# PUT Blockchain Implementation

Eng. K. Kłodziński, Eng. K. Baran

June 14, 2023

## Contents

## 1 Introduction

## 2 Documentation

This section contains a brief desciption of what most functions do, their parameters and return types.

### 2.1 block_chain.hpp

This files contains the main functions relative to the blockchain implementation. Transaction structure and block structure definition, functions to create transactions, add them to the chain, sign transaction and many more.

This file is namesapced as *put::blockchain::block_chain*.

#### 2.1.1 Constants

- **BLOCK_SIZE** = 10

#### 2.1.2 struct transaction_t

- *uint16_t transaction_id* – ID of the transaction
- *uint64_t sender_id* – ID of the sender
- *uint64_t recipient_id* – ID of the receiver
- *uint64_t transaction_amount* – PUT coins amount
- *char signature[256]* = {*0*} – transaction signature

#### 2.1.3 struct transaction_block_t

- *unsigned char previous_block_hash[SHA256_DIGEST_LENGTH]* – SHA256 of the previous block
- *transaction_t transactions[BLOCK_SIZE]* – transaction ledger
- *uint64_t proof_of_work* = *NULL* – proof-of-work for the current block

#### 2.1.4 class block_chain

Private fields:

- *RSA *private_key* = *nullptr* – RSA context with the private key file
- *uint64_t last_transaction_id* – ID of the last transaction wich will be later increased

- *std::vector¡transaction_t¿ transactions* – transactions on the current ledger appended later when new block will be generated
- *transaction_block_t newest_transaction_block* – last generated transaction block

Public fields:

**block_chain(uint64_t last_transaction_id)**
Constructor without private key. Necessary for blockchain miners as they do not need to add transactions to the ledger. Index of last transaction is used for incrementing transaction id when creating transactions.

- *last_transaction_id* – index of last transaction

**block_chain(std::string private_key_file_name, uint64_t last_transaction_id)**
Constructor with private RSA key in *pem* format. This key is used to sign transactions. Index of last transaction is used for incrementing transaction id when creating transactions.

- *private_key_file_name* – string with path to private keyfile. Can be relative.
- *last_transaction_id* – index of last transaction

**void set_private_key(std::string private_key_file_name)**
This function reads the RSA private key in PEM format and sets the current RSA context with this key.

- *private_key_file_name* – string with path to private keyfile. Can be relative.

**transaction_t add_transaction(uint64_t sender_id, uint64_t recipient_id, uint64_t transaction_amount)**
Adds a transaction to the current ledger. Returns error if the ledger does not have space for new transactios. A new block should be manually generated to clear the ledger. Transactions is signed with the RSA private key.

- *sender_id* – the ID of the sender
- *recipient_id* – the ID of the recipient
- *transaction_amount* – the amount of PUT coins to send

**transaction_t create_transaction(uint64_t sender_id, uint64_t recipient_id, uint64_t transaction_amount)**
Creates a transaction but does not add it to the current ledger. Increases the transaction ID.

- *sender_id* – the ID of the sender
- *recipient_id* – the ID of the recipient
- *transaction_amount* – the amount of PUT coins to send

**void add_transaction(transaction_t transaction)**
Adds a previously generated transactionto the current block.

- *transaction* – the transaction to add to the ledger

**transaction_block_t create_transaction_block(unsigned char previous_block_hash[SHA256_DIGEST_LENGTH])**
Creates a new transaction block if enough transactions are on the ledger. It flushes the current transaction ledger and sets this block as the newest block. It requires the has of the previous block as a parameter. This can be getted with *get_transaction_hash*.

- *previous_block_hash[SHA256_DIGEST_LENGTH]* – hash of the previous transaction block

Returns the generated block.

**unsigned char *get_transaction_block_hash()**
Gets the last transaction block hash. It is generated anew every call as this has is not stored in the block.

**void get_transaction_block_hash(unsigned char * transaction_hash)**
Gets the last transaction block and saves it into *transaction_hash* memory block.

- *transaction_hash* – pointer to beginning of char array