PEPITAS CRYPTOCURRENCY

Generated by Doxygen 1.8.17

Chapter 1

CODING STYLE

- *Functions*, *variables* and *filenames* must be written in `snake_case`.
- *Structures* must be written in `PascalCase`.
- *Constants* or *MACRO* must be written in `UPPER_SNAKE_CASE`.

Chapter 2

PEPITAS NETWORK PROTOCOL

2.1 HEADERS

2.1.1 Sync Headers

1. CONNECTION TO NETWORK
2. CONNECTION TO NODE
3. GET BLOCKS
4. ACTUAL HEIGHT
5. SEND BLOCK
6. GET PENDING TRANSACTION LIST
7. REJECT DEMAND

2.1.2 Running Headers

1. SEND PENDING TRANSACTION

2.1.3 Validating Headers

1. SEND BLOCK EPOCHMAN
2. SEND VOTE

2.1.4 CONNECTION TO NETWORK

Message:

- char ∗ : "CONNECTION TO NETWORK\r\n\r\n"

Description Send a request to be accepted by a network door.

2.1.5 CONNECTION TO NODE

Message:

- char * : "CONNECTION TO NODE\r\n\r\n"

Description Send a request to be accepted by a network node.

2.1.6 GET BLOCKS

Message:

- char * : "GET BLOCKS\r\n\r\n"
- uint32_t : P_VERSION
- char : Number of demand (max 50)
- size_t * : Block height

Description Send a request to a server for getting blocks. If the genesis block (height 0) is demand then the number of the actual blockchain height is return with "ACTUAL HEIGHT" header. If not, SEND BLOCK or REJECT DEMAND messages are returned.

2.1.7 ACTUAL HEIGHT

Message:

- char * : "ACTUAL HEIGHT\r\n\r\n"
- size_t : Block height

Description Send my actual blockchain height.

2.1.8 SEND BLOCK

Message:

- char * : "SEND BLOCK\r\n\r\n"
- size_t : Block height
- size_t : Block size
- char * : Block struct

Description The block of height demand by "GET BLOCKS".

2.1.9 GET PENDING TRANSACTION LIST

Message

- char * : "GET PENDING TRANSACTION LIST\r\n\r\n"

Description Call "SEND PENDING TRANSACTION LIST".

2.1.10 SEND PENDING TRANSACTION LIST

Message

- char * : "GET PENDING TRANSACTION LIST\r\n\r\n"
- size_t : Number of Transaction id
- time_t * : Transaction id

Description Send PDT list.

2.1.11 REJECT DEMAND

Message:

- char * : "REJECT DEMAND\r\n\r\n"

Description Reject a demand if can't reply. For example a "GET BLOCKS" of a not existing block.

2.1.12 GET PENDING TRANSACTION

Message:

- char * : "GET PENDING TRANSACTION\r\n\r\n"
- time_t : Transaction id

Description Demand a PENDING TRANSACTION.

2.1.13 SEND PENDING TRANSACTION

Message:

- char * : "SEND PENDING TRANSACTION\r\n\r\n"
- size_t : Transaction id
- size_t : Transaction struct size octet
- char * : Transaction struct

Description Send the PENDING TRANSACTION demand by SEND PENDING TRANSACTION.

2.1.14 SEND EPOCH BLOCK

Message:

- char * : "SEND EPOCH BLOCK\r\n\r\n"
- int : Epoch id
- size_t : Block height
- char * : Block struct

Description Send the epoch block of a committee member.

2.1.15 SEND VOTE

Message:

- char * : "SEND VOTE\r\n\r\n"
- size_t : size epoch creator pk
- char * : Epoch creator pk
- size_t : block height
- int : epoch_id
- char : 0 = False 1 = True
- char * : signature of vote precedent vars but not "SEND VOTE\r\n\r\n"

Description Send the vote of a committee member.

Chapter 3

README

3.1 PEPITAS, a C cryptocurrency

PEPITAS is an EPITA project made in the last semester of the preparatory cycle. This cryptocurrency is based on the *proof of stake*, the new validation concensus (used in Etherum 2.0).

With PEPITAS, you can test whatever a modern proof of stake based money can do :

- Send money
- Receive money
- Invest on the stake
- Validate transactions
- Earn transactions fees

All of these features are obviously secured by a 2048-bits RSA based protocol.

3.1.1 Requirements

- A Linux system (Ubuntu, Arch,...)
- GNU Make
- OpenSSL
- GTK

3.1.2 Installation

1. Download the last version of the project : `PEPITAS-Cryptocurrency`.
2. Exctract the archive and go to the exctracted directory
3. Open a terminal in the current directory, type `make`
4. Go to the `build` directory
5. Execute `client.elf`, with an argument : the IP address of an existing client, or without argument if you are the first node of the network

3.1.3 Some explainations about how the program works

When launched, the client will try to connect to the host you provided as an argument (if provided). In the case where no argument are given to the ELF program, the client will try to etablish a connect to a *serverdoor* (a node which have executed the program `serverdoor.elf`). A serverdoor is a program that provides IP adresses to a node in order to let him have a connection with the corresponding hosts.

NB : serverdoors IP adresses are stored in the `HARD_CODED_ADDR` macro in `src/core/network/network.c` and may not work if Maxence and/or Nathan decide to close the non-localhost serverdoor (currently hosted in a Google Cloud instance). If you really want to use a serverdoor, you also can refactor the `HARD_CODED_ADDR` macro and then run the program `serverdoor.elf`

If you want more information about the peer-to-peer or the validation protocol, you also can read the P2P_PROTOCOL.md) or the VALIDATION_PROTOCOL.md documentation.

Also, don't hesitate to check our Doxygen `code documentation`!

3.1.4 Contributors

- **Nathan RABET**, project leader, in charge of the blockchain and the validation protocol.
- **Maxence ODEN**, in charge of the networking and cryptographic part of this project.
- **Souleymane SENTICI**, responsible for the user interface.
- **Luca SAINGIER**, responsible for the web implementation.

Chapter 4

PEPITAS VALIDATION PROTOCOL

4.1 Prerequisites

To understand this documentation, you need to have a good understanding of the blockchain data structure used in cryptocurrencies and the concept of the proof of stake.

4.2 Introduction

PEPITAS is a C written cryptocurrency. At the beginning of cryptocurrencies, the method (or concensus) used to guarantee the network security was the *proof of work*, users computers had to calculate some hashes to validate transactions (also called *mining*). These calculations ensure a good security,but are not eco-frendly (because of the huge amount of CPU's and GPU's in charge of calculating hashes). This issue enrolled a new concensus : the *proof of stake*. This type of validation protocol doesn't use calculations to prove a transaction validity, but the money users putted in a bank, named the *stake*. The more a user send money to the stake, the more he has a chance to be selected to create a new block, and by the time, to earn money as a reward. It is important to note that if the other users of the network detect that a validator validated fraudulent transactions, the corresponding validator will lose some part of his stake. This punishment ensure that all users have more interest to validate valid transactions instead of fraudulent ones.

4.3 Definitions

4.3.1 VALIDATOR

Members of the network who can validate and create block. Each of there **STAKE** must contains at least 50 PEPITAS.

4.3.2 COMMITTEE

A list containing public keys, correpsonding to some accounts on the network. Each account in this list is allowed to participate to the validation and the creation of a new block for the blockchain network. A committee is pseudo-randomly selected and is known by every node of the network. It changes every time a block is added to the blockchain. The more a user puts money in his stake, the more he has a chance to appear and have a low ID in a committee.

4.3.3 EPOCH MAN

The committee (list) ID of the block creator. The EPOCH MAN is selected by priority order in a committee with this rule : $min(awaken_validator_ID)$ For example if the committee contains 10 members and the first awaken member is the third, EPOCH MAN is the third member of this comitte. An awaken member is a committee member who broadcast a block to the network or a committee member that send a verdict on a broadcasted block.

4.3.4 COMITAL

If the committee contains n members and the selected EPOCH MAN is the validator with the ID m, the comital members ID are from 0 to m (excluded) and from $m+1$ to n (excluded).

4.3.5 VOTE

A vote is a validator judgment about a the validity of a certain block. If a validator think that a block is valid, he will send a **postive** vote, otherwise, he will send a **negative** one. Note that the block at height 0 (genesis block) is considered as valid by default.

4.3.6 PLÈBE

All non-validators members. Each of there **STAKE** are under 50 PEPITAS.

4.3.7 MEMPOOL

A directory where all pending transactions (transactions that are not in a block) are stocked.

4.4 How EPOCH MAN creates a block

Lets admit that the current validated block is at height n.

To create a block, EPOCH MAN do several things :

4.4.1 Last block validity checking

- First, he creates a new empty block (at height $n+1$).
- Then, he check if the validators votes ratio of the block at height n.
  - If the block at height n has more positive than negative votes.
    - Writes on the block at height $n+1$ that the block at height n is valid.
    - Flushes the transactions in the block at height n from the mempool.
  - Else
    - Writes on the block at height $n+1$ that the block at height n is not valid.

4.4.2 Rewards and punishments attribution

To motivate the network to do the job correctly (without corruption), EPOCH MAN will create new special transactions called *rewards* and *punishments*. Rewards are transactions that "∗send∗" money to a validator (actually this transaction creates money) and punishments that *take of* money from a validator (this transaction delete money from an account). Before this step, EPOCH MAN checked the validators votes ratio of the block at height n, then, he will create rewards transactions for the majority and punishments for the others. If there is equality on votes, the block is considered as non-valid and the same rule is applied.

4.4.3 Broadcast

After all these steps, EPOCH MAN broadcasts his new block.

4.5 How COMITAL send their verdicts

1. A validator waits for a block from a validator that has a lower ID than him in the next committee. If it receive one, he will start computing it.
2. Verify the validity of the received block.
3. Send a verdict.
4. Reiterate if the validator receive a block from another EPOCH MAN with an ID lower than the previous EPOCH MAN, for a certain amount of time.

Note that if a member of the COMITAL doesn't send any verdict, he will be punished by the next EPOCH MAN.

4.6 How PLÈBE adhere blocks

1. A node waits for a block from a validator
2. Adhere all verdicts from the next committee
3. Reiterate for a certain amount of time, using the same rule as the COMITAL.
4. Keep the received block
5. If the previous block is valid (info stored in the received block), then flushes the transactions in the previous block from the mempool.

Chapter 5

Deprecated List

Global delete_epochs (size_t height)

Global gen_blockchain_header (infos_st *infos)

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Data Structure Documentation

8.1 Block Struct Reference

`#include <block.h>`

Collaboration diagram for Block:

Data Fields

- uint16_t chunk_id
- BlockData block_data
- char block_signature [256]
- char validators_votes [NB_VOTES_BITMAP]
- char vote_signature [MAX_VALIDATORS_PER_BLOCK - 1][256]

8.1.1 Detailed Description

Definition at line 80 of file block.h.

8.1.2 Field Documentation

8.1.2.1 block_data

`BlockData block_data`

Definition at line 83 of file block.h.

8.1.2.2 block_signature

```
char block_signature[256]
```

Definition at line 85 of file block.h.

8.1.2.3 chunk_id

```
uint16_t chunk_id
```

Definition at line 82 of file block.h.

8.1.2.4 validators_votes

```
char validators_votes[NB_VOTES_BITMAP]
```

Definition at line 88 of file block.h.

8.1.2.5 vote_signature

```
char vote_signature[MAX_VALIDATORS_PER_BLOCK - 1][256]
```

Definition at line 89 of file block.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h

8.2 BlockData Struct Reference

```
#include <block.h>
```

Collaboration diagram for BlockData:

Data Fields

- char magic
- int epoch_id
- char is_prev_block_valid
- char previous_block_hash [SHA384_DIGEST_LENGTH ∗2+1]
- size_t height
- uint16_t nb_transactions
- Transaction ∗∗ transactions
- int nb_validators
- RSA ∗ validators_public_keys [MAX_VALIDATORS_PER_BLOCK]
- time_t block_timestamp
- char prev_validators_votes [NB_VOTES_BITMAP]

8.2.1 Detailed Description

Definition at line 61 of file block.h.

8.2.2 Field Documentation

8.2.2.1 block_timestamp

```
time_t block_timestamp
```

Definition at line 75 of file block.h.

8.2.2.2 epoch_id

```
int epoch_id
```

Definition at line 64 of file block.h.

8.2.2.3 height

```
size_t height
```

Definition at line 67 of file block.h.

8.2.2.4 is_prev_block_valid

```
char is_prev_block_valid
```

Definition at line 65 of file block.h.

8.2.2.5 magic

```
char magic
```

Definition at line 63 of file block.h.

8.2.2.6 nb_transactions

```
uint16_t nb_transactions
```

Definition at line 69 of file block.h.

8.2.2.7 nb_validators

```
int nb_validators
```

Definition at line 73 of file block.h.

8.2.2.8 prev_validators_votes

```
char prev_validators_votes[NB_VOTES_BITMAP]
```

Definition at line 77 of file block.h.

8.2.2.9 previous_block_hash

```
char previous_block_hash[SHA384_DIGEST_LENGTH *2+1]
```

Definition at line 66 of file block.h.

8.2.2.10 transactions

```
Transaction** transactions
```

Definition at line 70 of file block.h.

8.2.2.11 validators_public_keys

```
RSA* validators_public_keys[MAX_VALIDATORS_PER_BLOCK]
```

Definition at line 74 of file block.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h

8.3 blockinfo Struct Reference

`#include <ui.h>`

Data Fields

- size_t height
- size_t transactions

8.3.1 Detailed Description

Definition at line 26 of file ui.h.

8.3.2 Field Documentation

8.3.2.1 height

`size_t height`

Definition at line 28 of file ui.h.

8.3.2.2 transactions

`size_t transactions`

Definition at line 29 of file ui.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h

8.4 ChunkBlockchain Struct Reference

`#include <block.h>`

Collaboration diagram for ChunkBlockchain:

Data Fields

- size_t chunk_nb
- Block ** chunk
- int16_t nb_blocks

8.4.1 Detailed Description

Definition at line 92 of file block.h.

8.4.2 Field Documentation

8.4.2.1 chunk

```
Block** chunk
```

Definition at line 95 of file block.h.

8.4.2.2 chunk_nb

```
size_t chunk_nb
```

Definition at line 94 of file block.h.

8.4.2.3 nb_blocks

```
int16_t nb_blocks
```

Definition at line 96 of file block.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h

8.5 connection Struct Reference

```
#include <network.h>
```

Data Fields

- pthread_t thread
- sem_t lock
- int demand
- int clientfd
- size_t Payloadsize
- void * Payload
- size_t actual_client_height

8.5.1 Detailed Description

Definition at line 44 of file network.h.

8.5.2 Field Documentation

8.5.2.1 actual_client_height

```
size_t actual_client_height
```

Definition at line 52 of file network.h.

8.5.2.2 clientfd

```
int clientfd
```

Definition at line 49 of file network.h.

8.5.2.3 demand

```
int demand
```

Definition at line 48 of file network.h.

8.5.2.4 lock

```
sem_t lock
```

Definition at line 47 of file network.h.

8.5.2.5 Payload

```
void* Payload
```

Definition at line 51 of file network.h.

8.5.2.6 Payloadsize

```
size_t Payloadsize
```

Definition at line 50 of file network.h.

8.5.2.7 thread

```
pthread_t thread
```

Definition at line 46 of file network.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h

8.6 infos_st Struct Reference

```
#include <network.h>
```

Data Fields

- char as_epoch
- char is_validator
- int validator_id
- size_t actual_height
- size_t pdt
- char serv_type
- char is_sychronize

8.6.1 Detailed Description

Definition at line 55 of file network.h.

8.6.2 Field Documentation

8.6.2.1 actual_height

```
size_t actual_height
```

Definition at line 60 of file network.h.

8.6.2.2 as_epoch

```
char as_epoch
```

Definition at line 57 of file network.h.

8.6.2.3 is_sychronize

```
char is_sychronize
```

Definition at line 10 of file unit_testing.c.

8.6.2.4 is_validator

```
char is_validator
```

Definition at line 58 of file network.h.

8.6.2.5 pdt

```
size_t pdt
```

Definition at line 61 of file network.h.

8.6.2.6 serv_type

```
char serv_type
```

Definition at line 62 of file network.h.

8.6.2.7 validator_id

```
int validator_id
```

Definition at line 59 of file network.h.

The documentation for this struct was generated from the following files:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit_testing.c

8.7 Neighbour Struct Reference

```
#include <network.h>
```

Data Fields

- int family
- char * hostname

8.7.1 Detailed Description

Definition at line 33 of file network.h.

8.7.2 Field Documentation

8.7.2.1 family

```
int family
```

Definition at line 35 of file network.h.

8.7.2.2 hostname

```
char* hostname
```

Definition at line 36 of file network.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h

8.8 Node Struct Reference

```
#include <network.h>
```

Collaboration diagram for Node:

Data Fields

- Neighbour * neighbours

8.8.1 Detailed Description

Definition at line 39 of file network.h.

8.8.2 Field Documentation

8.8.2.1 neighbours

```
Neighbour* neighbours
```

Definition at line 41 of file network.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h

8.9 th_arg Struct Reference

```
#include <network.h>
```

Collaboration diagram for th_arg:

Data Fields

- infos_st * infos
- connection * client_con

8.9.1 Detailed Description

Definition at line 64 of file network.h.

8.9.2 Field Documentation

8.9.2.1 client_con

```
connection* client_con
```

Definition at line 67 of file network.h.

8.9.2.2 infos

```
infos_st* infos
```

Definition at line 66 of file network.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h

8.10 Transaction Struct Reference

```
#include <block.h>
```

Collaboration diagram for Transaction:

Data Fields

- TransactionData transaction_data
- char transaction_signature [256]

8.10.1 Detailed Description

Definition at line 51 of file block.h.

8.10.2 Field Documentation

8.10.2.1 transaction_data

```
TransactionData transaction_data
```

Definition at line 53 of file block.h.

8.10.2.2 transaction_signature

```
char transaction_signature
```

Definition at line 55 of file block.h.

The documentation for this struct was generated from the following files:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h

8.11 TransactionData Struct Reference

```
#include <block.h>
```

Data Fields

- char magic
- char type
- RSA * sender_public_key
- RSA * receiver_public_key
- size_t amount
- size_t sender_remaining_money
- size_t receiver_remaining_money
- time_t transaction_timestamp
- char cause [512]
- char asset [512]

8.11.1 Detailed Description

Definition at line 32 of file block.h.

8.11.2 Field Documentation

8.11.2.1 amount

```
size_t amount
```

Definition at line 40 of file block.h.

8.11.2.2 asset

```
char asset
```

Definition at line 48 of file block.h.

8.11.2.3 cause

```
char cause
```

Definition at line 47 of file block.h.

8.11.2.4 magic

```
char magic
```

Definition at line 34 of file block.h.

8.11.2.5 receiver_public_key

```
RSA * receiver_public_key
```

Definition at line 39 of file block.h.

8.11.2.6 receiver_remaining_money

```
size_t receiver_remaining_money
```

Definition at line 42 of file block.h.

8.11.2.7 sender_public_key

```
RSA * sender_public_key
```

Definition at line 38 of file block.h.

8.11.2.8 sender_remaining_money

```
size_t sender_remaining_money
```

Definition at line 41 of file block.h.

8.11.2.9 transaction_timestamp

```
time_t transaction_timestamp
```

Definition at line 43 of file block.h.

8.11.2.10 type

```
char type
```

Definition at line 35 of file block.h.

The documentation for this struct was generated from the following files:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h

8.12 validators_state_header Struct Reference

```
#include <validators.h>
```

Data Fields

- size_t nb_validators
- size_t total_stake
- size_t block_height_validity

8.12.1 Detailed Description

Definition at line 14 of file validators.h.

8.12.2 Field Documentation

8.12.2.1 block_height_validity

```
size_t block_height_validity
```

Definition at line 18 of file validators.h.

8.12.2.2 nb_validators

```
size_t nb_validators
```

Definition at line 16 of file validators.h.

8.12.2.3 total_stake

```
size_t total_stake
```

Definition at line 17 of file validators.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h

8.13 validators_state_item Struct Reference

```
#include <validators.h>
```

Data Fields

- char validator_pkey [RSA_KEY_SIZE]
- size_t user_stake
- size_t validator_power

8.13.1 Detailed Description

Definition at line 21 of file validators.h.

8.13.2 Field Documentation

8.13.2.1 user_stake

```
size_t user_stake
```

Definition at line 24 of file validators.h.

8.13.2.2 validator_pkey

```
char validator_pkey[RSA_KEY_SIZE]
```

Definition at line 23 of file validators.h.

8.13.2.3 validator_power

```
size_t validator_power
```

Definition at line 25 of file validators.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h

8.14 Wallet Struct Reference

```
#include <wallet.h>
```

Data Fields

- RSA * priv_key
- RSA * pub_key
- size_t amount
- size_t stake_amount

8.14.1 Detailed Description

Definition at line 10 of file wallet.h.

8.14.2 Field Documentation

8.14.2.1 amount

```
size_t amount
```

Definition at line 15 of file wallet.h.

8.14.2.2 priv_key

```
RSA* priv_key
```

Definition at line 12 of file wallet.h.

8.14.2.3 pub_key

```
RSA* pub_key
```

Definition at line 13 of file wallet.h.

8.14.2.4 stake_amount

```
size_t stake_amount
```

Definition at line 16 of file wallet.h.

The documentation for this struct was generated from the following file:

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/wallet.h

Chapter 9

File Documentation

9.1 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/CODING_STYLE.md File Reference

9.2 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h File Reference

```
#include <string.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <err.h>
#include <errno.h>
#include <openssl/sha.h>
#include <openssl/pem.h>
#include <openssl/rsa.h>
#include <openssl/crypto.h>
#include <fcntl.h>
#include <sys/types.h>
#include "client.h"
#include "transaction.h"
#include "misc/files.h"
#include "blockchain/wallet.h"
#include "cryptosystem/rsa.h"
```

Include dependency graph for block.h:

9.3 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/blockchain_header.h File Reference

```
#include "network/network.h"
#include "blockchain/block.h"
```

```
#include "cryptosystem/rsa.h"
#include "validation/validators.h"
#include <sys/stat.h>
#include <stdio.h>
```

Include dependency graph for blockchain_header.h: This graph shows which files directly or indirectly include this file:

Functions

- void gen_blockchain_header (infos_st *infos)

  Generate block shared information.
- size_t get_receiver_remaining_money (infos_st *infos, RSA *receiver_public_key)

  Get the receiver remaining money.

9.3.1 Function Documentation

9.3.1.1 gen_blockchain_header()

```
void gen_blockchain_header (
            infos_st * infos )
```

Generate block shared information.

Deprecated

Parameters

infos	The information

Definition at line 9 of file blockchain_header.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.3.1.2 get_receiver_remaining_money()

```
size_t get_receiver_remaining_money (
            infos_st * infos,
            RSA * receiver_public_key )
```

Get the receiver remaining money.

Parameters

infos	Threads shared information
receiver_public_key	The RSA public key of the receiver

Returns

size_t

Definition at line 40 of file blockchain_header.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h File Reference

```
#include <string.h>
#include <stdlib.h>
#include <openssl/rsa.h>
#include <openssl/sha.h>
#include <openssl/pem.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <err.h>
#include "blockchain/wallet.h"
#include "blockchain/blockchain_header.h"
```

Include dependency graph for transaction.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct TransactionData
- struct Transaction

Macros

- #define TRANSACTION_DATA_SIZE sizeof(size_t) ∗ 3 + sizeof(time_t) + (512 ∗ 2)
- #define TRANSACTION_SIZE sizeof(size_t) + 2048 + TRANSACTION_DATA_SIZE
- #define T_TYPE_DEFAULT 0
- #define T_TYPE_ADD_STAKE 1
- #define T_TYPE_WITHDRAW_STAKE 2
- #define T_TYPE_REWARD_STAKE 3
- #define T_TYPE_PUNISH_STAKE 4
- #define TRANS_T

Typedefs

- typedef struct TransactionData TransactionData
- typedef struct Transaction Transaction

Functions

- int send_money (size_t amount, u_int64_t receiver_public_key)

  Send 'amount' money to 'receiver_public_key'. This will broadcast a transaction to the network.
- void write_transactiondata (TransactionData *transaction, int fd)

  Serialize a TransactionData structure.*
- void write_transaction (Transaction *transaction, int fd)

  Serialize a Transaction structure.*
- void get_transaction_data (Transaction *trans, char **buff, size_t *index)

  Get the transaction data object.
- void convert_data_to_transactiondata (TransactionData *transactiondata, int fd)

  Convert serialized TransactionData to TransactionData*.*
- void load_transaction (Transaction *transaction, int fd)

  Load a serialized Transaction structure.*
- Transaction * load_pending_transaction (time_t timestamp)

  Load a transaction in the pending transaction (pdt) directory.
- void add_pending_transaction (Transaction *transaction)

  Add a transaction to the pending transaction (pdt) directory.
- Transaction create_new_transaction (infos_st *infos, char type, RSA *receiver_public_key, size_t amount, char cause[512], char asset[512])

  Create a new transaction.
- void flush_pending_transactions (Transaction **transactions, size_t nb_transactions)

  Delete block transactions in the pending transaction (pdt) directory if the block is valid.

9.4.1 Macro Definition Documentation

9.4.1.1 T_TYPE_ADD_STAKE

```
#define T_TYPE_ADD_STAKE 1
```

Definition at line 22 of file transaction.h.

9.4.1.2 T_TYPE_DEFAULT

```
#define T_TYPE_DEFAULT 0
```

Definition at line 21 of file transaction.h.

9.4.1.3 T_TYPE_PUNISH_STAKE

```
#define T_TYPE_PUNISH_STAKE 4
```

Definition at line 25 of file transaction.h.

9.4.1.4 T_TYPE_REWARD_STAKE

```
#define T_TYPE_REWARD_STAKE 3
```

Definition at line 24 of file transaction.h.

9.4.1.5 T_TYPE_WITHDRAW_STAKE

```
#define T_TYPE_WITHDRAW_STAKE 2
```

Definition at line 23 of file transaction.h.

9.4.1.6 TRANS_T

```
#define TRANS_T
```

Definition at line 28 of file transaction.h.

9.4.1.7 TRANSACTION_DATA_SIZE

```
#define TRANSACTION_DATA_SIZE sizeof(size_t) * 3 + sizeof(time_t) + (512 * 2)
```

Definition at line 18 of file transaction.h.

9.4.1.8 TRANSACTION_SIZE

```
#define TRANSACTION_SIZE sizeof(size_t) + 2048 + TRANSACTION_DATA_SIZE
```

Definition at line 19 of file transaction.h.

9.4.2 Typedef Documentation

9.4.2.1 Transaction

```
typedef struct Transaction Transaction
```

9.4.2.2 TransactionData

```
typedef struct TransactionData TransactionData
```

9.4.3 Function Documentation

9.4.3.1 add_pending_transaction()

```
void add_pending_transaction (
            Transaction * transaction )
```

Add a transaction to the pending transaction (pdt) directory.

Parameters

transaction	The transaction to add

Definition at line 140 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4.3.2 convert_data_to_transactiondata()

```
void convert_data_to_transactiondata (
            TransactionData * transactiondata,
            int fd )
```

Convert serialized TransactionData∗ to TransactionData∗.

Parameters

transactiondata	The returned TransactionData∗
fd	The serialized TransactionData FD

Definition at line 88 of file transaction.c.

Here is the caller graph for this function:

9.4.3.3 create_new_transaction()

```
Transaction create_new_transaction (
            infos_st * infos,
            char type,
            RSA * receiver_public_key,
            size_t amount,
```

```
            char cause[512],
            char asset[512] )
```

Create a new transaction.

Parameters

infos	Shared information object
type	The type of transaction
receiver_public_key	The receiver pkey
amount	The amount of PEPITAS
cause	The cause (deprecated)
asset	The asset (deprecated)

Returns

Transaction

Definition at line 157 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4.3.4 flush_pending_transactions()

```
void flush_pending_transactions (
            Transaction ** transactions,
            size_t nb_transactions )
```

Delete block transactions in the pending transaction (pdt) directory if the block is valid.

Parameters

transactions	block.blockdata.transactions
nb_transactions	number of transactions

Definition at line 204 of file transaction.c.

9.4.3.5 get_transaction_data()

```
void get_transaction_data (
            Transaction * trans,
            char ** buff,
            size_t * index )
```

Get the transaction data object.

Parameters

trans	The returned transaction
buff	The buffer with the serialized data
index	The buffer starting offset

Definition at line 40 of file transaction.c.

Here is the caller graph for this function:

9.4.3.6 load_pending_transaction()

```
Transaction* load_pending_transaction (
            time_t timestamp )
```

Load a transaction in the pending transaction (pdt) directory.

Parameters

timestamp	The timestamp of the transaction

Returns

Transaction*

Definition at line 127 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4.3.7 load_transaction()

```
void load_transaction (
            Transaction * transaction,
            int fd )
```

Load a serialized Transaction* structure.

Parameters

transaction	The returned Transaction*
fd	The serialized Transaction FD

Definition at line 117 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4.3.8 send_money()

```
int send_money (
```

```
            size_t amount,
            u_int64_t receiver_public_key )
```

Send 'amount' money to 'receiver_public_key'. This will broadcast a transaction to the network.

Parameters

amount	The amount to send
receiver_public_key	The receiver public key

Returns

returns 0 if the broadcast succeeds, -1 otherwise

9.4.3.9 write_transaction()

```
void write_transaction (
            Transaction * transaction,
            int fd )
```

Serialize a Transaction∗ structure.

Parameters

transaction	The Transaction∗ structure to serialize
fd	The output file FD

Definition at line 34 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.4.3.10 write_transactiondata()

```
void write_transactiondata (
            TransactionData * transaction,
            int fd )
```

Serialize a TransactionData∗ structure.

Parameters

transaction	The TransactionData∗ structure to serialize
fd	The output file FD

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

9.5 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/wallet.h File Reference

```
#include <openssl/rsa.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "ui/labels.h"
```

Include dependency graph for wallet.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct Wallet

Typedefs

- typedef struct Wallet Wallet

Functions

- Wallet * get_my_wallet ()
  Get my wallet object.
- int create_account ()
  Creates an account in local and broadcasts the creation to the network.
- void add_money_to_wallet (size_t money)
  Add money to my wallet.
- void remove_money_from_wallet (size_t money)
  Remove money from my wallet.
- void add_money_to_stake (size_t money)
  Add money to my stake.
- void remove_money_from_stake (size_t money)
  Withdraw money from my stake.

9.5.1 Typedef Documentation

9.5.1.1 Wallet

```
typedef struct Wallet Wallet
```

9.5.2 Function Documentation

9.5.2.1 add_money_to_stake()

```
void add_money_to_stake (
            size_t money )
```

Add money to my stake.

Parameters

money	The amount of PEPITAS

Definition at line 45 of file wallet.c.

Here is the call graph for this function:

9.5.2.2 add_money_to_wallet()

```
void add_money_to_wallet (
             size_t money )
```

Add money to my wallet.

Parameters

money	The amount of PEPITAS

Definition at line 26 of file wallet.c.

Here is the call graph for this function:

9.5.2.3 create_account()

```
int create_account ( )
```

Creates an account in local and broadcasts the creation to the network.

Returns

0 if the broadcast succeeds, otherwise 1

Definition at line 18 of file wallet.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.5.2.4 get_my_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

Returns

Wallet

Definition at line 6 of file wallet.c.

Here is the caller graph for this function:

9.5.2.5 remove_money_from_stake()

```
void remove_money_from_stake (
             size_t money )
```

Withdraw money from my stake.

Parameters

money	The amount of PEPITAS

Definition at line 54 of file wallet.c.

Here is the call graph for this function:

9.5.2.6 remove_money_from_wallet()

```
void remove_money_from_wallet (
            size_t money )
```

Remove money from my wallet.

Parameters

money	The amount of PEPITAS

Definition at line 34 of file wallet.c.

Here is the call graph for this function:

9.6 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/client.h File Reference

```
#include <signal.h>
#include <stdlib.h>
#include <string.h>
#include "network/network.h"
```

Include dependency graph for client.h:

Functions

- void new_transaction (char type, char *rc_pk, size_t amount, char cause[512], char asset[512])
- infos_st * get_infos ()
- void update_pdt (int number)
- void move_file (char *src, char *dest)
- void Validate ()
- void join_network_door (infos_st *infos)
- void connection_to_others (infos_st *infos)
- size_t update_blockchain_height (infos_st *infos)
- void update_blockchain (infos_st *infos, size_t index_client)
- void clear_transactions ()
- void clear_epochs ()
- void update_pending_transactions_list ()

9.6.1 Function Documentation

9.6.1.1 clear_epochs()

```
void clear_epochs ( )
```

Definition at line 335 of file atrier.c.

Here is the caller graph for this function:

9.6.1.2 clear_transactions()

```
void clear_transactions ( )
```

Definition at line 312 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.6.1.3 connection_to_others()

```
void connection_to_others (
            infos_st * infos )
```

Definition at line 228 of file atrier.c.

Here is the call graph for this function:

9.6.1.4 get_infos()

```
infos_st* get_infos ( )
```

Definition at line 16 of file atrier.c.

Here is the caller graph for this function:

9.6.1.5 join_network_door()

```
void join_network_door (
            infos_st * infos )
```

Definition at line 210 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.6.1.6 move_file()

```
void move_file (
            char * src,
            char * dest )
```

Definition at line 27 of file atrier.c.

Here is the call graph for this function:

9.6.1.7 new_transaction()

```
void new_transaction (
            char type,
            char * rc_pk,
            size_t amount,
            char cause[512],
            char asset[512] )
```

Definition at line 148 of file atrier.c.

Here is the caller graph for this function:

9.6.1.8 update_blockchain()

```
void update_blockchain (
            infos_st * infos,
            size_t index_client )
```

Definition at line 285 of file atrier.c.

9.6.1.9 update_blockchain_height()

```
size_t update_blockchain_height (
            infos_st * infos )
```

Definition at line 249 of file atrier.c.

Here is the call graph for this function:

9.6.1.10 update_pdt()

```
void update_pdt (
            int number )
```

Definition at line 20 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.6.1.11 update_pending_transactions_list()

```
void update_pending_transactions_list ( )
```

Definition at line 354 of file atrier.c.

Here is the call graph for this function:

9.6.1.12 Validate()

```
void Validate ( )
```

Definition at line 62 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h File Reference

```
#include "network/network.h"
#include "network/server.h"
#include "network/get_data.h"
#include "network/send_data.h"
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <sys/stat.h>
#include <unistd.h>
#include <err.h>
#include <errno.h>
#include <semaphore.h>
#include <stddef.h>
```

Include dependency graph for client.h: This graph shows which files directly or indirectly include this file:

Functions

- Node * get_my_node (char who)
  Get the my node object.
- int set_neighbour (char who, char *hostname, int family)
  Sets a neighbour in the client.neightbours section.
- void remove_neighbour (char who, int index)
  Remove a neighbour in the client.neightbours section.
- int number_neighbours (char who)
  Return the nb of neighbour in the client.neightbours section.
- void print_neighbours (char who, char mask)
  Print neighbours list.
- void save_neighbours (char who)

Save neighbours list in .neighbours/neighbours.

- void load_neighbours (char who)

  Load neighbours list from .neighbours/neighbours.

- connection * listen_to (infos_st *infos, Neighbour neighbour, char *connection_type, connection *connection)

  Tries to connect to the peer-to-peer network via a node in the Node structure.

- int find_empty_connection (int max, connection *connection)

  Find if connection has any empty field.

- int is_in_neighbours (char who, char *hostname)

  Check if `hostname` is in `client.neightbours`

- void * client_thread (void *args)

  Create a client thread.

9.7.1 Function Documentation

9.7.1.1 client_thread()

```
void* client_thread (
            void * args )
```

Create a client thread.

Parameters

args	

Returns

void*

Definition at line 268 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.2 find_empty_connection()

```
int find_empty_connection (
            int max,
            connection * connection )
```

Find if connection has any empty field.

Parameters

max	The number of maximum connections
connection	The connection* buffer

Returns

int

Definition at line 258 of file client.c.

Here is the caller graph for this function:

9.7.1.3 get_my_node()

```
Node* get_my_node (
            char who )
```

Get the my node object.

Parameters

who	Tells if it is the server or the client side

Returns

Node*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

9.7.1.4 is_in_neighbours()

```
int is_in_neighbours (
            char who,
            char * hostname )
```

Check if `hostname` is in `client.neightbours`

Parameters

who	Tells if it is the server or the client side
hostname	The IP adress to check

Returns

int

Definition at line 149 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.5 listen_to()

```
connection* listen_to (
            infos_st * infos,
            Neighbour neighbour,
            char * connection_type,
            connection * connection )
```

Tries to connect to the peer-to-peer network via a node in the Node structure.

Parameters

infos	Some shared information
neighbour	The neighbour to connect with
connection_type	The type of connection
connection	The connection* structure

Returns

socket FD or -1 if an error occurs

Definition at line 172 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.6 load_neighbours()

```
void load_neighbours (
            char who )
```

Load neighbours list from .neighbours/neighbours.

Parameters

who	Tells if it is the server or the client side

Definition at line 113 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.7 number_neighbours()

```
int number_neighbours (
            char who )
```

Return the nb of neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side

Definition at line 160 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.8 print_neighbours()

```
void print_neighbours (
             char who,
             char mask )
```

Print neighbours list.

Parameters

who	Tells if it is the server or the client side
mask	

Definition at line 58 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.9 remove_neighbour()

```
void remove_neighbour (
             char who,
             int index )
```

Remove a neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side
index	The index of the neigbour to remove in client.neightbours

Definition at line 47 of file client.c.

Here is the call graph for this function:

9.7.1.10 save_neighbours()

```
void save_neighbours (
             char who )
```

Save neighbours list in .neighbours/neighbours.

Parameters

who	Tells if it is the server or the client side

Definition at line 74 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.7.1.11 set_neighbour()

```
int set_neighbour (
            char who,
            char * hostname,
            int family )
```

Sets a neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side
hostname	The neighbour IP adress
family	The type of IP adress

Returns

0 if sucess, -1 otherwise if full

Definition at line 19 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.8 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h File Reference

```
#include <stdlib.h>
#include "blockchain/block.h"
```

Include dependency graph for hash.h: This graph shows which files directly or indirectly include this file:

Functions

- char * sha384_data (void *data, size_t len_data)

  Apply the SHA384 algorithm on a 'data' of size 'len_data'.
- char * hash_block_transactions (Block *block)

  Apply the SHA384 to all block transactions.

9.8.1 Function Documentation

9.8.1.1 hash_block_transactions()

```
char* hash_block_transactions (
            Block * block )
```

Apply the SHA384 to all block transactions.

Parameters

block	The block to deal with

Returns

sha384[SHA384_DIGEST_LENGTH]

Definition at line 24 of file hash.c.

Here is the call graph for this function:

9.8.1.2 sha384_data()

```
char* sha384_data (
            void * data,
            size_t len_data )
```

Apply the SHA384 algorithm on a 'data' of size 'len_data'.

Parameters

data	The buffer to hash
len_data	The length of the buffer

Returns

char[97] (on heap)

Definition at line 6 of file hash.c.

Here is the caller graph for this function:

9.9 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h File Reference

```
#include "blockchain/wallet.h"
#include <stdio.h>
#include <stdlib.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <err.h>
#include <errno.h>
#include <openssl/bn.h>
#include <openssl/crypto.h>
#include <string.h>
```

Include dependency graph for rsa.h: This graph shows which files directly or indirectly include this file:

Macros

- #define RSA_KEY_SIZE 366
- #define RSA_FILE_TOTAL_SIZE 426
- #define RSA_BEGIN_SIZE 31
- #define RSA_END_SIZE 29

Functions

- void get_keys (char *password)

  Get the keys object.

9.9.1 Macro Definition Documentation

9.9.1.1 RSA_BEGIN_SIZE

```
#define RSA_BEGIN_SIZE 31
```

Definition at line 21 of file rsa.h.

9.9.1.2 RSA_END_SIZE

```
#define RSA_END_SIZE 29
```

Definition at line 22 of file rsa.h.

9.9.1.3 RSA_FILE_TOTAL_SIZE

```
#define RSA_FILE_TOTAL_SIZE 426
```

Definition at line 20 of file rsa.h.

9.9.1.4 RSA_KEY_SIZE

```
#define RSA_KEY_SIZE 366
```

Definition at line 19 of file rsa.h.

9.9.2 Function Documentation

9.9.2.1 get_keys()

```
void get_keys (
            char * password )
```

Get the keys object.

Here is the caller graph for this function:

9.10 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h File Reference

```
#include <stdlib.h>
#include <err.h>
#include <string.h>
#include <openssl/crypto.h>
#include <openssl/ssl3.h>
#include <openssl/rsa.h>
#include <openssl/err.h>
#include "blockchain/wallet.h"
#include "blockchain/block.h"
#include "validation/epoch_man.h"
```

Include dependency graph for signature.h: This graph shows which files directly or indirectly include this file:

Functions

- char * sign_message (char *data, size_t len_data, void *buffer)

  buffer <- encrypt(SHA284(msg,len_data),wallet_priv_key)
- char * sign_message_with_key (char *data, size_t len_data, RSA *key, void *buffer)

  encrypt(SHA284(msg,len_data),key) buffer <- encrypt(SHA284(msg,len_data),key)
- int verify_signature (void *data, size_t data_len, char *signature, RSA *pub_key)

  Verifies if SHA384(data) == decrypt(signature,pub_key)
- int verify_block_signature (Block block)

  Verifies if a block signature is valid.
- int verify_transaction_signature (Transaction *transaction)

  Verifies if a transaction signature is valid.
- void get_transaction_data (Transaction *trans, char **buff, size_t *size)

  *Converts transactions to char * buffer.*
- void write_blockdata (BlockData blockdata, int fd)

  Writes blockdata in a file.
- void write_block (Block block, int fd)

  Writes a block in a file.
- void sign_block (Block *block)

Signs a block with my private key.

- void sign_block_with_key (Block *block, RSA *key)

  Signs a block.
- void sign_transaction (Transaction *transaction)

  Signs a transaction with my private key.
- void sign_transaction_with_key (Transaction *transaction, RSA *key)

  Signs a transaction.
- void sign_block_transactions (Block *block)

  Signs all transactions of a block with my private key.

9.10.1 Function Documentation

9.10.1.1 get_transaction_data()

```
void get_transaction_data (
            Transaction * trans,
            char ** buff,
            size_t * size )
```

Converts transactions to char * buffer.

Parameters

transactions	The transaction array
buff	The buffer that receives the transactions
size	The number of transactions in the array

Returns

The buffer allocated (Must be freed)

Converts transactions to char * buffer.

Parameters

trans	The returned transaction
buff	The buffer with the serialized data
index	The buffer starting offset

Definition at line 40 of file transaction.c.

9.10.1.2 sign_block()

```
void sign_block (
            Block * block )
```

Signs a block with my private key.

Parameters

block	The block to sign

Definition at line 108 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.3 sign_block_transactions()

```
void sign_block_transactions (
            Block * block )
```

Signs all transactions of a block with my private key.

Parameters

block	The block to sign

Definition at line 138 of file signature.c.

Here is the call graph for this function:

9.10.1.4 sign_block_with_key()

```
void sign_block_with_key (
            Block * block,
            RSA * key )
```

Signs a block.

Parameters

block	The block to sign
key	The key to use for the signature

Definition at line 115 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.5 sign_message()

```
char* sign_message (
            char * data,
            size_t len_data,
            void * buffer )
```

buffer <- encrypt(SHA284(msg,len_data),wallet_priv_key)

If buffer == NULL, return a new allocated buffer

Parameters

data	The data to sign
len_data	The length of the data
buffer	The buffer to put signature into

Returns

char*

Definition at line 10 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.6 sign_message_with_key()

```
char* sign_message_with_key (
            char * data,
            size_t len_data,
            RSA * key,
            void * buffer )
```

encrypt(SHA284(msg,len_data),key) buffer <- encrypt(SHA284(msg,len_data),key)

If buffer == NULL, return a new allocated buffer

Parameters

data	The data to sign
len_data	The length of the data
key	The key to use for the signature
buffer	The buffer to put signature into

Returns

char*

Definition at line 34 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.7 sign_transaction()

```
void sign_transaction (
            Transaction * transaction )
```

Signs a transaction with my private key.

Parameters

transaction	The transaction to sign

Definition at line 122 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.8 sign_transaction_with_key()

```
void sign_transaction_with_key (
            Transaction * transaction,
            RSA * key )
```

Signs a transaction.

Parameters

transaction	The transaction to sign
key	The key to use for the signature

Definition at line 130 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.9 verify_block_signature()

```
int verify_block_signature (
            Block block )
```

Verifies if a block signature is valid.

Parameters

block	The block to verify

Returns

1 if valid, 0 otherwise

Definition at line 83 of file signature.c.

Here is the call graph for this function:

9.10.1.10 verify_signature()

```
int verify_signature (
            void * data,
```

```
            size_t data_len,
            char * signature,
            RSA * pub_key )
```

Verifies if SHA384(data) == decrypt(signature,pub_key)

Parameters

data	The buffer to verify
data_len	The length of the buffer
signature	The signature to compare with SHA384(data, len_data)
pub_key	The RSA public key used for the decryption

Returns

int

Definition at line 57 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.11 verify_transaction_signature()

```
int verify_transaction_signature (
            Transaction * transaction )
```

Verifies if a transaction signature is valid.

Parameters

transaction	The transaction to verify

Returns

1 if valid, 0 otherwise

Definition at line 95 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.12 write_block()

```
void write_block (
            Block block,
            int fd )
```

Writes a block in a file.

Parameters

block	The block to write
fd	the file descriptor of the file in which the block is written

Definition at line 228 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.10.1.13 write_blockdata()

```
void write_blockdata (
            BlockData blockdata,
            int fd )
```

Writes blockdata in a file.

Parameters

blockdata	The blockdata to write
fd	The file descriptor of the file in which the blockdata is written

Definition at line 196 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.11 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/bits.h File Reference

This graph shows which files directly or indirectly include this file:

9.12 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- char * last_file_in_folder (char folder_path[ ])

  Return the last file (reverse alphabetical order) of a folder path.

9.12.1 Function Documentation

9.12.1.1 last_file_in_folder()

```
char* last_file_in_folder (
            char folder_path[] )
```

Return the last file (reverse alphabetical order) of a folder path.

Parameters

folder_path	The path of the folder

Returns

char*, return NULL if any error, must be freed !

Definition at line 7 of file files.c.

9.13 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- #define MIN(a, b) ((a) < (b)) ? (a) : (b)
- #define MAX(a, b) ((a) > (b)) ? (a) : (b)

9.13.1 Macro Definition Documentation

9.13.1.1 MAX

```
#define MAX(
            a,
            b ) ((a) > (b)) ?  (a) :  (b)
```

Definition at line 10 of file math.h.

9.13.1.2 MIN

```
#define MIN(
             a,
             b ) ((a) < (b)) ?  (a) :  (b)
```

Definition at line 9 of file math.h.

9.14 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <err.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
```

Include dependency graph for safe.h: This graph shows which files directly or indirectly include this file:

Functions

- int safe_write (int fd, const void *buf, ssize_t count)

  Writes safely to a file descriptor.
- int safe_send (int fd, const void *buf, ssize_t count)

  Send safely to a file descriptor.
- ssize_t safe_read (int fd, const void **buf, size_t *bufsize)

  Reads safely in a file descriptor until '\r\n\r\n'.
- ssize_t safe_fread (void *buffer, const size_t size, const size_t n, FILE *file)

  Calls 'fread' but safely !

9.14.1 Function Documentation

9.14.1.1 safe_fread()

```
ssize_t safe_fread (
             void * buffer,
             const size_t size,
             const size_t n,
             FILE * file )
```

Calls 'fread' but safely !

Parameters

buffer	The buffer to write on
size	The size of 1 read element
n	The number of elements to read
file	The IO FILE

Returns

ssize_t, -1 if error or the number of read items

Definition at line 58 of file safe.c.

Here is the caller graph for this function:

9.14.1.2 safe_read()

```
ssize_t safe_read (
            int fd,
            const void ** buf,
            size_t * bufsize )
```

Reads safely in a file descriptor until '\r\n\r\n'.

Parameters

fd	The file descriptor
buf	The buffer which contains the message

Returns

The number of byte the file 'fd', if -1 error

Definition at line 31 of file safe.c.

Here is the caller graph for this function:

9.14.1.3 safe_send()

```
int safe_send (
            int fd,
            const void * buf,
            ssize_t count )
```

Send safely to a file descriptor.

Parameters

fd	The file descriptor
buf	The buffer to write
count	The number of byte to write in fd

Returns

Error code

Definition at line 17 of file safe.c.

Here is the caller graph for this function:

9.14.1.4 safe_write()

```
int safe_write (
            int fd,
            const void * buf,
            ssize_t count )
```

Writes safely to a file descriptor.

Parameters

fd	The file descriptor
buf	The buffer to write
count	The number of byte to write in fd

Returns

Error code

Definition at line 3 of file safe.c.

Here is the caller graph for this function:

9.15 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h File Reference

```
#include <string.h>
#include "network/network.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "validation/validation_engine.h"
#include "ui/ui.h"
```

Include dependency graph for get_data.h: This graph shows which files directly or indirectly include this file:

Functions

- size_t read_header (int sockfd, infos_st *infos)

  Waits a header in 'sockfd', reads it and processes it.

- int fetch_client_list (char who, int fd)

  Fetches the client list from a socket fd.
- int read_get_blocks (int fd, infos_st ∗infos)

  Read blocks from a sock fd.
- size_t read_actual_height (int fd)

  Get the actual height of a node via its sock fd.
- int read_send_block (int fd)

  Read a socket sended block.
- int read_vote (int fd, infos_st ∗infos)

  Read a socket sended vote.
- int read_epoch_block (int fd)

  Read a socket sended epoch block.
- int read_get_pending_transaction (int fd)

  Get a socket sended pending transaction.
- int read_send_pending_transaction (int fd, infos_st ∗infos)

  Read a socket sended pending transaction.
- int read_send_pending_transaction_list (int fd, infos_st ∗infos)

  Read a socket sended pending transaction list.
- int epoch_validation_process (int blockfile, size_t height, int id)

  Epoch validation protocol.

9.15.1 Function Documentation

9.15.1.1 epoch_validation_process()

```
int epoch_validation_process (
            int blockfile,
            size_t height,
            int id )
```

Epoch validation protocol.

Parameters

blockfile	The epoch FD
height	The epoch height
id	The epoch ID

Returns

int

Definition at line 482 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.2 fetch_client_list()

```
int fetch_client_list (
            char who,
            int fd )
```

Fetches the client list from a socket fd.

Parameters

who	Tells if it is the server or the client side
fd	The socket fd

Returns

0 if sucess, -1 otherwise

Definition at line 107 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.3 read_actual_height()

```
size_t read_actual_height (
            int fd )
```

Get the actual height of a node via its sock fd.

Parameters

fd	The sock fd

Returns

size_t

Definition at line 186 of file get_data.c.

Here is the caller graph for this function:

9.15.1.4 read_epoch_block()

```
int read_epoch_block (
            int fd )
```

Read a socket sended epoch block.

Parameters

fd	The socket fd

Returns

int

Definition at line 420 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.5 read_get_blocks()

```
int read_get_blocks (
            int fd,
            infos_st * infos )
```

Read blocks from a sock fd.

Parameters

fd	The sock fd
infos	Shared information

Returns

int

Definition at line 155 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.6 read_get_pending_transaction()

```
int read_get_pending_transaction (
            int fd )
```

Get a socket sended pending transaction.

Parameters

fd	The socket fd

Returns

int

Definition at line 629 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.7 read_header()

```
size_t read_header (
            int sockfd,
            infos_st * infos )
```

Waits a header in 'sockfd', reads it and processes it.

Parameters

sockfd	The sock FD
infos	Shared information

Returns

0 if sucess, -1 otherwise

Definition at line 136 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.8 read_send_block()

```
int read_send_block (
            int fd )
```

Read a socket sended block.

Parameters

fd	The socket fd

Returns

int

Definition at line 193 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.9 read_send_pending_transaction()

```
int read_send_pending_transaction (
            int fd,
            infos_st * infos )
```

Read a socket sended pending transaction.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 571 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.10 read_send_pending_transaction_list()

```
int read_send_pending_transaction_list (
             int fd,
             infos_st * infos )
```

Read a socket sended pending transaction list.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 549 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.15.1.11 read_vote()

```
int read_vote (
             int fd,
             infos_st * infos )
```

Read a socket sended vote.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 279 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.16 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h File Reference

```
#include <pthread.h>
#include <semaphore.h>
#include <stdint.h>
```

Include dependency graph for network.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct Neighbour
- struct Node
- struct connection
- struct infos_st
- struct th_arg

Macros

- #define SIZE_OF_HOSTNAME 39
- #define NB_HARD_CODED_ADDR 2
- #define MAX_CONNECTION 5
- #define STATIC_PORT "4242"
- #define P_VERSION 0
- #define IM_SERVER 0
- #define IM_CLIENT 1
- #define MAX_NEIGHBOURS 64
- #define NODESERVER 0
- #define DOORSERVER 1
- #define MAX_SERVER 20
- #define MAX_VALIDATORS_PER_BLOCK 512
- #define SOL_TCP 6
- #define TCP_USER_TIMEOUT 18
- #define HD_GET_CLIENT_LIST "GET CLIENT LIST\r\n\r\n"
- #define HD_SEND_CLIENT_LIST "SEND CLIENT LIST\r\n\r\n"
- #define HD_CONNECTION_TO_NETWORK "CONNECTION TO NETWORK\r\n\r\n"
- #define HD_CONNECTION_TO_NODE "CONNECTION TO NODE\r\n\r\n"
- #define HD_GET_BLOCKS "GET BLOCKS\r\n\r\n"
- #define HD_ACTUAL_HEIGHT "ACTUAL HEIGHT\r\n\r\n"
- #define HD_SEND_BLOCK "SEND BLOCK\r\n\r\n"
- #define HD_GET_PENDING_TRANSACTION_LIST "GET PENDING TRANSACTION LIST\r\n\r\n"
- #define HD_SEND_PENDING_TRANSACTION_LIST "SEND PENDING TRANSACTION LIST\r\n\r\n"

- #define HD_REJECT_DEMAND "REJECT DEMAND\r\n\r\n"
- #define HD_GET_PENDING_TRANSACTION "GET PENDING TRANSACTION\r\n\r\n"
- #define HD_SEND_PENDING_TRANSACTION "SEND PENDING TRANSACTION\r\n\r\n"
- #define HD_SEND_EPOCH_BLOCK "SEND EPOCH BLOCK\r\n\r\n"
- #define HD_SEND_VOTE "SEND VOTE\r\n\r\n"
- #define DD_GET_HEIGHT 1
- #define DD_GET_BLOCKS 2
- #define DD_SEND_TRANSACTION 3
- #define DD_GET_TRANSACTION_LIST 4
- #define DD_SEND_VOTE 5
- #define DD_SEND_EPOCH 6
- #define SERVERMSG printf("\033[0;31m[S]:\033[0m ");
- #define CLIENTMSG printf("\033[0;34m[C]:\033[0m ");
- #define MANAGERMSG printf("\033[0;32m[M]:\033[0m ");
- #define WARNINGMSG(x) printf("\033[0;35m[W]: %s\033[0m\n", x);

Typedefs

- typedef struct Neighbour Neighbour
- typedef struct Node Node
- typedef struct connection connection
- typedef struct infos_st infos_st
- typedef struct th_arg th_arg

Functions

- struct __attribute__ ((__packed__)) get_blocks_t

Variables

- const Neighbour HARD_CODED_ADDR [ ]
- get_blocks_t

9.16.1 Macro Definition Documentation

9.16.1.1 CLIENTMSG

```
#define CLIENTMSG printf("\033[0;34m[C]:\033[0m ");
```

Definition at line 99 of file network.h.

9.16.1.2 DD_GET_BLOCKS

```
#define DD_GET_BLOCKS 2
```

Definition at line 90 of file network.h.

9.16.1.3 DD_GET_HEIGHT

```
#define DD_GET_HEIGHT 1
```

Definition at line 89 of file network.h.

9.16.1.4 DD_GET_TRANSACTION_LIST

```
#define DD_GET_TRANSACTION_LIST 4
```

Definition at line 92 of file network.h.

9.16.1.5 DD_SEND_EPOCH

```
#define DD_SEND_EPOCH 6
```

Definition at line 94 of file network.h.

9.16.1.6 DD_SEND_TRANSACTION

```
#define DD_SEND_TRANSACTION 3
```

Definition at line 91 of file network.h.

9.16.1.7 DD_SEND_VOTE

```
#define DD_SEND_VOTE 5
```

Definition at line 93 of file network.h.

9.16.1.8 DOORSERVER

```
#define DOORSERVER 1
```

Definition at line 23 of file network.h.

9.16.1.9 HD_ACTUAL_HEIGHT

```
#define HD_ACTUAL_HEIGHT "ACTUAL HEIGHT\r\n\r\n"
```

Definition at line 78 of file network.h.

9.16.1.10 HD_CONNECTION_TO_NETWORK

```
#define HD_CONNECTION_TO_NETWORK "CONNECTION TO NETWORK\r\n\r\n"
```

Definition at line 75 of file network.h.

9.16.1.11 HD_CONNECTION_TO_NODE

```
#define HD_CONNECTION_TO_NODE "CONNECTION TO NODE\r\n\r\n"
```

Definition at line 76 of file network.h.

9.16.1.12 HD_GET_BLOCKS

```
#define HD_GET_BLOCKS "GET BLOCKS\r\n\r\n"
```

Definition at line 77 of file network.h.

9.16.1.13 HD_GET_CLIENT_LIST

```
#define HD_GET_CLIENT_LIST "GET CLIENT LIST\r\n\r\n"
```

Definition at line 73 of file network.h.

9.16.1.14 HD_GET_PENDING_TRANSACTION

```
#define HD_GET_PENDING_TRANSACTION "GET PENDING TRANSACTION\r\n\r\n"
```

Definition at line 83 of file network.h.

9.16.1.15 HD_GET_PENDING_TRANSACTION_LIST

```
#define HD_GET_PENDING_TRANSACTION_LIST "GET PENDING TRANSACTION LIST\r\n\r\n"
```

Definition at line 80 of file network.h.

9.16.1.16 HD_REJECT_DEMAND

```
#define HD_REJECT_DEMAND "REJECT DEMAND\r\n\r\n"
```

Definition at line 82 of file network.h.

9.16.1.17 HD_SEND_BLOCK

```
#define HD_SEND_BLOCK "SEND BLOCK\r\n\r\n"
```

Definition at line 79 of file network.h.

9.16.1.18 HD_SEND_CLIENT_LIST

```
#define HD_SEND_CLIENT_LIST "SEND CLIENT LIST\r\n\r\n"
```

Definition at line 74 of file network.h.

9.16.1.19 HD_SEND_EPOCH_BLOCK

```
#define HD_SEND_EPOCH_BLOCK "SEND EPOCH BLOCK\r\n\r\n"
```

Definition at line 85 of file network.h.

9.16.1.20 HD_SEND_PENDING_TRANSACTION

```
#define HD_SEND_PENDING_TRANSACTION "SEND PENDING TRANSACTION\r\n\r\n"
```

Definition at line 84 of file network.h.

9.16.1.21 HD_SEND_PENDING_TRANSACTION_LIST

```
#define HD_SEND_PENDING_TRANSACTION_LIST "SEND PENDING TRANSACTION LIST\r\n\r\n"
```

Definition at line 81 of file network.h.

9.16.1.22 HD_SEND_VOTE

```
#define HD_SEND_VOTE "SEND VOTE\r\n\r\n"
```

Definition at line 86 of file network.h.

9.16.1.23 IM_CLIENT

```
#define IM_CLIENT 1
```

Definition at line 18 of file network.h.

9.16.1.24 IM_SERVER

```
#define IM_SERVER 0
```

Definition at line 17 of file network.h.

9.16.1.25 MANAGERMSG

```
#define MANAGERMSG printf("\033[0;32m[M]:\033[0m ");
```

Definition at line 100 of file network.h.

9.16.1.26 MAX_CONNECTION

```
#define MAX_CONNECTION 5
```

Definition at line 11 of file network.h.

9.16.1.27 MAX_NEIGHBOURS

```
#define MAX_NEIGHBOURS 64
```

Definition at line 20 of file network.h.

9.16.1.28 MAX_SERVER

```
#define MAX_SERVER 20
```

Definition at line 25 of file network.h.

9.16.1.29 MAX_VALIDATORS_PER_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 27 of file network.h.

9.16.1.30 NB_HARD_CODED_ADDR

```
#define NB_HARD_CODED_ADDR 2
```

Definition at line 10 of file network.h.

9.16.1.31 NODESERVER

```
#define NODESERVER 0
```

Definition at line 22 of file network.h.

9.16.1.32 P_VERSION

```
#define P_VERSION 0
```

Definition at line 15 of file network.h.

9.16.1.33 SERVERMSG

```
#define SERVERMSG printf("\033[0;31m[S]:\033[0m ");
```

Definition at line 98 of file network.h.

9.16.1.34 SIZE_OF_HOSTNAME

```
#define SIZE_OF_HOSTNAME 39
```

Definition at line 9 of file network.h.

9.16.1.35 SOL_TCP

```
#define SOL_TCP 6
```

Definition at line 29 of file network.h.

9.16.1.36 STATIC_PORT

```
#define STATIC_PORT "4242"
```

Definition at line 13 of file network.h.

9.16.1.37 TCP_USER_TIMEOUT

```
#define TCP_USER_TIMEOUT 18
```

Definition at line 30 of file network.h.

9.16.1.38 WARNINGMSG

```
#define WARNINGMSG(
             x ) printf("\033[0;35m[W]:  %s\033[0m\n", x);
```

Definition at line 101 of file network.h.

9.16.2 Typedef Documentation

9.16.2.1 connection

```
typedef struct connection connection
```

9.16.2.2 infos_st

```
typedef struct infos_st infos_st
```

9.16.2.3 Neighbour

```
typedef struct Neighbour Neighbour
```

9.16.2.4 Node

```
typedef struct Node Node
```

9.16.2.5 th_arg

```
typedef struct th_arg th_arg
```

9.16.3 Function Documentation

9.16.3.1 __attribute__()

```
struct __attribute__ (
            (__packed__)  )
```

Definition at line 103 of file network.h.

9.16.4 Variable Documentation

9.16.4.1 get_blocks_t

```
get_blocks_t
```

Definition at line 108 of file network.h.

9.16.4.2 HARD_CODED_ADDR

```
const Neighbour HARD_CODED_ADDR[]
```

Definition at line 5 of file network.c.

9.17 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_data.h File Reference

```
#include "network/server.h"
#include <dirent.h>
#include <stdio.h>
```

Include dependency graph for send_data.h: This graph shows which files directly or indirectly include this file:

Functions

- int send_client_list (char who, int sockfd, char *sockip)

  Sends my client list to a node via 'sockfd'.
- void send_get_blocks (connection *cc)

  Sends get blocks.
- void send_actual_height (int fd, infos_st *infos)
- void send_reject_demand (int fd)
- void send_send_block (int fd, size_t height)
- void send_pending_transaction_list (int fd)
- void send_send_pending_transaction (int fd, time_t txid)
- void send_get_pending_transaction (int fd, time_t txid)
- void send_epoch_block (connection *cc)
- void send_vote_fd (connection *cc)

9.17.1 Function Documentation

9.17.1.1 send_actual_height()

```
void send_actual_height (
            int fd,
            infos_st * infos )
```

Definition at line 58 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.2 send_client_list()

```
int send_client_list (
            char who,
            int sockfd,
            char * sockip )
```

Sends my client list to a node via 'sockfd'.

Parameters

sockfd	The sock FD

Returns

0 if success, -1 otherwise

Definition at line 3 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.3 send_epoch_block()

```
void send_epoch_block (
            connection * cc )
```

Definition at line 173 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.4 send_get_blocks()

```
void send_get_blocks (
            connection * cc )
```

Sends get blocks.

Definition at line 52 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.5 send_get_pending_transaction()

```
void send_get_pending_transaction (
            int fd,
            time_t txid )
```

Definition at line 165 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.6 send_pending_transaction_list()

```
void send_pending_transaction_list (
            int fd )
```

Definition at line 104 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.7 send_reject_demand()

```
void send_reject_demand (
            int fd )
```

Definition at line 65 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.8 send_send_block()

```
void send_send_block (
            int fd,
            size_t height )
```

Definition at line 71 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.9 send_send_pending_transaction()

```
void send_send_pending_transaction (
            int fd,
            time_t txid )
```

Definition at line 127 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.17.1.10 send_vote_fd()

```
void send_vote_fd (
            connection * cc )
```

Definition at line 209 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.18 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h File Reference

```
#include <sys/socket.h>
#include <sys/types.h>
#include <semaphore.h>
#include <netdb.h>
#include <arpa/inet.h>
#include "blockchain/block.h"
#include "network/client.h"
#include "network/get_data.h"
#include "network/send_data.h"
#include "network/network.h"
#include "misc/safe.h"
```

Include dependency graph for server.h: This graph shows which files directly or indirectly include this file:

Functions

- void * init_server (void *args)

  Launches a server instance, connected to the peer-to-peer network 'hostname'.

9.18.1 Function Documentation

9.18.1.1 init_server()

```
void* init_server (
            void * args )
```

Launches a server instance, connected to the peer-to-peer network 'hostname'.

Parameters

type	Type of the server

Returns

0 if success, -1 otherwise

Definition at line 106 of file server.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.19 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/labels.h File Reference

```
#include <gtk/gtk.h>
#include <stdio.h>
#include <string.h>
#include <err.h>
#include <time.h>
```

Include dependency graph for labels.h: This graph shows which files directly or indirectly include this file:

Functions

- void change_label_text (GtkLabel *label, char *text)
- void add_new_blockinfo (size_t height, size_t transaction)

Variables

- GtkLabel * balance_1
- GtkLabel * balance_2
- GtkLabel * stake_label1
- GtkLabel * stake_label2
- GtkLabel * stake_label3
- GtkLabel * synchro_label
- GtkLabel * block_amount_label
- GtkLabel * connections_label
- GtkLabel * mempool_label

9.19.1 Function Documentation

9.19.1.1 add_new_blockinfo()

```
void add_new_blockinfo (
            size_t height,
            size_t transaction )
```

Definition at line 322 of file ui.c.

9.19.1.2 change_label_text()

```
void change_label_text (
            GtkLabel * label,
            char * text )
```

Definition at line 233 of file ui.c.

Here is the caller graph for this function:

9.19.2 Variable Documentation

9.19.2.1 balance_1

```
GtkLabel* balance_1
```

Definition at line 24 of file ui.c.

9.19.2.2 balance_2

```
GtkLabel* balance_2
```

Definition at line 25 of file ui.c.

9.19.2.3 block_amount_label

```
GtkLabel* block_amount_label
```

Definition at line 30 of file ui.c.

9.19.2.4 connections_label

```
GtkLabel* connections_label
```

Definition at line 31 of file ui.c.

9.19.2.5 mempool_label

```
GtkLabel* mempool_label
```

Definition at line 32 of file ui.c.

9.19.2.6 stake_label1

```
GtkLabel* stake_label1
```

Definition at line 26 of file ui.c.

9.19.2.7 stake_label2

```
GtkLabel* stake_label2
```

Definition at line 27 of file ui.c.

9.19.2.8 stake_label3

```
GtkLabel* stake_label3
```

Definition at line 28 of file ui.c.

9.19.2.9 synchro_label

```
GtkLabel* synchro_label
```

Definition at line 29 of file ui.c.

9.20 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h File Reference

```
#include <dirent.h>
#include <gtk/gtk.h>
#include <stdio.h>
#include <string.h>
#include <err.h>
#include <time.h>
#include "network/network.h"
#include "cryptosystem/rsa.h"
#include "cryptosystem/hash.h"
#include "blockchain/wallet.h"
#include "blockchain/block.h"
#include "client.h"
```

Include dependency graph for ui.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct blockinfo

Functions

- void * setup (void *args)

  Setups the gtk widgets for the GUI.
- gboolean on_main_window_delete (GtkWidget *widget, __attribute__((unused)) gpointer data)

  Destroys the window when it is closed.
- void on_main_window_destroy (__attribute((unused)) GtkWidget *widget, __attribute__((unused)) gpointer data)

  Quits GTK when the program ends.
- gboolean on_transaction_button_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Will be used when the transaction function is ready.
- gboolean on_invest_button1_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Opens the invest window.
- gboolean on_invest_button2_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Resets the entry in the invest window and closes it, will later be used for the invest function.
- gboolean on_recover_button1_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Opens the recover window.
- gboolean on_recover_button2_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Resets the entry in the recover window and closes it, will later be used for the recover function.
- gboolean on_add_contact_button1_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Opens the contact window.
- gboolean add_contact (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Adds a contact to the treeview if the entrys weren't empty, and closes the contact window.
- void change_label_text (GtkLabel *label, char *text)
- gboolean on_create_key_but1_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)
- gboolean on_create_key_but2_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)
- gboolean on_connect_but_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)
- void add_contacts_from_file (char *name, char *public_key)
- void load_contacts_from_file ()
- void add_contact_to_combobox (char *name)
- void update_labels ()
- void add_transaction_with_pkey (double amount, char *public_key, char *date)
- void add_transaction_with_contact (double amount, char *public_key, char *date)
- void add_transaction_from_file (double amount, char *public_key, char *date)
- void load_transaction_from_file ()
- char * get_public_key_from_contacts (const char *name)
- void add_new_blockinfo (size_t height, size_t transaction)
- void update_sync (size_t actual, size_t final)
- gboolean set_block_viewer_plus (GtkWidget *widget, GdkEventKey *event, gpointer user_data)
- gboolean set_block_viewer_minus (GtkWidget *widget, GdkEventKey *event, gpointer user_data)
- void set_block_viewer (int height)

Variables

- GtkLabel * balance_1
- GtkLabel * balance_2
- GtkLabel * stake_label1
- GtkLabel * stake_label2
- GtkLabel * stake_label3
- GtkLabel * synchro_label
- GtkLabel * block_amount_label
- GtkLabel * connections_label
- GtkLabel * mempool_label
- struct blockinfo blocksinfo [3]

9.20.1 Function Documentation

9.20.1.1 add_contact()

```
gboolean add_contact (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Adds a contact to the treeview if the entrys weren't empty, and closes the contact window.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

9.20.1.2 add_contact_to_combobox()

```
void add_contact_to_combobox (
            char * name )
```

Definition at line 624 of file ui.c.

Here is the caller graph for this function:

9.20.1.3 add_contacts_from_file()

```
void add_contacts_from_file (
            char * name,
            char * public_key )
```

Definition at line 632 of file ui.c.

Here is the caller graph for this function:

9.20.1.4 add_new_blockinfo()

```
void add_new_blockinfo (
            size_t height,
            size_t transaction )
```

Definition at line 322 of file ui.c.

9.20.1.5 add_transaction_from_file()

```
void add_transaction_from_file (
            double amount,
            char * public_key,
            char * date )
```

Definition at line 480 of file ui.c.

Here is the caller graph for this function:

9.20.1.6 add_transaction_with_contact()

```
void add_transaction_with_contact (
            double amount,
            char * public_key,
            char * date )
```

Definition at line 460 of file ui.c.

Here is the caller graph for this function:

9.20.1.7 add_transaction_with_pkey()

```
void add_transaction_with_pkey (
            double amount,
            char * public_key,
            char * date )
```

Definition at line 440 of file ui.c.

Here is the caller graph for this function:

9.20.1.8 change_label_text()

```
void change_label_text (
            GtkLabel * label,
            char * text )
```

Definition at line 233 of file ui.c.

Here is the caller graph for this function:

9.20.1.9 get_public_key_from_contacts()

```
char* get_public_key_from_contacts (
            const char * name )
```

Definition at line 667 of file ui.c.

Here is the caller graph for this function:

9.20.1.10 load_contacts_from_file()

```
void load_contacts_from_file ( )
```

Definition at line 641 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.20.1.11 load_transaction_from_file()

```
void load_transaction_from_file ( )
```

9.20.1.12 on_add_contact_button1_press()

```
gboolean on_add_contact_button1_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Opens the contact window.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

9.20.1.13 on_connect_but_press()

```
gboolean on_connect_but_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

9.20.1.14 on_create_key_but1_press()

```
gboolean on_create_key_but1_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

9.20.1.15 on_create_key_but2_press()

```
gboolean on_create_key_but2_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

9.20.1.16 on_invest_button1_press()

```
gboolean on_invest_button1_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Opens the invest window.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean

9.20.1.17 on_invest_button2_press()

```
gboolean on_invest_button2_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Resets the entry in the invest window and closes it, will later be used for the invest function.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error Code

9.20.1.18 on_main_window_delete()

```
gboolean on_main_window_delete (
            GtkWidget * widget,
            __attribute__((unused)) gpointer data )
```

Destroys the window when it is closed.

Parameters

widget	The main window of the GUI

Returns

gboolean Error code

Definition at line 358 of file ui.c.

9.20.1.19 on_main_window_destroy()

```
void on_main_window_destroy (
            __attribute((unused)) GtkWidget * widget,
            __attribute__((unused)) gpointer data )
```

Quits GTK when the program ends.

9.20.1.20 on_recover_button1_press()

```
gboolean on_recover_button1_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Opens the recover window.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

9.20.1.21 on_recover_button2_press()

```
gboolean on_recover_button2_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Resets the entry in the recover window and closes it, will later be used for the recover function.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

9.20.1.22 on_transaction_button_press()

```
gboolean on_transaction_button_press (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

Will be used when the transaction function is ready.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

9.20.1.23 set_block_viewer()

```
void set_block_viewer (
            int height )
```

Definition at line 270 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.20.1.24 set_block_viewer_minus()

```
gboolean set_block_viewer_minus (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

9.20.1.25 set_block_viewer_plus()

```
gboolean set_block_viewer_plus (
            GtkWidget * widget,
            GdkEventKey * event,
            gpointer user_data )
```

9.20.1.26 setup()

```
void* setup (
            void * args )
```

Setups the gtk widgets for the GUI.

Returns

int Returns 1 if there is an error, 0 otherwise

Definition at line 80 of file ui.c.

Here is the caller graph for this function:

9.20.1.27 update_labels()

```
void update_labels ( )
```

Definition at line 796 of file ui.c.

Here is the call graph for this function:

9.20.1.28 update_sync()

```
void update_sync (
            size_t actual,
            size_t final )
```

Definition at line 339 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.20.2 Variable Documentation

9.20.2.1 balance_1

```
GtkLabel* balance_1
```

Definition at line 24 of file ui.c.

9.20.2.2 balance_2

```
GtkLabel* balance_2
```

Definition at line 25 of file ui.c.

9.20.2.3 block_amount_label

```
GtkLabel* block_amount_label
```

Definition at line 30 of file ui.c.

9.20.2.4 blocksinfo

```
struct blockinfo blocksinfo[3]
```

Definition at line 32 of file ui.h.

9.20.2.5 connections_label

```
GtkLabel* connections_label
```

Definition at line 31 of file ui.c.

9.20.2.6 mempool_label

```
GtkLabel* mempool_label
```

Definition at line 32 of file ui.c.

9.20.2.7 stake_label1

```
GtkLabel* stake_label1
```

Definition at line 26 of file ui.c.

9.20.2.8 stake_label2

```
GtkLabel* stake_label2
```

Definition at line 27 of file ui.c.

9.20.2.9 stake_label3

```
GtkLabel* stake_label3
```

Definition at line 28 of file ui.c.

9.20.2.10 synchro_label

```
GtkLabel* synchro_label
```

Definition at line 29 of file ui.c.

9.21 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/epoch_man.h File Reference

```
#include "blockchain/transaction.h"
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
#include "validation_engine.h"
#include "misc/bits.h"
#include "validators.h"
#include <openssl/rsa.h>
#include <dirent.h>
```

Include dependency graph for epoch_man.h: This graph shows which files directly or indirectly include this file:

Functions

- char * create_vote_data (Block *block, char vote, int validator_index, size_t *data_length)
- Block * create_epoch_block ()

  Create a block object with the previous block hash & votes.
- RSA * get_epoch_man_pkey (BlockData *block_data)

  Give the pkey of the creator of a block.
- void give_punishments_and_rewards (Block *prev_block, Block *current_block)

  Add punishmnent and reward transactions to validators of the 'prev_block' into 'current_block'.

9.21.1 Function Documentation

9.21.1.1 create_epoch_block()

```
Block* create_epoch_block ( )
```

Create a block object with the previous block hash & votes.

See also

The function create a block based on the local last block

Returns

Block∗

Definition at line 141 of file epoch_man.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.21.1.2 create_vote_data()

```
char* create_vote_data (
            Block * block,
            char vote,
            int validator_index,
            size_t * data_length )
```

Definition at line 10 of file epoch_man.c.

9.21.1.3 get_epoch_man_pkey()

```
RSA* get_epoch_man_pkey (
            BlockData * block_data )
```

Give the pkey of the creator of a block.

Parameters

block_data	The created block data

Returns

RSA∗, NULL if the data is corrupted

Definition at line 3 of file epoch_man.c.

Here is the caller graph for this function:

9.21.1.4 give_punishments_and_rewards()

```
void give_punishments_and_rewards (
            Block * prev_block,
            Block * current_block )
```

Add punishmnent and reward transactions to validators of the 'prev_block' into 'current_block'.

See also

Number of added transactions = number of validators in 'prev_block'

Parameters

prev_block	The last validated block
current_block	The current block (in creation)

Definition at line 31 of file epoch_man.c.

Here is the caller graph for this function:

9.22 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/plebe.h File Reference

```
#include "blockchain/block.h"
#include "validation/validation_engine.h"
```

Include dependency graph for plebe.h: This graph shows which files directly or indirectly include this file:

Functions

- int plebe_adhere_block (Block *block)

  Adhere a block, write it locally.

9.22.1 Function Documentation

9.22.1.1 plebe_adhere_block()

```
int plebe_adhere_block (
            Block * block )
```

Adhere a block, write it locally.

Parameters

block	The block to adhere

Returns

0 if success, 2 if need to sync error, 1 if data corrupted error

Definition at line 7 of file plebe.c.

Here is the call graph for this function:

9.23 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validation_engine.h File Reference

```
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/rsa.h"
#include "cryptosystem/hash.h"
#include "network/get_data.h"
#include "misc/math.h"
#include "misc/files.h"
#include "misc/bits.h"
#include "misc/safe.h"
#include <string.h>
#include <openssl/bio.h>
#include <stdbool.h>
#include <openssl/evp.h>
```

Include dependency graph for validation_engine.h: This graph shows which files directly or indirectly include this file:

Macros

- #define VERIDCT_NO 0
- #define VERIDCT_YES 1

Functions

- int send_verdict (Block *block, char verdict)

  Broadcast a verdict about a block validity to the network.
- Transaction ** validate_transactions (Transaction **transaction_to_validate, size_t nb_transactions, size_t *nb_returned_transactions)

  Validate some transactions.
- int comital_validate_block (Block *block)

  For the comital, check block validity.
- char plebe_verify_block (Block *block)

  For the plèbe, check block validity.

Variables

- connection * client_connections

9.23.1 Macro Definition Documentation

9.23.1.1 VERIDCT_NO

```
#define VERIDCT_NO 0
```

Definition at line 19 of file validation_engine.h.

9.23.1.2 VERIDCT_YES

```
#define VERIDCT_YES 1
```

Definition at line 20 of file validation_engine.h.

9.23.2 Function Documentation

9.23.2.1 comital_validate_block()

```
int comital_validate_block (
            Block * block )
```

For the comital, check block validity.

Parameters

block	The block to check

Returns

int

Definition at line 242 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.23.2.2 plebe_verify_block()

```
char plebe_verify_block (
            Block * block )
```

For the plèbe, check block validity.

Parameters

block	The block to check

Returns

int

Definition at line 199 of file validation_engine.c.

Here is the caller graph for this function:

9.23.2.3 send_verdict()

```
int send_verdict (
            Block * block,
            char verdict )
```

Broadcast a verdict about a block validity to the network.

Parameters

block	The block awaiting validation
verdict	The verdict : 0 -> "SHAME ! The block is not valid at all", 1 -> "The block is valid for me"

Returns

0 if the broadcast suceed, -1 if not

Definition at line 305 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.23.2.4 validate_transactions()

```
Transaction** validate_transactions (
            Transaction ** transaction_to_validate,
            size_t nb_transactions,
            size_t * nb_returned_transactions )
```

Validate some transactions.

See also

The verification must take into account :

- Sender != receiver
- If the sender signature is correct
- If the sender exists in the blockchain and has enough money
- If the receiver exists
- If sender and receiver remaining money fields are correct

Parameters

transaction_to_validate	The transactions to validate
nb_transactions	The number of transactions to validate
nb_returned_transactions	The number of returned (valid) transactions

Returns

Transaction**, the valid transactions

Definition at line 3 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.23.3 Variable Documentation

9.23.3.1 client_connections

```
connection* client_connections
```

Definition at line 4 of file client.c.

9.24 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h File Reference

```
#include <stdlib.h>
#include <string.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include "cryptosystem/hash.h"
#include "cryptosystem/rsa.h"
#include "misc/files.h"
#include "misc/safe.h"
#include "misc/math.h"
```

Include dependency graph for validators.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct validators_state_header
- struct validators_state_item

Macros

- #define MAX_VALIDATORS_PER_BLOCK 512

Functions

- void init_validators_state ()

  Init the `validators.state` file if it doesn't exists.
- RSA ** get_comittee (size_t block_height, int *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- RSA ** get_next_comittee (int *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- ssize_t get_validators_states_total_stake ()

  Get the total stake of the network (parse 'validators.state')
- ssize_t get_validators_states_nb_validators ()

  Get the number of validators of the network (parse 'validators.state')
- ssize_t get_validators_states_block_height_validity ()

  Get the validators states block height validity (parse 'validators.state')
- ssize_t get_validator_stake (size_t validator_id)

  Get a validator total stake (parse 'validators.state')
- ssize_t get_validator_power (size_t validator_id)

  Get a validator power (parse 'validators.state')
- RSA * get_validator_pkey (size_t validator_id)

  Get the validator pkey as RSA (parse 'validators.state')*
- ssize_t get_validator_id (RSA *pkey)

  Get the validator id in 'validators.state'.
- int i_am_commitee_member ()

  Check if the current user is a member of the next comitee.
- char update_validators_state (Block *block)

  Given a block, update the 'validators.state' with the transactions.

9.24.1 Macro Definition Documentation

9.24.1.1 MAX_VALIDATORS_PER_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 28 of file validators.h.

9.24.2 Function Documentation

9.24.2.1 get_comittee()

```
RSA** get_comittee (
            size_t block_height,
            int * nb_validators )
```

Get the a comittee RSA public keys on a specific epoch.

Parameters

block_height	The height of the block you want a comitte from
nb_validators	return value, the number of selected validators

See also

The 'next block' is referring to block after the last block available OFFLINE

Returns

[*RSA]

Definition at line 46 of file validators.c.

9.24.2.2 get_next_comittee()

```
RSA** get_next_comittee (
            int * nb_validators )
```

Get the a comittee RSA public keys on a specific epoch.

Parameters

nb_validators	return value, the number of selected validators

See also

The 'next block' is referring to block after the last block available OFFLINE

Returns

[*RSA]

Definition at line 135 of file validators.c.

Here is the caller graph for this function:

9.24.2.3 get_validator_id()

```
ssize_t get_validator_id (
            RSA * pkey )
```

Get the validator id in 'validators.state'.

Parameters

pkey	The RSA public key

Returns

ssize_t, the validator index

Definition at line 247 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.24.2.4 get_validator_pkey()

```
RSA* get_validator_pkey (
            size_t validator_id )
```

Get the validator pkey as RSA* (parse 'validators.state')

Parameters

validator_id	The id of the validator in 'validators.state'

Returns

RSA*

Definition at line 216 of file validators.c.

Here is the call graph for this function:

9.24.2.5 get_validator_power()

```
ssize_t get_validator_power (
            size_t validator_id )
```

Get a validator power (parse 'validators.state')

Parameters

validator_id	The id of the validator in 'validators.state'

Returns

ssize_t

Definition at line 199 of file validators.c.

Here is the call graph for this function:

9.24.2.6 get_validator_stake()

```
ssize_t get_validator_stake (
            size_t validator_id )
```

Get a validator total stake (parse 'validators.state')

Parameters

validator↩ _id	The id of the validator in 'validators.state'

Returns

ssize_t

Definition at line 182 of file validators.c.

Here is the call graph for this function:

9.24.2.7 get_validators_states_block_height_validity()

```
ssize_t get_validators_states_block_height_validity ( )
```

Get the validators states block height validity (parse 'validators.state')

Returns

ssize_t

Definition at line 168 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.24.2.8 get_validators_states_nb_validators()

```
ssize_t get_validators_states_nb_validators ( )
```

Get the number of validators of the network (parse 'validators.state')

Returns

ssize_t

Definition at line 154 of file validators.c.

Here is the call graph for this function:

9.24.2.9 get_validators_states_total_stake()

```
ssize_t get_validators_states_total_stake ( )
```

Get the total stake of the network (parse 'validators.state')

Returns

ssize_t

Definition at line 140 of file validators.c.

Here is the call graph for this function:

9.24.2.10 i_am_commitee_member()

```
int i_am_commitee_member ( )
```

Check if the current user is a member of the next comitee.

Returns

The id in the comittee, -1 if you are not member of the comittee

Definition at line 281 of file validators.c.

Here is the caller graph for this function:

9.24.2.11 init_validators_state()

```
void init_validators_state ( )
```

Init the `validators.state` file if it doesn't exists.

Definition at line 33 of file validators.c.

Here is the caller graph for this function:

9.24.2.12 update_validators_state()

```
char update_validators_state (
            Block * block )
```

Given a block, update the 'validators.state' with the transactions.

Parameters

block	

Returns

0, -1 if the given block height is not 'validators.state' height + 1

Definition at line 333 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.25 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/P2P_PROTOCOL.md File Reference

9.26 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md File Reference

9.27 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c File Reference

```
#include "blockchain/block.h"
#include "client.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
#include "misc/safe.h"
#include "blockchain/transaction.h"
#include <openssl/rsa.h>
#include "ui/ui.h"
#include "blockchain/blockchain_header.h"
```

Include dependency graph for client.c:

Functions

- int main (int argc, char **argv)

Variables

- connection * client_connections
- infos_st * ac_infos

9.27.1 Function Documentation

9.27.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 18 of file client.c.

Here is the call graph for this function:

9.27.2 Variable Documentation

9.27.2.1 ac_infos

```
infos_st* ac_infos
```

Definition at line 15 of file client.c.

9.27.2.2 client_connections

```
connection* client_connections
```

Definition at line 4 of file client.c.

9.28 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/client.c File Reference

```
#include "network/network.h"
#include "network/client.h"
```

Include dependency graph for client.c:

Functions

- Node * get_my_node (char who)

  Get the my node object.
- int set_neighbour (char who, char *hostname, int family)

  Sets a neighbour in the client.neightbours section.
- void remove_neighbour (char who, int index)

  Remove a neighbour in the client.neightbours section.
- void print_neighbours (char who, char mask)

  Print neighbours list.
- void save_neighbours (char who)

  Save neighbours list in .neighbours/neighbours.
- void load_neighbours (char who)

  Load neighbours list from .neighbours/neighbours.
- int is_in_neighbours (char who, char *hostname)

  Check if `hostname` is in `client.neightbours`
- int number_neighbours (char who)

  Return the nb of neighbour in the client.neightbours section.
- connection * listen_to (infos_st *infos, Neighbour neighbour, char *connection_type, connection *connection)

  Tries to connect to the peer-to-peer network via a node in the Node structure.
- int find_empty_connection (int max, connection *connections)

  Find if connection has any empty field.
- void * client_thread (void *args)

  Create a client thread.

Variables

- connection * client_connections = NULL

9.28.1 Function Documentation

9.28.1.1 client_thread()

```
void* client_thread (
            void * args )
```

Create a client thread.

Parameters

args	

Returns

void*

Definition at line 268 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.2 find_empty_connection()

```
int find_empty_connection (
            int max,
            connection * connection )
```

Find if connection has any empty field.

Parameters

max	The number of maximum connections
connection	The connection* buffer

Returns

int

Definition at line 258 of file client.c.

Here is the caller graph for this function:

9.28.1.3 get_my_node()

```
Node* get_my_node (
            char who )
```

Get the my node object.

Parameters

who	Tells if it is the server or the client side

Returns

Node*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

9.28.1.4 is_in_neighbours()

```
int is_in_neighbours (
            char who,
            char * hostname )
```

Check if `hostname` is in `client.neightbours`

Parameters

who	Tells if it is the server or the client side
hostname	The IP adress to check

Returns

int

Definition at line 149 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.5 listen_to()

```
connection* listen_to (
            infos_st * infos,
            Neighbour neighbour,
            char * connection_type,
            connection * connection )
```

Tries to connect to the peer-to-peer network via a node in the Node structure.

Parameters

infos	Some shared information
neighbour	The neighbour to connect with
connection_type	The type of connection
connection	The connection* structure

Returns

socket FD or -1 if an error occurs

Definition at line 172 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.6 load_neighbours()

```
void load_neighbours (
            char who )
```

Load neighbours list from .neighbours/neighbours.

Parameters

who	Tells if it is the server or the client side

Definition at line 113 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.7 number_neighbours()

```
int number_neighbours (
            char who )
```

Return the nb of neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side

Definition at line 160 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.8 print_neighbours()

```
void print_neighbours (
            char who,
            char mask )
```

Print neighbours list.

Parameters

who	Tells if it is the server or the client side
mask	

Definition at line 58 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.9 remove_neighbour()

```
void remove_neighbour (
            char who,
            int index )
```

Remove a neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side
index	The index of the neigbour to remove in client.neightbours

Definition at line 47 of file client.c.

Here is the call graph for this function:

9.28.1.10 save_neighbours()

```
void save_neighbours (
            char who )
```

Save neighbours list in .neighbours/neighbours.

Parameters

who	Tells if it is the server or the client side

Definition at line 74 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.1.11 set_neighbour()

```
int set_neighbour (
            char who,
            char * hostname,
            int family )
```

Sets a neighbour in the client.neightbours section.

Parameters

who	Tells if it is the server or the client side
hostname	The neighbour IP adress
family	The type of IP adress

Returns

0 if sucess, -1 otherwise if full

Definition at line 19 of file client.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.28.2 Variable Documentation

9.28.2.1 client_connections

```
connection* client_connections = NULL
```

Definition at line 4 of file client.c.

9.29 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/atrier.c File Reference

```
#include "blockchain/block.h"
#include "client.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
#include "misc/safe.h"
#include "blockchain/transaction.h"
#include <openssl/rsa.h>
#include "ui/ui.h"
#include "blockchain/blockchain_header.h"
```

Include dependency graph for atrier.c:

Functions

- infos_st * get_infos ()
- void update_pdt (int number)
- void move_file (char *src, char *dest)
- void Validate ()
- void new_transaction (char type, char *rc_pk, size_t amount, char cause[512], char asset[512])
- void join_network_door (infos_st *infos)
- void connection_to_others (infos_st *infos)
- size_t update_blockchain_height (infos_st *infos)
- void update_blockchain (infos_st *infos, size_t index_client)
- void clear_transactions ()
- void clear_epochs ()
- void update_pending_transactions_list ()

Variables

- connection * client_connections
- infos_st * ac_infos

9.29.1 Function Documentation

9.29.1.1 clear_epochs()

```
void clear_epochs ( )
```

Definition at line 335 of file atrier.c.

Here is the caller graph for this function:

9.29.1.2 clear_transactions()

```
void clear_transactions ( )
```

Definition at line 312 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.29.1.3 connection_to_others()

```
void connection_to_others (
            infos_st * infos )
```

Definition at line 228 of file atrier.c.

Here is the call graph for this function:

9.29.1.4 get_infos()

```
infos_st* get_infos ( )
```

Definition at line 16 of file atrier.c.

Here is the caller graph for this function:

9.29.1.5 join_network_door()

```
void join_network_door (
            infos_st * infos )
```

Definition at line 210 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.29.1.6 move_file()

```
void move_file (
            char * src,
            char * dest )
```

Definition at line 27 of file atrier.c.

Here is the call graph for this function:

9.29.1.7 new_transaction()

```
void new_transaction (
            char type,
            char * rc_pk,
            size_t amount,
            char cause[512],
            char asset[512] )
```

Definition at line 148 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.29.1.8 update_blockchain()

```
void update_blockchain (
            infos_st * infos,
            size_t index_client )
```

Definition at line 285 of file atrier.c.

9.29.1.9 update_blockchain_height()

```
size_t update_blockchain_height (
            infos_st * infos )
```

Definition at line 249 of file atrier.c.

Here is the call graph for this function:

9.29.1.10 update_pdt()

```
void update_pdt (
            int number )
```

Definition at line 20 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.29.1.11 update_pending_transactions_list()

```
void update_pending_transactions_list ( )
```

Definition at line 354 of file atrier.c.

Here is the call graph for this function:

9.29.1.12 Validate()

```
void Validate ( )
```

Definition at line 62 of file atrier.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.29.2 Variable Documentation

9.29.2.1 ac_infos

```
infos_st* ac_infos
```

Definition at line 14 of file atrier.c.

9.29.2.2 client_connections

```
connection* client_connections
```

Definition at line 4 of file client.c.

9.30 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c File Reference

```
#include "blockchain/block.h"
```
Include dependency graph for block.c:

Functions

- ChunkBlockchain * load_blockchain (size_t nb_chunk)

  Loads a blockchain object with a padding of 'nb_chunk'.
- ChunkBlockchain * load_last_blockchain ()

  Load the last local blockchain chunk.
- void write_block_file (Block block)

  Writes a block struct in a file.
- void convert_data_to_blockdata (BlockData *blockdata, int fd)
- void convert_data_to_block (Block *block, int fd)

  Convert serialized data to Block.*
- Block * get_block (size_t block_height)

  Get a block object.
- void free_block (Block *block)

  Free a block structure.
- Block * get_next_block (Block *block)

  For a block of height `h`, returns the block of height `h+1`
- Block * get_prev_block (Block *block)

  For a block of height `h`, return the block of height `h-1`
- char * get_blockdata_data (Block *block, size_t *size)

  Get the blockdata data object.
- void write_blockdata (BlockData blockdata, int fd)

  Writes blockdata in a file.
- void write_block (Block block, int fd)

  Writes a block in a file.
- void update_wallet_with_block (Block block)

  Update the Wallet structure with the transactions in a block.*
- void delete_epochs (size_t height)

  Delete specific epoches (draft blocks)
- Block * get_epoch (int id, size_t height)

  Get the epoch object.
- void clear_block (Block *block)

  Free block data, without deleting it structure.

9.30.1 Function Documentation

9.30.1.1 clear_block()

```
void clear_block (
            Block * block )
```

Free block data, without deleting it structure.

Parameters

block	The block to free

Definition at line 337 of file block.c.

Here is the caller graph for this function:

9.30.1.2 convert_data_to_block()

```
void convert_data_to_block (
            Block * block,
            int fd )
```

Convert serialized data to Block∗.

Parameters

block	The return Block∗
fd	The file descriptor where data are serialized

Definition at line 103 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.3 convert_data_to_blockdata()

```
void convert_data_to_blockdata (
            BlockData * blockdata,
            int fd )
```

Definition at line 70 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.4 delete_epochs()

```
void delete_epochs (
            size_t height )
```

Delete specific epoches (draft blocks)

Deprecated

Parameters

height	The height of the epochs

Definition at line 301 of file block.c.

Here is the caller graph for this function:

9.30.1.5 free_block()

```
void free_block (
            Block * block )
```

Free a block structure.

Parameters

block	The block to free

Definition at line 133 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.6 get_block()

```
Block* get_block (
            size_t block_height )
```

Get a block object.

Parameters

block_height	The height of the block

Returns

Block∗

Definition at line 111 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.7 get_blockdata_data()

```
char* get_blockdata_data (
            Block * block,
            size_t * size )
```

Get the blockdata data object.

Parameters

block	The block
size	The size of the block

Returns

char*

Definition at line 159 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.8 get_epoch()

```
Block* get_epoch (
            int id,
            size_t height )
```

Get the epoch object.

Parameters

id	The ID of the epoch
height	The height of the epoch

Returns

Block*

Definition at line 316 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.9 get_next_block()

```
Block* get_next_block (
            Block * block )
```

For a block of height `h`, returns the block of height `h+1`

Parameters

block	The base block

Returns

The next Block*

Definition at line 139 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.10 get_prev_block()

```
Block* get_prev_block (
            Block * block )
```

For a block of height `h`, return the block of height `h-1`

Parameters

block	The base block

Returns

The previous Block*

Definition at line 149 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.11 load_blockchain()

```
ChunkBlockchain* load_blockchain (
            size_t nb_chunk )
```

Loads a blockchain object with a padding of 'nb_chunk'.

Parameters

nb_chunk	The chunk nb, if 0 : return the current blockchain object without modification

Returns

ChunkBlockchain*, NULL if the ChunkBlockchain is empty after switching

Definition at line 3 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.12 load_last_blockchain()

```
ChunkBlockchain* load_last_blockchain ( )
```

Load the last local blockchain chunk.

Parameters

nb_chunk	

Returns

ChunkBlockchain∗, NULL if the ChunkBlockchain is empty after switching

Definition at line 47 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.13 update_wallet_with_block()

```
void update_wallet_with_block (
            Block block )
```

Update the Wallet∗ structure with the transactions in a block.

Parameters

block	The block to fetch update from

Definition at line 236 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.14 write_block()

```
void write_block (
            Block block,
            int fd )
```

Writes a block in a file.

Parameters

block	The block to write
fd	the file descriptor of the file in which the block is written

Definition at line 228 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.15 write_block_file()

```
void write_block_file (
            Block block )
```

Writes a block struct in a file.

Parameters

block	The block to write

Definition at line 52 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.30.1.16 write_blockdata()

```
void write_blockdata (
            BlockData blockdata,
            int fd )
```

Writes blockdata in a file.

Parameters

blockdata	The blockdata to write
fd	The file descriptor of the file in which the blockdata is written

Definition at line 196 of file block.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.31 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/blockchain_header.c File Reference

```
#include "blockchain/blockchain_header.h"
```
Include dependency graph for blockchain_header.c:

Functions

- void write_block_header (FILE *fd, Block *block, size_t height)
- void gen_blockchain_header (infos_st *infos)

  Generate block shared information.
- size_t get_receiver_remaining_money (infos_st *infos, RSA *receiver_public_key)

  Get the receiver remaining money.

9.31.1 Function Documentation

9.31.1.1 gen_blockchain_header()

```
void gen_blockchain_header (
            infos_st * infos )
```

Generate block shared information.

Deprecated

Parameters

infos	The information

Definition at line 9 of file blockchain_header.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.31.1.2 get_receiver_remaining_money()

```
size_t get_receiver_remaining_money (
            infos_st * infos,
            RSA * receiver_public_key )
```

Get the receiver remaining money.

Parameters

infos	Threads shared information
receiver_public_key	The RSA public key of the receiver

Returns

size_t

Definition at line 40 of file blockchain_header.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.31.1.3 write_block_header()

```
void write_block_header (
            FILE * fd,
            Block * block,
            size_t height )
```

Definition at line 3 of file blockchain_header.c.

Here is the caller graph for this function:

9.32 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/transaction.c File Reference

```
#include "blockchain/transaction.h"
```

Include dependency graph for transaction.c:

Functions

- void write_transactiondata (TransactionData *transaction, int fd)

  Serialize a TransactionData structure.*
- void write_transaction (Transaction *transaction, int fd)

  Serialize a Transaction structure.*
- void get_transaction_data (Transaction *trans, char **buff, size_t *index)

  Get the transaction data object.
- void convert_data_to_transactiondata (TransactionData *transactiondata, int fd)

  Convert serialized TransactionData to TransactionData*.*
- void load_transaction (Transaction *transaction, int fd)

  Load a serialized Transaction structure.*
- Transaction * load_pending_transaction (time_t timestamp)

  Load a transaction in the pending transaction (pdt) directory.
- void add_pending_transaction (Transaction *transaction)

  Add a transaction to the pending transaction (pdt) directory.
- Transaction create_new_transaction (infos_st *infos, char type, RSA *receiver_public_key, size_t amount, char cause[512], char asset[512])

  Create a new transaction.
- void flush_pending_transactions (Transaction **transactions, size_t nb_transactions)

  Delete block transactions in the pending transaction (pdt) directory if the block is valid.

9.32.1 Function Documentation

9.32.1.1 add_pending_transaction()

```
void add_pending_transaction (
            Transaction * transaction )
```

Add a transaction to the pending transaction (pdt) directory.

Parameters

transaction	The transaction to add

Definition at line 140 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.32.1.2 convert_data_to_transactiondata()

```
void convert_data_to_transactiondata (
            TransactionData * transactiondata,
            int fd )
```

Convert serialized TransactionData* to TransactionData*.

Parameters

transactiondata	The returned TransactionData*
fd	The serialized TransactionData FD

Definition at line 88 of file transaction.c.

Here is the caller graph for this function:

9.32.1.3 create_new_transaction()

```
Transaction create_new_transaction (
            infos_st * infos,
            char type,
            RSA * receiver_public_key,
            size_t amount,
            char cause[512],
            char asset[512] )
```

Create a new transaction.

Parameters

infos	Shared information object
type	The type of transaction
receiver_public_key	The receiver pkey
amount	The amount of PEPITAS
cause	The cause (deprecated)
asset	The asset (deprecated)

Returns

Transaction

Definition at line 157 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.32.1.4 flush_pending_transactions()

```
void flush_pending_transactions (
            Transaction ** transactions,
            size_t nb_transactions )
```

Delete block transactions in the pending transaction (pdt) directory if the block is valid.

Parameters

transactions	block.blockdata.transactions
nb_transactions	number of transactions

Definition at line 204 of file transaction.c.

9.32.1.5 get_transaction_data()

```
void get_transaction_data (
            Transaction * trans,
            char ** buff,
            size_t * index )
```

Get the transaction data object.

Converts transactions to char * buffer.

Parameters

trans	The returned transaction
buff	The buffer with the serialized data
index	The buffer starting offset

Definition at line 40 of file transaction.c.

Here is the caller graph for this function:

9.32.1.6 load_pending_transaction()

```
Transaction* load_pending_transaction (
            time_t timestamp )
```

Load a transaction in the pending transaction (pdt) directory.

Parameters

timestamp	The timestamp of the transaction

Returns

Transaction*

Definition at line 127 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.32.1.7 load_transaction()

```
void load_transaction (
            Transaction * transaction,
            int fd )
```

Load a serialized Transaction* structure.

Parameters

transaction	The returned Transaction*
fd	The serialized Transaction FD

Definition at line 117 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.32.1.8 write_transaction()

```
void write_transaction (
            Transaction * transaction,
            int fd )
```

Serialize a Transaction* structure.

Parameters

transaction	The Transaction* structure to serialize
fd	The output file FD

Definition at line 34 of file transaction.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.32.1.9 write_transactiondata()

```
void write_transactiondata (
            TransactionData * transaction,
            int fd )
```

Serialize a TransactionData* structure.

Parameters

transaction	The TransactionData* structure to serialize
fd	The output file FD

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

9.33 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c File Reference

```
#include <time.h>
#include "blockchain/wallet.h"
```

```
#include "cryptosystem/rsa.h"
```

Include dependency graph for wallet.c:

Functions

- Wallet * get_my_wallet ()

  Get my wallet object.
- int create_account ()

  Creates an account in local and broadcasts the creation to the network.
- void add_money_to_wallet (size_t money)

  Add money to my wallet.
- void remove_money_from_wallet (size_t money)

  Remove money from my wallet.
- void add_money_to_stake (size_t money)

  Add money to my stake.
- void remove_money_from_stake (size_t money)

  Withdraw money from my stake.

9.33.1 Function Documentation

9.33.1.1 add_money_to_stake()

```
void add_money_to_stake (
            size_t money )
```

Add money to my stake.

Parameters

money	The amount of PEPITAS

Definition at line 45 of file wallet.c.

Here is the call graph for this function:

9.33.1.2 add_money_to_wallet()

```
void add_money_to_wallet (
            size_t money )
```

Add money to my wallet.

Parameters

money	The amount of PEPITAS

Definition at line 26 of file wallet.c.

Here is the call graph for this function:

9.33.1.3 create_account()

```
int create_account ( )
```

Creates an account in local and broadcasts the creation to the network.

Returns

0 if the broadcast succeeds, otherwise 1

Definition at line 18 of file wallet.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.33.1.4 get_my_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

Returns

Wallet

Definition at line 6 of file wallet.c.

Here is the caller graph for this function:

9.33.1.5 remove_money_from_stake()

```
void remove_money_from_stake (
            size_t money )
```

Withdraw money from my stake.

Parameters

money	The amount of PEPITAS

Definition at line 54 of file wallet.c.

Here is the call graph for this function:

9.33.1.6 remove_money_from_wallet()

```
void remove_money_from_wallet (
            size_t money )
```

Remove money from my wallet.

Parameters

money	The amount of PEPITAS

Definition at line 34 of file wallet.c.

Here is the call graph for this function:

9.34 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/hash.c File Reference

```
#include <openssl/sha.h>
#include "cryptosystem/hash.h"
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
```

Include dependency graph for hash.c:

Functions

- char * sha384_data (void *data, size_t len_data)

  Apply the SHA384 algorithm on a 'data' of size 'len_data'.
- char * hash_block_transactions (Block *block)

  Apply the SHA384 to all block transactions.

9.34.1 Function Documentation

9.34.1.1 hash_block_transactions()

```
char* hash_block_transactions (
            Block * block )
```

Apply the SHA384 to all block transactions.

Parameters

block	The block to deal with

Returns

sha384[SHA384_DIGEST_LENGTH]

Definition at line 24 of file hash.c.

Here is the call graph for this function:

9.34.1.2 sha384_data()

```
char* sha384_data (
            void * data,
            size_t len_data )
```

Apply the SHA384 algorithm on a 'data' of size 'len_data'.

Parameters

data	The buffer to hash
len_data	The length of the buffer

Returns

char[97] (on heap)

Definition at line 6 of file hash.c.

Here is the caller graph for this function:

9.35 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/rsa.c File Reference

```
#include "cryptosystem/rsa.h"
```
Include dependency graph for rsa.c:

Macros

- #define RSA_NUM_E 3

Functions

- void get_keys (__attribute__((unused)) char *password)

9.35.1 Macro Definition Documentation

9.35.1.1 RSA_NUM_E

```
#define RSA_NUM_E 3
```

Definition at line 2 of file rsa.c.

9.35.2 Function Documentation

9.35.2.1 get_keys()

```
void get_keys (
            __attribute__((unused)) char * password )
```

Definition at line 7 of file rsa.c.

Here is the call graph for this function:

9.36 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/signature.c File Reference

```
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/hash.h"
#include <openssl/bio.h>
#include <openssl/rsa.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
```

Include dependency graph for signature.c:

Functions

- char * sign_message (char *data, size_t len_data, void *buffer)
  buffer <- encrypt(SHA284(msg,len_data),wallet_priv_key)
- char * sign_message_with_key (char *data, size_t len_data, RSA *key, void *buffer)
  encrypt(SHA284(msg,len_data),key) buffer <- encrypt(SHA284(msg,len_data),key)
- int verify_signature (void *data, size_t data_len, char *signature, RSA *pub_key)
  Verifies if SHA384(data) == decrypt(signature,pub_key)
- int verify_block_signature (Block block)
  Verifies if a block signature is valid.
- int verify_transaction_signature (Transaction *transaction)
  Verifies if a transaction signature is valid.
- void sign_block (Block *block)
  Signs a block with my private key.
- void sign_block_with_key (Block *block, RSA *key)
  Signs a block.
- void sign_transaction (Transaction *transaction)
  Signs a transaction with my private key.
- void sign_transaction_with_key (Transaction *transaction, RSA *key)
  Signs a transaction.
- void sign_block_transactions (Block *block)
  Signs all transactions of a block with my private key.

9.36.1 Function Documentation

9.36.1.1 sign_block()

```
void sign_block (
            Block * block )
```

Signs a block with my private key.

Parameters

block	The block to sign

Definition at line 108 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.2 sign_block_transactions()

```
void sign_block_transactions (
            Block * block )
```

Signs all transactions of a block with my private key.

Parameters

block	The block to sign

Definition at line 138 of file signature.c.

Here is the call graph for this function:

9.36.1.3 sign_block_with_key()

```
void sign_block_with_key (
            Block * block,
            RSA * key )
```

Signs a block.

Parameters

block	The block to sign
key	The key to use for the signature

Definition at line 115 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.4 sign_message()

```
char* sign_message (
            char * data,
            size_t len_data,
            void * buffer )
```

buffer <- encrypt(SHA284(msg,len_data),wallet_priv_key)

If buffer == NULL, return a new allocated buffer

Parameters

data	The data to sign
len_data	The length of the data
buffer	The buffer to put signature into

Returns

char*

Definition at line 10 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.5 sign_message_with_key()

```
char* sign_message_with_key (
            char * data,
            size_t len_data,
            RSA * key,
            void * buffer )
```

encrypt(SHA284(msg,len_data),key) buffer <- encrypt(SHA284(msg,len_data),key)

If buffer == NULL, return a new allocated buffer

Parameters

data	The data to sign
len_data	The length of the data
key	The key to use for the signature
buffer	The buffer to put signature into

Returns

char*

Definition at line 34 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.6 sign_transaction()

```
void sign_transaction (
            Transaction * transaction )
```

Signs a transaction with my private key.

Parameters

transaction	The transaction to sign

Definition at line 122 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.7 sign_transaction_with_key()

```
void sign_transaction_with_key (
            Transaction * transaction,
            RSA * key )
```

Signs a transaction.

Parameters

transaction	The transaction to sign
key	The key to use for the signature

Definition at line 130 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.8 verify_block_signature()

```
int verify_block_signature (
            Block block )
```

Verifies if a block signature is valid.

Parameters

block	The block to verify

Returns

1 if valid, 0 otherwise

Definition at line 83 of file signature.c.

Here is the call graph for this function:

9.36.1.9 verify_signature()

```
int verify_signature (
            void * data,
            size_t data_len,
            char * signature,
            RSA * pub_key )
```

Verifies if SHA384(data) == decrypt(signature,pub_key)

Parameters

data	The buffer to verify
data_len	The length of the buffer
signature	The signature to compare with SHA384(data, len_data)
pub_key	The RSA public key used for the decryption

Returns

int

Definition at line 57 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.36.1.10 verify_transaction_signature()

```
int verify_transaction_signature (
            Transaction * transaction )
```

Verifies if a transaction signature is valid.

Parameters

transaction	The transaction to verify

Returns

1 if valid, 0 otherwise

Definition at line 95 of file signature.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.37 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/files.c File Reference

```
#include "misc/files.h"
#include <dirent.h>
#include <string.h>
#include <stdlib.h>
```
Include dependency graph for files.c:

Macros

- #define _GNU_SOURCE

Functions

- char * last_file_in_folder (char folder_path[])
  Return the last file (reverse alphabetical order) of a folder path.

9.37.1 Macro Definition Documentation

9.37.1.1 _GNU_SOURCE

```
#define _GNU_SOURCE
```

Definition at line 1 of file files.c.

9.37.2 Function Documentation

9.37.2.1 last_file_in_folder()

```
char* last_file_in_folder (
            char folder_path[] )
```

Return the last file (reverse alphabetical order) of a folder path.

Parameters

folder_path	The path of the folder

Returns

char∗, return NULL if any error, must be freed !

Definition at line 7 of file files.c.

9.38 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/safe.c File Reference

`#include "misc/safe.h"`
Include dependency graph for safe.c:

Functions

- int safe_write (int fd, const void ∗buf, ssize_t count)
  Writes safely to a file descriptor.
- int safe_send (int fd, const void ∗buf, ssize_t count)
  Send safely to a file descriptor.
- ssize_t safe_read (int fd, const void ∗∗buf, size_t ∗bufsize)
  Reads safely in a file descriptor until '\r\n\r\n'.
- ssize_t safe_fread (void ∗buffer, const size_t size, const size_t n, FILE ∗file)
  Calls 'fread' but safely !

9.38.1 Function Documentation

9.38.1.1 safe_fread()

```
ssize_t safe_fread (
            void * buffer,
            const size_t size,
            const size_t n,
            FILE * file )
```

Calls 'fread' but safely !

Parameters

buffer	The buffer to write on
size	The size of 1 read element
n	The number of elements to read
file	The IO FILE

Returns

ssize_t, -1 if error or the number of read items

Definition at line 58 of file safe.c.

Here is the caller graph for this function:

9.38.1.2 safe_read()

```
ssize_t safe_read (
            int fd,
            const void ** buf,
            size_t * bufsize )
```

Reads safely in a file descriptor until '\r\n\r\n'.

Parameters

fd	The file descriptor
buf	The buffer which contains the message

Returns

The number of byte the file 'fd', if -1 error

Definition at line 31 of file safe.c.

Here is the caller graph for this function:

9.38.1.3 safe_send()

```
int safe_send (
            int fd,
            const void * buf,
            ssize_t count )
```

Send safely to a file descriptor.

Parameters

fd	The file descriptor
buf	The buffer to write
count	The number of byte to write in fd

Returns

Error code

Definition at line 17 of file safe.c.

Here is the caller graph for this function:

9.38.1.4 safe_write()

```
int safe_write (
            int fd,
            const void * buf,
            ssize_t count )
```

Writes safely to a file descriptor.

Parameters

fd	The file descriptor
buf	The buffer to write
count	The number of byte to write in fd

Returns

Error code

Definition at line 3 of file safe.c.

Here is the caller graph for this function:

9.39 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/get_data.c File Reference

```
#include "network/get_data.h"
```
Include dependency graph for get_data.c:

Functions

- size_t process_header (char *header, int sockfd, infos_st *infos)
- int fetch_client_list (char who, int fd)

  Fetches the client list from a socket fd.
- size_t read_header (int sockfd, infos_st *infos)

  Waits a header in 'sockfd', reads it and processes it.
- int read_get_blocks (int fd, infos_st *infos)

  Read blocks from a sock fd.
- size_t read_actual_height (int fd)

  Get the actual height of a node via its sock fd.
- int read_send_block (int fd)

  Read a socket sended block.
- int read_vote (int fd, infos_st *infos)

  Read a socket sended vote.
- int read_epoch_block (int fd)

  Read a socket sended epoch block.
- int epoch_validation_process (int blockfile, size_t height, int id)

Epoch validation protocol.

- int read_send_pending_transaction_list (int fd, infos_st *infos)

  Read a socket sended pending transaction list.

- int read_send_pending_transaction (int fd, infos_st *infos)

  Read a socket sended pending transaction.

- int read_get_pending_transaction (int fd)

  Get a socket sended pending transaction.

9.39.1 Function Documentation

9.39.1.1 epoch_validation_process()

```
int epoch_validation_process (
            int blockfile,
            size_t height,
            int id )
```

Epoch validation protocol.

Parameters

blockfile	The epoch FD
height	The epoch height
id	The epoch ID

Returns

int

Definition at line 482 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.2 fetch_client_list()

```
int fetch_client_list (
            char who,
            int fd )
```

Fetches the client list from a socket fd.

Parameters

who	Tells if it is the server or the client side
fd	The socket fd

Returns

0 if sucess, -1 otherwise

Definition at line 107 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.3 process_header()

```
size_t process_header (
            char * header,
            int sockfd,
            infos_st * infos )
```

Definition at line 3 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.4 read_actual_height()

```
size_t read_actual_height (
            int fd )
```

Get the actual height of a node via its sock fd.

Parameters

fd	The sock fd

Returns

size_t

Definition at line 186 of file get_data.c.

Here is the caller graph for this function:

9.39.1.5 read_epoch_block()

```
int read_epoch_block (
            int fd )
```

Read a socket sended epoch block.

Parameters

fd	The socket fd

Returns

int

Definition at line 420 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.6 read_get_blocks()

```
int read_get_blocks (
            int fd,
            infos_st * infos )
```

Read blocks from a sock fd.

Parameters

fd	The sock fd
infos	Shared information

Returns

int

Definition at line 155 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.7 read_get_pending_transaction()

```
int read_get_pending_transaction (
            int fd )
```

Get a socket sended pending transaction.

Parameters

fd	The socket fd

Returns

int

Definition at line 629 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.8 read_header()

```
size_t read_header (
            int sockfd,
            infos_st * infos )
```

Waits a header in 'sockfd', reads it and processes it.

Parameters

sockfd	The sock FD
infos	Shared information

Returns

0 if sucess, -1 otherwise

Definition at line 136 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.9 read_send_block()

```
int read_send_block (
            int fd )
```

Read a socket sended block.

Parameters

fd	The socket fd

Returns

int

Definition at line 193 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.10 read_send_pending_transaction()

```
int read_send_pending_transaction (
            int fd,
            infos_st * infos )
```

Read a socket sended pending transaction.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 571 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.11 read_send_pending_transaction_list()

```
int read_send_pending_transaction_list (
            int fd,
            infos_st * infos )
```

Read a socket sended pending transaction list.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 549 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.39.1.12 read_vote()

```
int read_vote (
            int fd,
            infos_st * infos )
```

Read a socket sended vote.

Parameters

fd	The socket fd
infos	Shared information

Returns

int

Definition at line 279 of file get_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/network.c File Reference

```
#include "network/client.h"
#include "network/network.h"
#include <arpa/inet.h>
```

Include dependency graph for network.c:

Variables

- const Neighbour HARD_CODED_ADDR [ ]

9.40.1 Variable Documentation

9.40.1.1 HARD_CODED_ADDR

```
const Neighbour HARD_CODED_ADDR[]
```

Initial value:

```
=
{
    {AF_INET, "34.72.117.116"},
    {AF_INET, "127.0.0.1"}
}
```

Definition at line 5 of file network.c.

9.41 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/send_data.c File Reference

```
#include "network/send_data.h"
```

Include dependency graph for send_data.c:

Functions

- int send_client_list (char who, int sockfd, char ∗sockip)

  Sends my client list to a node via 'sockfd'.
- void send_get_blocks (connection ∗cc)

  Sends get blocks.
- void send_actual_height (int fd, infos_st ∗infos)
- void send_reject_demand (int fd)
- void send_send_block (int fd, size_t height)
- void send_pending_transaction_list (int fd)
- void send_send_pending_transaction (int fd, time_t txid)
- void send_get_pending_transaction (int fd, time_t txid)
- void send_epoch_block (connection ∗cc)
- void send_vote_fd (connection ∗cc)

9.41.1 Function Documentation

9.41.1.1 send_actual_height()

```
void send_actual_height (
            int fd,
            infos_st * infos )
```

Definition at line 58 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.2 send_client_list()

```
int send_client_list (
            char who,
            int sockfd,
            char * sockip )
```

Sends my client list to a node via 'sockfd'.

Parameters

sockfd	The sock FD

Returns

0 if success, -1 otherwise

Definition at line 3 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.3 send_epoch_block()

```
void send_epoch_block (
            connection * cc )
```

Definition at line 173 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.4 send_get_blocks()

```
void send_get_blocks (
            connection * cc )
```

Sends get blocks.

Definition at line 52 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.5 send_get_pending_transaction()

```
void send_get_pending_transaction (
            int fd,
            time_t txid )
```

Definition at line 165 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.6 send_pending_transaction_list()

```
void send_pending_transaction_list (
            int fd )
```

Definition at line 104 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.7 send_reject_demand()

```
void send_reject_demand (
            int fd )
```

Definition at line 65 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.8 send_send_block()

```
void send_send_block (
            int fd,
            size_t height )
```

Definition at line 71 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.9 send_send_pending_transaction()

```
void send_send_pending_transaction (
            int fd,
            time_t txid )
```

Definition at line 127 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.41.1.10 send_vote_fd()

```
void send_vote_fd (
            connection * cc )
```

Definition at line 209 of file send_data.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.42 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/server.c File Reference

```
#include "network/server.h"
```
Include dependency graph for server.c:

Functions

- void * accept_connection (void *args)
- void * redirect_connection (void *arg)
- void * init_server (void *args)

  Launches a server instance, connected to the peer-to-peer network 'hostname'.

9.42.1 Function Documentation

9.42.1.1 accept_connection()

```
void* accept_connection (
            void * args )
```

Definition at line 3 of file server.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.42.1.2 init_server()

```
void* init_server (
            void * args )
```

Launches a server instance, connected to the peer-to-peer network 'hostname'.

Parameters

type	Type of the server

Returns

0 if success, -1 otherwise

Definition at line 106 of file server.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.42.1.3 redirect_connection()

```
void* redirect_connection (
            void * arg )
```

Definition at line 72 of file server.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.43 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/ui/ui.c File Reference

```
#include "ui/ui.h"
```

Include dependency graph for ui.c:

Functions

- void * setup (void *args)

  Setups the gtk widgets for the GUI.
- void change_label_text (GtkLabel *label, char *text)
- gboolean set_block_viewer_plus (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean set_block_viewer_minus (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- void set_block_viewer (int height)
- void add_new_blockinfo (size_t height, size_t transaction)
- void update_sync (size_t actual, size_t final)
- gboolean on_main_window_delete (GtkWidget *widget, __attribute__((unused)) gpointer data)

  Destroys the window when it is closed.
- void on_main_window_destroy (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) gpointer data)
- gboolean on_transaction_button_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- void add_transaction_with_pkey (double amount, char *public_key, char *date)
- void add_transaction_with_contact (double amount, char *public_key, char *date)
- void add_transaction_from_file (double amount, char *public_key, char *date)
- void load_transactions_from_file ()
- gboolean on_invest_button1_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_invest_button2_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_recover_button1_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_recover_button2_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_add_contact_button1_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean add_contact (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- void add_contact_to_combobox (char *name)
- void add_contacts_from_file (char *name, char *public_key)
- void load_contacts_from_file ()
- char * get_public_key_from_contacts (const char *name)
- gboolean on_create_key_but1_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_create_key_but2_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- gboolean on_connect_but_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- void update_labels ()

Variables

- GtkLabel * balance_1
- GtkLabel * balance_2
- GtkLabel * stake_label1
- GtkLabel * stake_label2
- GtkLabel * stake_label3
- GtkLabel * synchro_label

- GtkLabel * block_amount_label
- GtkLabel * connections_label
- GtkLabel * mempool_label
- GtkLabel * public_key_label
- GtkLabel * password_error_label
- GtkLabel * latest_block_name1
- GtkLabel * latest_block_name2
- GtkLabel * latest_block_name3
- GtkLabel * error_label
- GtkLabel * block_height_label
- GtkLabel * transa_number_label
- GtkLabel * total_transa_label
- GtkLabel * magic_label
- GtkLabel * prev_block_valid_label
- GtkLabel * nb_validators_label
- GtkLabel * block_error_label
- GtkLabel * block_time_label
- GtkLabel * validators_votes_label
- GtkEntry * transa_amount
- GtkEntry * recipient_key
- GtkEntry * asset_entry
- GtkEntry * cause_entry
- GtkEntry * invest_entry
- GtkEntry * recover_entry
- GtkEntry * name_entry_con
- GtkEntry * public_key_entry_con
- GtkEntry * password_entry1
- GtkEntry * password_entry2
- GtkEntry * key_entry
- GtkTreeView * tv_con
- GtkTreeStore * ts_con
- GtkTreeViewColumn * cx1_con
- GtkTreeViewColumn * cx2_con
- GtkCellRenderer * cr1_con
- GtkCellRenderer * cr2_con
- GtkTreeView * tv_th
- GtkTreeStore * ts_th
- GtkTreeViewColumn * cx1_th
- GtkTreeViewColumn * cx2_th
- GtkTreeViewColumn * cx3_th
- GtkCellRenderer * cr1_th
- GtkCellRenderer * cr2_th
- GtkCellRenderer * cr3_th
- GtkComboBox * contacts_combo
- GtkListStore * ls_combo
- GtkCellRenderer * cr1_combo
- GtkProgressBar * progress_bar_blockchain
- size_t block_height = 0

9.43.1 Function Documentation

9.43.1.1 add_contact()

```
gboolean add_contact (
             __attribute__((unused)) GtkWidget * widget,
             __attribute__((unused)) GdkEventKey * event,
             __attribute__((unused)) gpointer user_data )
```

Definition at line 595 of file ui.c.

Here is the call graph for this function:

9.43.1.2 add_contact_to_combobox()

```
void add_contact_to_combobox (
             char * name )
```

Definition at line 624 of file ui.c.

Here is the caller graph for this function:

9.43.1.3 add_contacts_from_file()

```
void add_contacts_from_file (
             char * name,
             char * public_key )
```

Definition at line 632 of file ui.c.

Here is the caller graph for this function:

9.43.1.4 add_new_blockinfo()

```
void add_new_blockinfo (
             size_t height,
             size_t transaction )
```

Definition at line 322 of file ui.c.

9.43.1.5 add_transaction_from_file()

```
void add_transaction_from_file (
             double amount,
             char * public_key,
             char * date )
```

Definition at line 480 of file ui.c.

Here is the caller graph for this function:

9.43.1.6 add_transaction_with_contact()

```
void add_transaction_with_contact (
            double amount,
            char * public_key,
            char * date )
```

Definition at line 460 of file ui.c.

Here is the caller graph for this function:

9.43.1.7 add_transaction_with_pkey()

```
void add_transaction_with_pkey (
            double amount,
            char * public_key,
            char * date )
```

Definition at line 440 of file ui.c.

Here is the caller graph for this function:

9.43.1.8 change_label_text()

```
void change_label_text (
            GtkLabel * label,
            char * text )
```

Definition at line 233 of file ui.c.

Here is the caller graph for this function:

9.43.1.9 get_public_key_from_contacts()

```
char* get_public_key_from_contacts (
            const char * name )
```

Definition at line 667 of file ui.c.

Here is the caller graph for this function:

9.43.1.10 load_contacts_from_file()

```
void load_contacts_from_file ( )
```

Definition at line 641 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.43.1.11 load_transactions_from_file()

```
void load_transactions_from_file ( )
```

Definition at line 490 of file ui.c.

Here is the call graph for this function:

9.43.1.12 on_add_contact_button1_press()

```
gboolean on_add_contact_button1_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 586 of file ui.c.

9.43.1.13 on_connect_but_press()

```
gboolean on_connect_but_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 746 of file ui.c.

Here is the call graph for this function:

9.43.1.14 on_create_key_but1_press()

```
gboolean on_create_key_but1_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 686 of file ui.c.

9.43.1.15 on_create_key_but2_press()

```
gboolean on_create_key_but2_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 701 of file ui.c.

Here is the call graph for this function:

9.43.1.16 on_invest_button1_press()

```
gboolean on_invest_button1_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 525 of file ui.c.

9.43.1.17 on_invest_button2_press()

```
gboolean on_invest_button2_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 534 of file ui.c.

Here is the call graph for this function:

9.43.1.18 on_main_window_delete()

```
gboolean on_main_window_delete (
            GtkWidget * widget,
            __attribute__((unused)) gpointer data )
```

Destroys the window when it is closed.

Parameters

widget	The main window of the GUI

Returns

gboolean Error code

Definition at line 358 of file ui.c.

9.43.1.19 on_main_window_destroy()

```
void on_main_window_destroy (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) gpointer data )
```

Definition at line 367 of file ui.c.

9.43.1.20 on_recover_button1_press()

```
gboolean on_recover_button1_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 555 of file ui.c.

9.43.1.21 on_recover_button2_press()

```
gboolean on_recover_button2_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 564 of file ui.c.

Here is the call graph for this function:

9.43.1.22 on_transaction_button_press()

```
gboolean on_transaction_button_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 374 of file ui.c.

Here is the call graph for this function:

9.43.1.23 set_block_viewer()

```
void set_block_viewer (
            int height )
```

Definition at line 270 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.43.1.24 set_block_viewer_minus()

```
gboolean set_block_viewer_minus (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 253 of file ui.c.

Here is the call graph for this function:

9.43.1.25 set_block_viewer_plus()

```
gboolean set_block_viewer_plus (
             __attribute__((unused)) GtkWidget * widget,
             __attribute__((unused)) GdkEventKey * event,
             __attribute__((unused)) gpointer user_data )
```

Definition at line 238 of file ui.c.

Here is the call graph for this function:

9.43.1.26 setup()

```
void* setup (
             void * args )
```

Setups the gtk widgets for the GUI.

Returns

int Returns 1 if there is an error, 0 otherwise

Definition at line 80 of file ui.c.

Here is the caller graph for this function:

9.43.1.27 update_labels()

```
void update_labels ( )
```

Definition at line 796 of file ui.c.

Here is the call graph for this function:

9.43.1.28 update_sync()

```
void update_sync (
             size_t actual,
             size_t final )
```

Definition at line 339 of file ui.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.43.2 Variable Documentation

9.43.2.1 asset_entry

```
GtkEntry* asset_entry
```

Definition at line 50 of file ui.c.

9.43.2.2 balance_1

```
GtkLabel* balance_1
```

Definition at line 24 of file ui.c.

9.43.2.3 balance_2

```
GtkLabel* balance_2
```

Definition at line 25 of file ui.c.

9.43.2.4 block_amount_label

```
GtkLabel* block_amount_label
```

Definition at line 30 of file ui.c.

9.43.2.5 block_error_label

```
GtkLabel* block_error_label
```

Definition at line 45 of file ui.c.

9.43.2.6 block_height

```
size_t block_height = 0
```

Definition at line 78 of file ui.c.

9.43.2.7 block_height_label

```
GtkLabel* block_height_label
```

Definition at line 39 of file ui.c.

9.43.2.8 block_time_label

```
GtkLabel* block_time_label
```

Definition at line 46 of file ui.c.

9.43.2.9 cause_entry

```
GtkEntry* cause_entry
```

Definition at line 51 of file ui.c.

9.43.2.10 connections_label

```
GtkLabel* connections_label
```

Definition at line 31 of file ui.c.

9.43.2.11 contacts_combo

```
GtkComboBox* contacts_combo
```

Definition at line 73 of file ui.c.

9.43.2.12 cr1_combo

```
GtkCellRenderer* cr1_combo
```

Definition at line 75 of file ui.c.

9.43.2.13 cr1_con

```
GtkCellRenderer* cr1_con
```

Definition at line 63 of file ui.c.

9.43.2.14 cr1_th

```
GtkCellRenderer* cr1_th
```

Definition at line 70 of file ui.c.

9.43.2.15 cr2_con

```
GtkCellRenderer* cr2_con
```

Definition at line 64 of file ui.c.

9.43.2.16 cr2_th

```
GtkCellRenderer* cr2_th
```

Definition at line 71 of file ui.c.

9.43.2.17 cr3_th

```
GtkCellRenderer* cr3_th
```

Definition at line 72 of file ui.c.

9.43.2.18 cx1_con

```
GtkTreeViewColumn* cx1_con
```

Definition at line 61 of file ui.c.

9.43.2.19 cx1_th

```
GtkTreeViewColumn* cx1_th
```

Definition at line 67 of file ui.c.

9.43.2.20 cx2_con

```
GtkTreeViewColumn* cx2_con
```

Definition at line 62 of file ui.c.

9.43.2.21 cx2_th

```
GtkTreeViewColumn* cx2_th
```

Definition at line 68 of file ui.c.

9.43.2.22 cx3_th

```
GtkTreeViewColumn* cx3_th
```

Definition at line 69 of file ui.c.

9.43.2.23 error_label

```
GtkLabel* error_label
```

Definition at line 38 of file ui.c.

9.43.2.24 invest_entry

```
GtkEntry* invest_entry
```

Definition at line 52 of file ui.c.

9.43.2.25 key_entry

```
GtkEntry* key_entry
```

Definition at line 58 of file ui.c.

9.43.2.26 latest_block_name1

```
GtkLabel* latest_block_name1
```

Definition at line 35 of file ui.c.

9.43.2.27 latest_block_name2

```
GtkLabel* latest_block_name2
```

Definition at line 36 of file ui.c.

9.43.2.28 latest_block_name3

```
GtkLabel* latest_block_name3
```

Definition at line 37 of file ui.c.

9.43.2.29 ls_combo

```
GtkListStore* ls_combo
```

Definition at line 74 of file ui.c.

9.43.2.30 magic_label

```
GtkLabel* magic_label
```

Definition at line 42 of file ui.c.

9.43.2.31 mempool_label

```
GtkLabel* mempool_label
```

Definition at line 32 of file ui.c.

9.43.2.32 name_entry_con

```
GtkEntry* name_entry_con
```

Definition at line 54 of file ui.c.

9.43.2.33 nb_validators_label

```
GtkLabel* nb_validators_label
```

Definition at line 44 of file ui.c.

9.43.2.34 password_entry1

```
GtkEntry* password_entry1
```

Definition at line 56 of file ui.c.

9.43.2.35 password_entry2

```
GtkEntry* password_entry2
```

Definition at line 57 of file ui.c.

9.43.2.36 password_error_label

```
GtkLabel* password_error_label
```

Definition at line 34 of file ui.c.

9.43.2.37 prev_block_valid_label

```
GtkLabel* prev_block_valid_label
```

Definition at line 43 of file ui.c.

9.43.2.38 progress_bar_blockchain

```
GtkProgressBar* progress_bar_blockchain
```

Definition at line 76 of file ui.c.

9.43.2.39 public_key_entry_con

```
GtkEntry* public_key_entry_con
```

Definition at line 55 of file ui.c.

9.43.2.40 public_key_label

```
GtkLabel* public_key_label
```

Definition at line 33 of file ui.c.

9.43.2.41 recipient_key

```
GtkEntry* recipient_key
```

Definition at line 49 of file ui.c.

9.43.2.42 recover_entry

```
GtkEntry* recover_entry
```

Definition at line 53 of file ui.c.

9.43.2.43 stake_label1

```
GtkLabel* stake_label1
```

Definition at line 26 of file ui.c.

9.43.2.44 stake_label2

```
GtkLabel* stake_label2
```

Definition at line 27 of file ui.c.

9.43.2.45 stake_label3

```
GtkLabel* stake_label3
```

Definition at line 28 of file ui.c.

9.43.2.46 synchro_label

```
GtkLabel* synchro_label
```

Definition at line 29 of file ui.c.

9.43.2.47 total_transa_label

```
GtkLabel* total_transa_label
```

Definition at line 41 of file ui.c.

9.43.2.48 transa_amount

```
GtkEntry* transa_amount
```

Definition at line 48 of file ui.c.

9.43.2.49 transa_number_label

```
GtkLabel* transa_number_label
```

Definition at line 40 of file ui.c.

9.43.2.50 ts_con

```
GtkTreeStore* ts_con
```

Definition at line 60 of file ui.c.

9.43.2.51 ts_th

```
GtkTreeStore* ts_th
```

Definition at line 66 of file ui.c.

9.43.2.52 tv_con

```
GtkTreeView* tv_con
```

Definition at line 59 of file ui.c.

9.43.2.53 tv_th

```
GtkTreeView* tv_th
```

Definition at line 65 of file ui.c.

9.43.2.54 validators_votes_label

```
GtkLabel* validators_votes_label
```

Definition at line 47 of file ui.c.

9.44 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/epoch_man.c File Reference

```
#include "validation/epoch_man.h"
```
Include dependency graph for epoch_man.c:

Functions

- RSA * get_epoch_man_pkey (BlockData *block_data)

  Give the pkey of the creator of a block.
- char * create_vote_data (Block *block, char vote, int validator_index, size_t *data_length)
- void give_punishments_and_rewards (Block *last_block, Block *current_block)

  Add punishmnent and reward transactions to validators of the 'prev_block' into 'current_block'.
- void add_pdt_to_block (Block *block)
- Block * create_epoch_block ()

  Create a block object with the previous block hash & votes.

9.44.1 Function Documentation

9.44.1.1 add_pdt_to_block()

```
void add_pdt_to_block (
            Block * block )
```

Definition at line 94 of file epoch_man.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.44.1.2 create_epoch_block()

```
Block* create_epoch_block ( )
```

Create a block object with the previous block hash & votes.

See also

The function create a block based on the local last block

Returns

Block*

Definition at line 141 of file epoch_man.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.44.1.3 create_vote_data()

```
char* create_vote_data (
            Block * block,
            char vote,
            int validator_index,
            size_t * data_length )
```

Definition at line 10 of file epoch_man.c.

9.44.1.4 get_epoch_man_pkey()

```
RSA* get_epoch_man_pkey (
            BlockData * block_data )
```

Give the pkey of the creator of a block.

Parameters

block_data	The created block data

Returns

RSA*, NULL if the data is corrupted

Definition at line 3 of file epoch_man.c.

Here is the caller graph for this function:

9.44.1.5 give_punishments_and_rewards()

```
void give_punishments_and_rewards (
            Block * prev_block,
            Block * current_block )
```

Add punishmnent and reward transactions to validators of the 'prev_block' into 'current_block'.

See also

Number of added transactions = number of validators in 'prev_block'

Parameters

prev_block	The last validated block
current_block	The current block (in creation)

Definition at line 31 of file epoch_man.c.

Here is the caller graph for this function:

9.45 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/plebe.c File Reference

`#include "validation/plebe.h"`
Include dependency graph for plebe.c:

Functions

- int plebe_adhere_block (Block *block)
  Adhere a block, write it locally.

9.45.1 Function Documentation

9.45.1.1 plebe_adhere_block()

```
int plebe_adhere_block (
            Block * block )
```

Adhere a block, write it locally.

Parameters

block	The block to adhere

Returns

0 if success, 2 if need to sync error, 1 if data corrupted error

Definition at line 7 of file plebe.c.

Here is the call graph for this function:

9.46 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validation_engine.c File Reference

`#include "validation/validation_engine.h"`
Include dependency graph for validation_engine.c:

Functions

- Transaction ** validate_transactions (Transaction **transaction_to_validate, size_t nb_transactions, size_t *nb_returned_transactions)

  Validate some transactions.
- char plebe_verify_block (Block *block)

  For the plèbe, check block validity.
- int comital_validate_block (Block *block)

  For the comital, check block validity.
- int send_verdict (Block *block, char verdict)

  Broadcast a verdict about a block validity to the network.

9.46.1 Function Documentation

9.46.1.1 comital_validate_block()

```
int comital_validate_block (
            Block * block )
```

For the comital, check block validity.

Parameters

block	The block to check

Returns

int

Definition at line 242 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.46.1.2 plebe_verify_block()

```
char plebe_verify_block (
            Block * block )
```

For the plèbe, check block validity.

Parameters

block	The block to check

Returns

int

Definition at line 199 of file validation_engine.c.

Here is the caller graph for this function:

9.46.1.3 send_verdict()

```
int send_verdict (
            Block * block,
            char verdict )
```

Broadcast a verdict about a block validity to the network.

Parameters

block	The block awaiting validation
verdict	The verdict : 0 -> "SHAME ! The block is not valid at all", 1 -> "The block is valid for me"

Returns

0 if the broadcast suceed, -1 if not

Definition at line 305 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.46.1.4 validate_transactions()

```
Transaction** validate_transactions (
            Transaction ** transaction_to_validate,
            size_t nb_transactions,
            size_t * nb_returned_transactions )
```

Validate some transactions.

See also

The verification must take into account :

- Sender != receiver
- If the sender signature is correct
- If the sender exists in the blockchain and has enough money
- If the receiver exists
- If sender and receiver remaining money fields are correct

Parameters

transaction_to_validate	The transactions to validate
nb_transactions	The number of transactions to validate
nb_returned_transactions	The number of returned (valid) transactions

Returns

Transaction**, the valid transactions

Definition at line 3 of file validation_engine.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.47 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validators.c File Reference

`#include "validation/validators.h"`
Include dependency graph for validators.c:

Macros

- #define NB_RSA_CHUNK 2048 / 64
- #define HEADER_VALIDATORS_STATE_SIZE 3 * sizeof(size_t) + sizeof(char) + (RSA_KEY_SIZE + 2 * sizeof(size_t) + sizeof(char)) * validator_id

Functions

- int define_nb_validators (size_t n)
- char * hash_block_transactions_epoch (Block *block)
- void init_validators_state ()

  Init the `validators.state` file if it doesn't exists.
- RSA ** get_comittee (size_t block_height, int *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- RSA ** get_next_comittee (int *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- ssize_t get_validators_states_total_stake ()

  Get the total stake of the network (parse 'validators.state')
- ssize_t get_validators_states_nb_validators ()

  Get the number of validators of the network (parse 'validators.state')
- ssize_t get_validators_states_block_height_validity ()

  Get the validators states block height validity (parse 'validators.state')
- ssize_t get_validator_stake (size_t validator_id)

  Get a validator total stake (parse 'validators.state')
- ssize_t get_validator_power (size_t validator_id)

  Get a validator power (parse 'validators.state')

- RSA * get_validator_pkey (size_t validator_id)

  Get the validator pkey as RSA (parse 'validators.state')*
- ssize_t get_validator_id (RSA *pkey)

  Get the validator id in 'validators.state'.
- int i_am_commitee_member ()

  Check if the current user is a member of the next comitee.
- ssize_t _create_validator_item (FILE *validators_states, struct validators_state_header *updated_validators_state_header, Transaction *transaction, bool is_key_on_sender)
- char update_validators_state (Block *block)

  Given a block, update the 'validators.state' with the transactions.

9.47.1 Macro Definition Documentation

9.47.1.1 HEADER_VALIDATORS_STATE_SIZE

```
#define HEADER_VALIDATORS_STATE_SIZE 3 * sizeof(size_t) + sizeof(char) + (RSA_KEY_SIZE + 2 * sizeof(size_t) + sizeof(char)) * validator_id
```

Definition at line 4 of file validators.c.

9.47.1.2 NB_RSA_CHUNK

```
#define NB_RSA_CHUNK 2048 / 64
```

Definition at line 3 of file validators.c.

9.47.2 Function Documentation

9.47.2.1 _create_validator_item()

```
ssize_t _create_validator_item (
            FILE * validators_states,
            struct validators_state_header * updated_validators_state_header,
            Transaction * transaction,
            bool is_key_on_sender )
```

Definition at line 296 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.47.2.2 define_nb_validators()

```
int define_nb_validators (
            size_t n )
```

Definition at line 6 of file validators.c.

9.47.2.3 get_comittee()

```
RSA** get_comittee (
            size_t block_height,
            int * nb_validators )
```

Get the a comittee RSA public keys on a specific epoch.

Parameters

block_height	The height of the block you want a comitte from
nb_validators	return value, the number of selected validators

See also

The 'next block' is referring to block after the last block available OFFLINE

Returns

[∗RSA]

Definition at line 46 of file validators.c.

9.47.2.4 get_next_comittee()

```
RSA** get_next_comittee (
            int * nb_validators )
```

Get the a comittee RSA public keys on a specific epoch.

Parameters

nb_validators	return value, the number of selected validators

See also

The 'next block' is referring to block after the last block available OFFLINE

Returns

[∗RSA]

Definition at line 135 of file validators.c.

Here is the caller graph for this function:

9.47.2.5 get_validator_id()

```
ssize_t get_validator_id (
            RSA * pkey )
```

Get the validator id in 'validators.state'.

Parameters

pkey	The RSA public key

Returns

ssize_t, the validator index

Definition at line 247 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.47.2.6 get_validator_pkey()

```
RSA* get_validator_pkey (
            size_t validator_id )
```

Get the validator pkey as RSA∗ (parse 'validators.state')

Parameters

validator↩ _id	The id of the validator in 'validators.state'

Returns

RSA∗

Definition at line 216 of file validators.c.

Here is the call graph for this function:

9.47.2.7 get_validator_power()

```
ssize_t get_validator_power (
            size_t validator_id )
```

Get a validator power (parse 'validators.state')

Parameters

validator↩ _id	The id of the validator in 'validators.state'

Returns

ssize_t

Definition at line 199 of file validators.c.

Here is the call graph for this function:

9.47.2.8 get_validator_stake()

```
ssize_t get_validator_stake (
            size_t validator_id )
```

Get a validator total stake (parse 'validators.state')

Parameters

validator↩ _id	The id of the validator in 'validators.state'

Returns

ssize_t

Definition at line 182 of file validators.c.

Here is the call graph for this function:

9.47.2.9 get_validators_states_block_height_validity()

```
ssize_t get_validators_states_block_height_validity ( )
```

Get the validators states block height validity (parse 'validators.state')

Returns

ssize_t

Definition at line 168 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.47.2.10 get_validators_states_nb_validators()

```
ssize_t get_validators_states_nb_validators ( )
```

Get the number of validators of the network (parse 'validators.state')

Returns

ssize_t

Definition at line 154 of file validators.c.

Here is the call graph for this function:

9.47.2.11 get_validators_states_total_stake()

```
ssize_t get_validators_states_total_stake ( )
```

Get the total stake of the network (parse 'validators.state')

Returns

ssize_t

Definition at line 140 of file validators.c.

Here is the call graph for this function:

9.47.2.12 hash_block_transactions_epoch()

```
char* hash_block_transactions_epoch (
            Block * block )
```

Definition at line 21 of file validators.c.

Here is the call graph for this function:

9.47.2.13 i_am_commitee_member()

```
int i_am_commitee_member ( )
```

Check if the current user is a member of the next comitee.

Returns

The id in the comittee, -1 if you are not member of the comittee

Definition at line 281 of file validators.c.

Here is the caller graph for this function:

9.47.2.14 init_validators_state()

```
void init_validators_state ( )
```

Init the `validators.state` file if it doesn't exists.

Definition at line 33 of file validators.c.

Here is the caller graph for this function:

9.47.2.15 update_validators_state()

```
char update_validators_state (
             Block * block )
```

Given a block, update the 'validators.state' with the transactions.

Parameters

block	

Returns

0, -1 if the given block height is not 'validators.state' height + 1

Definition at line 333 of file validators.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.48 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/genesis.c File Reference

```
#include "client.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
#include "misc/safe.h"
#include <openssl/rsa.h>
#include "blockchain/transaction.h"
#include "blockchain/block.h"
#include "ui/ui.h"
#include "blockchain/blockchain_header.h"
```

Include dependency graph for genesis.c:

Functions

- infos_st * get_infos ()
- void new_transaction (char type, char *rc_pk, size_t amount, char cause[512], char asset[512])
- int main ()

Variables

- connection * client_connections
- infos_st * ac_infos

9.48.1 Function Documentation

9.48.1.1 get_infos()

```
infos_st* get_infos ( )
```

Definition at line 16 of file atrier.c.

9.48.1.2 main()

```
int main ( )
```

Definition at line 69 of file genesis.c.

Here is the call graph for this function:

9.48.1.3 new_transaction()

```
void new_transaction (
            char type,
            char * rc_pk,
            size_t amount,
            char cause[512],
            char asset[512] )
```

Definition at line 148 of file atrier.c.

Here is the call graph for this function:

9.48.2 Variable Documentation

9.48.2.1 ac_infos

```
infos_st* ac_infos
```

Definition at line 15 of file genesis.c.

9.48.2.2 client_connections

```
connection* client_connections
```

Definition at line 4 of file client.c.

9.49 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/serverdoor.c File Reference

```
#include "network/server.h"
#include "network/client.h"
#include "cryptosystem/signature.h"
#include "blockchain/block.h"
#include <time.h>
```

Include dependency graph for serverdoor.c:

Functions

- int main ()

9.49.1 Function Documentation

9.49.1.1 main()

```
int main ( )
```

Definition at line 10 of file serverdoor.c.

Here is the call graph for this function:

9.50 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_blockchain_files.c File Reference

```
#include "tests_macros.h"
#include "blockchain/block.h"
#include "blockchain/transaction.h"
```

Include dependency graph for GEN_blockchain_files.c: This graph shows which files directly or indirectly include this file:

Macros

- #define GEN_BLC_F_C

Functions

- void rand_data (size_t size, char *buffer)
- void gen_blockchain (size_t nb_blocks)

9.50.1 Macro Definition Documentation

9.50.1.1 GEN_BLC_F_C

```
#define GEN_BLC_F_C
```

Definition at line 2 of file GEN_blockchain_files.c.

9.50.2 Function Documentation

9.50.2.1 gen_blockchain()

```
void gen_blockchain (
            size_t nb_blocks )
```

Definition at line 22 of file GEN_blockchain_files.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.50.2.2 rand_data()

```
void rand_data (
            size_t size,
            char * buffer )
```

Definition at line 8 of file GEN_blockchain_files.c.

Here is the caller graph for this function:

9.51 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_validators_file.c File Reference

```
#include <stdio.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include "tests_macros.h"
#include "validation/validators.h"
#include "cryptosystem/rsa.h"
```

Include dependency graph for GEN_validators_file.c: This graph shows which files directly or indirectly include this file:

Macros

- #define GEN_VALIDATORS_FILE_H
- #define NB_FAKE_VALIDATORS 10
- #define str(x) #x

Functions

- void gen_validators_file (char path[])

  Generate a mock validators states file.

9.51.1 Macro Definition Documentation

9.51.1.1 GEN_VALIDATORS_FILE_H

```
#define GEN_VALIDATORS_FILE_H
```

Definition at line 2 of file GEN_validators_file.c.

9.51.1.2 NB_FAKE_VALIDATORS

```
#define NB_FAKE_VALIDATORS 10
```

Definition at line 15 of file GEN_validators_file.c.

9.51.1.3 str

```
#define str(
              x ) #x
```

Definition at line 16 of file GEN_validators_file.c.

9.51.2 Function Documentation

9.51.2.1 gen_validators_file()

```
void gen_validators_file (
             char path[] )
```

Generate a mock validators states file.

Parameters

path	The path of the output file

See also

For one stake transaction, power += amount / (block_height + 1) + amount Foreach stake withdraw, power -= power ∗ withdraw_stake / user_total_stake

validators states file description Header : nb_validators[sizeof(size_t)], total_stake[sizeof(size_t)], block_height_validity[sizeof(size_t)] '
'[sizeof(char)] For each 'nb_validators' : validator_pkey[RSA_KEY_SIZE], user_stake[sizeof(size_t)] ,validator_power[sizeof(size_t)], '
'[sizeof(char)]

Definition at line 32 of file GEN_validators_file.c.

Here is the caller graph for this function:

9.52 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/blockchain/block_test.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void block_test (void)

9.52.1 Function Documentation

9.52.1.1 block_test()

```
void block_test (
            void  )
```

Definition at line 13 of file block_test.c.

Here is the call graph for this function:

9.53 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/rsa_test.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void get_keys_test ()
- void get_keys_equality_test ()

9.53.1 Function Documentation

9.53.1.1 get_keys_equality_test()

```
void get_keys_equality_test ( )
```

Definition at line 32 of file rsa_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.53.1.2 get_keys_test()

```
void get_keys_test ( )
```

Definition at line 18 of file rsa_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.54 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/signature_test.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void verify_sign_test ()

9.54.1 Function Documentation

9.54.1.1 verify_sign_test()

```
void verify_sign_test ( )
```

Definition at line 4 of file signature_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.55 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/network/client_test.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void network_test ()

9.55.1 Function Documentation

9.55.1.1 network_test()

```
void network_test ( )
```

Definition at line 15 of file client_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.56 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/validation/validations_test.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void validations_test ()

9.56.1 Function Documentation

9.56.1.1 validations_test()

```
void validations_test ( )
```

Definition at line 6 of file validations_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.57 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main_test.c File Reference

```
#include "blockchain/wallet.h"
```
Include dependency graph for main_test.c:

Macros

- #define MAIN_TEST_C

Functions

- int main ()

9.57.1 Macro Definition Documentation

9.57.1.1 MAIN_TEST_C

```
#define MAIN_TEST_C
```

Definition at line 2 of file main_test.c.

9.57.2 Function Documentation

9.57.2.1 main()

```
int main ( )
```

Definition at line 5 of file main_test.c.

Here is the call graph for this function:

9.58 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/blockchain/block_test.c File Reference

```
#include "tests_macros.h"
#include "blockchain/block.h"
#include "blockchain/transaction.h"
#include "gen/GEN_blockchain_files.c"
```
Include dependency graph for block_test.c:

Macros

- #define BLOCK_TEST_C
- #define NB_BLOCK_PER_CHUNK 10
- #define NB_MOCK_BLOCKS 13

Functions

- void block_test (void)

9.58.1 Macro Definition Documentation

9.58.1.1 BLOCK_TEST_C

```
#define BLOCK_TEST_C
```

Definition at line 2 of file block_test.c.

9.58.1.2 NB_BLOCK_PER_CHUNK

```
#define NB_BLOCK_PER_CHUNK 10
```

Definition at line 9 of file block_test.c.

9.58.1.3 NB_MOCK_BLOCKS

```
#define NB_MOCK_BLOCKS 13
```

Definition at line 11 of file block_test.c.

9.58.2 Function Documentation

9.58.2.1 block_test()

```
void block_test (
            void  )
```

Definition at line 13 of file block_test.c.

Here is the call graph for this function:

9.59 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa_test.c File Reference

```
#include "tests_macros.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/rsa.h"
#include "blockchain/wallet.h"
#include "misc/math.h"
#include <stdio.h>
#include <unistd.h>
#include <openssl/sha.h>
#include "misc/safe.h"
#include <fcntl.h>
#include <sys/stat.h>
```

Include dependency graph for rsa_test.c:

Macros

- #define RSA_SIZE_C

Functions

- void get_keys_test ()
- void get_keys_equality_test ()

9.59.1 Macro Definition Documentation

9.59.1.1 RSA_SIZE_C

```
#define RSA_SIZE_C
```

Definition at line 2 of file rsa_test.c.

9.59.2 Function Documentation

9.59.2.1 get_keys_equality_test()

```
void get_keys_equality_test ( )
```

Definition at line 32 of file rsa_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.59.2.2 get_keys_test()

```
void get_keys_test ( )
```

Definition at line 18 of file rsa_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.60 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/signature_test.c File Reference

```
#include "tests_macros.h"
#include "cryptosystem/signature.h"
```
Include dependency graph for signature_test.c:

Functions

- void verify_sign_test ()

9.60.1 Function Documentation

9.60.1.1 verify_sign_test()

```
void verify_sign_test ( )
```

Definition at line 4 of file signature_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.61 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/client_test.c File Reference

```
#include <signal.h>
#include "tests_macros.h"
#include "network/network.h"
#include "network/server.h"
#include "network/client.h"
#include "network/send_data.h"
#include "network/get_data.h"
```
Include dependency graph for client_test.c:

Macros

- #define CLIENT_TEST_C

Functions

- void network_test ()

Variables

- connection * client_connections

9.61.1 Macro Definition Documentation

9.61.1.1 CLIENT_TEST_C

```
#define CLIENT_TEST_C
```

Definition at line 2 of file client_test.c.

9.61.2 Function Documentation

9.61.2.1 network_test()

```
void network_test ( )
```

Definition at line 15 of file client_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.61.3 Variable Documentation

9.61.3.1 client_connections

```
connection* client_connections
```

Definition at line 4 of file client.c.

9.62 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/validation/validations_test.c File Reference

```
#include "gen/GEN_validators_file.c"
#include "validation/validators.h"
#include "tests_macros.h"
```

Include dependency graph for validations_test.c:

Functions

- void validations_test ()

9.62.1 Function Documentation

9.62.1.1 validations_test()

```
void validations_test ( )
```

Definition at line 6 of file validations_test.c.

Here is the call graph for this function: Here is the caller graph for this function:

9.63 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/tests_macros.h File Reference

```
#include <stdio.h>
```

Include dependency graph for tests_macros.h: This graph shows which files directly or indirectly include this file:

Macros

- #define DEBUG(function)
- #define LOG(str...)
- #define TEST_PASSED(name...)
- #define TEST_FAILED(name, reason...)
- #define TEST_WARNING(name, reason...)

9.63.1 Macro Definition Documentation

9.63.1.1 DEBUG

```
#define DEBUG(
              function )
```

Value:

```
    printf("Testing '%s'...\n", #function); \
    function()
```

Definition at line 5 of file tests_macros.h.

9.63.1.2 LOG

```
#define LOG(
              str...  )
```

Value:

```
    printf("\033[0;34m[-]  "); \
    printf(str);              \
    printf("\033[0m\n")
```

Definition at line 9 of file tests_macros.h.

9.63.1.3 TEST_FAILED

```
#define TEST_FAILED(
              name,
              reason...  )
```

Value:

```
    printf("\033[0;31m[X] TEST '%s' failed\n\t-> REASON : ", name); \
    printf(reason);                                                 \
    printf("\033[0m\n");                                            \
    exit(1)
```

Definition at line 19 of file tests_macros.h.

9.63.1.4 TEST_PASSED

```
#define TEST_PASSED(
              name...  )
```

Value:

```
    printf("\033[0;32m[OK] TEST -> '"); \
    printf(name);                      \
    printf("' success\033[0m\n")
```

Definition at line 14 of file tests_macros.h.

9.63.1.5 TEST_WARNING

```
#define TEST_WARNING(
              name,
              reason...  )
```

Value:

```
    printf("\033[0;33m[!] WARNING '%s'\n\t-> BECAUSE : ", name); \
    printf(reason);                                               \
    printf("\033[0m\n")
```

Definition at line 25 of file tests_macros.h.

9.64 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit_testing.c File Reference

```
#include "tests_macros.h"
#include "cryptosystem/rsa_test.h"
#include "cryptosystem/signature_test.h"
#include "network/client_test.h"
#include "blockchain/block_test.h"
#include "validation/validations_test.h"
```

Include dependency graph for unit_testing.c:

Data Structures

- struct infos_st

Typedefs

- typedef struct infos_st infos_st

Functions

- infos_st * get_infos ()
- int main ()

Variables

- infos_st * ac_infos

9.64.1 Typedef Documentation

9.64.1.1 infos_st

```
typedef struct infos_st infos_st
```

9.64.2 Function Documentation

9.64.2.1 get_infos()

```
infos_st* get_infos ( )
```

Definition at line 16 of file atrier.c.

9.64.2.2 main()

```
int main ( )
```

Definition at line 22 of file unit_testing.c.

Here is the call graph for this function:

9.64.3 Variable Documentation

9.64.3.1 ac_infos

```
infos_st* ac_infos
```

Definition at line 18 of file unit_testing.c.

9.65 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/VALIDATION_PROTOCOL.md File Reference

Index