 8
Message (*class in bioblockchain.message*), 8
message_handler() (*bioblockchain.pbft.PBFT method*), 16
MessageLogged (*class in bioblockchain.message_log*), 9
MIN_WEIGHT_COMMIT (*in module bioblockchain.config*), 8
MIN_WEIGHT_PREPARE (*in module bioblockchain.config*), 8
MIN_WEIGHT_REPLY (*in module bioblockchain.config*), 8
module
    bioblockchain, 20
    bioblockchain.bioblockchain, 3
    bioblockchain.block, 4
    bioblockchain.blockchain, 6
    bioblockchain.chain_utils, 17
    bioblockchain.config, 8
    bioblockchain.main, 8
    bioblockchain.message, 8
    bioblockchain.message_log, 9
    bioblockchain.node, 9
    bioblockchain.parser, 15
    bioblockchain.pbft, 15
    bioblockchain.time_utils, 19
    bioblockchain.transaction, 18
    bioblockchain.wallet, 19
my_date() (*bioblockchain.time_utils.TimeUtils static method*), 19
MyParser (*class in bioblockchain.parser*), 15

N

Node (*class in bioblockchain.node*), 10
NUM_PARTICIPATING_NODES (*in module bioblockchain.config*), 8

P

PBFT (*class in bioblockchain.pbft*), 15
PBFT_Message (*class in bioblockchain.message*), 9
PRE_PREPARE (*bioblockchain.message.PBFT_Message attribute*), 9
PREPARE (*bioblockchain.message.PBFT_Message attribute*), 9
previous_hash (*bioblockchain.block.Block property*), 5
previous_hash_hexdigest (*bioblockchain.block.Block property*), 5
private_key (*bioblockchain.wallet.Wallet property*), 19

Q

query_yes_no() (*in module bioblockchain.main*), 8

R

received_commit() (*bioblockchain.node.Node method*), 13
received_prepare() (*bioblockchain.node.Node method*), 13
received_reply() (*bioblockchain.node.Node method*), 13
REPLY (*bioblockchain.message.PBFT_Message attribute*), 9
REQUEST (*bioblockchain.message.PBFT_Message attribute*), 9
ROUND_CHANGE (*bioblockchain.message.PBFT_Message attribute*), 9
run() (*in module bioblockchain.main*), 8
run_authentication() (*bioblockchain.bioblockchain.BioBlockchain method*), 3
run_authentication_no_feature_extraction() (*bioblockchain.bioblockchain.BioBlockchain method*), 3
run_enrollment() (*bioblockchain.bioblockchain.BioBlockchain method*), 4

S

search_by_process() (*bioblockchain.block.Block method*), 5
search_by_process() (*bioblockchain.blockchain.Blockchain method*), 7
search_log_msg() (*bioblockchain.node.Node method*), 13
search_log_msghash() (*bioblockchain.node.Node method*), 13
search_user_in_database() (*bioblockchain.node.Node method*), 14
seq_number (*bioblockchain.block.Block attribute*), 6
set_seq_number() (*bioblockchain.node.Node method*), 14
sign() (*bioblockchain.wallet.Wallet method*), 20
sign_block_hash() (*bioblockchain.block.Block static method*), 6
store_features() (*bioblockchain.node.Node method*), 14
string_from_verifkey() (*bioblockchain.chain_utils.ChainUtils static method*), 17

T

TimeUtils (*class in bioblockchain.time_utils*), 19
toJSON() (*bioblockchain.message.Message method*), 8
toJSON() (*bioblockchain.transaction.Transaction method*), 18
Transaction (*class in bioblockchain.transaction*), 18
transactionlist_to_json() (*bioblockchain.chain_utils.ChainUtils static method*), 17

U

update_nums() (*bioblockchain.message_log.MessageLogged method*), 9
update_transaction() (*bioblockchain.transaction.Transaction method*), 19

V

validate_block() (*bioblockchain.pbft.PBFT method*), 16
validate_decision() (*bioblockchain.pbft.PBFT method*), 16
verif_key (*bioblockchain.wallet.Wallet property*), 20
VERIFICATION (*bioblockchain.node.Biometric_Processes attribute*), 10
verify() (*bioblockchain.message.Message method*), 8
verify_block() (*bioblockchain.block.Block method*), 6
verify_block() (*bioblockchain.node.Node method*), 14
verify_decision() (*bioblockchain.node.Node method*), 14
verify_proposer() (*bioblockchain.block.Block method*), 6
verify_signature() (*bioblockchain.chain_utils.ChainUtils static method*), 18
verify_transaction() (*bioblockchain.transaction.Transaction method*), 19

W

Wallet (*class in bioblockchain.wallet*), 19