PEPITAS CRYPTOCURRENCY

Generated by Doxygen 1.8.17

Chapter 1

PEPITAS

C cryptocurrency.

1.1 CODING STYLE

1.1.1 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`.

1.1.2 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 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 Block Struct Reference

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

Collaboration diagram for Block:

Data Fields

- uint16_t chunk_id
- BlockData block_data
- size_t signature_len
- char ∗ block_signature

4.1.1 Detailed Description

Definition at line 31 of file block.h.

4.1.2 Field Documentation

4.1.2.1 block_data

```
BlockData block_data
```

Definition at line 34 of file block.h.

4.1.2.2 block_signature

```
char* block_signature
```

Definition at line 37 of file block.h.

4.1.2.3 chunk_id

```
uint16_t chunk_id
```

Definition at line 33 of file block.h.

4.1.2.4 signature_len

```
size_t signature_len
```

Definition at line 36 of file block.h.

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

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

4.2 BlockData Struct Reference

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

Collaboration diagram for BlockData:

Data Fields

- char magic
- char previous_block_hash [SHA384_DIGEST_LENGTH *2+1]
- size_t height
- uint16_t nb_transactions
- Transaction ** transactions
- RSA * validator_public_key
- time_t block_timestamp

4.2.1 Detailed Description

Definition at line 17 of file block.h.

4.2.2 Field Documentation

4.2.2.1 block_timestamp

```
time_t block_timestamp
```

Definition at line 28 of file block.h.

4.2.2.2 height

```
size_t height
```

Definition at line 21 of file block.h.

4.2.2.3 magic

```
char magic
```

Definition at line 19 of file block.h.

4.2.2.4 nb_transactions

```
uint16_t nb_transactions
```

Definition at line 23 of file block.h.

4.2.2.5 previous_block_hash

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

Definition at line 20 of file block.h.

4.2.2.6 transactions

```
Transaction** transactions
```

Definition at line 24 of file block.h.

4.2.2.7 validator_public_key

```
RSA* validator_public_key
```

Definition at line 27 of file block.h.

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

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

4.3 ChunkBlockchain Struct Reference

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

Collaboration diagram for ChunkBlockchain:

Data Fields

- size_t chunk_nb
- Block ** chunk

4.3.1 Detailed Description

Definition at line 41 of file block.h.

4.3.2 Field Documentation

4.3.2.1 chunk

```
Block** chunk
```

Definition at line 44 of file block.h.

4.3.2.2 chunk_nb

```
size_t chunk_nb
```

Definition at line 43 of file block.h.

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

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

4.4 client_connection Struct Reference

```
#include <server.h>
```

Data Fields

- struct addrinfo info
- int socket

4.4.1 Detailed Description

Definition at line 8 of file server.h.

4.4.2 Field Documentation

4.4.2.1 info

```
struct addrinfo info
```

Definition at line 10 of file server.h.

4.4.2.2 socket

```
int socket
```

Definition at line 11 of file server.h.

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

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

4.5 Neighbour Struct Reference

```
#include <client.h>
```

Data Fields

- int family
- char ∗ hostname
- int server_sockfd
- int client_sockfd

4.5.1 Detailed Description

Definition at line 8 of file client.h.

4.5.2 Field Documentation

4.5.2.1 client_sockfd

```
int client_sockfd
```

Definition at line 13 of file client.h.

4.5.2.2 family

```
int family
```

Definition at line 10 of file client.h.

4.5.2.3 hostname

```
char* hostname
```

Definition at line 11 of file client.h.

4.5.2.4 server_sockfd

```
int server_sockfd
```

Definition at line 12 of file client.h.

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

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

4.6 Node Struct Reference

```
#include <client.h>
```

Collaboration diagram for Node:

Data Fields

- Neighbour ∗ neighbours

4.6.1 Detailed Description

Definition at line 16 of file client.h.

4.6.2 Field Documentation

4.6.2.1 neighbours

```
Neighbour∗ neighbours
```

Definition at line 18 of file client.h.

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

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

4.7 Transaction Struct Reference

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

Collaboration diagram for Transaction:

Data Fields

- TransactionData ∗ transaction_data
- size_t signature_len
- char ∗ transaction_signature

4.7.1 Detailed Description

Definition at line 28 of file transaction.h.

4.7.2 Field Documentation

4.7.2.1 signature_len

```
size_t signature_len
```

Definition at line 32 of file transaction.h.

4.7.2.2 transaction_data

```
TransactionData* transaction_data
```

Definition at line 30 of file transaction.h.

4.7.2.3 transaction_signature

```
char* transaction_signature
```

Definition at line 33 of file transaction.h.

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

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

4.8 TransactionData Struct Reference

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

Data Fields

- 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]

4.8.1 Detailed Description

Definition at line 11 of file transaction.h.

4.8.2 Field Documentation

4.8.2.1 amount

```
size_t amount
```

Definition at line 17 of file transaction.h.

4.8.2.2 asset

```
char asset[512]
```

Definition at line 25 of file transaction.h.

4.8.2.3 cause

```
char cause[512]
```

Definition at line 24 of file transaction.h.

4.8.2.4 organisation_public_key

```
RSA* organisation_public_key
```

Definition at line 16 of file transaction.h.

4.8.2.5 receiver_public_key

```
RSA* receiver_public_key
```

Definition at line 15 of file transaction.h.

4.8.2.6 receiver_remaining_money

```
size_t receiver_remaining_money
```

Definition at line 19 of file transaction.h.

4.8.2.7 sender_public_key

```
RSA* sender_public_key
```

Definition at line 14 of file transaction.h.

4.8.2.8 sender_remaining_money

```
size_t sender_remaining_money
```

Definition at line 18 of file transaction.h.

4.8.2.9 transaction_timestamp

```
time_t transaction_timestamp
```

Definition at line 20 of file transaction.h.

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

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

4.9 Wallet Struct Reference

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

Data Fields

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

4.9.1 Detailed Description

Definition at line 10 of file wallet.h.

4.9.2 Field Documentation

4.9.2.1 amount

```
size_t amount
```

Definition at line 15 of file wallet.h.

4.9.2.2 is_validator

```
char is_validator
```

Definition at line 16 of file wallet.h.

4.9.2.3 priv_key

```
RSA* priv_key
```

Definition at line 12 of file wallet.h.

4.9.2.4 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/core/blockchain/wallet.h

Chapter 5

File Documentation

5.1 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/headers/core/blockchain/block.h File Reference

```
#include <stdlib.h>
#include <openssl/sha.h>
#include "transaction.h"
```
Include dependency graph for block.h:

5.2 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/headers/core/blockchain/transaction.h File Reference

```
#include <stdlib.h>
#include <openssl/rsa.h>
#include <openssl/sha.h>
#include <time.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

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.

5.2.1 Macro Definition Documentation

5.2.1.1 TRANSACTION_DATA_SIZE

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

Definition at line 9 of file transaction.h.

5.2.1.2 TRANSACTION_SIZE

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

Definition at line 10 of file transaction.h.

5.2.2 Typedef Documentation

5.2.2.1 Transaction

```
typedef struct Transaction Transaction
```

5.2.2.2 TransactionData

```
typedef struct TransactionData TransactionData
```

5.2.3 Function Documentation

5.2.3.1 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

5.3 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/headers/core/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.

5.3.1 Typedef Documentation

5.3.1.1 Wallet

```
typedef struct Wallet Wallet
```

5.3.2 Function Documentation

5.3.2.1 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:

5.3.2.2 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:

5.4 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/headers/core/validation/stake.h File Reference

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

Include dependency graph for stake.h:

Functions

- int push_stake (size_t amount)

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

  Pops an amount on the stake.

5.4.1 Function Documentation

5.4.1.1 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

5.4.1.2 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

5.5 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/headers/core/validation/validations.h File Reference

```
#include <stdlib.h>
#include <openssl/rsa.h>
```
Include dependency graph for validations.h: This graph shows which files directly or indirectly include this file:

Functions

- RSA ∗∗ get_next_committee (size_t ∗nb_validators)

    Get the 'next block' validators RSA public keys.
- ssize_t get_amount (RSA ∗public_key)

    Searches how much money 'public_key' has.

5.5.1 Function Documentation

5.5.1.1 get_amount()

```
ssize_t get_amount (
            RSA * public_key )
```

Searches how much money 'public_key' has.

Parameters

public_key	The RSA public key

Returns

The amount, or -1 in case of an error

5.5.1.2 get_next_committee()

```
RSA** get_next_committee (
            size_t * nb_validators )
```

Get the 'next block' validators RSA public keys.

Parameters

nb_validators	return value, the number of selected validators

See also

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

Returns

[∗RSA]

Definition at line 31 of file validations.c.

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

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

```
#include <stdlib.h>
#include "core/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.

5.6.1 Function Documentation

5.6.1.1 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:

5.6.1.2 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:

5.7 /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 ()

  Get the keys object.

5.7.1 Macro Definition Documentation

5.7.1.1 RSA_BEGIN_SIZE

```
#define RSA_BEGIN_SIZE 31
```

Definition at line 6 of file rsa.h.

5.7.1.2 RSA_END_SIZE

```
#define RSA_END_SIZE 29
```

Definition at line 7 of file rsa.h.

5.7.1.3 RSA_FILE_TOTAL_SIZE

```
#define RSA_FILE_TOTAL_SIZE 426
```

Definition at line 5 of file rsa.h.

5.7.1.4 RSA_KEY_SIZE

```
#define RSA_KEY_SIZE 366
```

Definition at line 4 of file rsa.h.

5.7.2 Function Documentation

5.7.2.1 get_keys()

```
void get_keys ( )
```

Get the keys object.

Definition at line 21 of file rsa.c.

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

5.8 /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 "core/blockchain/wallet.h"
#include "core/blockchain/block.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, size_t ∗signature_len)

  encrypt(SHA284(msg,len_data),priv_key)
- int verify_signature (void ∗data, size_t data_len, char ∗signature, size_t signature_len, RSA ∗pub_key)

  Apply the SHA384 algorithm on a 'data' of size 'len_data' and verifies if SHA384(data, len_data) == 'signature'.
- 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.
- char ∗ get_blockdata_data (Block ∗block, size_t ∗size)

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

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

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

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

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

  Signs transactions of a block.

5.8.1 Function Documentation

5.8.1.1 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 144 of file signature.c.

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

5.8.1.2 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 93 of file signature.c.

Here is the caller graph for this function:

5.8.1.3 sign_block()

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

Signs a block.

Parameters

block	The block to sign

Definition at line 233 of file signature.c.

Here is the call graph for this function:

5.8.1.4 sign_block_transactions()

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

Signs transactions of a block.

Parameters

block	The block to sign

Definition at line 258 of file signature.c.

Here is the call graph for this function:

5.8.1.5 sign_message()

```
char* sign_message (
             char * data,
             size_t len_data,
             size_t * signature_len )
```

encrypt(SHA284(msg,len_data),priv_key)

Parameters

data	The data to sign
len_data	The length of the data
signature_len	The length of the data signature

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:

5.8.1.6 sign_transaction()

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

Sign a transaction.

Parameters

transaction	The transaction to sign

Definition at line 245 of file signature.c.

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

5.8.1.7 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 206 of file signature.c.

Here is the call graph for this function:

5.8.1.8 verify_signature()

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

Apply the SHA384 algorithm on a 'data' of size 'len_data' and verifies if SHA384(data, len_data) == 'signature'.

Parameters

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

Returns

int

Definition at line 31 of file signature.c.

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

5.8.1.9 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 219 of file signature.c.

Here is the call graph for this function:

5.8.1.10 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 199 of file signature.c.

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

5.8.1.11 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 174 of file signature.c.

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

5.9 /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.

5.9.1 Function Documentation

5.9.1.1 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:

5.10 /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)

5.10.1 Macro Definition Documentation

5.10.1.1 MAX

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

Definition at line 2 of file math.h.

5.10.1.2 MIN

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

Definition at line 1 of file math.h.

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

```
#include <stdlib.h>
#include <err.h>
#include <unistd.h>
#include <string.h>
#include <errno.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.
- 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 !

5.11.1 Function Documentation

5.11.1.1 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 40 of file safe.c.

Here is the caller graph for this function:

5.11.1.2 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 18 of file safe.c.

Here is the caller graph for this function:

5.11.1.3 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 4 of file safe.c.

Here is the caller graph for this function:

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

```
#include <stddef.h>
```

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

Data Structures

- struct Neighbour
- struct Node

Macros

- #define MAX_NEIGHBOURS 64

Typedefs

- typedef struct Neighbour Neighbour
- typedef struct Node Node

Functions

- Node ∗ get_my_node ()

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

  Sets a neighbour in the client.neightbours section.
- int listen_to (size_t neighbour_id)

  Tries to connect to the peer-to-peer network via a node in the Node structure.
- int ping_client (size_t neighbour_id)

  Pings the client side of 'neighbour_id' and deletes it from struct Node if there is no response.

5.12.1 Macro Definition Documentation

5.12.1.1 MAX_NEIGHBOURS

```
#define MAX_NEIGHBOURS 64
```

Definition at line 6 of file client.h.

5.12.2 Typedef Documentation

5.12.2.1 Neighbour

```
typedef struct Neighbour Neighbour
```

5.12.2.2 Node

```
typedef struct Node Node
```

5.12.3 Function Documentation

5.12.3.1 get_my_node()

```
Node* get_my_node ( )
```

Get the my node object.

Returns

Node*

Definition at line 5 of file client.c.

Here is the caller graph for this function:

5.12.3.2 listen_to()

```
int listen_to (
             size_t neighbour_id )
```

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

Parameters

neighbour↩ _id	The neighbour's index (in struct Node) to connect with

Returns

socket FD or -1 if an error occurs

Definition at line 57 of file client.c.

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

5.12.3.3 ping_client()

```
int ping_client (
             size_t neighbour_id )
```

Pings the client side of 'neighbour_id' and deletes it from struct Node if there is no response.

Parameters

neighbour↩ _id	

Returns

0 if sucess, -1 otherwise

5.12.3.4 set_neighbour()

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

Sets a neighbour in the client.neightbours section.

Returns

0 if sucess, -1 otherwise

Definition at line 14 of file client.c.

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

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

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

Functions

- int read_header (int sockfd)

  Waits a header in 'sockfd', reads it and processes it.
- int fetch_client_list (int neighbour_id)

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

5.13.1 Function Documentation

5.13.1.1 fetch_client_list()

```
int fetch_client_list (
            int neighbour_id )
```

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

Parameters

neighbour↩ _id	The id of the neighbour list to merge

Returns

0 if sucess, -1 otherwise

Definition at line 32 of file get_data.c.

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

5.13.1.2 read_header()

```
int read_header (
            int sockfd )
```

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

Parameters

sockfd	The sock FD

Returns

0 if sucess, -1 otherwise

Definition at line 86 of file get_data.c.

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

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

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/un.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <err.h>
#include <string.h>
#include <pthread.h>
#include <arpa/inet.h>
#include "misc/safe.h"
#include "client.h"
```

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

Macros

- #define NB_HARD_CODED_ADDR 2
- #define STATIC_PORT "4242"
- #define HD_GET_CLIENT_LIST "GET CLIENT LIST\r\n\r\n"
- #define HD_SEND_CLIENT_LIST "SEND CLIENT LIST\n"
- #define HD_GET_BLOCKCHAIN "GET BLOCKCHAIN\r\n\r\n"
- #define HD_SEND_BLOCKCHAIN "SEND BLOCKCHAIN\n"

Variables

- const Neighbour HARD_CODED_ADDR [ ]

5.14.1 Macro Definition Documentation

5.14.1.1 HD_GET_BLOCKCHAIN

```
#define HD_GET_BLOCKCHAIN "GET BLOCKCHAIN\r\n\r\n"
```

Definition at line 25 of file network.h.

5.14.1.2 HD_GET_CLIENT_LIST

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

Definition at line 23 of file network.h.

5.14.1.3 HD_SEND_BLOCKCHAIN

```
#define HD_SEND_BLOCKCHAIN "SEND BLOCKCHAIN\n"
```

Definition at line 26 of file network.h.

5.14.1.4 HD_SEND_CLIENT_LIST

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

Definition at line 24 of file network.h.

5.14.1.5 NB_HARD_CODED_ADDR

```
#define NB_HARD_CODED_ADDR 2
```

Definition at line 17 of file network.h.

5.14.1.6 STATIC_PORT

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

Definition at line 20 of file network.h.

5.14.2 Variable Documentation

5.14.2.1 HARD_CODED_ADDR

```
const Neighbour HARD_CODED_ADDR[ ]
```

Definition at line 4 of file network.c.

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

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

Functions

- int send_client_list (int sockfd)
  Sends my client list to a node via 'sockfd'.

5.15.1 Function Documentation

5.15.1.1 send_client_list()

```
int send_client_list (
             int sockfd )
```

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:

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

```
#include <sys/socket.h>
#include "network.h"
#include "core/blockchain/block.h"
```

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

Data Structures

- struct client_connection

Typedefs

- typedef struct client_connection client_connection

Functions

- int init_server ()
  Launches a server instance, connected to the peer-to-peer network 'hostname'.
- int send_block (Block block, int sockfd)
  Sends a block to a user via a socket FD.

5.16.1 Typedef Documentation

5.16.1.1 client_connection

```
typedef struct client_connection client_connection
```

5.16.2 Function Documentation

5.16.2.1 init_server()

```
int init_server ( )
```

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

Returns

0 if success, -1 otherwise

Definition at line 30 of file server.c.

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

5.16.2.2 send_block()

```
int send_block (
            Block block,
            int sockfd )
```

Sends a block to a user via a socket FD.

Parameters

sockfd	The socket FD
block	The block to send

Returns

int

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

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

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

Functions

- int setup ()

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.

5.17.1 Function Documentation

5.17.1.1 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

5.17.1.2 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

5.17.1.3 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

5.17.1.4 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

5.17.1.5 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 126 of file ui.c.

5.17.1.6 on_main_window_destroy()

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

Quits GTK when the program ends.

5.17.1.7 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

5.17.1.8 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

5.17.1.9 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

5.17.1.10 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

5.17.1.11 setup()

```
int setup ( )
```

Setups the gtk widgets for the GUI.

Returns

int Returns 1 if there is an error, 0 otherwise

Definition at line 45 of file ui.c.

Here is the caller graph for this function:

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

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

```
#include <signal.h>
#include <stdlib.h>
#include "network/network.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
```
Include dependency graph for client.c:

Functions

- int main ()

5.19.1 Function Documentation

5.19.1.1 main()

```
int main ( )
```

Definition at line 10 of file client.c.

Here is the call graph for this function:

5.20 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/network/client.c File Reference

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

Functions

- Node ∗ get_my_node ()

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

  Sets a neighbour in the client.neightbours section.
- int listen_to (size_t neighbour_id)

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

5.20.1 Function Documentation

5.20.1.1 get_my_node()

```
Node* get_my_node ( )
```

Get the my node object.

Returns

Node∗

Definition at line 5 of file client.c.

Here is the caller graph for this function:

5.20.1.2 listen_to()

```
int listen_to (
            size_t neighbour_id )
```

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

Parameters

neighbour↩_id	The neighbour's index (in struct Node) to connect with

Returns

socket FD or -1 if an error occurs

Definition at line 57 of file client.c.

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

5.20.1.3 set_neighbour()

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

Sets a neighbour in the client.neightbours section.

Returns

0 if sucess, -1 otherwise

Definition at line 14 of file client.c.

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

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

```
#include "core/blockchain/block.h"
#include "cryptosystem/signature.h"
#include <sys/stat.h>
#include <unistd.h>
#include <err.h>
#include <errno.h>
#include <openssl/rsa.h>
#include <openssl/crypto.h>
#include <fcntl.h>
#include <sys/types.h>
```

Include dependency graph for block.c:

Functions

- ChunkBlockchain ∗ get_blockchain (size_t nb_chunk)

  Loads a blockchain object with a padding of 'nb_chunk'.
- void write_block_file (Block block)

  Writes a block struct in a file.
- void convert_data_to_transactiondata (TransactionData ∗transactiondata, FILE ∗blockfile)
- void convert_data_to_transaction (Transaction ∗transaction, FILE ∗blockfile)
- void convert_data_to_blockdata (BlockData ∗blockdata, FILE ∗blockfile)
- void convert_data_to_block (Block ∗block, FILE ∗blockfile)
- Block ∗ get_block (size_t block_height)
- 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 $h-1$

5.21.1 Function Documentation

5.21.1.1 convert_data_to_block()

```
void convert_data_to_block (
            Block * block,
            FILE * blockfile )
```

Definition at line 142 of file block.c.

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

5.21.1.2 convert_data_to_blockdata()

```
void convert_data_to_blockdata (
            BlockData * blockdata,
            FILE * blockfile )
```

Definition at line 116 of file block.c.

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

5.21.1.3 convert_data_to_transaction()

```
void convert_data_to_transaction (
            Transaction * transaction,
            FILE * blockfile )
```

Definition at line 106 of file block.c.

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

5.21.1.4 convert_data_to_transactiondata()

```
void convert_data_to_transactiondata (
            TransactionData * transactiondata,
            FILE * blockfile )
```

Definition at line 69 of file block.c.

Here is the caller graph for this function:

5.21.1.5 free_block()

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

Free a block struct.

Parameters

block	The block to free

Definition at line 168 of file block.c.

Here is the caller graph for this function:

5.21.1.6 get_block()

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

Definition at line 150 of file block.c.

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

5.21.1.7 get_blockchain()

```
ChunkBlockchain* get_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 12 of file block.c.

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

5.21.1.8 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 184 of file block.c.

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

5.21.1.9 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 194 of file block.c.

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

5.21.1.10 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 51 of file block.c.

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

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

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

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

5.22.1 Function Documentation

5.22.1.1 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:

5.22.1.2 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:

5.23 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validations.c File Reference

```
#include "core/validation/validations.h"
#include "core/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 dependency graph for validations.c:

Macros

- #define NB_RSA_CHUNK 2048 / 64
- #define MAX_VALIDATORS_PER_BLOCK 10000

Functions

- uint16_t define_nb_validators (size_t n)
- RSA ** get_next_committee (size_t *nb_validators)

  Get the 'next block' validators RSA public keys.

5.23.1 Macro Definition Documentation

5.23.1.1 MAX_VALIDATORS_PER_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 10000
```

Definition at line 14 of file validations.c.

5.23.1.2 NB_RSA_CHUNK

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

Definition at line 13 of file validations.c.

5.23.2 Function Documentation

5.23.2.1 define_nb_validators()

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

Definition at line 16 of file validations.c.

Here is the caller graph for this function:

5.23.2.2 get_next_committee()

```
RSA** get_next_committee (
            size_t * nb_validators )
```

Get the 'next block' validators RSA public keys.

Parameters

nb_validators	return value, the number of selected validators

See also

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

Returns

[∗RSA]

Definition at line 31 of file validations.c.

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

5.24 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-←Cryptocurrency/src/cryptosystem/hash.c File Reference

```
#include <openssl/sha.h>
#include "cryptosystem/hash.h"
#include "core/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.

5.24.1 Function Documentation

5.24.1.1 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:

5.24.1.2 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:

5.25 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/cryptosystem/rsa.c File Reference

```
#include "cryptosystem/rsa.h"
#include "core/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 ()

  Get the keys object.

5.25.1 Macro Definition Documentation

5.25.1.1 RSA_NUM_E

```
#define RSA_NUM_E 3
```

Definition at line 16 of file rsa.c.

5.25.2 Function Documentation

5.25.2.1 get_keys()

```
void get_keys ( )
```

Get the keys object.

Definition at line 21 of file rsa.c.

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

5.26 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/cryptosystem/signature.c File Reference

```
#include "core/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, size_t ∗signature_len)

    encrypt(SHA284(msg,len_data),priv_key)
- int verify_signature (void ∗data, size_t data_len, char ∗signature, size_t signature_len, RSA ∗pub_key)

    Apply the SHA384 algorithm on a 'data' of size 'len_data' and verifies if SHA384(data, len_data) == 'signature'.
- 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.
- 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.
- 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_transaction (Transaction ∗transaction)

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

    Signs transactions of a block.

5.26.1 Function Documentation

5.26.1.1 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 144 of file signature.c.

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

5.26.1.2 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 93 of file signature.c.

Here is the caller graph for this function:

5.26.1.3 sign_block()

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

Signs a block.

Parameters

block	The block to sign

Definition at line 233 of file signature.c.

Here is the call graph for this function:

5.26.1.4 sign_block_transactions()

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

Signs transactions of a block.

Parameters

block	The block to sign

Definition at line 258 of file signature.c.

Here is the call graph for this function:

5.26.1.5 sign_message()

```
char* sign_message (
             char * data,
             size_t len_data,
             size_t * signature_len )
```

encrypt(SHA284(msg,len_data),priv_key)

Parameters

data	The data to sign
len_data	The length of the data
signature_len	The length of the data signature

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:

5.26.1.6 sign_transaction()

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

Sign a transaction.

Parameters

transaction	The transaction to sign

Definition at line 245 of file signature.c.

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

5.26.1.7 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 206 of file signature.c.

Here is the call graph for this function:

5.26.1.8 verify_signature()

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

Apply the SHA384 algorithm on a 'data' of size 'len_data' and verifies if SHA384(data, len_data) == 'signature'.

Parameters

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

Returns

int

Definition at line 31 of file signature.c.

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

5.26.1.9 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 219 of file signature.c.

Here is the call graph for this function:

5.26.1.10 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 199 of file signature.c.

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

5.26.1.11 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 174 of file signature.c.

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

5.26.1.12 write_transaction()

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

Definition at line 86 of file signature.c.

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

5.26.1.13 write_transactiondata()

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

Definition at line 50 of file signature.c.

Here is the caller graph for this function:

5.27 /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)

5.27.1 Function Documentation

5.27.1.1 main()

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

Definition at line 3 of file gui.c.

Here is the call graph for this function:

5.28 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↵ Cryptocurrency/src/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.

5.28.1 Macro Definition Documentation

5.28.1.1 _GNU_SOURCE

```
#define _GNU_SOURCE
```

Definition at line 1 of file files.c.

5.28.2 Function Documentation

5.28.2.1 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:

5.29 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/misc/safe.c File Reference

```
#include <stdio.h>
#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.
- 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 !

5.29.1 Function Documentation

5.29.1.1 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 40 of file safe.c.

Here is the caller graph for this function:

5.29.1.2 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 18 of file safe.c.

Here is the caller graph for this function:

5.29.1.3 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 4 of file safe.c.

Here is the caller graph for this function:

5.30 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/network/get_data.c File Reference

```
#include "network/client.h"
#include "network/server.h"
#include "network/network.h"
#include "network/send_data.h"
#include "network/get_data.h"
```

Include dependency graph for get_data.c:

Functions

- int process_header (char ∗header, int sockfd)
- int fetch_client_list (int neighbour_id)

  Merges my neighbours list with the one sent by 'neighbour_id'.
- int read_header (int sockfd)

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

5.30.1 Function Documentation

5.30.1.1 fetch_client_list()

```
int fetch_client_list (
            int neighbour_id )
```

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

Parameters

neighbour↩ _id	The id of the neighbour list to merge

Returns

0 if sucess, -1 otherwise

Definition at line 32 of file get_data.c.

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

5.30.1.2 process_header()

```
int process_header (
            char * header,
            int sockfd )
```

Definition at line 7 of file get_data.c.

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

5.30.1.3 read_header()

```
int read_header (
            int sockfd )
```

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

Parameters

sockfd	The sock FD

Returns

0 if sucess, -1 otherwise

Definition at line 86 of file get_data.c.

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

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

```
#include "network/client.h"
#include <arpa/inet.h>
```
Include dependency graph for network.c:

Variables

- const Neighbour HARD_CODED_ADDR [ ]

5.31.1 Variable Documentation

5.31.1.1 HARD_CODED_ADDR

```
const Neighbour HARD_CODED_ADDR[ ]
```

Initial value:

```
=
{
    {AF_INET, "34.72.117.116", 0, 0},
    {AF_INET, "127.0.0.1", 0, 0}
}
```

Definition at line 4 of file network.c.

5.32 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/network/send_data.c File Reference

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

Include dependency graph for send_data.c:

Functions

- int send_client_list (int sockfd)

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

5.32.1 Function Documentation

5.32.1.1 send_client_list()

```
int send_client_list (
            int sockfd )
```

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:

5.33 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/network/server.c File Reference

```
#include "network/server.h"
#include "network/client.h"
#include "network/get_data.h"
#include "network/network.h"
#include "misc/safe.h"
```

Include dependency graph for server.c:

Functions

- void ∗ accept_connection (void ∗arg)
- int init_server ()

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

5.33.1 Function Documentation

5.33.1.1 accept_connection()

```
void* accept_connection (
            void * arg )
```

Definition at line 7 of file server.c.

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

5.33.1.2 init_server()

```
int init_server ( )
```

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

Returns

0 if success, -1 otherwise

Definition at line 30 of file server.c.

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

5.34 /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 "core/blockchain/block.h"
#include <time.h>
```

Include dependency graph for server.c:

Functions

- int main ()

5.34.1 Function Documentation

5.34.1.1 main()

```
int main ( )
```

Definition at line 7 of file server.c.

Here is the call graph for this function:

5.35 /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 ()

5.35.1 Function Documentation

5.35.1.1 main()

```
int main ( )
```

Definition at line 10 of file sign.c.

Here is the call graph for this function:

5.36 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/src/ui/ui.c File Reference

```
#include "ui/ui.h"
```

Include dependency graph for ui.c:

Functions

- int setup ()

    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)
- gboolean on_transaction_button_press (__attribute__((unused)) GtkWidget ∗widget, __attribute__((unused)) GdkEventKey ∗event, __attribute__((unused)) gpointer user_data)
- 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)

Variables

- GtkLabel ∗ private_key_label
- GtkLabel ∗ stake_label1
- GtkLabel ∗ stake_label2
- GtkLabel ∗ stake_label3
- GtkEntry ∗ transa_amount
- GtkEntry ∗ recipient_key
- GtkEntry ∗ invest_entry
- GtkEntry ∗ recover_entry
- GtkEntry ∗ name_entry_con
- GtkEntry ∗ public_key_entry_con
- 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
- GtkTreeViewColumn ∗ cx4_th
- GtkCellRenderer ∗ cr1_th
- GtkCellRenderer ∗ cr2_th
- GtkCellRenderer ∗ cr3_th
- GtkCellRenderer ∗ cr4_th

5.36.1 Function Documentation

5.36.1.1 add_contact()

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

Definition at line 215 of file ui.c.

5.36.1.2 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 206 of file ui.c.

5.36.1.3 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 167 of file ui.c.

5.36.1.4 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 176 of file ui.c.

5.36.1.5 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 126 of file ui.c.

5.36.1.6 on_main_window_destroy()

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

Definition at line 135 of file ui.c.

5.36.1.7 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 149 of file ui.c.

5.36.1.8 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 186 of file ui.c.

5.36.1.9 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 195 of file ui.c.

5.36.1.10 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 142 of file ui.c.

5.36.1.11 setup()

```
int setup ( )
```

Setups the gtk widgets for the GUI.

Returns

int Returns 1 if there is an error, 0 otherwise

Definition at line 45 of file ui.c.

Here is the caller graph for this function:

5.36.2 Variable Documentation

5.36.2.1 cr1_con

```
GtkCellRenderer* cr1_con
```

Definition at line 31 of file ui.c.

5.36.2.2 cr1_th

```
GtkCellRenderer* cr1_th
```

Definition at line 39 of file ui.c.

5.36.2.3 cr2_con

```
GtkCellRenderer* cr2_con
```

Definition at line 32 of file ui.c.

5.36.2.4 cr2_th

```
GtkCellRenderer* cr2_th
```

Definition at line 40 of file ui.c.

5.36.2.5 cr3_th

```
GtkCellRenderer* cr3_th
```

Definition at line 41 of file ui.c.

5.36.2.6 cr4_th

```
GtkCellRenderer* cr4_th
```

Definition at line 42 of file ui.c.

5.36.2.7 cx1_con

```
GtkTreeViewColumn* cx1_con
```

Definition at line 29 of file ui.c.

5.36.2.8 cx1_th

```
GtkTreeViewColumn* cx1_th
```

Definition at line 35 of file ui.c.

5.36.2.9 cx2_con

```
GtkTreeViewColumn* cx2_con
```

Definition at line 30 of file ui.c.

5.36.2.10 cx2_th

```
GtkTreeViewColumn* cx2_th
```

Definition at line 36 of file ui.c.

5.36.2.11 cx3_th

```
GtkTreeViewColumn* cx3_th
```

Definition at line 37 of file ui.c.

5.36.2.12 cx4_th

```
GtkTreeViewColumn* cx4_th
```

Definition at line 38 of file ui.c.

5.36.2.13 invest_entry

```
GtkEntry* invest_entry
```

Definition at line 23 of file ui.c.

5.36.2.14 name_entry_con

```
GtkEntry* name_entry_con
```

Definition at line 25 of file ui.c.

5.36.2.15 private_key_label

```
GtkLabel* private_key_label
```

Definition at line 17 of file ui.c.

5.36.2.16 public_key_entry_con

```
GtkEntry* public_key_entry_con
```

Definition at line 26 of file ui.c.

5.36.2.17 recipient_key

```
GtkEntry* recipient_key
```

Definition at line 22 of file ui.c.

5.36.2.18 recover_entry

```
GtkEntry* recover_entry
```

Definition at line 24 of file ui.c.

5.36.2.19 stake_label1

```
GtkLabel* stake_label1
```

Definition at line 18 of file ui.c.

5.36.2.20 stake_label2

```
GtkLabel* stake_label2
```

Definition at line 19 of file ui.c.

5.36.2.21 stake_label3

```
GtkLabel* stake_label3
```

Definition at line 20 of file ui.c.

5.36.2.22 transa_amount

```
GtkEntry* transa_amount
```

Definition at line 21 of file ui.c.

5.36.2.23 ts_con

```
GtkTreeStore* ts_con
```

Definition at line 28 of file ui.c.

5.36.2.24 ts_th

```
GtkTreeStore* ts_th
```

Definition at line 34 of file ui.c.

5.36.2.25 tv_con

```
GtkTreeView* tv_con
```

Definition at line 27 of file ui.c.

5.36.2.26 tv_th

```
GtkTreeView* tv_th
```

Definition at line 33 of file ui.c.

5.37 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/gen/GEN_blockchain_files.c File Reference

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

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

Functions

- void ∗ rand_data (size_t size)
- void gen_blockhain (size_t nb_blocks)

5.37.1 Function Documentation

5.37.1.1 gen_blockhain()

```
void gen_blockhain (
             size_t nb_blocks )
```

Definition at line 20 of file GEN_blockchain_files.c.

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

5.37.1.2 rand_data()

```
void* rand_data (
            size_t size )
```

Definition at line 5 of file GEN_blockchain_files.c.

Here is the caller graph for this function:

5.38 /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 "cryptosystem/rsa.h"
```

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

Macros

- #define NB_FAKE_VALIDATORS 10
- #define str(x) #x

Functions

- void gen_validators_file (char path[ ])

  Generate a mock validators states file.

5.38.1 Macro Definition Documentation

5.38.1.1 NB_FAKE_VALIDATORS

```
#define NB_FAKE_VALIDATORS 10
```

Definition at line 11 of file GEN_validators_file.c.

5.38.1.2 str

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

Definition at line 12 of file GEN_validators_file.c.

5.38.2 Function Documentation

5.38.2.1 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 28 of file GEN_validators_file.c.

Here is the caller graph for this function:

5.39 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩Cryptocurrency/tests/main_test.c File Reference

```
#include "gen/GEN_validators_file.c"
```

Include dependency graph for main_test.c:

Functions

- int main ()

5.39.1 Function Documentation

5.39.1.1 main()

```
int main ( )
```

Definition at line 3 of file main_test.c.

Here is the call graph for this function:

5.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/core/blockchain/block_test.c File Reference

```
#include "tests_macros.h"
#include "core/blockchain/block.h"
#include "core/blockchain/transaction.h"
#include "gen/GEN_blockchain_files.c"
```

Include dependency graph for block_test.c:

Macros

- #define NB_BLOCK_PER_CHUNK 10
- #define NB_MOCK_BLOCKS 13

Functions

- void block_test (void)

5.40.1 Macro Definition Documentation

5.40.1.1 NB_BLOCK_PER_CHUNK

```
#define NB_BLOCK_PER_CHUNK 10
```

Definition at line 7 of file block_test.c.

5.40.1.2 NB_MOCK_BLOCKS

```
#define NB_MOCK_BLOCKS 13
```

Definition at line 9 of file block_test.c.

5.40.2 Function Documentation

5.40.2.1 block_test()

```
void block_test (
             void  )
```

Definition at line 11 of file block_test.c.

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

5.41 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/core/blockchain/block_test.h File Reference

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

Functions

- void block_test (void)

5.41.1 Function Documentation

5.41.1.1 block_test()

```
void block_test (
             void  )
```

Definition at line 11 of file block_test.c.

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

5.42 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/core/validation/validations_test.c File Reference

```
#include "gen/GEN_validators_file.c"
#include "core/validation/validations.h"
#include "tests_macros.h"
```
Include dependency graph for validations_test.c: This graph shows which files directly or indirectly include this file:

Functions

- void validations_test ()

5.42.1 Function Documentation

5.42.1.1 validations_test()

```
void validations_test ( )
```

Definition at line 5 of file validations_test.c.

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

5.43 /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 "core/blockchain/wallet.h"
#include <stdio.h>
#include <unistd.h>
#include <openssl/sha.h>
#include "misc/safe.h"
#include <fcntl.h>
#include <math.h>
#include <sys/stat.h>
```
Include dependency graph for rsa_test.c:

Macros

- #define MAX(a, b)

Functions

- void get_keys_test ()
- void get_keys_equality_test ()

5.43.1 Macro Definition Documentation

5.43.1.1 MAX

```
#define MAX(
               a,
               b )
```

Value:
```
    ({ __typeof__ (a) _a = (a); \
       __typeof__ (b) _b = (b); \
     _a > _b ? _a : _b; })
```

5.43.2 Function Documentation

5.43.2.1 get_keys_equality_test()

```
void get_keys_equality_test ( )
```

Definition at line 28 of file rsa_test.c.

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

5.43.2.2 get_keys_test()

```
void get_keys_test ( )
```

Definition at line 14 of file rsa_test.c.

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

5.44 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/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 ()

5.44.1 Function Documentation

5.44.1.1 get_keys_equality_test()

```
void get_keys_equality_test ( )
```

Definition at line 28 of file rsa_test.c.

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

5.44.1.2 get_keys_test()

```
void get_keys_test ( )
```

Definition at line 14 of file rsa_test.c.

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

5.45 /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 ()

5.45.1 Function Documentation

5.45.1.1 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:

5.46 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/cryptosystem/signature_test.h File Reference

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

Functions

- void verify_sign_test ()

5.46.1 Function Documentation

5.46.1.1 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:

5.47 /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/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
```

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

Functions

- void network_test ()

5.47.1 Function Documentation

5.47.1.1 network_test()

```
void network_test ( )
```

Definition at line 10 of file client_test.c.

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

5.48 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/src/network/server_test.c File Reference

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

Include dependency graph for server_test.c:

Functions

- int main ()

5.48.1 Function Documentation

5.48.1.1 main()

```
int main ( )
```

Definition at line 4 of file server_test.c.

Here is the call graph for this function:

5.49 /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...)

5.49.1 Macro Definition Documentation

5.49.1.1 DEBUG

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

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

Definition at line 5 of file tests_macros.h.

5.49.1.2 LOG

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

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

Definition at line 9 of file tests_macros.h.

5.49.1.3 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.

5.49.1.4 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.

5.49.1.5 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.

5.50 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-↩ Cryptocurrency/tests/unit_testing.c File Reference

```
#include "tests_macros.h"
#include "cryptosystem/signature_test.h"
#include "cryptosystem/rsa_test.h"
#include "network/client_test.c"
#include "core/blockchain/block_test.h"
#include "core/validation/validations_test.c"
```
Include dependency graph for unit_testing.c:

Functions

- int main ()

5.50.1 Function Documentation

5.50.1.1 main()

```
int main ( )
```

Definition at line 8 of file unit_testing.c.

Here is the call graph for this function:

Index