PEPITAS CRYPTOCURRENCY

Generated by Doxygen 1.8.17

Chapter 1

PEPITAS NETWORK PROTOCOL

1.1 HEADERS

1.1.1 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

1.1.2 Running Headers

1. SEND PENDING TRANSACTION

1.1.3 Validating Headers

1. SEND BLOCK EPOCHMAN
2. SEND VOTE

1.1.4 CONNECTION TO NETWORK

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

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

1.1.5 CONNECTION TO NODE

Message: char ∗ : "CONNECTION TO NODE\r\n\r\n"

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

1.1.6 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.

1.1.7 ACTUAL HEIGHT

Message: char ∗ : "ACTUAL HEIGHT\r\n\r\n" size_t : Block height

Description Send my actual blockchain height.

1.1.8 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".

1.1.9 GET PENDING TRANSACTION LIST

Message char ∗ : "GET PENDING TRANSACTION LIST\r\n\r\n"

Description Call "SEND PENDING TRANSACTION".

1.1.10 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.

1.1.11 SEND PENDING TRANSACTION

Message: char ∗ : "SEND PENDING TRANSACTION\r\n\r\n"

Description Send the epoch block of a committee member.

1.1.12 SEND EPOCH BLOCK

Message: char ∗ : "SEND EPOCH BLOCK\r\n\r\n" char ∗ : Block struct

Description Send the epoch block of a committee member.

1.1.13 SEND VOTE

Message: char ∗ : "SEND VOTE\r\n\r\n" char ∗: Epoch creator pk size_t : block height int epoch_id: creator char : 0 = False 1 = True char ∗ : signature of vote precedent vars

Description Send the vote of a committee member.

Chapter 2

PEPITAS

C cryptocurrency.

2.1 CODING STYLE

2.1.1 Coding case

- *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`.

2.1.2 Tests

Each function must be tested before **marked as done**. To create a test function, you must write it in the `test/` directory and call the file `filename_test.c` and its functions `functionname_test`. Note that the test file must be at the same relative place than his real function

exemple : if you want to test `init_server()` in the file `network/client.c`, you must write the test in `test/network/client_test.c` and call the test function `init_server_test()`

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Structure Documentation

5.1 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 [MAX_VALIDATORS_PER_BLOCK/8]
- char vote_signature [MAX_VALIDATORS_PER_BLOCK - 1][256]

5.1.1 Detailed Description

Definition at line 51 of file block.h.

5.1.2 Field Documentation

5.1.2.1 block_data

`BlockData block_data`

Definition at line 54 of file block.h.

5.1.2.2 block_signature

```
char block_signature[256]
```

Definition at line 56 of file block.h.

5.1.2.3 chunk_id

```
uint16_t chunk_id
```

Definition at line 53 of file block.h.

5.1.2.4 validators_votes

```
char validators_votes[MAX_VALIDATORS_PER_BLOCK/8]
```

Definition at line 59 of file block.h.

5.1.2.5 vote_signature

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

Definition at line 60 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

5.2 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]
- char prev_validators_votes [MAX_VALIDATORS_PER_BLOCK/8]
- time_t block_timestamp

5.2.1 Detailed Description

Definition at line 33 of file block.h.

5.2.2 Field Documentation

5.2.2.1 block_timestamp

```
time_t block_timestamp
```

Definition at line 48 of file block.h.

5.2.2.2 epoch_id

```
int epoch_id
```

Definition at line 36 of file block.h.

5.2.2.3 height

```
size_t height
```

Definition at line 39 of file block.h.

5.2.2.4 is_prev_block_valid

```
char is_prev_block_valid
```

Definition at line 37 of file block.h.

5.2.2.5 magic

```
char magic
```

Definition at line 35 of file block.h.

5.2.2.6 nb_transactions

```
uint16_t nb_transactions
```

Definition at line 41 of file block.h.

5.2.2.7 nb_validators

```
int nb_validators
```

Definition at line 45 of file block.h.

5.2.2.8 prev_validators_votes

```
char prev_validators_votes[MAX_VALIDATORS_PER_BLOCK/8]
```

Definition at line 47 of file block.h.

5.2.2.9 previous_block_hash

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

Definition at line 38 of file block.h.

5.2.2.10 transactions

```
Transaction** transactions
```

Definition at line 42 of file block.h.

5.2.2.11 validators_public_keys

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

Definition at line 46 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

5.3 blockinfo Struct Reference

`#include <ui.h>`

Data Fields

- size_t height
- size_t transactions

5.3.1 Detailed Description

Definition at line 17 of file ui.h.

5.3.2 Field Documentation

5.3.2.1 height

`size_t height`

Definition at line 19 of file ui.h.

5.3.2.2 transactions

`size_t transactions`

Definition at line 20 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

5.4 ChunkBlockchain Struct Reference

`#include <block.h>`

Collaboration diagram for ChunkBlockchain:

Data Fields

- size_t chunk_nb
- Block ** chunk
- int16_t nb_blocks

5.4.1 Detailed Description

Definition at line 63 of file block.h.

5.4.2 Field Documentation

5.4.2.1 chunk

```
Block** chunk
```

Definition at line 66 of file block.h.

5.4.2.2 chunk_nb

```
size_t chunk_nb
```

Definition at line 65 of file block.h.

5.4.2.3 nb_blocks

```
int16_t nb_blocks
```

Definition at line 67 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

5.5 client_connection Struct Reference

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

Data Fields

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

5.5.1 Detailed Description

Definition at line 44 of file network.h.

5.5.2 Field Documentation

5.5.2.1 actual_client_height

```
size_t actual_client_height
```

Definition at line 52 of file network.h.

5.5.2.2 clientfd

```
int clientfd
```

Definition at line 49 of file network.h.

5.5.2.3 demand

```
int demand
```

Definition at line 48 of file network.h.

5.5.2.4 lock

```
sem_t lock
```

Definition at line 47 of file network.h.

5.5.2.5 Payload

```
void* Payload
```

Definition at line 51 of file network.h.

5.5.2.6 Playloadsize

```
size_t Playloadsize
```

Definition at line 50 of file network.h.

5.5.2.7 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

5.6 infos_st Struct Reference

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

Data Fields

- char is_sychronize
- size_t actual_height
- char serv_type

5.6.1 Detailed Description

Definition at line 55 of file network.h.

5.6.2 Field Documentation

5.6.2.1 actual_height

```
size_t actual_height
```

Definition at line 58 of file network.h.

5.6.2.2 is_sychronize

```
char is_sychronize
```

Definition at line 57 of file network.h.

5.6.2.3 serv_type

```
char serv_type
```

Definition at line 59 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

5.7 Neighbour Struct Reference

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

Data Fields

- int family
- char * hostname

5.7.1 Detailed Description

Definition at line 33 of file network.h.

5.7.2 Field Documentation

5.7.2.1 family

```
int family
```

Definition at line 35 of file network.h.

5.7.2.2 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

5.8 Node Struct Reference

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

Collaboration diagram for Node:

Data Fields

- Neighbour * neighbours

5.8.1 Detailed Description

Definition at line 39 of file network.h.

5.8.2 Field Documentation

5.8.2.1 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

5.9 th_arg Struct Reference

`#include <network.h>`

Collaboration diagram for th_arg:

Data Fields

- infos_st * infos
- client_connection * client_con

5.9.1 Detailed Description

Definition at line 61 of file network.h.

5.9.2 Field Documentation

5.9.2.1 client_con

```
client_connection* client_con
```

Definition at line 64 of file network.h.

5.9.2.2 infos

```
infos_st* infos
```

Definition at line 63 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

5.10 Transaction Struct Reference

`#include <transaction.h>`

Collaboration diagram for Transaction:

Data Fields

- TransactionData transaction_data
- char transaction_signature [256]

5.10.1 Detailed Description

Definition at line 43 of file transaction.h.

5.10.2 Field Documentation

5.10.2.1 transaction_data

```
TransactionData transaction_data
```

Definition at line 45 of file transaction.h.

5.10.2.2 transaction_signature

```
char transaction_signature[256]
```

Definition at line 47 of file transaction.h.

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

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

5.11 TransactionData Struct Reference

```
#include <transaction.h>
```

Data Fields

- char magic
- char type
- RSA * sender_public_key
- RSA * receiver_public_key
- RSA * organisation_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]

5.11.1 Detailed Description

Definition at line 24 of file transaction.h.

5.11.2 Field Documentation

5.11.2.1 amount

```
size_t amount
```

Definition at line 32 of file transaction.h.

5.11.2.2 asset

```
char asset[512]
```

Definition at line 40 of file transaction.h.

5.11.2.3 cause

```
char cause[512]
```

Definition at line 39 of file transaction.h.

5.11.2.4 magic

```
char magic
```

Definition at line 26 of file transaction.h.

5.11.2.5 organisation_public_key

```
RSA* organisation_public_key
```

Definition at line 31 of file transaction.h.

5.11.2.6 receiver_public_key

```
RSA* receiver_public_key
```

Definition at line 30 of file transaction.h.

5.11.2.7 receiver_remaining_money

```
size_t receiver_remaining_money
```

Definition at line 34 of file transaction.h.

5.11.2.8 sender_public_key

```
RSA* sender_public_key
```

Definition at line 29 of file transaction.h.

5.11.2.9 sender_remaining_money

```
size_t sender_remaining_money
```

Definition at line 33 of file transaction.h.

5.11.2.10 transaction_timestamp

```
time_t transaction_timestamp
```

Definition at line 35 of file transaction.h.

5.11.2.11 type

```
char type
```

Definition at line 27 of file transaction.h.

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

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

5.12 Wallet Struct Reference

`#include <wallet.h>`

Data Fields

- RSA * priv_key
- RSA * pub_key
- size_t amount
- char is_validator

5.12.1 Detailed Description

Definition at line 10 of file wallet.h.

5.12.2 Field Documentation

5.12.2.1 amount

`size_t amount`

Definition at line 15 of file wallet.h.

5.12.2.2 is_validator

`char is_validator`

Definition at line 16 of file wallet.h.

5.12.2.3 priv_key

`RSA* priv_key`

Definition at line 12 of file wallet.h.

5.12.2.4 pub_key

`RSA* pub_key`

Definition at line 13 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 6

File Documentation

6.1 /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 "transaction.h"
#include "misc/files.h"
```

Include dependency graph for block.h:

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

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

6.2.1 Function Documentation

6.2.1.1 gen_blockchain_header()

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

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:

6.3 /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 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_STAKE_TO_STAKE 3

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)
- void write_transaction (Transaction *transaction, int fd)
- void get_transaction_data (Transaction *trans, char **buff, size_t *index)
- void convert_data_to_transactiondata (TransactionData *transactiondata, int fd)
- void load_transaction (Transaction *transaction, int fd)
- Transaction * load_pending_transaction (time_t timestamp)
- void add_pending_transaction (Transaction *transaction)

6.3.1 Macro Definition Documentation

6.3.1.1 T_TYPE_ADD_STAKE

```
#define T_TYPE_ADD_STAKE 1
```

Definition at line 20 of file transaction.h.

6.3.1.2 T_TYPE_DEFAULT

```
#define T_TYPE_DEFAULT 0
```

Definition at line 19 of file transaction.h.

6.3.1.3 T_TYPE_STAKE_TO_STAKE

```
#define T_TYPE_STAKE_TO_STAKE 3
```

Definition at line 22 of file transaction.h.

6.3.1.4 T_TYPE_WITHDRAW_STAKE

```
#define T_TYPE_WITHDRAW_STAKE 2
```

Definition at line 21 of file transaction.h.

6.3.1.5 TRANSACTION_DATA_SIZE

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

Definition at line 16 of file transaction.h.

6.3.1.6 TRANSACTION_SIZE

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

Definition at line 17 of file transaction.h.

6.3.2 Typedef Documentation

6.3.2.1 Transaction

```
typedef struct Transaction Transaction
```

6.3.2.2 TransactionData

```
typedef struct TransactionData TransactionData
```

6.3.3 Function Documentation

6.3.3.1 add_pending_transaction()

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

Definition at line 158 of file transaction.c.

Here is the call graph for this function:

6.3.3.2 convert_data_to_transactiondata()

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

Definition at line 99 of file transaction.c.

Here is the caller graph for this function:

6.3.3.3 get_transaction_data()

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

Definition at line 48 of file transaction.c.

Here is the caller graph for this function:

6.3.3.4 load_pending_transaction()

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

Definition at line 145 of file transaction.c.

Here is the call graph for this function:

6.3.3.5 load_transaction()

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

Definition at line 135 of file transaction.c.

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

6.3.3.6 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

6.3.3.7 write_transaction()

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

Definition at line 42 of file transaction.c.

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

6.3.3.8 write_transactiondata()

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

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

6.4 /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 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.

6.4.1 Typedef Documentation

6.4.1.1 Wallet

```
typedef struct Wallet Wallet
```

6.4.2 Function Documentation

6.4.2.1 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 19 of file wallet.c.

Here is the call graph for this function:

6.4.2.2 get_my_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

Returns

Wallet

Definition at line 7 of file wallet.c.

Here is the caller graph for this function:

6.5 /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.

6.5.1 Function Documentation

6.5.1.1 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: Here is the caller graph for this function:

6.5.1.2 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:

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

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.

6.6.1 Macro Definition Documentation

6.6.1.1 RSA_BEGIN_SIZE

```
#define RSA_BEGIN_SIZE 31
```

Definition at line 6 of file rsa.h.

6.6.1.2 RSA_END_SIZE

```
#define RSA_END_SIZE 29
```

Definition at line 7 of file rsa.h.

6.6.1.3 RSA_FILE_TOTAL_SIZE

```
#define RSA_FILE_TOTAL_SIZE 426
```

Definition at line 5 of file rsa.h.

6.6.1.4 RSA_KEY_SIZE

```
#define RSA_KEY_SIZE 366
```

Definition at line 4 of file rsa.h.

6.6.2 Function Documentation

6.6.2.1 get_keys()

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

Get the keys object.

Here is the caller graph for this function:

6.7 /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)

  *Convert 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.
- void sign_block_with_key (Block *block, RSA *key)
- void sign_transaction (Transaction *transaction)
- void sign_transaction_with_key (Transaction *transaction, RSA *key)

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

  Signs transactions of a block.

6.7.1 Function Documentation

6.7.1.1 get_transaction_data()

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

Convert 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)

Definition at line 48 of file transaction.c.

6.7.1.2 sign_block()

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

Signs a block.

Parameters

block	The block to sign

Definition at line 102 of file signature.c.

Here is the call graph for this function:

6.7.1.3 sign_block_transactions()

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

Signs transactions of a block.

Parameters

block	The block to sign

Definition at line 132 of file signature.c.

Here is the call graph for this function:

6.7.1.4 sign_block_with_key()

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

Definition at line 109 of file signature.c.

Here is the call graph for this function:

6.7.1.5 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:

6.7.1.6 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:

6.7.1.7 sign_transaction()

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

Definition at line 116 of file signature.c.

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

6.7.1.8 sign_transaction_with_key()

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

Sign a transaction.

Parameters

transaction	The transaction to sign

Definition at line 124 of file signature.c.

Here is the call graph for this function:

6.7.1.9 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 77 of file signature.c.

Here is the call graph for this function:

6.7.1.10 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:

6.7.1.11 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 89 of file signature.c.

Here is the call graph for this function:

6.7.1.12 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 309 of file block.c.

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

6.7.1.13 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 277 of file block.c.

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

6.8 /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.

6.8.1 Function Documentation

6.8.1.1 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.

Here is the caller graph for this function:

6.9 /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)

6.9.1 Macro Definition Documentation

6.9.1.1 MAX

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

Definition at line 2 of file math.h.

6.9.1.2 MIN

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

Definition at line 1 of file math.h.

6.10 /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 !

6.10.1 Function Documentation

6.10.1.1 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:

6.10.1.2 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:

6.10.1.3 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:

6.10.1.4 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:

6.11 /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.
- client_connection * listen_to (infos_st *infos, Neighbour neighbour, char *connection_type)
  Tries to connect to the peer-to-peer network via a node in the Node structure.
- int find_empty_connection (int max)
- void * client_thread (void *args)

6.11.1 Function Documentation

6.11.1.1 client_thread()

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

Definition at line 247 of file client.c.

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

6.11.1.2 find_empty_connection()

```
int find_empty_connection (
            int max )
```

Definition at line 237 of file client.c.

Here is the caller graph for this function:

6.11.1.3 get_my_node()

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

Get the my node object.

Returns

Node*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

6.11.1.4 listen_to()

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

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

Parameters

neighbour	The neighbour to connect with

Returns

socket FD or -1 if an error occurs

Definition at line 161 of file client.c.

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

6.11.1.5 load_neighbours()

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

Load neighbours list from .neighbours/neighbours.

Definition at line 113 of file client.c.

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

6.11.1.6 number_neighbours()

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

return the nb of neighbour in the client.neightbours section

Definition at line 149 of file client.c.

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

6.11.1.7 print_neighbours()

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

Print neighbours list.

Definition at line 58 of file client.c.

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

6.11.1.8 remove_neighbour()

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

Remove a neighbour in the client.neightbours section.

Definition at line 47 of file client.c.

Here is the call graph for this function:

6.11.1.9 save_neighbours()

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

Save neighbours list in .neighbours/neighbours.

Definition at line 74 of file client.c.

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

6.11.1.10 set_neighbour()

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

Sets a neighbour in the client.neightbours section.

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:

6.12 /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 "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)

  Merges my neighbours list with the one sent by 'neighbour_id'.
- int read_get_blocks (int fd, infos_st *infos)
- size_t read_actual_height (int fd)
- int read_send_block (int fd)
- int read_vote (int fd)
- int read_pending_transaction_list (int fd)
- int read_epoch_block (int fd)

6.12.1 Function Documentation

6.12.1.1 fetch_client_list()

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

Merges my neighbours list with the one sent by 'neighbour_id'.

Parameters

sockfd	The sockfd to read

Returns

0 if sucess, -1 otherwise

Definition at line 95 of file get_data.c.

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

6.12.1.2 read_actual_height()

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

Definition at line 172 of file get_data.c.

Here is the caller graph for this function:

6.12.1.3 read_epoch_block()

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

Definition at line 222 of file get_data.c.

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

6.12.1.4 read_get_blocks()

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

Definition at line 144 of file get_data.c.

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

6.12.1.5 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

Returns

0 if sucess, -1 otherwise

Definition at line 125 of file get_data.c.

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

6.12.1.6 read_pending_transaction_list()

```
int read_pending_transaction_list (
            int fd )
```

6.12.1.7 read_send_block()

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

Definition at line 178 of file get_data.c.

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

6.12.1.8 read_vote()

```
int read_vote (
            int fd )
```

6.13 /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 client_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_REJECT_DEMAND "REJECT DEMAND\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 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 client_connection client_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

6.13.1 Macro Definition Documentation

6.13.1.1 CLIENTMSG

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

Definition at line 90 of file network.h.

6.13.1.2 DD_GET_BLOCKS

```
#define DD_GET_BLOCKS 2
```

Definition at line 85 of file network.h.

6.13.1.3 DD_GET_HEIGHT

```
#define DD_GET_HEIGHT 1
```

Definition at line 84 of file network.h.

6.13.1.4 DOORSERVER

```
#define DOORSERVER 1
```

Definition at line 23 of file network.h.

6.13.1.5 HD_ACTUAL_HEIGHT

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

Definition at line 75 of file network.h.

6.13.1.6 HD_CONNECTION_TO_NETWORK

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

Definition at line 72 of file network.h.

6.13.1.7 HD_CONNECTION_TO_NODE

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

Definition at line 73 of file network.h.

6.13.1.8 HD_GET_BLOCKS

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

Definition at line 74 of file network.h.

6.13.1.9 HD_GET_CLIENT_LIST

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

Definition at line 70 of file network.h.

6.13.1.10 HD_GET_PENDING_TRANSACTION_LIST

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

Definition at line 77 of file network.h.

6.13.1.11 HD_REJECT_DEMAND

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

Definition at line 78 of file network.h.

6.13.1.12 HD_SEND_BLOCK

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

Definition at line 76 of file network.h.

6.13.1.13 HD_SEND_CLIENT_LIST

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

Definition at line 71 of file network.h.

6.13.1.14 HD_SEND_EPOCH_BLOCK

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

Definition at line 80 of file network.h.

6.13.1.15 HD_SEND_PENDING_TRANSACTION

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

Definition at line 79 of file network.h.

6.13.1.16 HD_SEND_VOTE

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

Definition at line 81 of file network.h.

6.13.1.17 IM_CLIENT

```
#define IM_CLIENT 1
```

Definition at line 18 of file network.h.

6.13.1.18 IM_SERVER

```
#define IM_SERVER 0
```

Definition at line 17 of file network.h.

6.13.1.19 MANAGERMSG

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

Definition at line 91 of file network.h.

6.13.1.20 MAX_CONNECTION

```
#define MAX_CONNECTION 5
```

Definition at line 11 of file network.h.

6.13.1.21 MAX_NEIGHBOURS

```
#define MAX_NEIGHBOURS 64
```

Definition at line 20 of file network.h.

6.13.1.22 MAX_SERVER

```
#define MAX_SERVER 20
```

Definition at line 25 of file network.h.

6.13.1.23 MAX_VALIDATORS_PER_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 27 of file network.h.

6.13.1.24 NB_HARD_CODED_ADDR

```
#define NB_HARD_CODED_ADDR 2
```

Definition at line 10 of file network.h.

6.13.1.25 NODESERVER

```
#define NODESERVER 0
```

Definition at line 22 of file network.h.

6.13.1.26 P_VERSION

```
#define P_VERSION 0
```

Definition at line 15 of file network.h.

6.13.1.27 SERVERMSG

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

Definition at line 89 of file network.h.

6.13.1.28 SIZE_OF_HOSTNAME

```
#define SIZE_OF_HOSTNAME 39
```

Definition at line 9 of file network.h.

6.13.1.29 SOL_TCP

```
#define SOL_TCP 6
```

Definition at line 29 of file network.h.

6.13.1.30 STATIC_PORT

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

Definition at line 13 of file network.h.

6.13.1.31 TCP_USER_TIMEOUT

```
#define TCP_USER_TIMEOUT 18
```

Definition at line 30 of file network.h.

6.13.1.32 WARNINGMSG

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

Definition at line 92 of file network.h.

6.13.2 Typedef Documentation

6.13.2.1 client_connection

```
typedef struct client_connection client_connection
```

6.13.2.2 infos_st

```
typedef struct infos_st infos_st
```

6.13.2.3 Neighbour

```
typedef struct Neighbour Neighbour
```

6.13.2.4 Node

```
typedef struct Node Node
```

6.13.2.5 th_arg

```
typedef struct th_arg th_arg
```

6.13.3 Function Documentation

6.13.3.1 __attribute__()

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

Definition at line 94 of file network.h.

Here is the caller graph for this function:

6.13.4 Variable Documentation

6.13.4.1 get_blocks_t

```
get_blocks_t
```

Definition at line 99 of file network.h.

6.13.4.2 HARD_CODED_ADDR

```
const Neighbour HARD_CODED_ADDR[]
```

Definition at line 5 of file network.c.

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

```
#include "network/server.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 (client_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 sockfd)

6.14.1 Function Documentation

6.14.1.1 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:

6.14.1.2 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:

6.14.1.3 send_get_blocks()

```
void send_get_blocks (
            client_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:

6.14.1.4 send_pending_transaction_list()

```
void send_pending_transaction_list (
            int sockfd )
```

Here is the caller graph for this function:

6.14.1.5 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:

6.14.1.6 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:

6.15 /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'.

6.15.1 Function Documentation

6.15.1.1 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 77 of file server.c.

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

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

```
#include <gtk/gtk.h>
#include <stdio.h>
#include <string.h>
#include <err.h>
#include <time.h>
#include "cryptosystem/rsa.h"
#include "cryptosystem/hash.h"
#include "blockchain/wallet.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_pkey_button_press (GtkWidget *widget, GdkEventKey *event, gpointer user_data)

  Hides the private key of the user, or shows it if it was already hidden.
- 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)

Variables

- GtkLabel * synchro_label
- GtkLabel * block_amount_label
- GtkLabel * connections_label
- GtkLabel * mempool_label
- struct blockinfo blocksinfo [3]

6.16.1 Function Documentation

6.16.1.1 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

6.16.1.2 add_contact_to_combobox()

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

Definition at line 466 of file ui.c.

Here is the caller graph for this function:

6.16.1.3 add_contacts_from_file()

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

Definition at line 474 of file ui.c.

Here is the caller graph for this function:

6.16.1.4 add_new_blockinfo()

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

Definition at line 199 of file ui.c.

6.16.1.5 add_transaction_from_file()

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

Definition at line 324 of file ui.c.

Here is the caller graph for this function:

6.16.1.6 add_transaction_with_contact()

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

Definition at line 304 of file ui.c.

Here is the caller graph for this function:

6.16.1.7 add_transaction_with_pkey()

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

Definition at line 283 of file ui.c.

Here is the caller graph for this function:

6.16.1.8 change_label_text()

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

Definition at line 194 of file ui.c.

Here is the caller graph for this function:

6.16.1.9 get_public_key_from_contacts()

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

Definition at line 511 of file ui.c.

Here is the caller graph for this function:

6.16.1.10 load_contacts_from_file()

```
void load_contacts_from_file ( )
```

Definition at line 483 of file ui.c.

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

6.16.1.11 load_transaction_from_file()

```
void load_transaction_from_file ( )
```

6.16.1.12 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

6.16.1.13 on_connect_but_press()

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

6.16.1.14 on_create_key_but1_press()

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

6.16.1.15 on_create_key_but2_press()

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

6.16.1.16 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

6.16.1.17 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

6.16.1.18 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 233 of file ui.c.

6.16.1.19 on_main_window_destroy()

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

Quits GTK when the program ends.

6.16.1.20 on_pkey_button_press()

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

Hides the private key of the user, or shows it if it was already hidden.

Parameters

widget	unused
event	unused
user_data	unused

Returns

gboolean Error code

6.16.1.21 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

6.16.1.22 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

6.16.1.23 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

6.16.1.24 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 63 of file ui.c.

Here is the caller graph for this function:

6.16.1.25 update_labels()

```
void update_labels ( )
```

Definition at line 658 of file ui.c.

Here is the call graph for this function:

6.16.1.26 update_sync()

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

Definition at line 214 of file ui.c.

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

6.16.2 Variable Documentation

6.16.2.1 block_amount_label

```
GtkLabel* block_amount_label
```

Definition at line 14 of file ui.h.

6.16.2.2 blocksinfo

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

Definition at line 23 of file ui.h.

6.16.2.3 connections_label

```
GtkLabel* connections_label
```

Definition at line 15 of file ui.h.

6.16.2.4 mempool_label

```
GtkLabel* mempool_label
```

Definition at line 16 of file ui.h.

6.16.2.5 synchro_label

```
GtkLabel* synchro_label
```

Definition at line 13 of file ui.h.

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

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

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

Functions

- 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'.
- int flush_pending_transactions (Block *block)

  Delete block transactions in pdt if the block is valid.

6.17.1 Function Documentation

6.17.1.1 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*

6.17.1.2 flush_pending_transactions()

```
int flush_pending_transactions (
            Block * block )
```

Delete block transactions in pdt if the block is valid.

Parameters

block	

Returns

1 if the flush proceed, 0 if not

6.17.1.3 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*

Definition at line 3 of file epoch_man.c.

Here is the caller graph for this function:

6.17.1.4 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	
current_block	

6.18 /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 "misc/math.h"
#include "misc/files.h"
#include "misc/safe.h"
#include <string.h>
#include <openssl/bio.h>
#include <stdbool.h>
```

Include dependency graph for validation_engine.h:

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.

6.18.1 Function Documentation

6.18.1.1 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

6.18.1.2 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

6.19 /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/rsa.h"
#include "cryptosystem/hash.h"
#include "misc/files.h"
#include "misc/safe.h"
```

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

Macros

- #define MAX_VALIDATORS_PER_BLOCK 512

Functions

- RSA ** get_comittee (size_t block_height, size_t *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- RSA ** get_next_comittee (size_t *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 (char pkey[ ])

  Get the validator id in 'validators.state'.
- int push_stake (size_t amount)

  Push an amount on the stake.
- int pop_stake (size_t amount)

  Pops an amount on the stake.

6.19.1 Macro Definition Documentation

6.19.1.1 MAX_VALIDATORS_PER_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 13 of file validators.h.

6.19.2 Function Documentation

6.19.2.1 get_comittee()

```
RSA** get_comittee (
            size_t block_height,
            size_t * 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 32 of file validators.c.

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

6.19.2.2 get_next_comittee()

```
RSA** get_next_comittee (
            size_t * 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 124 of file validators.c.

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

6.19.2.3 get_validator_id()

```
ssize_t get_validator_id (
             char pkey[] )
```

Get the validator id in 'validators.state'.

Parameters

pkey	The string public key (without -–—BEGIN RSA KEY-–— & -–—END RSA KEY-–—)

Returns

ssize_t, the validator index

Definition at line 209 of file validators.c.

Here is the call graph for this function:

6.19.2.4 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 188 of file validators.c.

Here is the call graph for this function:

6.19.2.5 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 176 of file validators.c.

Here is the call graph for this function:

6.19.2.6 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 164 of file validators.c.

Here is the call graph for this function:

6.19.2.7 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 152 of file validators.c.

Here is the call graph for this function:

6.19.2.8 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 141 of file validators.c.

Here is the call graph for this function:

6.19.2.9 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 129 of file validators.c.

Here is the call graph for this function:

6.19.2.10 pop_stake()

```
int pop_stake (
            size_t amount )
```

Pops an amount on the stake.

This will broadcast a stake pop on the network.

See also

The stake account public key is '1'

Parameters

amount	The amount to pop

Returns

0 if the broadcast succeeds, else returns -1

6.19.2.11 push_stake()

```
int push_stake (
            size_t amount )
```

Push an amount on the stake.

This will broadcast a stake push on the network.

See also

The stake account public key is '1'

Parameters

amount	The amount to push

Returns

0 if the broadcast succeeds, else returns -1

6.20 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/P2P_Protocol.md File Reference

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

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

```
#include <signal.h>
#include <stdlib.h>
```

```
#include <string.h>
#include "network/network.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/blockchain_header.h"
#include "ui/ui.h"
```

Include dependency graph for client.c:

Functions

- 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)
- int main ()

Variables

- client_connection * client_connections

6.22.1 Function Documentation

6.22.1.1 connection_to_others()

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

Definition at line 39 of file client.c.

Here is the call graph for this function:

6.22.1.2 join_network_door()

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

Definition at line 19 of file client.c.

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

6.22.1.3 main()

```
int main ( )
```

Definition at line 121 of file client.c.

Here is the call graph for this function:

6.22.1.4 update_blockchain()

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

Definition at line 93 of file client.c.

6.22.1.5 update_blockchain_height()

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

Definition at line 57 of file client.c.

Here is the call graph for this function:

6.22.2 Variable Documentation

6.22.2.1 client_connections

```
client_connection* client_connections
```

Definition at line 4 of file client.c.

6.23 /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 number_neighbours (char who)

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

  Tries to connect to the peer-to-peer network via a node in the Node structure.
- int find_empty_connection (int max)
- void * client_thread (void *args)

Variables

- client_connection * client_connections = NULL

6.23.1 Function Documentation

6.23.1.1 client_thread()

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

Definition at line 247 of file client.c.

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

6.23.1.2 find_empty_connection()

```
int find_empty_connection (
            int max )
```

Definition at line 237 of file client.c.

Here is the caller graph for this function:

6.23.1.3 get_my_node()

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

Get the my node object.

Returns

Node*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

6.23.1.4 listen_to()

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

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

Parameters

neighbour	The neighbour to connect with

Returns

socket FD or -1 if an error occurs

Definition at line 161 of file client.c.

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

6.23.1.5 load_neighbours()

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

Load neighbours list from .neighbours/neighbours.

Definition at line 113 of file client.c.

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

6.23.1.6 number_neighbours()

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

return the nb of neighbour in the client.neightbours section

Definition at line 149 of file client.c.

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

6.23.1.7 print_neighbours()

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

Print neighbours list.

Definition at line 58 of file client.c.

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

6.23.1.8 remove_neighbour()

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

Remove a neighbour in the client.neightbours section.

Definition at line 47 of file client.c.

Here is the call graph for this function:

6.23.1.9 save_neighbours()

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

Save neighbours list in .neighbours/neighbours.

Definition at line 74 of file client.c.

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

6.23.1.10 set_neighbour()

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

Sets a neighbour in the client.neightbours section.

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:

6.23.2 Variable Documentation

6.23.2.1 client_connections

```
client_connection* client_connections = NULL
```

Definition at line 4 of file client.c.

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

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

Macros

- #define GENESIS_RSA_PUB_1 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAwcXVgJ6Hy9nry↩ AmSFGpRYxLtPJ1VcI9XTbV34hniNMtztMGpwSTG\nCQ28WIWiD43qjmHxvY4Y26mLYXPjlJ2HiwneSo↩ ZcLtY+gJfObGcclpI1DSA0vE72\neTBbDz8enRbJqFWenwopKDoBjvf7nwc/fqRwD0ptLC7xwIPccRiLGd↩ OvP/IusLY0\nLCP6A9R50H7tGsbaAQfGoHYezY8p05K6XRankb7I8wsLFdU6Ew6OghX1tq02liP4\ns5↩ DrIoSsxi1mJtW7d+vln0D/a7t2bz4jI+OMtD5M5jldGMyQpzq3D8ZJokMyh6K2\nNLwrAiqDKZiIHJTw8FZid↩ A9/yuzlRpxNHQIBAw==\n-----END RSA PUBLIC KEY-----\n"
- #define GENESIS_RSA_PUB_2 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAsrHwAjOQzmoguF↩ CpWTEe/3x/T6KSr7dF1zYFnCq3V3v6OQFAcyt7\nQG0q138XFasRM70Hc0k589s5nKfPYSz5MCa6iDD1↩ IKo1qrGSyF9CPfW87DwZuLXW\nhShifhsLu+VfkbjYx5h/SGmC5WSedro3cTrex7V1BbZkCeKqRMYCgtT↩ PucyYE4pP\nqEnFQtMVAssyaDckjpWzpwun9wsoZ3qkqAAREwNecR7i2ojyUBJ8L5ZUryqmxi4F\ngwvF↩ LnlhAeoraWqk40L3bSdnGH1u/YV59f4/MSyVmTezI6DhFx2E3Pld/Kar5PnF\nrJSEQjtjwg+OVdGnrT46S↩ Kq8JQQlgFVZzwIBAw==\n-----END RSA PUBLIC KEY-----\n"

Functions

- Block * get_genesis_block ()
- 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.
- size_t get_last_block_height ()

  Get the last block height.
- 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)
- Block * get_block (size_t block_height)

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

  Free a block struct.
- 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.

6.24.1 Macro Definition Documentation

6.24.1.1 GENESIS_RSA_PUB_1

```
#define GENESIS_RSA_PUB_1 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAwcXVgJ6Hy9nryAmSFGpR↩
YxLtPJ1VcI9XTbV34hniNMtztMGpwSTG\nCQ28WIWiD43qjmHxvY4Y26mLYXPjlJ2HiwneSoZcLtY+gJfObGcclpI1D↩
SA0vE72\neTBbDz8enRbJqFWenwopKDoBjvf7nwc/fqRwD0ptLC7xwlPccRiLGdOvP/IusLY0\nLCP6A9R50H7tGsba↩
AQfGoHYezY8p05K6XRankb7I8wsLFdU6Ew6OghX1tq02liP4\ns5DrloSsxi1mJtW7d+vln0D/a7t2bz4jI+OMtD5↩
M5jldGMyQpzq3D8ZJokMyh6K2\nNLwrAiqDKZiIHJTw8FZidA9/yuzlRpxNHQIBAw==\n-----END RSA PUBLIC K↩
EY-----\n"
```

Definition at line 3 of file block.c.

6.24.1.2 GENESIS_RSA_PUB_2

```
#define GENESIS_RSA_PUB_2 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAsrHwAjOQzmoguFCpWT↩
Ee/3x/T6KSr7dF1zYFnCq3V3v6OQFAcyt7\nQG0q138XFasRM70Hc0k589s5nKfPYSz5MCa6iDD1IKo1qrGSyF9CPf↩
W87DwZuLXW\nhShifhsLu+VfkbjYx5h/SGmC5WSedro3cTrex7V1BbZkCeKqRMYCgtTPucyYE4pP\nqEnFQtMVAssya↩
DckjpWzpwun9wsoZ3qkqAAREwNecR7i2ojyUBJ8L5ZUryqmxi4F\ngwvFLnlhAeoraWqk40L3bSdnGH1u/YV59f4/M↩
SyVmTezI6DhFx2E3Pld/Kar5PnF\nrJSEQjtjwg+OVdGnrT46SKq8JQQlgFVZzwIBAw==\n-----END RSA PUBLIC K↩
EY-----\n"
```

Definition at line 33 of file block.c.

6.24.2 Function Documentation

6.24.2.1 convert_data_to_block()

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

Definition at line 172 of file block.c.

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

6.24.2.2 convert_data_to_blockdata()

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

Definition at line 139 of file block.c.

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

6.24.2.3 free_block()

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

Free a block struct.

Parameters

block	The block to free

Definition at line 202 of file block.c.

Here is the caller graph for this function:

6.24.2.4 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 180 of file block.c.

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

6.24.2.5 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 241 of file block.c.

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

6.24.2.6 get_genesis_block()

```
Block* get_genesis_block ( )
```

Definition at line 35 of file block.c.

Here is the call graph for this function:

6.24.2.7 get_last_block_height()

```
size_t get_last_block_height ( )
```

Get the last block height.

Returns

size_t

Definition at line 115 of file block.c.

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

6.24.2.8 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 221 of file block.c.

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

6.24.2.9 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 next Block*

Definition at line 231 of file block.c.

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

6.24.2.10 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 69 of file block.c.

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

6.24.2.11 load_last_blockchain()

```
ChunkBlockchain* load_last_blockchain ( )
```

Load the last local blockchain chunk.

Parameters

nb_chunk	

Returns

ChunkBlockchain*

Definition at line 110 of file block.c.

Here is the call graph for this function:

6.24.2.12 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 309 of file block.c.

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

6.24.2.13 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 121 of file block.c.

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

6.24.2.14 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 277 of file block.c.

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

6.25 /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)

6.25.1 Function Documentation

6.25.1.1 gen_blockchain_header()

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

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:

6.25.1.2 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:

6.26 /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)
- void write_transaction (Transaction *transaction, int fd)
- void get_transaction_data (Transaction *trans, char **buff, size_t *index)

  *Convert transactions to char * buffer.*
- void convert_data_to_transactiondata (TransactionData *transactiondata, int fd)
- void load_transaction (Transaction *transaction, int fd)
- Transaction * load_pending_transaction (time_t timestamp)
- void add_pending_transaction (Transaction *transaction)

6.26.1 Function Documentation

6.26.1.1 add_pending_transaction()

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

Definition at line 158 of file transaction.c.

Here is the call graph for this function:

6.26.1.2 convert_data_to_transactiondata()

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

Definition at line 99 of file transaction.c.

Here is the caller graph for this function:

6.26.1.3 get_transaction_data()

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

Convert 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)

Definition at line 48 of file transaction.c.

Here is the caller graph for this function:

6.26.1.4 load_pending_transaction()

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

Definition at line 145 of file transaction.c.

Here is the call graph for this function:

6.26.1.5 load_transaction()

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

Definition at line 135 of file transaction.c.

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

6.26.1.6 write_transaction()

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

Definition at line 42 of file transaction.c.

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

6.26.1.7 write_transactiondata()

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

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

6.27 /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 "blockchain/transaction.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.

6.27.1 Function Documentation

6.27.1.1 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 19 of file wallet.c.

Here is the call graph for this function:

6.27.1.2 get_my_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

Returns

Wallet

Definition at line 7 of file wallet.c.

Here is the caller graph for this function:

6.28 /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.

6.28.1 Function Documentation

6.28.1.1 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: Here is the caller graph for this function:

6.28.1.2 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:

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

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

Macros

- #define RSA_NUM_E 3

Functions

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

6.29.1 Macro Definition Documentation

6.29.1.1 RSA_NUM_E

```
#define RSA_NUM_E 3
```

Definition at line 16 of file rsa.c.

6.29.2 Function Documentation

6.29.2.1 get_keys()

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

Definition at line 21 of file rsa.c.

Here is the call graph for this function:

6.30 /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.
- void sign_block_with_key (Block *block, RSA *key)
- void sign_transaction (Transaction *transaction)
- void sign_transaction_with_key (Transaction *transaction, RSA *key)

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

  Signs transactions of a block.

6.30.1 Function Documentation

6.30.1.1 sign_block()

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

Signs a block.

Parameters

block	The block to sign

Definition at line 102 of file signature.c.

Here is the call graph for this function:

6.30.1.2 sign_block_transactions()

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

Signs transactions of a block.

Parameters

block	The block to sign

Definition at line 132 of file signature.c.

Here is the call graph for this function:

6.30.1.3 sign_block_with_key()

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

Definition at line 109 of file signature.c.

Here is the call graph for this function:

6.30.1.4 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:

6.30.1.5 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:

6.30.1.6 sign_transaction()

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

Definition at line 116 of file signature.c.

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

6.30.1.7 sign_transaction_with_key()

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

Sign a transaction.

Parameters

transaction	The transaction to sign

Definition at line 124 of file signature.c.

Here is the call graph for this function:

6.30.1.8 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 77 of file signature.c.

Here is the call graph for this function:

6.30.1.9 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:

6.30.1.10 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 89 of file signature.c.

Here is the call graph for this function:

6.31 /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.

6.31.1 Macro Definition Documentation

6.31.1.1 _GNU_SOURCE

```
#define _GNU_SOURCE
```

Definition at line 1 of file files.c.

6.31.2 Function Documentation

6.31.2.1 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.

Here is the caller graph for this function:

6.32 /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 !

6.32.1 Function Documentation

6.32.1.1 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:

6.32.1.2 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:

6.32.1.3 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:

6.32.1.4 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:

6.33 /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)

  Merges my neighbours list with the one sent by 'neighbour_id'.
- 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)
- size_t read_actual_height (int fd)
- int read_send_block (int fd)
- int read_vote (__attribute__((unused)) int fd)
- int read_epoch_block (int fd)
- int read_pending_transaction_list (__attribute__((unused)) int fd)

6.33.1 Function Documentation

6.33.1.1 fetch_client_list()

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

Merges my neighbours list with the one sent by 'neighbour_id'.

Parameters

sockfd	The sockfd to read

Returns

0 if sucess, -1 otherwise

Definition at line 95 of file get_data.c.

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

6.33.1.2 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:

6.33.1.3 read_actual_height()

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

Definition at line 172 of file get_data.c.

Here is the caller graph for this function:

6.33.1.4 read_epoch_block()

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

Definition at line 222 of file get_data.c.

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

6.33.1.5 read_get_blocks()

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

Definition at line 144 of file get_data.c.

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

6.33.1.6 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

Returns

0 if sucess, -1 otherwise

Definition at line 125 of file get_data.c.

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

6.33.1.7 read_pending_transaction_list()

```
int read_pending_transaction_list (
            __attribute__((unused)) int fd )
```

Definition at line 229 of file get_data.c.

Here is the caller graph for this function:

6.33.1.8 read_send_block()

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

Definition at line 178 of file get_data.c.

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

6.33.1.9 read_vote()

```
int read_vote (
            __attribute__((unused)) int fd )
```

Definition at line 217 of file get_data.c.

Here is the caller graph for this function:

6.34 /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 [ ]

6.34.1 Variable Documentation

6.34.1.1 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.

6.35 /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 (client_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 (__attribute__((unused)) int fd)

6.35.1 Function Documentation

6.35.1.1 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:

6.35.1.2 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:

6.35.1.3 send_get_blocks()

```
void send_get_blocks (
            client_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:

6.35.1.4 send_pending_transaction_list()

```
void send_pending_transaction_list (
            __attribute__((unused)) int fd )
```

Definition at line 104 of file send_data.c.

6.35.1.5 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:

6.35.1.6 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:

6.36 /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'.

6.36.1 Function Documentation

6.36.1.1 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:

6.36.1.2 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 77 of file server.c.

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

6.36.1.3 redirect_connection()

```
void* redirect_connection (
            void * arg )
```

Definition at line 43 of file server.c.

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

6.37 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.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 server.c:

Functions

- int main ()

6.37.1 Function Documentation

6.37.1.1 main()

```
int main ( )
```

Definition at line 7 of file server.c.

Here is the call graph for this function:

6.38 /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)
- 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_pkey_button_press (__attribute__((unused)) GtkWidget *widget, __attribute__((unused)) GdkEventKey *event, __attribute__((unused)) gpointer user_data)
- 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 * private_key_label
- GtkLabel * stake_label1
- GtkLabel * stake_label2
- GtkLabel * stake_label3
- GtkLabel * password_error_label
- GtkLabel * latest_block_name1
- GtkLabel * latest_block_name2
- GtkLabel * latest_block_name3
- GtkEntry * transa_amount
- GtkEntry * recipient_key
- 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

6.38.1 Function Documentation

6.38.1.1 add_contact()

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

Definition at line 437 of file ui.c.

Here is the call graph for this function:

6.38.1.2 add_contact_to_combobox()

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

Definition at line 466 of file ui.c.

Here is the caller graph for this function:

6.38.1.3 add_contacts_from_file()

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

Definition at line 474 of file ui.c.

Here is the caller graph for this function:

6.38.1.4 add_new_blockinfo()

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

Definition at line 199 of file ui.c.

6.38.1.5 add_transaction_from_file()

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

Definition at line 324 of file ui.c.

Here is the caller graph for this function:

6.38.1.6 add_transaction_with_contact()

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

Definition at line 304 of file ui.c.

Here is the caller graph for this function:

6.38.1.7 add_transaction_with_pkey()

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

Definition at line 283 of file ui.c.

Here is the caller graph for this function:

6.38.1.8 change_label_text()

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

Definition at line 194 of file ui.c.

Here is the caller graph for this function:

6.38.1.9 get_public_key_from_contacts()

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

Definition at line 511 of file ui.c.

Here is the caller graph for this function:

6.38.1.10 load_contacts_from_file()

```
void load_contacts_from_file ( )
```

Definition at line 483 of file ui.c.

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

6.38.1.11 load_transactions_from_file()

```
void load_transactions_from_file ( )
```

Definition at line 334 of file ui.c.

Here is the call graph for this function:

6.38.1.12 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 428 of file ui.c.

6.38.1.13 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 608 of file ui.c.

Here is the call graph for this function:

6.38.1.14 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 548 of file ui.c.

6.38.1.15 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 563 of file ui.c.

Here is the call graph for this function:

6.38.1.16 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 389 of file ui.c.

6.38.1.17 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 398 of file ui.c.

6.38.1.18 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 233 of file ui.c.

6.38.1.19 on_main_window_destroy()

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

Definition at line 242 of file ui.c.

6.38.1.20 on_pkey_button_press()

```
gboolean on_pkey_button_press (
            __attribute__((unused)) GtkWidget * widget,
            __attribute__((unused)) GdkEventKey * event,
            __attribute__((unused)) gpointer user_data )
```

Definition at line 371 of file ui.c.

6.38.1.21 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 408 of file ui.c.

6.38.1.22 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 417 of file ui.c.

6.38.1.23 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 249 of file ui.c.

Here is the call graph for this function:

6.38.1.24 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 63 of file ui.c.

Here is the caller graph for this function:

6.38.1.25 update_labels()

```
void update_labels ( )
```

Definition at line 658 of file ui.c.

Here is the call graph for this function:

6.38.1.26 update_sync()

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

Definition at line 214 of file ui.c.

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

6.38.2 Variable Documentation

6.38.2.1 balance_1

```
GtkLabel* balance_1
```

Definition at line 23 of file ui.c.

6.38.2.2 balance_2

```
GtkLabel* balance_2
```

Definition at line 24 of file ui.c.

6.38.2.3 contacts_combo

```
GtkComboBox* contacts_combo
```

Definition at line 56 of file ui.c.

6.38.2.4 cr1_combo

```
GtkCellRenderer* cr1_combo
```

Definition at line 58 of file ui.c.

6.38.2.5 cr1_con

```
GtkCellRenderer* cr1_con
```

Definition at line 46 of file ui.c.

6.38.2.6 cr1_th

```
GtkCellRenderer* cr1_th
```

Definition at line 53 of file ui.c.

6.38.2.7 cr2_con

```
GtkCellRenderer* cr2_con
```

Definition at line 47 of file ui.c.

6.38.2.8 cr2_th

```
GtkCellRenderer* cr2_th
```

Definition at line 54 of file ui.c.

6.38.2.9 cr3_th

```
GtkCellRenderer* cr3_th
```

Definition at line 55 of file ui.c.

6.38.2.10 cx1_con

```
GtkTreeViewColumn* cx1_con
```

Definition at line 44 of file ui.c.

6.38.2.11 cx1_th

```
GtkTreeViewColumn* cx1_th
```

Definition at line 50 of file ui.c.

6.38.2.12 cx2_con

```
GtkTreeViewColumn* cx2_con
```

Definition at line 45 of file ui.c.

6.38.2.13 cx2_th

```
GtkTreeViewColumn* cx2_th
```

Definition at line 51 of file ui.c.

6.38.2.14 cx3_th

```
GtkTreeViewColumn* cx3_th
```

Definition at line 52 of file ui.c.

6.38.2.15 invest_entry

```
GtkEntry* invest_entry
```

Definition at line 35 of file ui.c.

6.38.2.16 key_entry

```
GtkEntry* key_entry
```

Definition at line 41 of file ui.c.

6.38.2.17 latest_block_name1

```
GtkLabel* latest_block_name1
```

Definition at line 30 of file ui.c.

6.38.2.18 latest_block_name2

```
GtkLabel* latest_block_name2
```

Definition at line 31 of file ui.c.

6.38.2.19 latest_block_name3

```
GtkLabel* latest_block_name3
```

Definition at line 32 of file ui.c.

6.38.2.20 ls_combo

```
GtkListStore* ls_combo
```

Definition at line 57 of file ui.c.

6.38.2.21 name_entry_con

```
GtkEntry* name_entry_con
```

Definition at line 37 of file ui.c.

6.38.2.22 password_entry1

```
GtkEntry* password_entry1
```

Definition at line 39 of file ui.c.

6.38.2.23 password_entry2

```
GtkEntry* password_entry2
```

Definition at line 40 of file ui.c.

6.38.2.24 password_error_label

```
GtkLabel* password_error_label
```

Definition at line 29 of file ui.c.

6.38.2.25 private_key_label

```
GtkLabel* private_key_label
```

Definition at line 25 of file ui.c.

6.38.2.26 progress_bar_blockchain

```
GtkProgressBar* progress_bar_blockchain
```

Definition at line 59 of file ui.c.

6.38.2.27 public_key_entry_con

```
GtkEntry* public_key_entry_con
```

Definition at line 38 of file ui.c.

6.38.2.28 recipient_key

```
GtkEntry* recipient_key
```

Definition at line 34 of file ui.c.

6.38.2.29 recover_entry

```
GtkEntry* recover_entry
```

Definition at line 36 of file ui.c.

6.38.2.30 stake_label1

```
GtkLabel* stake_label1
```

Definition at line 26 of file ui.c.

6.38.2.31 stake_label2

```
GtkLabel* stake_label2
```

Definition at line 27 of file ui.c.

6.38.2.32 stake_label3

```
GtkLabel* stake_label3
```

Definition at line 28 of file ui.c.

6.38.2.33 transa_amount

```
GtkEntry* transa_amount
```

Definition at line 33 of file ui.c.

6.38.2.34 ts_con

```
GtkTreeStore* ts_con
```

Definition at line 43 of file ui.c.

6.38.2.35 ts_th

```
GtkTreeStore* ts_th
```

Definition at line 49 of file ui.c.

6.38.2.36 tv_con

```
GtkTreeView* tv_con
```

Definition at line 42 of file ui.c.

6.38.2.37 tv_th

```
GtkTreeView* tv_th
```

Definition at line 48 of file ui.c.

6.39 /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.

6.39.1 Function Documentation

6.39.1.1 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*

Definition at line 3 of file epoch_man.c.

Here is the caller graph for this function:

6.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validation_engine.c File Reference

6.41 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validators.c File Reference

```
#include "validation/validators.h"
#include "misc/math.h"
```

Include dependency graph for validators.c:

Macros

- #define NB_RSA_CHUNK 2048/64

Functions

- uint16_t define_nb_validators (size_t n)
- char * hash_block_transactions_epoch (Block *block)
- RSA ** get_comittee (size_t block_height, size_t *nb_validators)

  Get the a comittee RSA public keys on a specific epoch.
- RSA ** get_next_comittee (size_t *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 (char pkey[ ])

  Get the validator id in 'validators.state'.

6.41.1 Macro Definition Documentation

6.41.1.1 NB_RSA_CHUNK

```
#define NB_RSA_CHUNK 2048/64
```

Definition at line 4 of file validators.c.

6.41.2 Function Documentation

6.41.2.1 define_nb_validators()

```
uint16_t define_nb_validators (
            size_t n )
```

Definition at line 6 of file validators.c.

Here is the caller graph for this function:

6.41.2.2 get_comittee()

```
RSA** get_comittee (
            size_t block_height,
            size_t * 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 32 of file validators.c.

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

6.41.2.3 get_next_comittee()

```
RSA** get_next_comittee (
            size_t * 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 124 of file validators.c.

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

6.41.2.4 get_validator_id()

```
ssize_t get_validator_id (
            char pkey[] )
```

Get the validator id in 'validators.state'.

Parameters

pkey	The string public key (without -–—BEGIN RSA KEY-–— & -–—END RSA KEY-–—)

Returns

ssize_t, the validator index

Definition at line 209 of file validators.c.

Here is the call graph for this function:

6.41.2.5 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 188 of file validators.c.

Here is the call graph for this function:

6.41.2.6 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 176 of file validators.c.

Here is the call graph for this function:

6.41.2.7 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 164 of file validators.c.

Here is the call graph for this function:

6.41.2.8 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 152 of file validators.c.

Here is the call graph for this function:

6.41.2.9 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 141 of file validators.c.

Here is the call graph for this function:

6.41.2.10 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 129 of file validators.c.

Here is the call graph for this function:

6.41.2.11 hash_block_transactions_epoch()

```
char* hash_block_transactions_epoch (
            Block * block )
```

Definition at line 20 of file validators.c.

Here is the call graph for this function:

6.42 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c File Reference

```
#include "ui/ui.h"
```
Include dependency graph for gui.c:

Functions

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

6.42.1 Function Documentation

6.42.1.1 main()

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

Definition at line 3 of file gui.c.

Here is the call graph for this function:

6.43 /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 ()

6.43.1 Function Documentation

6.43.1.1 main()

```
int main ( )
```

Definition at line 10 of file serverdoor.c.

Here is the call graph for this function:

6.44 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c File Reference

```
#include "network/network.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/rsa.h"
#include "cryptosystem/hash.h"
```

Include dependency graph for sign.c:

Functions

- int main ()

6.44.1 Function Documentation

6.44.1.1 main()

```
int main ( )
```

Definition at line 10 of file sign.c.

Here is the call graph for this function:

6.45 /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)

6.45.1 Macro Definition Documentation

6.45.1.1 GEN_BLC_F_C

```
#define GEN_BLC_F_C
```

Definition at line 2 of file GEN_blockchain_files.c.

6.45.2 Function Documentation

6.45.2.1 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:

6.45.2.2 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:

6.46 /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 "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.

6.46.1 Macro Definition Documentation

6.46.1.1 GEN_VALIDATORS_FILE_H

```
#define GEN_VALIDATORS_FILE_H
```

Definition at line 2 of file GEN_validators_file.c.

6.46.1.2 NB_FAKE_VALIDATORS

```
#define NB_FAKE_VALIDATORS 10
```

Definition at line 14 of file GEN_validators_file.c.

6.46.1.3 str

```
#define str(
              x ) #x
```

Definition at line 15 of file GEN_validators_file.c.

6.46.2 Function Documentation

6.46.2.1 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 + 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 31 of file GEN_validators_file.c.

Here is the caller graph for this function:

6.47 /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)

6.47.1 Function Documentation

6.47.1.1 block_test()

```
void block_test (
            void  )
```

Definition at line 13 of file block_test.c.

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

6.48 /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 ()

6.48.1 Function Documentation

6.48.1.1 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:

6.48.1.2 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:

6.49 /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 ()

6.49.1 Function Documentation

6.49.1.1 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:

6.50 /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 ()

6.50.1 Function Documentation

6.50.1.1 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:

6.51 /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 ()

6.51.1 Function Documentation

6.51.1.1 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:

6.52 /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 ()

6.52.1 Macro Definition Documentation

6.52.1.1 MAIN_TEST_C

```
#define MAIN_TEST_C
```

Definition at line 2 of file main_test.c.

6.52.2 Function Documentation

6.52.2.1 main()

```
int main ( )
```

Definition at line 5 of file main_test.c.

Here is the call graph for this function:

6.53 /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)

6.53.1 Macro Definition Documentation

6.53.1.1 BLOCK_TEST_C

```
#define BLOCK_TEST_C
```

Definition at line 2 of file block_test.c.

6.53.1.2 NB_BLOCK_PER_CHUNK

```
#define NB_BLOCK_PER_CHUNK 10
```

Definition at line 9 of file block_test.c.

6.53.1.3 NB_MOCK_BLOCKS

```
#define NB_MOCK_BLOCKS 13
```

Definition at line 11 of file block_test.c.

6.53.2 Function Documentation

6.53.2.1 block_test()

```
void block_test (
            void  )
```

Definition at line 13 of file block_test.c.

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

6.54 /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 ()

6.54.1 Macro Definition Documentation

6.54.1.1 RSA_SIZE_C

```
#define RSA_SIZE_C
```

Definition at line 2 of file rsa_test.c.

6.54.2 Function Documentation

6.54.2.1 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:

6.54.2.2 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:

6.55 /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 ()

6.55.1 Function Documentation

6.55.1.1 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:

6.56 /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

- client_connection * client_connections

6.56.1 Macro Definition Documentation

6.56.1.1 CLIENT_TEST_C

```
#define CLIENT_TEST_C
```

Definition at line 2 of file client_test.c.

6.56.2 Function Documentation

6.56.2.1 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:

6.56.3 Variable Documentation

6.56.3.1 client_connections

```
client_connection* client_connections
```

Definition at line 4 of file client.c.

6.57 /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 ()

6.57.1 Function Documentation

6.57.1.1 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:

6.58 /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...)

6.58.1 Macro Definition Documentation

6.58.1.1 DEBUG

```
#define DEBUG(
              function )
```

Value:

```
    printf("Testing '%s'...\n", #function); \
    function()
```

Definition at line 5 of file tests_macros.h.

6.58.1.2 LOG

```
#define LOG(
              str...  )
```

Value:

```
    printf("\033[0;34m[-]  "); \
    printf(str);               \
    printf("\033[0m\n")
```

Definition at line 9 of file tests_macros.h.

6.58.1.3 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.

6.58.1.4 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.

6.58.1.5 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.

6.59 /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:

Functions

- int main ()

6.59.1 Function Documentation

6.59.1.1 main()

```
int main ( )
```

Definition at line 8 of file unit_testing.c.

Here is the call graph for this function:

Index