

PEPITAS CRYPTOCURRENCY

Generated by Doxygen 1.8.17

1 PEPITAS	1
1.1 CODING STYLE	1
1.1.1 Coding case	1
1.1.2 Tests	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 Block Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 block_data	7
4.1.2.2 block_signature	8
4.1.2.3 chunk_id	8
4.1.2.4 signature_len	8
4.2 BlockData Struct Reference	8
4.2.1 Detailed Description	8
4.2.2 Field Documentation	9
4.2.2.1 block_timestamp	9
4.2.2.2 height	9
4.2.2.3 magic	9
4.2.2.4 nb_transactions	9
4.2.2.5 previous_block_hash	9
4.2.2.6 transactions	10
4.2.2.7 validator_public_key	10
4.3 ChunkBlockchain Struct Reference	10
4.3.1 Detailed Description	10
4.3.2 Field Documentation	10
4.3.2.1 chunk	10
4.3.2.2 chunk_nb	11
4.4 client_connection Struct Reference	11
4.4.1 Detailed Description	11
4.4.2 Field Documentation	11
4.4.2.1 info	11
4.4.2.2 socket	11
4.5 Neighbour Struct Reference	12
4.5.1 Detailed Description	12
4.5.2 Field Documentation	12
4.5.2.1 client_sockfd	12

4.5.2.2 family	12
4.5.2.3 hostname	12
4.5.2.4 server_sockfd	13
4.6 Node Struct Reference	13
4.6.1 Detailed Description	13
4.6.2 Field Documentation	13
4.6.2.1 neighbours	13
4.7 Transaction Struct Reference	13
4.7.1 Detailed Description	14
4.7.2 Field Documentation	14
4.7.2.1 signature_len	14
4.7.2.2 transaction_data	14
4.7.2.3 transaction_signature	14
4.8 TransactionData Struct Reference	14
4.8.1 Detailed Description	15
4.8.2 Field Documentation	15
4.8.2.1 amount	15
4.8.2.2 asset	15
4.8.2.3 cause	15
4.8.2.4 organisation_public_key	16
4.8.2.5 receiver_public_key	16
4.8.2.6 receiver_remaining_money	16
4.8.2.7 sender_public_key	16
4.8.2.8 sender_remaining_money	16
4.8.2.9 transaction_timestamp	16
4.9 Wallet Struct Reference	17
4.9.1 Detailed Description	17
4.9.2 Field Documentation	17
4.9.2.1 amount	17
4.9.2.2 is_validator	17
4.9.2.3 priv_key	17
4.9.2.4 pub_key	17
5 File Documentation	19
5.1 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/block.h	
File Reference	19
5.2 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/transaction.h	
File Reference	19
5.2.1 Macro Definition Documentation	20
5.2.1.1 TRANSACTION_DATA_SIZE	20
5.2.1.2 TRANSACTION_SIZE	20
5.2.2 Typedef Documentation	20
5.2.2.1 Transaction	20

5.2.2.2 TransactionData	20
5.2.3 Function Documentation	20
5.2.3.1 send_money()	20
5.3 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/wallet.h	
File Reference	21
5.3.1 Typedef Documentation	21
5.3.1.1 Wallet	21
5.3.2 Function Documentation	22
5.3.2.1 create_account()	22
5.3.2.2 get_my_wallet()	22
5.4 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/stake.h	
File Reference	22
5.4.1 Function Documentation	23
5.4.1.1 pop_stake()	23
5.4.1.2 push_stake()	23
5.5 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/validations.h	
File Reference	24
5.5.1 Function Documentation	24
5.5.1.1 get_amount()	24
5.5.1.2 get_next_committee()	24
5.6 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h	
File Reference	25
5.6.1 Function Documentation	25
5.6.1.1 hash_block_transactions()	25
5.6.1.2 sha384_data()	26
5.7 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h	
File Reference	26
5.7.1 Macro Definition Documentation	26
5.7.1.1 RSA_BEGIN_SIZE	27
5.7.1.2 RSA_END_SIZE	27
5.7.1.3 RSA_FILE_TOTAL_SIZE	27
5.7.1.4 RSA_KEY_SIZE	27
5.7.2 Function Documentation	27
5.7.2.1 get_keys()	27
5.8 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h	
File Reference	28
5.8.1 Function Documentation	28
5.8.1.1 get_blockdata_data()	28
5.8.1.2 get_transaction_data()	29
5.8.1.3 sign_block()	29
5.8.1.4 sign_block_transactions()	30
5.8.1.5 sign_message()	30
5.8.1.6 sign_transaction()	30
5.8.1.7 verify_block_signature()	31

5.8.1.8	verify_signature()	31
5.8.1.9	verify_transaction_signature()	32
5.8.1.10	write_block()	32
5.8.1.11	write_blockdata()	32
5.9	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h File Reference	34
5.9.1	Function Documentation	34
5.9.1.1	last_file_in_folder()	34
5.10	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h File Reference	35
5.10.1	Macro Definition Documentation	35
5.10.1.1	MAX	35
5.10.1.2	MIN	35
5.11	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h File Reference	35
5.11.1	Function Documentation	36
5.11.1.1	safe_fread()	36
5.11.1.2	safe_read()	36
5.11.1.3	safe_write()	37
5.12	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h File Reference	37
5.12.1	Macro Definition Documentation	38
5.12.1.1	MAX_NEIGHBOURS	38
5.12.2	Typedef Documentation	38
5.12.2.1	Neighbour	38
5.12.2.2	Node	38
5.12.3	Function Documentation	39
5.12.3.1	get_my_node()	39
5.12.3.2	listen_to()	39
5.12.3.3	ping_client()	39
5.12.3.4	set_neighbour()	40
5.13	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h File Reference	40
5.13.1	Function Documentation	40
5.13.1.1	fetch_client_list()	40
5.13.1.2	read_header()	41
5.14	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h File Reference	41
5.14.1	Macro Definition Documentation	42
5.14.1.1	HD_GET_BLOCKCHAIN	42
5.14.1.2	HD_GET_CLIENT_LIST	42
5.14.1.3	HD_SEND_BLOCKCHAIN	42
5.14.1.4	HD_SEND_CLIENT_LIST	42
5.14.1.5	NB_HARD_CODED_ADDR	43

5.14.1.6	STATIC_PORT	43
5.14.2	Variable Documentation	43
5.14.2.1	HARD_CODED_ADDR	43
5.15	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_↵ data.h File Reference	43
5.15.1	Function Documentation	43
5.15.1.1	send_client_list()	43
5.16	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h File Reference	44
5.16.1	Typedef Documentation	44
5.16.1.1	client_connection	44
5.16.2	Function Documentation	45
5.16.2.1	init_server()	45
5.16.2.2	send_block()	45
5.17	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h File Reference	45
5.17.1	Function Documentation	46
5.17.1.1	add_contact()	46
5.17.1.2	on_add_contact_button1_press()	47
5.17.1.3	on_invest_button1_press()	47
5.17.1.4	on_invest_button2_press()	47
5.17.1.5	on_main_window_delete()	48
5.17.1.6	on_main_window_destroy()	48
5.17.1.7	on_pkey_button_press()	48
5.17.1.8	on_recover_button1_press()	49
5.17.1.9	on_recover_button2_press()	49
5.17.1.10	on_transaction_button_press()	50
5.17.1.11	setup()	50
5.18	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md File Reference	51
5.19	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c File Reference	51
5.19.1	Function Documentation	51
5.19.1.1	main()	51
5.20	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/client.c File Reference	51
5.20.1	Function Documentation	52
5.20.1.1	get_my_node()	52
5.20.1.2	listen_to()	52
5.20.1.3	set_neighbour()	53
5.21	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c File Reference	53
5.21.1	Function Documentation	54
5.21.1.1	convert_data_to_block()	54
5.21.1.2	convert_data_to_blockdata()	54
5.21.1.3	convert_data_to_transaction()	54

5.21.1.4	convert_data_to_transactiondata()	54
5.21.1.5	free_block()	54
5.21.1.6	get_block()	55
5.21.1.7	get_blockchain()	55
5.21.1.8	get_next_block()	55
5.21.1.9	get_prev_block()	56
5.21.1.10	write_block_file()	56
5.22	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c	
	File Reference	56
5.22.1	Function Documentation	57
5.22.1.1	create_account()	57
5.22.1.2	get_my_wallet()	57
5.23	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validations.c	
	File Reference	58
5.23.1	Macro Definition Documentation	58
5.23.1.1	MAX_VALIDATORS_PER_BLOCK	58
5.23.1.2	NB_RSA_CHUNK	58
5.23.2	Function Documentation	59
5.23.2.1	define_nb_validators()	59
5.23.2.2	get_next_committee()	59
5.24	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/hash.c	
	File Reference	59
5.24.1	Function Documentation	60
5.24.1.1	hash_block_transactions()	60
5.24.1.2	sha384_data()	60
5.25	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/rsa.c	
	File Reference	61
5.25.1	Macro Definition Documentation	61
5.25.1.1	RSA_NUM_E	61
5.25.2	Function Documentation	61
5.25.2.1	get_keys()	62
5.26	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/signature.c	
	File Reference	62
5.26.1	Function Documentation	63
5.26.1.1	get_blockdata_data()	63
5.26.1.2	get_transaction_data()	63
5.26.1.3	sign_block()	64
5.26.1.4	sign_block_transactions()	64
5.26.1.5	sign_message()	64
5.26.1.6	sign_transaction()	65
5.26.1.7	verify_block_signature()	65
5.26.1.8	verify_signature()	65
5.26.1.9	verify_transaction_signature()	66

5.26.1.10 write_block()	66
5.26.1.11 write_blockdata()	67
5.26.1.12 write_transaction()	67
5.26.1.13 write_transactiondata()	67
5.27 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c File Reference	67
5.27.1 Function Documentation	68
5.27.1.1 main()	68
5.28 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/files.c File Reference	68
5.28.1 Macro Definition Documentation	68
5.28.1.1 _GNU_SOURCE	68
5.28.2 Function Documentation	69
5.28.2.1 last_file_in_folder()	69
5.29 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/safe.c File Reference	69
5.29.1 Function Documentation	69
5.29.1.1 safe_fread()	70
5.29.1.2 safe_read()	71
5.29.1.3 safe_write()	71
5.30 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/get_data.c File Reference	72
5.30.1 Function Documentation	72
5.30.1.1 fetch_client_list()	72
5.30.1.2 process_header()	73
5.30.1.3 read_header()	73
5.31 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/network.c File Reference	73
5.31.1 Variable Documentation	74
5.31.1.1 HARD_CODED_ADDR	74
5.32 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/send_data.c File Reference	74
5.32.1 Function Documentation	74
5.32.1.1 send_client_list()	74
5.33 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/server.c File Reference	75
5.33.1 Function Documentation	75
5.33.1.1 accept_connection()	75
5.33.1.2 init_server()	75
5.34 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c File Reference	76
5.34.1 Function Documentation	76
5.34.1.1 main()	76
5.35 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c File Reference	76
5.35.1 Function Documentation	77
5.35.1.1 main()	77
5.36 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/ui/ui.c File Reference	77

5.36.1 Function Documentation	78
5.36.1.1 add_contact()	78
5.36.1.2 on_add_contact_button1_press()	78
5.36.1.3 on_invest_button1_press()	79
5.36.1.4 on_invest_button2_press()	79
5.36.1.5 on_main_window_delete()	79
5.36.1.6 on_main_window_destroy()	79
5.36.1.7 on_pkey_button_press()	80
5.36.1.8 on_recover_button1_press()	80
5.36.1.9 on_recover_button2_press()	80
5.36.1.10 on_transaction_button_press()	80
5.36.1.11 setup()	80
5.36.2 Variable Documentation	81
5.36.2.1 cr1_con	81
5.36.2.2 cr1_th	81
5.36.2.3 cr2_con	81
5.36.2.4 cr2_th	81
5.36.2.5 cr3_th	81
5.36.2.6 cr4_th	82
5.36.2.7 cx1_con	82
5.36.2.8 cx1_th	82
5.36.2.9 cx2_con	82
5.36.2.10 cx2_th	82
5.36.2.11 cx3_th	82
5.36.2.12 cx4_th	83
5.36.2.13 invest_entry	83
5.36.2.14 name_entry_con	83
5.36.2.15 private_key_label	83
5.36.2.16 public_key_entry_con	83
5.36.2.17 recipient_key	83
5.36.2.18 recover_entry	84
5.36.2.19 stake_label1	84
5.36.2.20 stake_label2	84
5.36.2.21 stake_label3	84
5.36.2.22 transa_amount	84
5.36.2.23 ts_con	84
5.36.2.24 ts_th	85
5.36.2.25 tv_con	85
5.36.2.26 tv_th	85
5.37 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_blockchain↵ _files.c File Reference	85
5.37.1 Function Documentation	85

5.37.1.1 gen_blockchain()	85
5.37.1.2 rand_data()	86
5.38 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_validators↔ _file.c File Reference	86
5.38.1 Macro Definition Documentation	86
5.38.1.1 NB_FAKE_VALIDATORS	86
5.38.1.2 str	87
5.38.2 Function Documentation	87
5.38.2.1 gen_validators_file()	87
5.39 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main_test.c File Refer- ence	87
5.39.1 Function Documentation	88
5.39.1.1 main()	88
5.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/blockchain/block↔ _test.c File Reference	88
5.40.1 Macro Definition Documentation	88
5.40.1.1 NB_BLOCK_PER_CHUNK	88
5.40.1.2 NB MOCK_BLOCKS	89
5.40.2 Function Documentation	89
5.40.2.1 block_test()	89
5.41 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/blockchain/block↔ _test.h File Reference	89
5.41.1 Function Documentation	89
5.41.1.1 block_test()	89
5.42 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/validation/validations↔ _test.c File Reference	90
5.42.1 Function Documentation	90
5.42.1.1 validations_test()	90
5.43 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa↔ _test.c File Reference	90
5.43.1 Macro Definition Documentation	91
5.43.1.1 MAX	91
5.43.2 Function Documentation	91
5.43.2.1 get_keys_equality_test()	91
5.43.2.2 get_keys_test()	91
5.44 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa↔ _test.h File Reference	91
5.44.1 Function Documentation	92
5.44.1.1 get_keys_equality_test()	92
5.44.1.2 get_keys_test()	92
5.45 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/signature↔ _test.c File Reference	92
5.45.1 Function Documentation	92
5.45.1.1 verify_sign_test()	93

5.46 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/signature↔ _test.h File Reference	93
5.46.1 Function Documentation	93
5.46.1.1 verify_sign_test()	93
5.47 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/client↔ test.c File Reference	93
5.47.1 Function Documentation	94
5.47.1.1 network_test()	94
5.48 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/server↔ test.c File Reference	94
5.48.1 Function Documentation	94
5.48.1.1 main()	94
5.49 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/tests_macros.h File Reference	94
5.49.1 Macro Definition Documentation	95
5.49.1.1 DEBUG	95
5.49.1.2 LOG	95
5.49.1.3 TEST_FAILED	95
5.49.1.4 TEST_PASSED	96
5.49.1.5 TEST_WARNING	96
5.50 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit_testing.c File Ref- erence	96
5.50.1 Function Documentation	96
5.50.1.1 main()	96
Index	97

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:

Block	7
BlockData	8
ChunkBlockchain	10
client_connection	11
Neighbour	12
Node	13
Transaction	13
TransactionData	14
Wallet	17

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/block.h	19
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/transaction.h	19
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/wallet.h	21
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/stake.h	22
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/validations.h	24
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h	25
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h	26
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h	28
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h	34
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h	35
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h	35
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h	37
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h	40
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h	41
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_data.h	43
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h	44
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h	45
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c	51
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c	67
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c	76
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c	76
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c	53
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c	56
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validations.c	58
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/hash.c	59
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/rsa.c	61
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/signature.c	62
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/files.c	68
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/safe.c	69
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/client.c	51
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/get_data.c	72
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/network.c	73

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/ send_data.c	74
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/ server.c	75
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/ui/ ui.c	77
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/ main_test.c	87
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/ tests_macros.h	94
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/ unit_testing.c	96
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/ GEN_blockchain_files.c 85	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/ GEN_validators_file.c 86	86
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/blockchain/ block_test.c 88	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/blockchain/ block_test.h 89	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/validation/ validations_test.c 90	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/ rsa_test.c 90	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/ rsa_test.h 91	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/ signature_test.c 92	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/ signature_test.h 93	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/ client_test.c .	93
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/ server_test.c .	94

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.

5.3

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

File Reference

21

Parameters

<i>amount</i>	The amount to send
<i>receiver_public_key</i>	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

<i>amount</i>	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

<i>amount</i>	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

<i>public_key</i>	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

<i>nb_validators</i>	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

<i>block</i>	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

<i>data</i>	The buffer to hash
<i>len_data</i>	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

<i>block</i>	The block
<i>size</i>	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

<i>transactions</i>	The transaction array
<i>buff</i>	The buffer that receives the transactions
<i>size</i>	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

<i>block</i>	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

<i>block</i>	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

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>signature_len</i>	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

<i>transaction</i>	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

<i>block</i>	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

<i>data</i>	The buffer to verify
<i>data_len</i>	The length of the buffer
<i>signature</i>	The signature to compare with SHA384(data, len_data)
<i>signature_len</i>	The length of the signature
<i>pub_key</i>	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

<i>transaction</i>	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

<i>block</i>	The block to write
<i>fd</i>	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

<i>blockdata</i>	The blockdata to write
<i>fd</i>	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

<i>folder_path</i>	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

<i>buffer</i>	The buffer to write on
<i>size</i>	The size of 1 read element
<i>n</i>	The number of elements to read
<i>file</i>	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

<i>fd</i>	The file descriptor
<i>buf</i>	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

<i>fd</i>	The file descriptor
<i>buf</i>	The buffer to write
<i>count</i>	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.neighbours 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

<i>neighbour_id</i>	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

<i>neighbour_id</i>	
---------------------	--

Returns

0 if success, -1 otherwise

5.12.3.4 set_neighbour()

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

Sets a neighbour in the client.neighbours section.

Returns

0 if success, -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

<code>neighbour↔ _id</code>	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

<code>sockfd</code>	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

<code>sockfd</code>	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

<i>sockfd</i>	The socket FD
<i>block</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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

<i>widget</i>	unused
<i>event</i>	unused
<i>user_data</i>	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.neighbours 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

<code>neighbour↔ _id</code>	The neighbour's index (in struct <code>Node</code>) 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.neighbours section.

Returns

0 if success, -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

<i>block</i>	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

<i>nb_chunk</i>	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

<i>block</i>	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

<i>block</i>	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

<i>block</i>	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

<i>nb_validators</i>	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

<i>block</i>	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

<i>data</i>	The buffer to hash
<i>len_data</i>	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

<i>block</i>	The block
<i>size</i>	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

<i>transactions</i>	The transaction array
<i>buff</i>	The buffer that receives the transactions
<i>size</i>	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

<i>block</i>	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

<i>block</i>	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

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>signature_len</i>	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

<i>transaction</i>	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

<i>block</i>	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

<i>data</i>	The buffer to verify
<i>data_len</i>	The length of the buffer
<i>signature</i>	The signature to compare with SHA384(data, len_data)
<i>signature_len</i>	The length of the signature
<i>pub_key</i>	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

<i>transaction</i>	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

<i>block</i>	The block to write
<i>fd</i>	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

<i>blockdata</i>	The blockdata to write
<i>fd</i>	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

<i>folder_path</i>	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

<i>buffer</i>	The buffer to write on
<i>size</i>	The size of 1 read element
<i>n</i>	The number of elements to read
<i>file</i>	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 '\n\n\n'.

Parameters

<i>fd</i>	The file descriptor
<i>buf</i>	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

<i>fd</i>	The file descriptor
<i>buf</i>	The buffer to write
<i>count</i>	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

<i>neighbour_id</i>	The id of the neighbour list to merge
---------------------	---------------------------------------

Returns

0 if success, -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

<i>sockfd</i>	The sock FD
---------------	-------------

Returns

0 if success, -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

<i>sockfd</i>	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

<i>widget</i>	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_blockchain](#) (size_t nb_blocks)

5.37.1 Function Documentation

5.37.1.1 gen_blockchain()

```
void gen_blockchain (
    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

<i>path</i>	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("\n 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("\n\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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md, [51](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/block.h, [19](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/transaction.h, [19](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/blockchain/wallet.h, [21](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/stake.h, [22](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/core/validation/validations.h, [24](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h, [25](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h, [26](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h, [28](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h, [34](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h, [35](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h, [35](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h, [37](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h, [40](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h, [41](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_data.h, [43](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h, [44](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h, [45](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c, [51](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c, [53](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c, [56](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validations.c, [58](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/hash.c, [59](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/rsa.c, [61](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/cryptosystem/signature.c, [62](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c, [67](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/files.c, [68](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/misc/safe.c, [69](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/client.c, [51](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/get_data.c, [72](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/network.c, [73](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/send_data.c, [74](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/network/server.c, [75](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c, [76](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c, [76](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/ui/ui.c, [77](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_blockchain_files.c, [85](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_validators_file.c, [86](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main_test.c, [87](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/core/blockchain/block_test.c, [88](#)

- convert_data_to_transaction
 - block.c, [54](#)
- convert_data_to_transactiondata
 - block.c, [54](#)
- cr1_con
 - ui.c, [81](#)
- cr1_th
 - ui.c, [81](#)
- cr2_con
 - ui.c, [81](#)
- cr2_th
 - ui.c, [81](#)
- cr3_th
 - ui.c, [81](#)
- cr4_th
 - ui.c, [81](#)
- create_account
 - wallet.c, [57](#)
 - wallet.h, [22](#)
- cx1_con
 - ui.c, [82](#)
- cx1_th
 - ui.c, [82](#)
- cx2_con
 - ui.c, [82](#)
- cx2_th
 - ui.c, [82](#)
- cx3_th
 - ui.c, [82](#)
- cx4_th
 - ui.c, [82](#)
- DEBUG
 - tests_macros.h, [95](#)
- define_nb_validators
 - validations.c, [59](#)
- family
 - Neighbour, [12](#)
- fetch_client_list
 - get_data.c, [72](#)
 - get_data.h, [40](#)
- files.c
 - _GNU_SOURCE, [68](#)
 - last_file_in_folder, [69](#)
- files.h
 - last_file_in_folder, [34](#)
- free_block
 - block.c, [54](#)
- GEN_blockchain_files.c
 - gen_blockchain, [85](#)
 - rand_data, [85](#)
- gen_blockchain
 - GEN_blockchain_files.c, [85](#)
- gen_validators_file
 - GEN_validators_file.c, [87](#)
- GEN_validators_file.c
 - gen_validators_file, [87](#)
- NB_FAKE_VALIDATORS, [86](#)
- str, [86](#)
- get_amount
 - validations.h, [24](#)
- get_block
 - block.c, [55](#)
- get_blockchain
 - block.c, [55](#)
- get_blockdata_data
 - signature.c, [63](#)
 - signature.h, [28](#)
- get_data.c
 - fetch_client_list, [72](#)
 - process_header, [73](#)
 - read_header, [73](#)
- get_data.h
 - fetch_client_list, [40](#)
 - read_header, [41](#)
- get_keys
 - rsa.c, [61](#)
 - rsa.h, [27](#)
- get_keys_equality_test
 - rsa_test.c, [91](#)
 - rsa_test.h, [92](#)
- get_keys_test
 - rsa_test.c, [91](#)
 - rsa_test.h, [92](#)
- get_my_node
 - client.c, [52](#)
 - client.h, [39](#)
- get_my_wallet
 - wallet.c, [57](#)
 - wallet.h, [22](#)
- get_next_block
 - block.c, [55](#)
- get_next_committee
 - validations.c, [59](#)
 - validations.h, [24](#)
- get_prev_block
 - block.c, [56](#)
- get_transaction_data
 - signature.c, [63](#)
 - signature.h, [29](#)
- gui.c
 - main, [68](#)
- HARD_CODED_ADDR
 - network.c, [74](#)
 - network.h, [43](#)
- hash.c
 - hash_block_transactions, [60](#)
 - sha384_data, [60](#)
- hash.h
 - hash_block_transactions, [25](#)
 - sha384_data, [26](#)
- hash_block_transactions
 - hash.c, [60](#)
 - hash.h, [25](#)
- HD_GET_BLOCKCHAIN

- network.h, 42
- HD_GET_CLIENT_LIST
 - network.h, 42
- HD_SEND_BLOCKCHAIN
 - network.h, 42
- HD_SEND_CLIENT_LIST
 - network.h, 42
- height
 - BlockData, 9
- hostname
 - Neighbour, 12
- info
 - client_connection, 11
- init_server
 - server.c, 75
 - server.h, 45
- invest_entry
 - ui.c, 83
- is_validator
 - Wallet, 17
- last_file_in_folder
 - files.c, 69
 - files.h, 34
- listen_to
 - client.c, 52
 - client.h, 39
- LOG
 - tests_macros.h, 95
- magic
 - BlockData, 9
- main
 - client.c, 51
 - gui.c, 68
 - main_test.c, 88
 - server.c, 76
 - server_test.c, 94
 - sign.c, 77
 - unit_testing.c, 96
- main_test.c
 - main, 88
- math.h
 - MAX, 35
 - MIN, 35
- MAX
 - math.h, 35
 - rsa_test.c, 91
- MAX_NEIGHBOURS
 - client.h, 38
- MAX_VALIDATORS_PER_BLOCK
 - validations.c, 58
- MIN
 - math.h, 35
- name_entry_con
 - ui.c, 83
- NB_BLOCK_PER_CHUNK
 - block_test.c, 88
- NB_FAKE_VALIDATORS
 - GEN_validators_file.c, 86
- NB_HARD_CODED_ADDR
 - network.h, 42
- NB MOCK_BLOCKS
 - block_test.c, 88
- NB_RSA_CHUNK
 - validations.c, 58
- nb_transactions
 - BlockData, 9
- Neighbour, 12
 - client.h, 38
 - client_sockfd, 12
 - family, 12
 - hostname, 12
 - server_sockfd, 12
- neighbours
 - Node, 13
- network.c
 - HARD_CODED_ADDR, 74
- network.h
 - HARD_CODED_ADDR, 43
 - HD_GET_BLOCKCHAIN, 42
 - HD_GET_CLIENT_LIST, 42
 - HD_SEND_BLOCKCHAIN, 42
 - HD_SEND_CLIENT_LIST, 42
 - NB_HARD_CODED_ADDR, 42
 - STATIC_PORT, 43
- network_test
 - client_test.c, 94
- Node, 13
 - client.h, 38
 - neighbours, 13
- on_add_contact_button1_press
 - ui.c, 78
 - ui.h, 46
- on_invest_button1_press
 - ui.c, 78
 - ui.h, 47
- on_invest_button2_press
 - ui.c, 79
 - ui.h, 47
- on_main_window_delete
 - ui.c, 79
 - ui.h, 48
- on_main_window_destroy
 - ui.c, 79
 - ui.h, 48
- on_pkey_button_press
 - ui.c, 79
 - ui.h, 48
- on_recover_button1_press
 - ui.c, 80
 - ui.h, 49
- on_recover_button2_press
 - ui.c, 80
 - ui.h, 49

- on_transaction_button_press
 - ui.c, [80](#)
 - ui.h, [50](#)
- organisation_public_key
 - TransactionData, [15](#)
- ping_client
 - client.h, [39](#)
- pop_stake
 - stake.h, [23](#)
- previous_block_hash
 - BlockData, [9](#)
- priv_key
 - Wallet, [17](#)
- private_key_label
 - ui.c, [83](#)
- process_header
 - get_data.c, [73](#)
- pub_key
 - Wallet, [17](#)
- public_key_entry_con
 - ui.c, [83](#)
- push_stake
 - stake.h, [23](#)
- rand_data
 - GEN_blockchain_files.c, [85](#)
- read_header
 - get_data.c, [73](#)
 - get_data.h, [41](#)
- receiver_public_key
 - TransactionData, [16](#)
- receiver_remaining_money
 - TransactionData, [16](#)
- recipient_key
 - ui.c, [83](#)
- recover_entry
 - ui.c, [83](#)
- rsa.c
 - get_keys, [61](#)
 - RSA_NUM_E, [61](#)
- rsa.h
 - get_keys, [27](#)
 - RSA_BEGIN_SIZE, [26](#)
 - RSA_END_SIZE, [27](#)
 - RSA_FILE_TOTAL_SIZE, [27](#)
 - RSA_KEY_SIZE, [27](#)
- RSA_BEGIN_SIZE
 - rsa.h, [26](#)
- RSA_END_SIZE
 - rsa.h, [27](#)
- RSA_FILE_TOTAL_SIZE
 - rsa.h, [27](#)
- RSA_KEY_SIZE
 - rsa.h, [27](#)
- RSA_NUM_E
 - rsa.c, [61](#)
- rsa_test.c
 - get_keys_equality_test, [91](#)
 - get_keys_test, [91](#)
 - MAX, [91](#)
- rsa_test.h
 - get_keys_equality_test, [92](#)
 - get_keys_test, [92](#)
- safe.c
 - safe_fread, [69](#)
 - safe_read, [71](#)
 - safe_write, [71](#)
- safe.h
 - safe_fread, [36](#)
 - safe_read, [36](#)
 - safe_write, [37](#)
- safe_fread
 - safe.c, [69](#)
 - safe.h, [36](#)
- safe_read
 - safe.c, [71](#)
 - safe.h, [36](#)
- safe_write
 - safe.c, [71](#)
 - safe.h, [37](#)
- send_block
 - server.h, [45](#)
- send_client_list
 - send_data.c, [74](#)
 - send_data.h, [43](#)
- send_data.c
 - send_client_list, [74](#)
- send_data.h
 - send_client_list, [43](#)
- send_money
 - transaction.h, [20](#)
- sender_public_key
 - TransactionData, [16](#)
- sender_remaining_money
 - TransactionData, [16](#)
- server.c
 - accept_connection, [75](#)
 - init_server, [75](#)
 - main, [76](#)
- server.h
 - client_connection, [44](#)
 - init_server, [45](#)
 - send_block, [45](#)
- server_sockfd
 - Neighbour, [12](#)
- server_test.c
 - main, [94](#)
- set_neighbour
 - client.c, [52](#)
 - client.h, [40](#)
- setup
 - ui.c, [80](#)
 - ui.h, [50](#)
- sha384_data
 - hash.c, [60](#)
 - hash.h, [26](#)

- sign.c
 - main, 77
- sign_block
 - signature.c, 63
 - signature.h, 29
- sign_block_transactions
 - signature.c, 64
 - signature.h, 30
- sign_message
 - signature.c, 64
 - signature.h, 30
- sign_transaction
 - signature.c, 65
 - signature.h, 30
- signature.c
 - get_blockdata_data, 63
 - get_transaction_data, 63
 - sign_block, 63
 - sign_block_transactions, 64
 - sign_message, 64
 - sign_transaction, 65
 - verify_block_signature, 65
 - verify_signature, 65
 - verify_transaction_signature, 66
 - write_block, 66
 - write_blockdata, 67
 - write_transaction, 67
 - write_transactiondata, 67
- signature.h
 - get_blockdata_data, 28
 - get_transaction_data, 29
 - sign_block, 29
 - sign_block_transactions, 30
 - sign_message, 30
 - sign_transaction, 30
 - verify_block_signature, 31
 - verify_signature, 31
 - verify_transaction_signature, 32
 - write_block, 32
 - write_blockdata, 32
- signature_len
 - Block, 8
 - Transaction, 14
- signature_test.c
 - verify_sign_test, 92
- signature_test.h
 - verify_sign_test, 93
- socket
 - client_connection, 11
- stake.h
 - pop_stake, 23
 - push_stake, 23
- stake_label1
 - ui.c, 84
- stake_label2
 - ui.c, 84
- stake_label3
 - ui.c, 84
- STATIC_PORT
 - network.h, 43
- str
 - GEN_validators_file.c, 86
- TEST_FAILED
 - tests_macros.h, 95
- TEST_PASSED
 - tests_macros.h, 95
- TEST_WARNING
 - tests_macros.h, 96
- tests_macros.h
 - DEBUG, 95
 - LOG, 95
 - TEST_FAILED, 95
 - TEST_PASSED, 95
 - TEST_WARNING, 96
- transa_amount
 - ui.c, 84
- Transaction, 13
 - signature_len, 14
 - transaction.h, 20
 - transaction_data, 14
 - transaction_signature, 14
- transaction.h
 - send_money, 20
 - Transaction, 20
 - TRANSACTION_DATA_SIZE, 20
 - TRANSACTION_SIZE, 20
 - TransactionData, 20
- transaction_data
 - Transaction, 14
- TRANSACTION_DATA_SIZE
 - transaction.h, 20
- transaction_signature
 - Transaction, 14
- TRANSACTION_SIZE
 - transaction.h, 20
- transaction_timestamp
 - TransactionData, 16
- TransactionData, 14
 - amount, 15
 - asset, 15
 - cause, 15
 - organisation_public_key, 15
 - receiver_public_key, 16
 - receiver_remaining_money, 16
 - sender_public_key, 16
 - sender_remaining_money, 16
 - transaction.h, 20
 - transaction_timestamp, 16
- transactions
 - BlockData, 9
- ts_con
 - ui.c, 84
- ts_th
 - ui.c, 84
- tv_con
 - ui.c, 85

- tv_th
 - ui.c, [85](#)
- ui.c
 - add_contact, [78](#)
 - cr1_con, [81](#)
 - cr1_th, [81](#)
 - cr2_con, [81](#)
 - cr2_th, [81](#)
 - cr3_th, [81](#)
 - cr4_th, [81](#)
 - cx1_con, [82](#)
 - cx1_th, [82](#)
 - cx2_con, [82](#)
 - cx2_th, [82](#)
 - cx3_th, [82](#)
 - cx4_th, [82](#)
 - invest_entry, [83](#)
 - name_entry_con, [83](#)
 - on_add_contact_button1_press, [78](#)
 - on_invest_button1_press, [78](#)
 - on_invest_button2_press, [79](#)
 - on_main_window_delete, [79](#)
 - on_main_window_destroy, [79](#)
 - on_pkey_button_press, [79](#)
 - on_recover_button1_press, [80](#)
 - on_recover_button2_press, [80](#)
 - on_transaction_button_press, [80](#)
 - private_key_label, [83](#)
 - public_key_entry_con, [83](#)
 - recipient_key, [83](#)
 - recover_entry, [83](#)
 - setup, [80](#)
 - stake_label1, [84](#)
 - stake_label2, [84](#)
 - stake_label3, [84](#)
 - transa_amount, [84](#)
 - ts_con, [84](#)
 - ts_th, [84](#)
 - tv_con, [85](#)
 - tv_th, [85](#)
- ui.h
 - add_contact, [46](#)
 - on_add_contact_button1_press, [46](#)
 - on_invest_button1_press, [47](#)
 - on_invest_button2_press, [47](#)
 - on_main_window_delete, [48](#)
 - on_main_window_destroy, [48](#)
 - on_pkey_button_press, [48](#)
 - on_recover_button1_press, [49](#)
 - on_recover_button2_press, [49](#)
 - on_transaction_button_press, [50](#)
 - setup, [50](#)
- unit_testing.c
 - main, [96](#)
- validations.c
 - define_nb_validators, [59](#)
 - get_next_committee, [59](#)
 - MAX_VALIDATORS_PER_BLOCK, [58](#)
 - NB_RSA_CHUNK, [58](#)
- validations.h
 - get_amount, [24](#)
 - get_next_committee, [24](#)
- validations_test
 - validations_test.c, [90](#)
- validations_test.c
 - validations_test, [90](#)
- validator_public_key
 - BlockData, [10](#)
- verify_block_signature
 - signature.c, [65](#)
 - signature.h, [31](#)
- verify_sign_test
 - signature_test.c, [92](#)
 - signature_test.h, [93](#)
- verify_signature
 - signature.c, [65](#)
 - signature.h, [31](#)
- verify_transaction_signature
 - signature.c, [66](#)
 - signature.h, [32](#)
- Wallet, [17](#)
 - amount, [17](#)
 - is_validator, [17](#)
 - priv_key, [17](#)
 - pub_key, [17](#)
 - wallet.h, [21](#)
- wallet.c
 - create_account, [57](#)
 - get_my_wallet, [57](#)
- wallet.h
 - create_account, [22](#)
 - get_my_wallet, [22](#)
 - Wallet, [21](#)
- write_block
 - signature.c, [66](#)
 - signature.h, [32](#)
- write_block_file
 - block.c, [56](#)
- write_blockdata
 - signature.c, [67](#)
 - signature.h, [32](#)
- write_transaction
 - signature.c, [67](#)
- write_transactiondata
 - signature.c, [67](#)