

# PEPITAS CRYPTOCURRENCY

Generated by Doxygen 1.8.17



<b>1 PEPITAS NETWORK PROTOCOL</b>	<b>1</b>
1.1 HEADERS	1
1.1.1 Sync Headers	1
1.1.2 Running Headers	1
1.1.3 Validating Headers	1
1.1.4 CONNECTION TO NETWORK	1
1.1.5 CONNECTION TO NODE	2
1.1.6 GET BLOCKS	2
1.1.7 ACTUAL HEIGHT	2
1.1.8 SEND BLOCK	2
1.1.9 GET PENDING TRANSACTION LIST	2
1.1.10 REJECT DEMAND	2
1.1.11 SEND PENDING TRANSACTION	2
1.1.12 SEND EPOCH BLOCK	2
1.1.13 SEND VOTE	2
<b>2 PEPITAS</b>	<b>3</b>
2.1 CODING STYLE	3
2.1.1 Coding case	3
2.1.2 Tests	3
<b>3 Data Structure Index</b>	<b>5</b>
3.1 Data Structures	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Data Structure Documentation</b>	<b>9</b>
5.1 Block Struct Reference	9
5.1.1 Detailed Description	9
5.1.2 Field Documentation	9
5.1.2.1 block_data	9
5.1.2.2 block_signature	10
5.1.2.3 chunk_id	10
5.1.2.4 validators_votes	10
5.1.2.5 vote_signature	10
5.2 BlockData Struct Reference	10
5.2.1 Detailed Description	11
5.2.2 Field Documentation	11
5.2.2.1 block_timestamp	11
5.2.2.2 epoch_id	11
5.2.2.3 height	11
5.2.2.4 is_prev_block_valid	11
5.2.2.5 magic	11

5.2.2.6 nb_transactions	12
5.2.2.7 nb_validators	12
5.2.2.8 prev_validators_votes	12
5.2.2.9 previous_block_hash	12
5.2.2.10 transactions	12
5.2.2.11 validators_public_keys	12
5.3 blockinfo Struct Reference	13
5.3.1 Detailed Description	13
5.3.2 Field Documentation	13
5.3.2.1 height	13
5.3.2.2 transactions	13
5.4 ChunkBlockchain Struct Reference	13
5.4.1 Detailed Description	14
5.4.2 Field Documentation	14
5.4.2.1 chunk	14
5.4.2.2 chunk_nb	14
5.4.2.3 nb_blocks	14
5.5 client_connection Struct Reference	14
5.5.1 Detailed Description	15
5.5.2 Field Documentation	15
5.5.2.1 actual_client_height	15
5.5.2.2 clientfd	15
5.5.2.3 demand	15
5.5.2.4 lock	15
5.5.2.5 Payload	16
5.5.2.6 Payloadsize	16
5.5.2.7 thread	16
5.6 infos_st Struct Reference	16
5.6.1 Detailed Description	16
5.6.2 Field Documentation	16
5.6.2.1 actual_height	17
5.6.2.2 is_synchronize	17
5.6.2.3 serv_type	17
5.7 Neighbour Struct Reference	17
5.7.1 Detailed Description	17
5.7.2 Field Documentation	17
5.7.2.1 family	18
5.7.2.2 hostname	18
5.8 Node Struct Reference	18
5.8.1 Detailed Description	18
5.8.2 Field Documentation	18
5.8.2.1 neighbours	18

5.9 th_arg Struct Reference . . . . .	19
5.9.1 Detailed Description . . . . .	19
5.9.2 Field Documentation . . . . .	19
5.9.2.1 client_con . . . . .	19
5.9.2.2 infos . . . . .	19
5.10 Transaction Struct Reference . . . . .	19
5.10.1 Detailed Description . . . . .	20
5.10.2 Field Documentation . . . . .	20
5.10.2.1 transaction_data . . . . .	20
5.10.2.2 transaction_signature . . . . .	20
5.11 TransactionData Struct Reference . . . . .	20
5.11.1 Detailed Description . . . . .	21
5.11.2 Field Documentation . . . . .	21
5.11.2.1 amount . . . . .	21
5.11.2.2 asset . . . . .	21
5.11.2.3 cause . . . . .	21
5.11.2.4 magic . . . . .	21
5.11.2.5 organisation_public_key . . . . .	21
5.11.2.6 receiver_public_key . . . . .	22
5.11.2.7 receiver_remaining_money . . . . .	22
5.11.2.8 sender_public_key . . . . .	22
5.11.2.9 sender_remaining_money . . . . .	22
5.11.2.10 transaction_timestamp . . . . .	22
5.11.2.11 type . . . . .	22
5.12 Wallet Struct Reference . . . . .	23
5.12.1 Detailed Description . . . . .	23
5.12.2 Field Documentation . . . . .	23
5.12.2.1 amount . . . . .	23
5.12.2.2 is_validator . . . . .	23
5.12.2.3 priv_key . . . . .	23
5.12.2.4 pub_key . . . . .	23
<b>6 File Documentation . . . . .</b>	<b>25</b>
6.1 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h File Reference . . . . .	25
6.2 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/blockchain↵ _header.h File Reference . . . . .	25
6.2.1 Function Documentation . . . . .	26
6.2.1.1 gen_blockchain_header() . . . . .	26
6.3 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h File Reference . . . . .	26
6.3.1 Macro Definition Documentation . . . . .	27
6.3.1.1 T_TYPE_ADD_STAKE . . . . .	27

6.3.1.2 T_TYPE_DEFAULT . . . . .	27
6.3.1.3 T_TYPE_STAKE_TO_STAKE . . . . .	27
6.3.1.4 T_TYPE_WITHDRAW_STAKE . . . . .	27
6.3.1.5 TRANSACTION_DATA_SIZE . . . . .	28
6.3.1.6 TRANSACTION_SIZE . . . . .	28
6.3.2 Typedef Documentation . . . . .	28
6.3.2.1 Transaction . . . . .	28
6.3.2.2 TransactionData . . . . .	28
6.3.3 Function Documentation . . . . .	28
6.3.3.1 add_pending_transaction() . . . . .	28
6.3.3.2 convert_data_to_transactiondata() . . . . .	29
6.3.3.3 get_transaction_data() . . . . .	29
6.3.3.4 load_pending_transaction() . . . . .	29
6.3.3.5 load_transaction() . . . . .	29
6.3.3.6 send_money() . . . . .	29
6.3.3.7 write_transaction() . . . . .	30
6.3.3.8 write_transactiondata() . . . . .	30
6.4 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/wallet.h File Reference . . . . .	30
6.4.1 Typedef Documentation . . . . .	31
6.4.1.1 Wallet . . . . .	31
6.4.2 Function Documentation . . . . .	31
6.4.2.1 create_account() . . . . .	31
6.4.2.2 get_my_wallet() . . . . .	31
6.5 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h File Reference . . . . .	32
6.5.1 Function Documentation . . . . .	32
6.5.1.1 hash_block_transactions() . . . . .	32
6.5.1.2 sha384_data() . . . . .	32
6.6 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h File Reference . . . . .	33
6.6.1 Macro Definition Documentation . . . . .	33
6.6.1.1 RSA_BEGIN_SIZE . . . . .	33
6.6.1.2 RSA_END_SIZE . . . . .	34
6.6.1.3 RSA_FILE_TOTAL_SIZE . . . . .	34
6.6.1.4 RSA_KEY_SIZE . . . . .	34
6.6.2 Function Documentation . . . . .	34
6.6.2.1 get_keys() . . . . .	34
6.7 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h File Reference . . . . .	34
6.7.1 Function Documentation . . . . .	35
6.7.1.1 get_transaction_data() . . . . .	35
6.7.1.2 sign_block() . . . . .	36

6.7.1.3	<a href="#">sign_block_transactions()</a>	36
6.7.1.4	<a href="#">sign_block_with_key()</a>	36
6.7.1.5	<a href="#">sign_message()</a>	37
6.7.1.6	<a href="#">sign_message_with_key()</a>	38
6.7.1.7	<a href="#">sign_transaction()</a>	38
6.7.1.8	<a href="#">sign_transaction_with_key()</a>	39
6.7.1.9	<a href="#">verify_block_signature()</a>	39
6.7.1.10	<a href="#">verify_signature()</a>	39
6.7.1.11	<a href="#">verify_transaction_signature()</a>	40
6.7.1.12	<a href="#">write_block()</a>	40
6.7.1.13	<a href="#">write_blockdata()</a>	40
6.8	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h File Reference</a>	42
6.8.1	<a href="#">Function Documentation</a>	42
6.8.1.1	<a href="#">last_file_in_folder()</a>	42
6.9	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h File Reference</a>	43
6.9.1	<a href="#">Macro Definition Documentation</a>	43
6.9.1.1	<a href="#">MAX</a>	43
6.9.1.2	<a href="#">MIN</a>	43
6.10	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h File Reference</a>	43
6.10.1	<a href="#">Function Documentation</a>	44
6.10.1.1	<a href="#">safe_fread()</a>	44
6.10.1.2	<a href="#">safe_read()</a>	44
6.10.1.3	<a href="#">safe_send()</a>	45
6.10.1.4	<a href="#">safe_write()</a>	45
6.11	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h File Reference</a>	46
6.11.1	<a href="#">Function Documentation</a>	47
6.11.1.1	<a href="#">client_thread()</a>	47
6.11.1.2	<a href="#">find_empty_connection()</a>	47
6.11.1.3	<a href="#">get_my_node()</a>	47
6.11.1.4	<a href="#">listen_to()</a>	47
6.11.1.5	<a href="#">load_neighbours()</a>	48
6.11.1.6	<a href="#">number_neighbours()</a>	48
6.11.1.7	<a href="#">print_neighbours()</a>	48
6.11.1.8	<a href="#">remove_neighbour()</a>	48
6.11.1.9	<a href="#">save_neighbours()</a>	49
6.11.1.10	<a href="#">set_neighbour()</a>	49
6.12	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h File Reference</a>	49
6.12.1	<a href="#">Function Documentation</a>	50

6.12.1.1	<a href="#">fetch_client_list()</a>	50
6.12.1.2	<a href="#">read_actual_height()</a>	50
6.12.1.3	<a href="#">read_epoch_block()</a>	50
6.12.1.4	<a href="#">read_get_blocks()</a>	50
6.12.1.5	<a href="#">read_header()</a>	51
6.12.1.6	<a href="#">read_pending_transaction_list()</a>	52
6.12.1.7	<a href="#">read_send_block()</a>	52
6.12.1.8	<a href="#">read_vote()</a>	52
6.13	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h</a>	
	File Reference	52
6.13.1	Macro Definition Documentation	54
6.13.1.1	<a href="#">CLIENTMSG</a>	54
6.13.1.2	<a href="#">DD_GET_BLOCKS</a>	54
6.13.1.3	<a href="#">DD_GET_HEIGHT</a>	54
6.13.1.4	<a href="#">DOORSERVER</a>	54
6.13.1.5	<a href="#">HD_ACTUAL_HEIGHT</a>	54
6.13.1.6	<a href="#">HD_CONNECTION_TO_NETWORK</a>	55
6.13.1.7	<a href="#">HD_CONNECTION_TO_NODE</a>	55
6.13.1.8	<a href="#">HD_GET_BLOCKS</a>	55
6.13.1.9	<a href="#">HD_GET_CLIENT_LIST</a>	55
6.13.1.10	<a href="#">HD_GET_PENDING_TRANSACTION_LIST</a>	55
6.13.1.11	<a href="#">HD_REJECT_DEMAND</a>	55
6.13.1.12	<a href="#">HD_SEND_BLOCK</a>	56
6.13.1.13	<a href="#">HD_SEND_CLIENT_LIST</a>	56
6.13.1.14	<a href="#">HD_SEND_EPOCH_BLOCK</a>	56
6.13.1.15	<a href="#">HD_SEND_PENDING_TRANSACTION</a>	56
6.13.1.16	<a href="#">HD_SEND_VOTE</a>	56
6.13.1.17	<a href="#">IM_CLIENT</a>	56
6.13.1.18	<a href="#">IM_SERVER</a>	57
6.13.1.19	<a href="#">MANAGERMSG</a>	57
6.13.1.20	<a href="#">MAX_CONNECTION</a>	57
6.13.1.21	<a href="#">MAX_NEIGHBOURS</a>	57
6.13.1.22	<a href="#">MAX_SERVER</a>	57
6.13.1.23	<a href="#">MAX_VALIDATORS_PER_BLOCK</a>	57
6.13.1.24	<a href="#">NB_HARD_CODED_ADDR</a>	58
6.13.1.25	<a href="#">NODESERVER</a>	58
6.13.1.26	<a href="#">P_VERSION</a>	58
6.13.1.27	<a href="#">SERVERMSG</a>	58
6.13.1.28	<a href="#">SIZE_OF_HOSTNAME</a>	58
6.13.1.29	<a href="#">SOL_TCP</a>	58
6.13.1.30	<a href="#">STATIC_PORT</a>	59
6.13.1.31	<a href="#">TCP_USER_TIMEOUT</a>	59



6.13.1.32 WARNINGMSG	59
6.13.2 Typedef Documentation	59
6.13.2.1 client_connection	59
6.13.2.2 infos_st	59
6.13.2.3 Neighbour	59
6.13.2.4 Node	60
6.13.2.5 th_arg	60
6.13.3 Function Documentation	60
6.13.3.1 __attribute__()	60
6.13.4 Variable Documentation	60
6.13.4.1 get_blocks_t	60
6.13.4.2 HARD_CODED_ADDR	60
6.14 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_data.h File Reference	61
6.14.1 Function Documentation	61
6.14.1.1 send_actual_height()	61
6.14.1.2 send_client_list()	61
6.14.1.3 send_get_blocks()	62
6.14.1.4 send_pending_transaction_list()	62
6.14.1.5 send_reject_demand()	62
6.14.1.6 send_send_block()	62
6.15 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h File Reference	63
6.15.1 Function Documentation	63
6.15.1.1 init_server()	63
6.16 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h File Reference	64
6.16.1 Function Documentation	65
6.16.1.1 add_contact()	65
6.16.1.2 add_contact_to_combobox()	65
6.16.1.3 add_contacts_from_file()	66
6.16.1.4 add_new_blockinfo()	66
6.16.1.5 add_transaction_from_file()	66
6.16.1.6 add_transaction_with_contact()	66
6.16.1.7 add_transaction_with_pkey()	66
6.16.1.8 change_label_text()	67
6.16.1.9 get_public_key_from_contacts()	67
6.16.1.10 load_contacts_from_file()	67
6.16.1.11 load_transaction_from_file()	67
6.16.1.12 on_add_contact_button1_press()	67
6.16.1.13 on_connect_but_press()	68
6.16.1.14 on_create_key_but1_press()	68
6.16.1.15 on_create_key_but2_press()	68
6.16.1.16 on_invest_button1_press()	68

6.16.1.17 on_invest_button2_press()	69
6.16.1.18 on_main_window_delete()	69
6.16.1.19 on_main_window_destroy()	70
6.16.1.20 on_pkey_button_press()	70
6.16.1.21 on_recover_button1_press()	70
6.16.1.22 on_recover_button2_press()	71
6.16.1.23 on_transaction_button_press()	71
6.16.1.24 setup()	71
6.16.1.25 update_labels()	72
6.16.1.26 update_sync()	72
6.16.2 Variable Documentation	72
6.16.2.1 block_amount_label	72
6.16.2.2 blocksinfo	72
6.16.2.3 connections_label	73
6.16.2.4 mempool_label	73
6.16.2.5 synchro_label	73
6.17 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/epoch↔ _man.h File Reference	73
6.17.1 Function Documentation	73
6.17.1.1 create_epoch_block()	74
6.17.1.2 flush_pending_transactions()	74
6.17.1.3 get_epoch_man_pkey()	74
6.17.1.4 give_punishments_and_rewards()	75
6.18 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validation↔ _engine.h File Reference	75
6.18.1 Function Documentation	75
6.18.1.1 send_verdict()	76
6.18.1.2 validate_transactions()	76
6.19 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h File Reference	77
6.19.1 Macro Definition Documentation	78
6.19.1.1 MAX_VALIDATORS_PER_BLOCK	78
6.19.2 Function Documentation	78
6.19.2.1 get_comittee()	78
6.19.2.2 get_next_comittee()	78
6.19.2.3 get_validator_id()	79
6.19.2.4 get_validator_pkey()	79
6.19.2.5 get_validator_power()	80
6.19.2.6 get_validator_stake()	80
6.19.2.7 get_validators_states_block_height_validity()	81
6.19.2.8 get_validators_states_nb_validators()	81
6.19.2.9 get_validators_states_total_stake()	81
6.19.2.10 pop_stake()	81

6.19.2.11 push_stake()	82
6.20 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/P2P_Protocol.md File Reference	82
6.21 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md File Reference	82
6.22 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c File Reference	82
6.22.1 Function Documentation	83
6.22.1.1 connection_to_others()	83
6.22.1.2 join_network_door()	83
6.22.1.3 main()	84
6.22.1.4 update_blockchain()	84
6.22.1.5 update_blockchain_height()	84
6.22.2 Variable Documentation	84
6.22.2.1 client_connections	84
6.23 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/client.c File Reference	84
6.23.1 Function Documentation	85
6.23.1.1 client_thread()	85
6.23.1.2 find_empty_connection()	85
6.23.1.3 get_my_node()	86
6.23.1.4 listen_to()	86
6.23.1.5 load_neighbours()	86
6.23.1.6 number_neighbours()	87
6.23.1.7 print_neighbours()	87
6.23.1.8 remove_neighbour()	87
6.23.1.9 save_neighbours()	87
6.23.1.10 set_neighbour()	88
6.23.2 Variable Documentation	88
6.23.2.1 client_connections	88
6.24 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c File Reference	88
6.24.1 Macro Definition Documentation	89
6.24.1.1 GENESIS_RSA_PUB_1	89
6.24.1.2 GENESIS_RSA_PUB_2	90
6.24.2 Function Documentation	90
6.24.2.1 convert_data_to_block()	90
6.24.2.2 convert_data_to_blockdata()	90
6.24.2.3 free_block()	90
6.24.2.4 get_block()	91
6.24.2.5 get_blockdata_data()	91
6.24.2.6 get_genesis_block()	91
6.24.2.7 get_last_block_height()	92
6.24.2.8 get_next_block()	92
6.24.2.9 get_prev_block()	92

6.24.2.10 load_blockchain()	93
6.24.2.11 load_last_blockchain()	93
6.24.2.12 write_block()	93
6.24.2.13 write_block_file()	94
6.24.2.14 write_blockdata()	94
6.25 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/blockchain_header.c File Reference	94
6.25.1 Function Documentation	95
6.25.1.1 gen_blockchain_header()	95
6.25.1.2 write_block_header()	95
6.26 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/transaction.c File Reference	95
6.26.1 Function Documentation	95
6.26.1.1 add_pending_transaction()	96
6.26.1.2 convert_data_to_transactiondata()	96
6.26.1.3 get_transaction_data()	96
6.26.1.4 load_pending_transaction()	96
6.26.1.5 load_transaction()	97
6.26.1.6 write_transaction()	97
6.26.1.7 write_transactiondata()	97
6.27 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c File Reference	97
6.27.1 Function Documentation	98
6.27.1.1 create_account()	98
6.27.1.2 get_my_wallet()	98
6.28 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/hash.c File Reference	98
6.28.1 Function Documentation	99
6.28.1.1 hash_block_transactions()	99
6.28.1.2 sha384_data()	100
6.29 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/rsa.c File Reference	100
6.29.1 Macro Definition Documentation	101
6.29.1.1 RSA_NUM_E	101
6.29.2 Function Documentation	101
6.29.2.1 get_keys()	101
6.30 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/signature.c File Reference	101
6.30.1 Function Documentation	102
6.30.1.1 sign_block()	102
6.30.1.2 sign_block_transactions()	102
6.30.1.3 sign_block_with_key()	103
6.30.1.4 sign_message()	103
6.30.1.5 sign_message_with_key()	103

6.30.1.6	<a href="#">sign_transaction()</a>	105
6.30.1.7	<a href="#">sign_transaction_with_key()</a>	105
6.30.1.8	<a href="#">verify_block_signature()</a>	105
6.30.1.9	<a href="#">verify_signature()</a>	106
6.30.1.10	<a href="#">verify_transaction_signature()</a>	106
6.31	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/files.c</a> File Reference	107
6.31.1	Macro Definition Documentation	107
6.31.1.1	<a href="#">_GNU_SOURCE</a>	107
6.31.2	Function Documentation	107
6.31.2.1	<a href="#">last_file_in_folder()</a>	107
6.32	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/safe.c</a> File Reference	108
6.32.1	Function Documentation	108
6.32.1.1	<a href="#">safe_fread()</a>	108
6.32.1.2	<a href="#">safe_read()</a>	109
6.32.1.3	<a href="#">safe_send()</a>	109
6.32.1.4	<a href="#">safe_write()</a>	110
6.33	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/get_data.c</a> File Reference	110
6.33.1	Function Documentation	111
6.33.1.1	<a href="#">fetch_client_list()</a>	111
6.33.1.2	<a href="#">process_header()</a>	111
6.33.1.3	<a href="#">read_actual_height()</a>	111
6.33.1.4	<a href="#">read_epoch_block()</a>	111
6.33.1.5	<a href="#">read_get_blocks()</a>	112
6.33.1.6	<a href="#">read_header()</a>	112
6.33.1.7	<a href="#">read_pending_transaction_list()</a>	112
6.33.1.8	<a href="#">read_send_block()</a>	112
6.33.1.9	<a href="#">read_vote()</a>	113
6.34	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/network.c</a> File Reference	113
6.34.1	Variable Documentation	113
6.34.1.1	<a href="#">HARD_CODED_ADDR</a>	113
6.35	<a href="#">/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/send_data.c</a> File Reference	113
6.35.1	Function Documentation	114
6.35.1.1	<a href="#">send_actual_height()</a>	114
6.35.1.2	<a href="#">send_client_list()</a>	114
6.35.1.3	<a href="#">send_get_blocks()</a>	115
6.35.1.4	<a href="#">send_pending_transaction_list()</a>	115
6.35.1.5	<a href="#">send_reject_demand()</a>	115
6.35.1.6	<a href="#">send_send_block()</a>	115

6.36	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/server.c File Reference	115
6.36.1	Function Documentation	116
6.36.1.1	accept_connection()	116
6.36.1.2	init_server()	116
6.36.1.3	redirect_connection()	116
6.37	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c File Reference	117
6.37.1	Function Documentation	117
6.37.1.1	main()	117
6.38	/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/ui/ui.c File Reference	117
6.38.1	Function Documentation	119
6.38.1.1	add_contact()	119
6.38.1.2	add_contact_to_combobox()	119
6.38.1.3	add_contacts_from_file()	119
6.38.1.4	add_new_blockinfo()	120
6.38.1.5	add_transaction_from_file()	120
6.38.1.6	add_transaction_with_contact()	120
6.38.1.7	add_transaction_with_pkey()	120
6.38.1.8	change_label_text()	120
6.38.1.9	get_public_key_from_contacts()	121
6.38.1.10	load_contacts_from_file()	121
6.38.1.11	load_transactions_from_file()	121
6.38.1.12	on_add_contact_button1_press()	121
6.38.1.13	on_connect_but_press()	121
6.38.1.14	on_create_key_but1_press()	122
6.38.1.15	on_create_key_but2_press()	122
6.38.1.16	on_invest_button1_press()	122
6.38.1.17	on_invest_button2_press()	122
6.38.1.18	on_main_window_delete()	122
6.38.1.19	on_main_window_destroy()	123
6.38.1.20	on_pkey_button_press()	123
6.38.1.21	on_recover_button1_press()	123
6.38.1.22	on_recover_button2_press()	123
6.38.1.23	on_transaction_button_press()	124
6.38.1.24	setup()	124
6.38.1.25	update_labels()	124
6.38.1.26	update_sync()	124
6.38.2	Variable Documentation	124
6.38.2.1	balance_1	125
6.38.2.2	balance_2	125
6.38.2.3	contacts_combo	125
6.38.2.4	cr1_combo	125

6.38.2.5 cr1_con	125
6.38.2.6 cr1_th	125
6.38.2.7 cr2_con	126
6.38.2.8 cr2_th	126
6.38.2.9 cr3_th	126
6.38.2.10 cx1_con	126
6.38.2.11 cx1_th	126
6.38.2.12 cx2_con	126
6.38.2.13 cx2_th	127
6.38.2.14 cx3_th	127
6.38.2.15 invest_entry	127
6.38.2.16 key_entry	127
6.38.2.17 latest_block_name1	127
6.38.2.18 latest_block_name2	127
6.38.2.19 latest_block_name3	128
6.38.2.20 ls_combo	128
6.38.2.21 name_entry_con	128
6.38.2.22 password_entry1	128
6.38.2.23 password_entry2	128
6.38.2.24 password_error_label	128
6.38.2.25 private_key_label	129
6.38.2.26 progress_bar_blockchain	129
6.38.2.27 public_key_entry_con	129
6.38.2.28 recipient_key	129
6.38.2.29 recover_entry	129
6.38.2.30 stake_label1	129
6.38.2.31 stake_label2	130
6.38.2.32 stake_label3	130
6.38.2.33 transa_amount	130
6.38.2.34 ts_con	130
6.38.2.35 ts_th	130
6.38.2.36 tv_con	130
6.38.2.37 tv_th	131
6.39 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/epoch↔ _man.c File Reference	131
6.39.1 Function Documentation	131
6.39.1.1 get_epoch_man_pkey()	131
6.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validation↔ _engine.c File Reference	132
6.41 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validators.c File Reference	132
6.41.1 Macro Definition Documentation	132
6.41.1.1 NB_RSA_CHUNK	133

6.41.2 Function Documentation . . . . .	133
6.41.2.1 define_nb_validators() . . . . .	133
6.41.2.2 get_comittee() . . . . .	133
6.41.2.3 get_next_comittee() . . . . .	133
6.41.2.4 get_validator_id() . . . . .	134
6.41.2.5 get_validator_pkey() . . . . .	134
6.41.2.6 get_validator_power() . . . . .	135
6.41.2.7 get_validator_stake() . . . . .	135
6.41.2.8 get_validators_states_block_height_validity() . . . . .	136
6.41.2.9 get_validators_states_nb_validators() . . . . .	136
6.41.2.10 get_validators_states_total_stake() . . . . .	136
6.41.2.11 hash_block_transactions_epoch() . . . . .	136
6.42 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c File Reference . . . . .	137
6.42.1 Function Documentation . . . . .	137
6.42.1.1 main() . . . . .	137
6.43 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/serverdoor.c File Reference . . . . .	137
6.43.1 Function Documentation . . . . .	137
6.43.1.1 main() . . . . .	138
6.44 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c File Reference . . . . .	138
6.44.1 Function Documentation . . . . .	138
6.44.1.1 main() . . . . .	138
6.45 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_blockchain↵_files.c File Reference . . . . .	138
6.45.1 Macro Definition Documentation . . . . .	139
6.45.1.1 GEN_BLC_F_C . . . . .	139
6.45.2 Function Documentation . . . . .	139
6.45.2.1 gen_blockchain() . . . . .	139
6.45.2.2 rand_data() . . . . .	139
6.46 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_validators↵_file.c File Reference . . . . .	140
6.46.1 Macro Definition Documentation . . . . .	140
6.46.1.1 GEN_VALIDATORS_FILE_H . . . . .	140
6.46.1.2 NB_FAKE_VALIDATORS . . . . .	140
6.46.1.3 str . . . . .	141
6.46.2 Function Documentation . . . . .	141
6.46.2.1 gen_validators_file() . . . . .	141
6.47 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/blockchain/block↵_test.h File Reference . . . . .	141
6.47.1 Function Documentation . . . . .	142
6.47.1.1 block_test() . . . . .	142
6.48 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/rsa↵_test.h File Reference . . . . .	142



6.48.1 Function Documentation . . . . .	142
6.48.1.1 get_keys_equality_test() . . . . .	142
6.48.1.2 get_keys_test() . . . . .	142
6.49 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/signature↔ _test.h File Reference . . . . .	143
6.49.1 Function Documentation . . . . .	143
6.49.1.1 verify_sign_test() . . . . .	143
6.50 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/network/client↔ _test.h File Reference . . . . .	143
6.50.1 Function Documentation . . . . .	143
6.50.1.1 network_test() . . . . .	143
6.51 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/validation/validations↔ _test.h File Reference . . . . .	144
6.51.1 Function Documentation . . . . .	144
6.51.1.1 validations_test() . . . . .	144
6.52 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main_test.c File Refer- ence . . . . .	144
6.52.1 Macro Definition Documentation . . . . .	144
6.52.1.1 MAIN_TEST_C . . . . .	145
6.52.2 Function Documentation . . . . .	145
6.52.2.1 main() . . . . .	145
6.53 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/blockchain/block↔ _test.c File Reference . . . . .	145
6.53.1 Macro Definition Documentation . . . . .	145
6.53.1.1 BLOCK_TEST_C . . . . .	146
6.53.1.2 NB_BLOCK_PER_CHUNK . . . . .	146
6.53.1.3 NB MOCK_BLOCKS . . . . .	146
6.53.2 Function Documentation . . . . .	146
6.53.2.1 block_test() . . . . .	146
6.54 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa↔ _test.c File Reference . . . . .	146
6.54.1 Macro Definition Documentation . . . . .	147
6.54.1.1 RSA_SIZE_C . . . . .	147
6.54.2 Function Documentation . . . . .	147
6.54.2.1 get_keys_equality_test() . . . . .	147
6.54.2.2 get_keys_test() . . . . .	147
6.55 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/signature↔ _test.c File Reference . . . . .	147
6.55.1 Function Documentation . . . . .	148
6.55.1.1 verify_sign_test() . . . . .	148
6.56 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/client↔ test.c File Reference . . . . .	148
6.56.1 Macro Definition Documentation . . . . .	148
6.56.1.1 CLIENT_TEST_C . . . . .	149

6.56.2 Function Documentation . . . . .	149
6.56.2.1 network_test() . . . . .	149
6.56.3 Variable Documentation . . . . .	149
6.56.3.1 client_connections . . . . .	149
6.57 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/validation/validations_↵ _test.c File Reference . . . . .	149
6.57.1 Function Documentation . . . . .	149
6.57.1.1 validations_test() . . . . .	150
6.58 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/tests_macros.h File Reference . . . . .	150
6.58.1 Macro Definition Documentation . . . . .	150
6.58.1.1 DEBUG . . . . .	150
6.58.1.2 LOG . . . . .	150
6.58.1.3 TEST_FAILED . . . . .	151
6.58.1.4 TEST_PASSED . . . . .	151
6.58.1.5 TEST_WARNING . . . . .	151
6.59 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit_testing.c File Ref- erence . . . . .	151
6.59.1 Function Documentation . . . . .	152
6.59.1.1 main() . . . . .	152
<b>Index</b>	<b>153</b>

## Chapter 1

# PEPITAS NETWORK PROTOCOL

## 1.1 HEADERS

### 1.1.1 Sync Headers

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

### 1.1.2 Running Headers

1. SEND PENDING TRANSACTION

### 1.1.3 Validating Headers

1. SEND BLOCK EPOCHMAN
2. SEND VOTE

### 1.1.4 CONNECTION TO NETWORK

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

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

### 1.1.5 CONNECTION TO NODE

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

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

### 1.1.6 GET BLOCKS

**Message:** char \* : "GET BLOCKS\r\n\r\n" uint32\_t : P\_VERSION char : Number of demand (max 50) size\_t \* : [Block](#) height

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

### 1.1.7 ACTUAL HEIGHT

**Message:** char \* : "ACTUAL HEIGHT\r\n\r\n" size\_t : [Block](#) height

**Description** Send my actual blockchain height.

### 1.1.8 SEND BLOCK

**Message:** char \* : "SEND BLOCK\r\n\r\n" size\_t : [Block](#) height size\_t : [Block](#) size char \* : [Block](#) struct

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

### 1.1.9 GET PENDING TRANSACTION LIST

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

**Description** Call "SEND PENDING TRANSACTION".

### 1.1.10 REJECT DEMAND

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

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

### 1.1.11 SEND PENDING TRANSACTION

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

**Description** Send the epoch block of a committee member.

### 1.1.12 SEND EPOCH BLOCK

**Message:** char \* : "SEND EPOCH BLOCK\r\n\r\n" char \* : [Block](#) struct

**Description** Send the epoch block of a committee member.

### 1.1.13 SEND VOTE

**Message:** char \* : "SEND VOTE\r\n\r\n" char \*: Epoch creator pk size\_t : block height int epoch\_id: creator char : 0 = False 1 = True char \* : signature of vote precedent vars

**Description** Send the vote of a committee member.

## Chapter 2

# PEPITAS

C cryptocurrency.

## 2.1 CODING STYLE

### 2.1.1 Coding case

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

### 2.1.2 Tests

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

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



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

Block	9
BlockData	10
blockinfo	13
ChunkBlockchain	13
client_connection	14
infos_st	16
Neighbour	17
Node	18
th_arg	19
Transaction	19
TransactionData	20
Wallet	23





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/block.h . . .	25
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/blockchain_header.h	25
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h	26
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/wallet.h . . .	30
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h . . .	32
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h . . .	33
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/signature.h	34
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h . . . . .	42
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/math.h . . . . .	43
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/safe.h . . . . .	43
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h . . . . .	46
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get_data.h . . . . .	49
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h . . . . .	52
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send_data.h . . . . .	61
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h . . . . .	63
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h . . . . .	64
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/epoch_man.h	73
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validation_engine.h	75
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h . . . . .	77
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c . . . . .	82
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/gui.c . . . . .	137
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c . . . . .	117
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/serverdoor.c . . . . .	137
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/sign.c . . . . .	138
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/block.c . . . . .	88
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/blockchain_header.c	94
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/transaction.c	95
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c . . . . .	97
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/hash.c . . . . .	98
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/rsa.c . . . . .	100

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/signature.c	
101	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/files.c	107
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/safe.c	108
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/client.c	84
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/get_data.c	110
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/network.c	113
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/send_data.c	113
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/server.c	115
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/ui/ui.c	117
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/epoch_man.c	131
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validation_engine.c	
132	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validators.c	132
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main_test.c	144
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/tests_macros.h	150
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit_testing.c	151
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_blockchain_files.c	
138	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN_validators_file.c	140
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/blockchain/block_test.h	
141	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/rsa_test.h	
142	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/signature_test.h	
143	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/network/client_test.h	
143	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/validation/validations_test.h	
144	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/blockchain/block_test.c	145
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa_test.c	
146	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/signature_test.c	
147	
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/client_test.c	148
/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/validation/validations_test.c	
149	

## Chapter 5

# Data Structure Documentation

### 5.1 Block Struct Reference

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

Collaboration diagram for Block:

#### Data Fields

- uint16\_t [chunk\\_id](#)
- [BlockData](#) [block\\_data](#)
- char [block\\_signature](#) [256]
- char [validators\\_votes](#) [MAX\_VALIDATORS\_PER\_BLOCK/8]
- char [vote\\_signature](#) [MAX\_VALIDATORS\_PER\_BLOCK - 1][256]

#### 5.1.1 Detailed Description

Definition at line 51 of file block.h.

#### 5.1.2 Field Documentation

##### 5.1.2.1 [block\\_data](#)

[BlockData](#) [block\\_data](#)

Definition at line 54 of file block.h.

### 5.1.2.2 block\_signature

```
char block_signature[256]
```

Definition at line 56 of file block.h.

### 5.1.2.3 chunk\_id

```
uint16_t chunk_id
```

Definition at line 53 of file block.h.

### 5.1.2.4 validators\_votes

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

Definition at line 59 of file block.h.

### 5.1.2.5 vote\_signature

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

Definition at line 60 of file block.h.

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

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

## 5.2 BlockData Struct Reference

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

Collaboration diagram for BlockData:

### Data Fields

- char [magic](#)
- int [epoch\\_id](#)
- char [is\\_prev\\_block\\_valid](#)
- char [previous\\_block\\_hash](#) [SHA384\_DIGEST\_LENGTH \*2+1]
- size\_t [height](#)
- uint16\_t [nb\\_transactions](#)
- [Transaction](#) \*\* [transactions](#)
- int [nb\\_validators](#)
- RSA \* [validators\\_public\\_keys](#) [MAX\_VALIDATORS\_PER\_BLOCK]
- char [prev\\_validators\\_votes](#) [MAX\_VALIDATORS\_PER\_BLOCK/8]
- time\_t [block\\_timestamp](#)

### 5.2.1 Detailed Description

Definition at line 33 of file block.h.

### 5.2.2 Field Documentation

#### 5.2.2.1 block\_timestamp

```
time_t block_timestamp
```

Definition at line 48 of file block.h.

#### 5.2.2.2 epoch\_id

```
int epoch_id
```

Definition at line 36 of file block.h.

#### 5.2.2.3 height

```
size_t height
```

Definition at line 39 of file block.h.

#### 5.2.2.4 is\_prev\_block\_valid

```
char is_prev_block_valid
```

Definition at line 37 of file block.h.

#### 5.2.2.5 magic

```
char magic
```

Definition at line 35 of file block.h.

#### 5.2.2.6 nb\_transactions

```
uint16_t nb_transactions
```

Definition at line 41 of file block.h.

#### 5.2.2.7 nb\_validators

```
int nb_validators
```

Definition at line 45 of file block.h.

#### 5.2.2.8 prev\_validators\_votes

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

Definition at line 47 of file block.h.

#### 5.2.2.9 previous\_block\_hash

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

Definition at line 38 of file block.h.

#### 5.2.2.10 transactions

```
Transaction** transactions
```

Definition at line 42 of file block.h.

#### 5.2.2.11 validators\_public\_keys

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

Definition at line 46 of file block.h.

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

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

## 5.3 blockinfo Struct Reference

```
#include <ui.h>
```

### Data Fields

- `size_t` [height](#)
- `size_t` [transactions](#)

### 5.3.1 Detailed Description

Definition at line 17 of file `ui.h`.

### 5.3.2 Field Documentation

#### 5.3.2.1 height

```
size_t height
```

Definition at line 19 of file `ui.h`.

#### 5.3.2.2 transactions

```
size_t transactions
```

Definition at line 20 of file `ui.h`.

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

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

## 5.4 ChunkBlockchain Struct Reference

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

Collaboration diagram for `ChunkBlockchain`:

## Data Fields

- `size_t` [chunk\\_nb](#)
- `Block**` [chunk](#)
- `int16_t` [nb\\_blocks](#)

### 5.4.1 Detailed Description

Definition at line 63 of file `block.h`.

### 5.4.2 Field Documentation

#### 5.4.2.1 `chunk`

`Block**` `chunk`

Definition at line 66 of file `block.h`.

#### 5.4.2.2 `chunk_nb`

`size_t` `chunk_nb`

Definition at line 65 of file `block.h`.

#### 5.4.2.3 `nb_blocks`

`int16_t` `nb_blocks`

Definition at line 67 of file `block.h`.

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

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

## 5.5 `client_connection` Struct Reference

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



## Data Fields

- pthread\_t [thread](#)
- sem\_t [lock](#)
- int [demand](#)
- int [clientfd](#)
- size\_t [Payloadsize](#)
- void \* [Payload](#)
- size\_t [actual\\_client\\_height](#)

### 5.5.1 Detailed Description

Definition at line 44 of file network.h.

### 5.5.2 Field Documentation

#### 5.5.2.1 actual\_client\_height

```
size_t actual_client_height
```

Definition at line 52 of file network.h.

#### 5.5.2.2 clientfd

```
int clientfd
```

Definition at line 49 of file network.h.

#### 5.5.2.3 demand

```
int demand
```

Definition at line 48 of file network.h.

#### 5.5.2.4 lock

```
sem_t lock
```

Definition at line 47 of file network.h.

#### 5.5.2.5 Payload

```
void* Payload
```

Definition at line 51 of file network.h.

#### 5.5.2.6 Payloadsize

```
size_t Payloadsize
```

Definition at line 50 of file network.h.

#### 5.5.2.7 thread

```
pthread_t thread
```

Definition at line 46 of file network.h.

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

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

## 5.6 infos\_st Struct Reference

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

### Data Fields

- char [is\\_synchronize](#)
- size\_t [actual\\_height](#)
- char [serv\\_type](#)

#### 5.6.1 Detailed Description

Definition at line 55 of file network.h.

#### 5.6.2 Field Documentation

### 5.6.2.1 actual\_height

```
size_t actual_height
```

Definition at line 58 of file network.h.

### 5.6.2.2 is\_synchronize

```
char is_synchronize
```

Definition at line 57 of file network.h.

### 5.6.2.3 serv\_type

```
char serv_type
```

Definition at line 59 of file network.h.

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

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

## 5.7 Neighbour Struct Reference

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

### Data Fields

- int [family](#)
- char \* [hostname](#)

### 5.7.1 Detailed Description

Definition at line 33 of file network.h.

### 5.7.2 Field Documentation

### 5.7.2.1 family

```
int family
```

Definition at line 35 of file network.h.

### 5.7.2.2 hostname

```
char* hostname
```

Definition at line 36 of file network.h.

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

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

## 5.8 Node Struct Reference

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

Collaboration diagram for Node:

### Data Fields

- [Neighbour](#) \* [neighbours](#)

### 5.8.1 Detailed Description

Definition at line 39 of file network.h.

### 5.8.2 Field Documentation

#### 5.8.2.1 neighbours

```
Neighbour* neighbours
```

Definition at line 41 of file network.h.

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

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

## 5.9 th\_arg Struct Reference

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

Collaboration diagram for th\_arg:

### Data Fields

- [infos\\_st](#) \* [infos](#)
- [client\\_connection](#) \* [client\\_con](#)

### 5.9.1 Detailed Description

Definition at line 61 of file network.h.

### 5.9.2 Field Documentation

#### 5.9.2.1 client\_con

```
client\_connection* client\_con
```

Definition at line 64 of file network.h.

#### 5.9.2.2 infos

```
infos\_st* infos
```

Definition at line 63 of file network.h.

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

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

## 5.10 Transaction Struct Reference

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

Collaboration diagram for Transaction:

## Data Fields

- [TransactionData transaction\\_data](#)
- char [transaction\\_signature](#) [256]

### 5.10.1 Detailed Description

Definition at line 43 of file transaction.h.

### 5.10.2 Field Documentation

#### 5.10.2.1 transaction\_data

[TransactionData transaction\\_data](#)

Definition at line 45 of file transaction.h.

#### 5.10.2.2 transaction\_signature

char [transaction\\_signature](#)[256]

Definition at line 47 of file transaction.h.

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

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

## 5.11 TransactionData Struct Reference

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

## Data Fields

- char [magic](#)
- char [type](#)
- RSA \* [sender\\_public\\_key](#)
- RSA \* [receiver\\_public\\_key](#)
- RSA \* [organisation\\_public\\_key](#)
- size\_t [amount](#)
- size\_t [sender\\_remaining\\_money](#)
- size\_t [receiver\\_remaining\\_money](#)
- time\_t [transaction\\_timestamp](#)
- char [cause](#) [512]
- char [asset](#) [512]

### 5.11.1 Detailed Description

Definition at line 24 of file transaction.h.

### 5.11.2 Field Documentation

#### 5.11.2.1 amount

```
size_t amount
```

Definition at line 32 of file transaction.h.

#### 5.11.2.2 asset

```
char asset[512]
```

Definition at line 40 of file transaction.h.

#### 5.11.2.3 cause

```
char cause[512]
```

Definition at line 39 of file transaction.h.

#### 5.11.2.4 magic

```
char magic
```

Definition at line 26 of file transaction.h.

#### 5.11.2.5 organisation\_public\_key

```
RSA* organisation_public_key
```

Definition at line 31 of file transaction.h.

#### 5.11.2.6 receiver\_public\_key

`RSA* receiver_public_key`

Definition at line 30 of file transaction.h.

#### 5.11.2.7 receiver\_remaining\_money

`size_t receiver_remaining_money`

Definition at line 34 of file transaction.h.

#### 5.11.2.8 sender\_public\_key

`RSA* sender_public_key`

Definition at line 29 of file transaction.h.

#### 5.11.2.9 sender\_remaining\_money

`size_t sender_remaining_money`

Definition at line 33 of file transaction.h.

#### 5.11.2.10 transaction\_timestamp

`time_t transaction_timestamp`

Definition at line 35 of file transaction.h.

#### 5.11.2.11 type

`char type`

Definition at line 27 of file transaction.h.

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

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



## 5.12 Wallet Struct Reference

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

### Data Fields

- RSA \* [priv\\_key](#)
- RSA \* [pub\\_key](#)
- size\_t [amount](#)
- char [is\\_validator](#)

### 5.12.1 Detailed Description

Definition at line 10 of file wallet.h.

### 5.12.2 Field Documentation

#### 5.12.2.1 amount

```
size_t amount
```

Definition at line 15 of file wallet.h.

#### 5.12.2.2 is\_validator

```
char is_validator
```

Definition at line 16 of file wallet.h.

#### 5.12.2.3 priv\_key

```
RSA* priv_key
```

Definition at line 12 of file wallet.h.

#### 5.12.2.4 pub\_key

```
RSA* pub_key
```

Definition at line 13 of file wallet.h.

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

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



## Chapter 6

# File Documentation

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

```
#include <string.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <err.h>
#include <errno.h>
#include <openssl/sha.h>
#include <openssl/pem.h>
#include <openssl/rsa.h>
#include <openssl/crypto.h>
#include <fcntl.h>
#include <sys/types.h>
#include "transaction.h"
#include "misc/files.h"
Include dependency graph for block.h:
```

### 6.2 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/blockchain\_header.h File Reference

```
#include "blockchain/block.h"
#include "network/network.h"
#include <sys/stat.h>
#include <stdio.h>
```

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

#### Functions

- void [gen\\_blockchain\\_header](#) (infos\_st \*infos)

## 6.2.1 Function Documentation

### 6.2.1.1 `gen_blockchain_header()`

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

Definition at line 9 of file `blockchain_header.c`.

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

## 6.3 `/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/transaction.h` File Reference

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

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

## Data Structures

- struct [TransactionData](#)
- struct [Transaction](#)

## Macros

- #define [TRANSACTION\\_DATA\\_SIZE](#) `sizeof(size_t) * 3 + sizeof(time_t) + (512 * 2)`
- #define [TRANSACTION\\_SIZE](#) `sizeof(size_t) + 2048 + TRANSACTION_DATA_SIZE`
- #define [T\\_TYPE\\_DEFAULT](#) 0
- #define [T\\_TYPE\\_ADD\\_STAKE](#) 1
- #define [T\\_TYPE\\_WITHDRAW\\_STAKE](#) 2
- #define [T\\_TYPE\\_STAKE\\_TO\\_STAKE](#) 3

## Typedefs

- typedef struct [TransactionData](#) [TransactionData](#)
- typedef struct [Transaction](#) [Transaction](#)

**Functions**

- int [send\\_money](#) (size\_t amount, u\_int64\_t receiver\_public\_key)  
*Send 'amount' money to 'receiver\_public\_key'. This will broadcast a transaction to the network.*
- void [write\\_transactiondata](#) (TransactionData \*transaction, int fd)
- void [write\\_transaction](#) (Transaction \*transaction, int fd)
- void [get\\_transaction\\_data](#) (Transaction \*trans, char \*\*buff, size\_t \*index)
- void [convert\\_data\\_to\\_transactiondata](#) (TransactionData \*transactiondata, int fd)
- void [load\\_transaction](#) (Transaction \*transaction, int fd)
- Transaction \* [load\\_pending\\_transaction](#) (time\_t timestamp)
- void [add\\_pending\\_transaction](#) (Transaction \*transaction)

**6.3.1 Macro Definition Documentation****6.3.1.1 T\_TYPE\_ADD\_STAKE**

```
#define T_TYPE_ADD_STAKE 1
```

Definition at line 20 of file transaction.h.

**6.3.1.2 T\_TYPE\_DEFAULT**

```
#define T_TYPE_DEFAULT 0
```

Definition at line 19 of file transaction.h.

**6.3.1.3 T\_TYPE\_STAKE\_TO\_STAKE**

```
#define T_TYPE_STAKE_TO_STAKE 3
```

Definition at line 22 of file transaction.h.

**6.3.1.4 T\_TYPE\_WITHDRAW\_STAKE**

```
#define T_TYPE_WITHDRAW_STAKE 2
```

Definition at line 21 of file transaction.h.

#### 6.3.1.5 TRANSACTION\_DATA\_SIZE

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

Definition at line 16 of file transaction.h.

#### 6.3.1.6 TRANSACTION\_SIZE

```
#define TRANSACTION_SIZE sizeof(size_t) + 2048 + TRANSACTION\_DATA\_SIZE
```

Definition at line 17 of file transaction.h.

### 6.3.2 Typedef Documentation

#### 6.3.2.1 Transaction

```
typedef struct Transaction Transaction
```

#### 6.3.2.2 TransactionData

```
typedef struct TransactionData TransactionData
```

### 6.3.3 Function Documentation

#### 6.3.3.1 add\_pending\_transaction()

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

Definition at line 158 of file transaction.c.

Here is the call graph for this function:

### 6.3.3.2 convert\_data\_to\_transactiondata()

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

Definition at line 99 of file transaction.c.

Here is the caller graph for this function:

### 6.3.3.3 get\_transaction\_data()

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

Definition at line 48 of file transaction.c.

Here is the caller graph for this function:

### 6.3.3.4 load\_pending\_transaction()

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

Definition at line 145 of file transaction.c.

Here is the call graph for this function:

### 6.3.3.5 load\_transaction()

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

Definition at line 135 of file transaction.c.

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

### 6.3.3.6 send\_money()

```
int send_money (
    size_t amount,
    u_int64_t receiver_public_key )
```

Send 'amount' money to 'receiver\_public\_key'. This will broadcast a transaction to the network.

**Parameters**

<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

**6.3.3.7 write\_transaction()**

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

Definition at line 42 of file transaction.c.

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

**6.3.3.8 write\_transactiondata()**

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

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

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

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

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

**Data Structures**

- struct [Wallet](#)

**Typedefs**

- typedef struct [Wallet](#) [Wallet](#)



## Functions

- `Wallet * get_my_wallet ()`  
*Get my wallet object.*
- `int create_account ()`  
*Creates an account in local and broadcasts the creation to the network.*

### 6.4.1 Typedef Documentation

#### 6.4.1.1 Wallet

```
typedef struct Wallet Wallet
```

### 6.4.2 Function Documentation

#### 6.4.2.1 create\_account()

```
int create_account ( )
```

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

##### Returns

0 if the broadcast succeeds, otherwise 1

Definition at line 19 of file wallet.c.

Here is the call graph for this function:

#### 6.4.2.2 get\_my\_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

##### Returns

`Wallet`

Definition at line 7 of file wallet.c.

Here is the caller graph for this function:

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

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

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

### Functions

- char \* [sha384\\_data](#) (void \*data, size\_t len\_data)  
*Apply the SHA384 algorithm on a 'data' of size 'len\_data'.*
- char \* [hash\\_block\\_transactions](#) ([Block](#) \*block)  
*Apply the SHA384 to all block transactions.*

### 6.5.1 Function Documentation

#### 6.5.1.1 hash\_block\_transactions()

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

Apply the SHA384 to all block transactions.

##### Parameters

<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:

#### 6.5.1.2 sha384\_data()

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

Apply the SHA384 algorithm on a 'data' of size 'len\_data'.

## Parameters

<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:

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

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

### Macros

- #define [RSA\\_KEY\\_SIZE](#) 366
- #define [RSA\\_FILE\\_TOTAL\\_SIZE](#) 426
- #define [RSA\\_BEGIN\\_SIZE](#) 31
- #define [RSA\\_END\\_SIZE](#) 29

### Functions

- void [get\\_keys](#) (char \*password)  
*Get the keys object.*

#### 6.6.1 Macro Definition Documentation

##### 6.6.1.1 RSA\_BEGIN\_SIZE

```
#define RSA_BEGIN_SIZE 31
```

Definition at line 6 of file rsa.h.

### 6.6.1.2 RSA\_END\_SIZE

```
#define RSA_END_SIZE 29
```

Definition at line 7 of file rsa.h.

### 6.6.1.3 RSA\_FILE\_TOTAL\_SIZE

```
#define RSA_FILE_TOTAL_SIZE 426
```

Definition at line 5 of file rsa.h.

### 6.6.1.4 RSA\_KEY\_SIZE

```
#define RSA_KEY_SIZE 366
```

Definition at line 4 of file rsa.h.

## 6.6.2 Function Documentation

### 6.6.2.1 get\_keys()

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

Get the keys object.

Here is the caller graph for this function:

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

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

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

## Functions

- char \* [sign\\_message](#) (char \*data, size\_t len\_data, void \*buffer)  
*buffer <- encrypt(SHA284(msg,len\_data),wallet\_priv\_key)*
- char \* [sign\\_message\\_with\\_key](#) (char \*data, size\_t len\_data, RSA \*key, void \*buffer)  
*encrypt(SHA284(msg,len\_data),key) buffer <- encrypt(SHA284(msg,len\_data),key)*
- int [verify\\_signature](#) (void \*data, size\_t data\_len, char \*signature, RSA \*pub\_key)  
*Verifies if SHA384(data) == decrypt(signature,pub\_key)*
- int [verify\\_block\\_signature](#) (Block block)  
*Verifies if a block signature is valid.*
- int [verify\\_transaction\\_signature](#) (Transaction transaction)  
*Verifies if a transaction signature is valid.*
- void [get\\_transaction\\_data](#) (Transaction \*trans, char \*\*buff, size\_t \*size)  
*Convert transactions to char \* buffer.*
- void [write\\_blockdata](#) (BlockData blockdata, int fd)  
*Writes blockdata in a file.*
- void [write\\_block](#) (Block block, int fd)  
*Writes a block in a file.*
- void [sign\\_block](#) (Block \*block)  
*Signs a block.*
- void [sign\\_block\\_with\\_key](#) (Block \*block, RSA \*key)
- void [sign\\_transaction](#) (Transaction \*transaction)
- void [sign\\_transaction\\_with\\_key](#) (Transaction \*transaction, RSA \*key)  
*Sign a transaction.*
- void [sign\\_block\\_transactions](#) (Block \*block)  
*Signs transactions of a block.*

## 6.7.1 Function Documentation

### 6.7.1.1 [get\\_transaction\\_data\(\)](#)

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

Convert transactions to char \* buffer.

#### Parameters

<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 48 of file transaction.c.

#### 6.7.1.2 sign\_block()

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

Signs a block.

##### Parameters

<i>block</i>	The block to sign
--------------	-------------------

Definition at line 102 of file signature.c.

Here is the call graph for this function:

#### 6.7.1.3 sign\_block\_transactions()

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

Signs transactions of a block.

##### Parameters

<i>block</i>	The block to sign
--------------	-------------------

Definition at line 132 of file signature.c.

Here is the call graph for this function:

#### 6.7.1.4 sign\_block\_with\_key()

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

Definition at line 109 of file signature.c.

Here is the call graph for this function:

**6.7.1.5 sign\_message()**

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

buffer <- encrypt(SHA284(msg,len\_data),wallet\_priv\_key)

If buffer == NULL, return a new allocated buffer

**Parameters**

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>buffer</i>	The buffer to put signature into

**Returns**

char\*

Definition at line 10 of file signature.c.

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

**6.7.1.6 sign\_message\_with\_key()**

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

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

If buffer == NULL, return a new allocated buffer

**Parameters**

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>key</i>	The key to use for the signature
<i>buffer</i>	The buffer to put signature into

**Returns**

char\*

Definition at line 34 of file signature.c.

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

**6.7.1.7 sign\_transaction()**

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

Definition at line 116 of file signature.c.

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



**6.7.1.8 sign\_transaction\_with\_key()**

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

Sign a transaction.

**Parameters**

<i>transaction</i>	The transaction to sign
--------------------	-------------------------

Definition at line 124 of file signature.c.

Here is the call graph for this function:

**6.7.1.9 verify\_block\_signature()**

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

Verifies if a block signature is valid.

**Parameters**

<i>block</i>	The block to verify
--------------	---------------------

**Returns**

1 if valid, 0 otherwise

Definition at line 77 of file signature.c.

Here is the call graph for this function:

**6.7.1.10 verify\_signature()**

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

Verifies if SHA384(data) == decrypt(signature, pub\_key)

**Parameters**

<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>pub_key</i>	The RSA public key used for the decryption

**Returns**

int

Definition at line 57 of file signature.c.

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

**6.7.1.11 verify\_transaction\_signature()**

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

Verifies if a transaction signature is valid.

**Parameters**

<i>transaction</i>	The transaction to verify
--------------------	---------------------------

**Returns**

1 if valid, 0 otherwise

Definition at line 89 of file signature.c.

Here is the call graph for this function:

**6.7.1.12 write\_block()**

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

Writes a block in a file.

**Parameters**

<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 309 of file block.c.

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

**6.7.1.13 write\_blockdata()**

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

Writes blockdata in a file.

**Parameters**

<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 277 of file block.c.

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

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

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

**Functions**

- char \* [last\\_file\\_in\\_folder](#) (char folder\_path[])  
*Return the last file (reverse alphabetical order) of a folder path.*

### 6.8.1 Function Documentation

#### 6.8.1.1 last\_file\_in\_folder()

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

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

**Parameters**

<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:

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

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

### Macros

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

### 6.9.1 Macro Definition Documentation

#### 6.9.1.1 MAX

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

Definition at line 2 of file math.h.

#### 6.9.1.2 MIN

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

Definition at line 1 of file math.h.

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

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

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

## Functions

- int [safe\\_write](#) (int fd, const void \*buf, ssize\_t count)  
*Writes safely to a file descriptor.*
- int [safe\\_send](#) (int fd, const void \*buf, ssize\_t count)  
*Send safely to a file descriptor.*
- ssize\_t [safe\\_read](#) (int fd, const void \*\*buf, size\_t \*bufsize)  
*Reads safely in a file descriptor until '\r\n\r\n'.*
- ssize\_t [safe\\_fread](#) (void \*buffer, const size\_t size, const size\_t n, FILE \*file)  
*Calls 'fread' but safely !*

### 6.10.1 Function Documentation

#### 6.10.1.1 [safe\\_fread\(\)](#)

```

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

```

Calls 'fread' but safely !

##### Parameters

<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 58 of file safe.c.

Here is the caller graph for this function:

#### 6.10.1.2 [safe\\_read\(\)](#)

```

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

```

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

**Parameters**

<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 31 of file safe.c.

Here is the caller graph for this function:

**6.10.1.3 safe\_send()**

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

Send safely to a file descriptor.

**Parameters**

<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 17 of file safe.c.

Here is the caller graph for this function:

**6.10.1.4 safe\_write()**

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

Writes safely to a file descriptor.

**Parameters**

<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 3 of file safe.c.

Here is the caller graph for this function:

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

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

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

**Functions**

- [Node \\* get\\_my\\_node](#) (char who)  
*Get the my node object.*
- int [set\\_neighbour](#) (char who, char \*hostname, int family)  
*Sets a neighbour in the client.neighbours section.*
- void [remove\\_neighbour](#) (char who, int index)  
*Remove a neighbour in the client.neighbours section.*
- int [number\\_neighbours](#) (char who)  
*return the nb of neighbour in the client.neighbours section*
- void [print\\_neighbours](#) (char who, char mask)  
*Print neighbours list.*
- void [save\\_neighbours](#) (char who)  
*Save neighbours list in .neighbours/neighbours.*
- void [load\\_neighbours](#) (char who)  
*Load neighbours list from .neighbours/neighbours.*
- [client\\_connection \\* listen\\_to](#) ([infos\\_st](#) \*infos, [Neighbour](#) neighbour, char \*connection\_type)  
*Tries to connect to the peer-to-peer network via a node in the [Node](#) structure.*
- int [find\\_empty\\_connection](#) (int max)
- void \* [client\\_thread](#) (void \*args)



## 6.11.1 Function Documentation

### 6.11.1.1 `client_thread()`

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

Definition at line 247 of file client.c.

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

### 6.11.1.2 `find_empty_connection()`

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

Definition at line 237 of file client.c.

Here is the caller graph for this function:

### 6.11.1.3 `get_my_node()`

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

Get the my node object.

#### Returns

Node\*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

### 6.11.1.4 `listen_to()`

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

Tries to connect to the peer-to-peer network via a node in the [Node](#) structure.

#### Parameters

<i>neighbour</i>	The neighbour to connect with
------------------	-------------------------------

**Returns**

socket FD or -1 if an error occurs

Definition at line 161 of file client.c.

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

**6.11.1.5 load\_neighbours()**

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

Load neighbours list from .neighbours/neighbours.

Definition at line 113 of file client.c.

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

**6.11.1.6 number\_neighbours()**

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

return the nb of neighbour in the client.neighbours section

Definition at line 149 of file client.c.

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

**6.11.1.7 print\_neighbours()**

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

Print neighbours list.

Definition at line 58 of file client.c.

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

**6.11.1.8 remove\_neighbour()**

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

Remove a neighbour in the client.neighbours section.

Definition at line 47 of file client.c.

Here is the call graph for this function:

### 6.11.1.9 save\_neighbours()

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

Save neighbours list in .neighbours/neighbours.

Definition at line 74 of file client.c.

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

### 6.11.1.10 set\_neighbour()

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

Sets a neighbour in the client.neighbours section.

#### Returns

0 if success, -1 otherwise if full

Definition at line 19 of file client.c.

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

## 6.12 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get\_data.h File Reference

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

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

### Functions

- `size_t read_header` (int sockfd, `infos_st` \*infos)  
*Waits a header in 'sockfd', reads it and processes it.*
- `int fetch_client_list` (char who, int fd)  
*Merges my neighbours list with the one sent by 'neighbour\_id'.*
- `int read_get_blocks` (int fd, `infos_st` \*infos)
- `size_t read_actual_height` (int fd)
- `int read_send_block` (int fd)
- `int read_vote` (int fd)
- `int read_pending_transaction_list` (int fd)
- `int read_epoch_block` (int fd)

## 6.12.1 Function Documentation

### 6.12.1.1 `fetch_client_list()`

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

Merges my neighbours list with the one sent by 'neighbour\_id'.

#### Parameters

<i>sockfd</i>	The sockfd to read
---------------	--------------------

#### Returns

0 if success, -1 otherwise

Definition at line 95 of file `get_data.c`.

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

### 6.12.1.2 `read_actual_height()`

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

Definition at line 172 of file `get_data.c`.

Here is the caller graph for this function:

### 6.12.1.3 `read_epoch_block()`

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

Definition at line 222 of file `get_data.c`.

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

### 6.12.1.4 `read_get_blocks()`

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

Definition at line 144 of file `get_data.c`.

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

#### 6.12.1.5 read\_header()

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

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

**Parameters**

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

**Returns**

0 if success, -1 otherwise

Definition at line 125 of file get\_data.c.

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

**6.12.1.6 read\_pending\_transaction\_list()**

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

**6.12.1.7 read\_send\_block()**

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

Definition at line 178 of file get\_data.c.

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

**6.12.1.8 read\_vote()**

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

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

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

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

**Data Structures**

- struct [Neighbour](#)
- struct [Node](#)
- struct [client\\_connection](#)
- struct [infos\\_st](#)
- struct [th\\_arg](#)

## Macros

- `#define SIZE_OF_HOSTNAME 39`
- `#define NB_HARD_CODED_ADDR 2`
- `#define MAX_CONNECTION 5`
- `#define STATIC_PORT "4242"`
- `#define P_VERSION 0`
- `#define IM_SERVER 0`
- `#define IM_CLIENT 1`
- `#define MAX_NEIGHBOURS 64`
- `#define NODESERVER 0`
- `#define DOORSERVER 1`
- `#define MAX_SERVER 20`
- `#define MAX_VALIDATORS_PER_BLOCK 512`
- `#define SOL_TCP 6`
- `#define TCP_USER_TIMEOUT 18`
- `#define HD_GET_CLIENT_LIST "GET CLIENT LIST\r\n\r\n"`
- `#define HD_SEND_CLIENT_LIST "SEND CLIENT LIST\r\n\r\n"`
- `#define HD_CONNECTION_TO_NETWORK "CONNECTION TO NETWORK\r\n\r\n"`
- `#define HD_CONNECTION_TO_NODE "CONNECTION TO NODE\r\n\r\n"`
- `#define HD_GET_BLOCKS "GET BLOCKS\r\n\r\n"`
- `#define HD_ACTUAL_HEIGHT "ACTUAL HEIGHT\r\n\r\n"`
- `#define HD_SEND_BLOCK "SEND BLOCK\r\n\r\n"`
- `#define HD_GET_PENDING_TRANSACTION_LIST "GET PENDING TRANSACTION LIST\r\n\r\n"`
- `#define HD_REJECT_DEMAND "REJECT DEMAND\r\n\r\n"`
- `#define HD_SEND_PENDING_TRANSACTION "SEND PENDING TRANSACTION\r\n\r\n"`
- `#define HD_SEND_EPOCH_BLOCK "SEND EPOCH BLOCK\r\n\r\n"`
- `#define HD_SEND_VOTE "SEND VOTE\r\n\r\n"`
- `#define DD_GET_HEIGHT 1`
- `#define DD_GET_BLOCKS 2`
- `#define SERVERMSG printf("\033[0;31m[S]:\033[0m ");`
- `#define CLIENTMSG printf("\033[0;34m[C]:\033[0m ");`
- `#define MANAGERMSG printf("\033[0;32m[M]:\033[0m ");`
- `#define WARNINGMSG(x) printf("\033[0;35m[W]: %s\033[0m\n", x);`

## Typedefs

- `typedef struct Neighbour Neighbour`
- `typedef struct Node Node`
- `typedef struct client_connection client_connection`
- `typedef struct infos_st infos_st`
- `typedef struct th_arg th_arg`

## Functions

- `struct __attribute__((__packed__)) get_blocks_t`

## Variables

- `const Neighbour HARD_CODED_ADDR []`
- `get_blocks_t`

## 6.13.1 Macro Definition Documentation

### 6.13.1.1 CLIENTMSG

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

Definition at line 90 of file network.h.

### 6.13.1.2 DD\_GET\_BLOCKS

```
#define DD_GET_BLOCKS 2
```

Definition at line 85 of file network.h.

### 6.13.1.3 DD\_GET\_HEIGHT

```
#define DD_GET_HEIGHT 1
```

Definition at line 84 of file network.h.

### 6.13.1.4 DOORSERVER

```
#define DOORSERVER 1
```

Definition at line 23 of file network.h.

### 6.13.1.5 HD\_ACTUAL\_HEIGHT

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

Definition at line 75 of file network.h.



#### 6.13.1.6 HD\_CONNECTION\_TO\_NETWORK

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

Definition at line 72 of file network.h.

#### 6.13.1.7 HD\_CONNECTION\_TO\_NODE

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

Definition at line 73 of file network.h.

#### 6.13.1.8 HD\_GET\_BLOCKS

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

Definition at line 74 of file network.h.

#### 6.13.1.9 HD\_GET\_CLIENT\_LIST

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

Definition at line 70 of file network.h.

#### 6.13.1.10 HD\_GET\_PENDING\_TRANSACTION\_LIST

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

Definition at line 77 of file network.h.

#### 6.13.1.11 HD\_REJECT\_DEMAND

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

Definition at line 78 of file network.h.

#### 6.13.1.12 HD\_SEND\_BLOCK

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

Definition at line 76 of file network.h.

#### 6.13.1.13 HD\_SEND\_CLIENT\_LIST

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

Definition at line 71 of file network.h.

#### 6.13.1.14 HD\_SEND\_EPOCH\_BLOCK

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

Definition at line 80 of file network.h.

#### 6.13.1.15 HD\_SEND\_PENDING\_TRANSACTION

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

Definition at line 79 of file network.h.

#### 6.13.1.16 HD\_SEND\_VOTE

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

Definition at line 81 of file network.h.

#### 6.13.1.17 IM\_CLIENT

```
#define IM_CLIENT 1
```

Definition at line 18 of file network.h.

#### 6.13.1.18 IM\_SERVER

```
#define IM_SERVER 0
```

Definition at line 17 of file network.h.

#### 6.13.1.19 MANAGERMSG

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

Definition at line 91 of file network.h.

#### 6.13.1.20 MAX\_CONNECTION

```
#define MAX_CONNECTION 5
```

Definition at line 11 of file network.h.

#### 6.13.1.21 MAX\_NEIGHBOURS

```
#define MAX_NEIGHBOURS 64
```

Definition at line 20 of file network.h.

#### 6.13.1.22 MAX\_SERVER

```
#define MAX_SERVER 20
```

Definition at line 25 of file network.h.

#### 6.13.1.23 MAX\_VALIDATORS\_PER\_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 27 of file network.h.

**6.13.1.24 NB\_HARD\_CODED\_ADDR**

```
#define NB_HARD_CODED_ADDR 2
```

Definition at line 10 of file network.h.

**6.13.1.25 NODESERVER**

```
#define NODESERVER 0
```

Definition at line 22 of file network.h.

**6.13.1.26 P\_VERSION**

```
#define P_VERSION 0
```

Definition at line 15 of file network.h.

**6.13.1.27 SERVERMSG**

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

Definition at line 89 of file network.h.

**6.13.1.28 SIZE\_OF\_HOSTNAME**

```
#define SIZE_OF_HOSTNAME 39
```

Definition at line 9 of file network.h.

**6.13.1.29 SOL\_TCP**

```
#define SOL_TCP 6
```

Definition at line 29 of file network.h.

#### 6.13.1.30 STATIC\_PORT

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

Definition at line 13 of file network.h.

#### 6.13.1.31 TCP\_USER\_TIMEOUT

```
#define TCP_USER_TIMEOUT 18
```

Definition at line 30 of file network.h.

#### 6.13.1.32 WARNINGMSG

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

Definition at line 92 of file network.h.

### 6.13.2 Typedef Documentation

#### 6.13.2.1 client\_connection

```
typedef struct client\_connection client\_connection
```

#### 6.13.2.2 infos\_st

```
typedef struct infos\_st infos\_st
```

#### 6.13.2.3 Neighbour

```
typedef struct Neighbour Neighbour
```

#### 6.13.2.4 Node

```
typedef struct Node Node
```

#### 6.13.2.5 th\_arg

```
typedef struct th_arg th_arg
```

### 6.13.3 Function Documentation

#### 6.13.3.1 \_\_attribute\_\_()

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

Definition at line 94 of file network.h.

Here is the caller graph for this function:

### 6.13.4 Variable Documentation

#### 6.13.4.1 get\_blocks\_t

```
get_blocks_t
```

Definition at line 99 of file network.h.

#### 6.13.4.2 HARD\_CODED\_ADDR

```
const Neighbour HARD_CODED_ADDR[ ]
```

Definition at line 5 of file network.c.

## 6.14 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send\_data.h File Reference

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

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

### Functions

- int [send\\_client\\_list](#) (char who, int sockfd, char \*sockip)  
*Sends my client list to a node via 'sockfd'.*
- void [send\\_get\\_blocks](#) (client\_connection \*cc)  
*Sends get blocks.*
- void [send\\_actual\\_height](#) (int fd, infos\_st \*infos)
- void [send\\_reject\\_demand](#) (int fd)
- void [send\\_send\\_block](#) (int fd, size\_t height)
- void [send\\_pending\\_transaction\\_list](#) (int sockfd)

### 6.14.1 Function Documentation

#### 6.14.1.1 send\_actual\_height()

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

Definition at line 58 of file send\_data.c.

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

#### 6.14.1.2 send\_client\_list()

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

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

#### Parameters

<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:

**6.14.1.3 send\_get\_blocks()**

```
void send_get_blocks (
    client_connection * cc )
```

Sends get blocks.

Definition at line 52 of file send\_data.c.

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

**6.14.1.4 send\_pending\_transaction\_list()**

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

Here is the caller graph for this function:

**6.14.1.5 send\_reject\_demand()**

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

Definition at line 65 of file send\_data.c.

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

**6.14.1.6 send\_send\_block()**

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

Definition at line 71 of file send\_data.c.

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



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

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

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

### Functions

- void \* [init\\_server](#) (void \*args)

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

### 6.15.1 Function Documentation

#### 6.15.1.1 init\_server()

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

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

#### Parameters

<i>type</i>	Type of the server
-------------	--------------------

#### Returns

0 if success, -1 otherwise

Definition at line 77 of file server.c.

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

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

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

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

### Data Structures

- struct [blockinfo](#)

### Functions

- void \* [setup](#) (void \*args)  
*Setups the gtk widgets for the GUI.*
- gboolean [on\\_main\\_window\\_delete](#) (GtkWidget \*widget, [\\_\\_attribute\\_\\_\(\(unused\)\)](#) gpointer data)  
*Destroys the window when it is closed.*
- void [on\\_main\\_window\\_destroy](#) ([\\_\\_attribute\\_\\_\(\(unused\)\)](#) GtkWidget \*widget, [\\_\\_attribute\\_\\_\(\(unused\)\)](#) gpointer data)  
*Quits GTK when the program ends.*
- gboolean [on\\_transaction\\_button\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Will be used when the transaction function is ready.*
- gboolean [on\\_pkey\\_button\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Hides the private key of the user, or shows it if it was already hidden.*
- gboolean [on\\_invest\\_button1\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Opens the invest window.*
- gboolean [on\\_invest\\_button2\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Resets the entry in the invest window and closes it, will later be used for the invest function.*
- gboolean [on\\_recover\\_button1\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Opens the recover window.*
- gboolean [on\\_recover\\_button2\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Resets the entry in the recover window and closes it, will later be used for the recover function.*
- gboolean [on\\_add\\_contact\\_button1\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Opens the contact window.*
- gboolean [add\\_contact](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)  
*Adds a contact to the treeview if the entrys weren't empty, and closes the contact window.*
- void [change\\_label\\_text](#) (GtkLabel \*label, char \*text)
- gboolean [on\\_create\\_key\\_but1\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)
- gboolean [on\\_create\\_key\\_but2\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)
- gboolean [on\\_connect\\_but\\_press](#) (GtkWidget \*widget, GdkEventKey \*event, gpointer user\_data)
- void [add\\_contacts\\_from\\_file](#) (char \*name, char \*public\_key)
- void [load\\_contacts\\_from\\_file](#) ()
- void [add\\_contact\\_to\\_combobox](#) (char \*name)

- void [update\\_labels](#) ()
- void [add\\_transaction\\_with\\_pkey](#) (double amount, char \*public\_key, char \*date)
- void [add\\_transaction\\_with\\_contact](#) (double amount, char \*public\_key, char \*date)
- void [add\\_transaction\\_from\\_file](#) (double amount, char \*public\_key, char \*date)
- void [load\\_transaction\\_from\\_file](#) ()
- char \* [get\\_public\\_key\\_from\\_contacts](#) (const char \*name)
- void [add\\_new\\_blockinfo](#) (size\_t height, size\_t transaction)
- void [update\\_sync](#) (size\_t actual, size\_t final)

## Variables

- GtkWidget \* [synchro\\_label](#)
- GtkWidget \* [block\\_amount\\_label](#)
- GtkWidget \* [connections\\_label](#)
- GtkWidget \* [mempool\\_label](#)
- struct [blockinfo](#) [blocksinfo](#) [3]

## 6.16.1 Function Documentation

### 6.16.1.1 [add\\_contact\(\)](#)

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

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

#### Parameters

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

#### Returns

gboolean Error code

### 6.16.1.2 [add\\_contact\\_to\\_combobox\(\)](#)

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

Definition at line 466 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.3 add\_contacts\_from\_file()

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

Definition at line 474 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.4 add\_new\_blockinfo()

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

Definition at line 199 of file ui.c.

#### 6.16.1.5 add\_transaction\_from\_file()

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

Definition at line 324 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.6 add\_transaction\_with\_contact()

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

Definition at line 304 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.7 add\_transaction\_with\_pkey()

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

Definition at line 283 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.8 change\_label\_text()

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

Definition at line 194 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.9 get\_public\_key\_from\_contacts()

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

Definition at line 511 of file ui.c.

Here is the caller graph for this function:

#### 6.16.1.10 load\_contacts\_from\_file()

```
void load_contacts_from_file ( )
```

Definition at line 483 of file ui.c.

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

#### 6.16.1.11 load\_transaction\_from\_file()

```
void load_transaction_from_file ( )
```

#### 6.16.1.12 on\_add\_contact\_button1\_press()

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

Opens the contact window.

##### Parameters

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

**Returns**

gboolean Error code

**6.16.1.13 on\_connect\_but\_press()**

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

**6.16.1.14 on\_create\_key\_but1\_press()**

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

**6.16.1.15 on\_create\_key\_but2\_press()**

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

**6.16.1.16 on\_invest\_button1\_press()**

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

Opens the invest window.

**Parameters**

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

**Returns**

gboolean

**6.16.1.17 on\_invest\_button2\_press()**

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

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

**Parameters**

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

**Returns**

gboolean Error Code

**6.16.1.18 on\_main\_window\_delete()**

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

Destroys the window when it is closed.

**Parameters**

<i>widget</i>	The main window of the GUI
---------------	----------------------------

**Returns**

gboolean Error code

Definition at line 233 of file ui.c.

### 6.16.1.19 on\_main\_window\_destroy()

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

Quits GTK when the program ends.

### 6.16.1.20 on\_pkey\_button\_press()

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

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

#### Parameters

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

#### Returns

gboolean Error code

### 6.16.1.21 on\_recover\_button1\_press()

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

Opens the recover window.

#### Parameters

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

#### Returns

gboolean Error code



#### 6.16.1.22 on\_recover\_button2\_press()

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

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

##### Parameters

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

##### Returns

gboolean Error code

#### 6.16.1.23 on\_transaction\_button\_press()

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

Will be used when the transaction function is ready.

##### Parameters

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

##### Returns

gboolean Error code

#### 6.16.1.24 setup()

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

Setups the gtk widgets for the GUI.

**Returns**

int Returns 1 if there is an error, 0 otherwise

Definition at line 63 of file ui.c.

Here is the caller graph for this function:

**6.16.1.25 update\_labels()**

```
void update_labels ( )
```

Definition at line 658 of file ui.c.

Here is the call graph for this function:

**6.16.1.26 update\_sync()**

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

Definition at line 214 of file ui.c.

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

**6.16.2 Variable Documentation****6.16.2.1 block\_amount\_label**

```
GtkLabel* block_amount_label
```

Definition at line 14 of file ui.h.

**6.16.2.2 blocksinfo**

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

Definition at line 23 of file ui.h.

### 6.16.2.3 connections\_label

```
GtkLabel* connections_label
```

Definition at line 15 of file ui.h.

### 6.16.2.4 mempool\_label

```
GtkLabel* mempool_label
```

Definition at line 16 of file ui.h.

### 6.16.2.5 synchro\_label

```
GtkLabel* synchro_label
```

Definition at line 13 of file ui.h.

## 6.17 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/epoch\_man.h File Reference

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

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

### Functions

- [Block \\* create\\_epoch\\_block \(\)](#)  
*Create a block object with the previous block hash & votes.*
- [RSA \\* get\\_epoch\\_man\\_pkey \(BlockData \\*block\\_data\)](#)  
*Give the pkey of the creator of a block.*
- void [give\\_punishments\\_and\\_rewards \(Block \\*prev\\_block, Block \\*current\\_block\)](#)  
*Add punishment and reward transactions to validators of the 'prev\_block' into 'current\_block'.*
- int [flush\\_pending\\_transactions \(Block \\*block\)](#)  
*Delete block transactions in pdt if the block is valid.*

### 6.17.1 Function Documentation

### 6.17.1.1 create\_epoch\_block()

```
Block* create_epoch_block ( )
```

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

#### See also

The function create a block based on the local last block

#### Returns

Block\*

### 6.17.1.2 flush\_pending\_transactions()

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

Delete block transactions in pdt if the block is valid.

#### Parameters

<i>block</i>	
--------------	--

#### Returns

1 if the flush proceed, 0 if not

### 6.17.1.3 get\_epoch\_man\_pkey()

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

Give the pkey of the creator of a block.

#### Parameters

<i>block_data</i>	The created block data
-------------------	------------------------

#### Returns

RSA\*

Definition at line 3 of file epoch\_man.c.

Here is the caller graph for this function:

#### 6.17.1.4 give\_punishments\_and\_rewards()

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

Add punishment and reward transactions to validators of the 'prev\_block' into 'current\_block'.

#### See also

Number of added transactions = number of validators in 'prev\_block'

#### Parameters

<i>prev_block</i>	
<i>current_block</i>	

## 6.18 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/headers/validation/validation\_engine.h File Reference

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

Include dependency graph for validation\_engine.h:

### Functions

- int [send\\_verdict](#) (Block \*block, char verdict)  
*Broadcast a verdict about a block validity to the network.*
- Transaction \*\* [validate\\_transactions](#) (Transaction \*transaction\_to\_validate, size\_t nb\_transactions, size\_t \*nb\_returned\_transactions)  
*Validate some transactions.*

### 6.18.1 Function Documentation

### 6.18.1.1 send\_verdict()

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

Broadcast a verdict about a block validity to the network.

#### Parameters

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

#### Returns

0 if the broadcast succeed, -1 if not

### 6.18.1.2 validate\_transactions()

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

Validate some transactions.

#### See also

The verification must take into account :

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

#### Parameters

<i>transaction_to_validate</i>	The transactions to validate
<i>nb_transactions</i>	The number of transactions to validate
<i>nb_returned_transactions</i>	The number of returned (valid) transactions

## Returns

Transaction\*\*, the valid transactions

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

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

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

## Macros

- #define [MAX\\_VALIDATORS\\_PER\\_BLOCK](#) 512

## Functions

- RSA \*\* [get\\_comittee](#) (size\_t block\_height, size\_t \*nb\_validators)  
*Get the a comittee RSA public keys on a specific epoch.*
- RSA \*\* [get\\_next\\_comittee](#) (size\_t \*nb\_validators)  
*Get the a comittee RSA public keys on a specific epoch.*
- ssize\_t [get\\_validators\\_states\\_total\\_stake](#) ()  
*Get the total stake of the network (parse 'validators.state')*
- ssize\_t [get\\_validators\\_states\\_nb\\_validators](#) ()  
*Get the number of validators of the network (parse 'validators.state')*
- ssize\_t [get\\_validators\\_states\\_block\\_height\\_validity](#) ()  
*Get the validators states block height validity (parse 'validators.state')*
- ssize\_t [get\\_validator\\_stake](#) (size\_t validator\_id)  
*Get a validator total stake (parse 'validators.state')*
- ssize\_t [get\\_validator\\_power](#) (size\_t validator\_id)  
*Get a validator power (parse 'validators.state')*
- RSA \* [get\\_validator\\_pkey](#) (size\_t validator\_id)  
*Get the validator pkey as RSA\* (parse 'validators.state')*
- ssize\_t [get\\_validator\\_id](#) (char pkey[])  
*Get the validator id in 'validators.state'.*
- int [push\\_stake](#) (size\_t amount)  
*Push an amount on the stake.*
- int [pop\\_stake](#) (size\_t amount)  
*Pops an amount on the stake.*

## 6.19.1 Macro Definition Documentation

### 6.19.1.1 MAX\_VALIDATORS\_PER\_BLOCK

```
#define MAX_VALIDATORS_PER_BLOCK 512
```

Definition at line 13 of file validators.h.

## 6.19.2 Function Documentation

### 6.19.2.1 get\_comittee()

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

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

#### Parameters

<i>block_height</i>	The height of the block you want a comitte from
<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 32 of file validators.c.

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

### 6.19.2.2 get\_next\_comittee()

```
RSA** get_next_comittee (
    size_t * nb_validators )
```

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



**Parameters**

<i>block_height</i>	The height of the block you want a comitte from
<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 124 of file validators.c.

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

**6.19.2.3 get\_validator\_id()**

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

Get the validator id in 'validators.state'.

**Parameters**

<i>pkey</i>	The string public key (without ---BEGIN RSA KEY--- & ---END RSA KEY---)
-------------	---

**Returns**

ssize\_t, the validator index

Definition at line 209 of file validators.c.

Here is the call graph for this function:

**6.19.2.4 get\_validator\_pkey()**

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

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

**Parameters**

<i>validator_id</i>	The id of the validator in 'validators.state'
---------------------	---

**Returns**

RSA\*

Definition at line 188 of file validators.c.

Here is the call graph for this function:

**6.19.2.5 get\_validator\_power()**

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

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

**Parameters**

<i>validator</i> ↔ _id	The id of the validator in 'validators.state'
---------------------------	---

**Returns**

ssize\_t

Definition at line 176 of file validators.c.

Here is the call graph for this function:

**6.19.2.6 get\_validator\_stake()**

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

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

**Parameters**

<i>validator</i> ↔ _id	The id of the validator in 'validators.state'
---------------------------	---

**Returns**

ssize\_t

Definition at line 164 of file validators.c.

Here is the call graph for this function:

#### 6.19.2.7 `get_validators_states_block_height_validity()`

```
ssize_t get_validators_states_block_height_validity ( )
```

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

##### Returns

`ssize_t`

Definition at line 152 of file validators.c.

Here is the call graph for this function:

#### 6.19.2.8 `get_validators_states_nb_validators()`

```
ssize_t get_validators_states_nb_validators ( )
```

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

##### Returns

`ssize_t`

Definition at line 141 of file validators.c.

Here is the call graph for this function:

#### 6.19.2.9 `get_validators_states_total_stake()`

```
ssize_t get_validators_states_total_stake ( )
```

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

##### Returns

`ssize_t`

Definition at line 129 of file validators.c.

Here is the call graph for this function:

#### 6.19.2.10 `pop_stake()`

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

Pops an amount on the stake.

This will broadcast a stake pop on the network.

##### See also

The stake account public key is '1'

**Parameters**

<i>amount</i>	The amount to pop
---------------	-------------------

**Returns**

0 if the broadcast succeeds, else returns -1

**6.19.2.11 push\_stake()**

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

Push an amount on the stake.

This will broadcast a stake push on the network.

**See also**

The stake account public key is '1'

**Parameters**

<i>amount</i>	The amount to push
---------------	--------------------

**Returns**

0 if the broadcast succeeds, else returns -1

## 6.20 [/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/P2P\\_Protocol.md](#) File Reference

## 6.21 [/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md](#) File Reference

## 6.22 [/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c](#) File Reference

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

```
#include <string.h>
#include "network/network.h"
#include "network/client.h"
#include "network/server.h"
#include "network/send_data.h"
#include "network/get_data.h"
#include "misc/safe.h"
#include "blockchain/blockchain_header.h"
#include "ui/ui.h"
Include dependency graph for client.c:
```

## Functions

- void [join\\_network\\_door](#) ([infos\\_st](#) \*infos)
- void [connection\\_to\\_others](#) ([infos\\_st](#) \*infos)
- [size\\_t](#) [update\\_blockchain\\_height](#) ([infos\\_st](#) \*infos)
- void [update\\_blockchain](#) ([infos\\_st](#) \*infos, [size\\_t](#) index\_client)
- int [main](#) ()

## Variables

- [client\\_connection](#) \* [client\\_connections](#)

### 6.22.1 Function Documentation

#### 6.22.1.1 [connection\\_to\\_others\(\)](#)

```
void connection_to_others (
    infos\_st * infos )
```

Definition at line 39 of file client.c.

Here is the call graph for this function:

#### 6.22.1.2 [join\\_network\\_door\(\)](#)

```
void join_network_door (
    infos\_st * infos )
```

Definition at line 19 of file client.c.

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

### 6.22.1.3 main()

```
int main ( )
```

Definition at line 121 of file client.c.

Here is the call graph for this function:

### 6.22.1.4 update\_blockchain()

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

Definition at line 93 of file client.c.

### 6.22.1.5 update\_blockchain\_height()

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

Definition at line 57 of file client.c.

Here is the call graph for this function:

## 6.22.2 Variable Documentation

### 6.22.2.1 client\_connections

```
client_connection* client_connections
```

Definition at line 4 of file client.c.

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

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

## Functions

- `Node * get_my_node` (char who)  
*Get the my node object.*
- int `set_neighbour` (char who, char \*hostname, int family)  
*Sets a neighbour in the client.neighbours section.*
- void `remove_neighbour` (char who, int index)  
*Remove a neighbour in the client.neighbours section.*
- void `print_neighbours` (char who, char mask)  
*Print neighbours list.*
- void `save_neighbours` (char who)  
*Save neighbours list in .neighbours/neighbours.*
- void `load_neighbours` (char who)  
*Load neighbours list from .neighbours/neighbours.*
- int `number_neighbours` (char who)  
*return the nb of neighbour in the client.neighbours section*
- `client_connection * listen_to` (`infos_st *infos`, `Neighbour` neighbour, char \*connection\_type)  
*Tries to connect to the peer-to-peer network via a node in the `Node` structure.*
- int `find_empty_connection` (int max)
- void \* `client_thread` (void \*args)

## Variables

- `client_connection * client_connections` = NULL

### 6.23.1 Function Documentation

#### 6.23.1.1 client\_thread()

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

Definition at line 247 of file client.c.

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

#### 6.23.1.2 find\_empty\_connection()

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

Definition at line 237 of file client.c.

Here is the caller graph for this function:

### 6.23.1.3 get\_my\_node()

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

Get the my node object.

#### Returns

Node\*

Definition at line 6 of file client.c.

Here is the caller graph for this function:

### 6.23.1.4 listen\_to()

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

Tries to connect to the peer-to-peer network via a node in the [Node](#) structure.

#### Parameters

<i>neighbour</i>	The neighbour to connect with
------------------	-------------------------------

#### Returns

socket FD or -1 if an error occurs

Definition at line 161 of file client.c.

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

### 6.23.1.5 load\_neighbours()

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

Load neighbours list from .neighbours/neighbours.

Definition at line 113 of file client.c.

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



#### 6.23.1.6 number\_neighbours()

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

return the nb of neighbour in the client.neighbours section

Definition at line 149 of file client.c.

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

#### 6.23.1.7 print\_neighbours()

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

Print neighbours list.

Definition at line 58 of file client.c.

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

#### 6.23.1.8 remove\_neighbour()

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

Remove a neighbour in the client.neighbours section.

Definition at line 47 of file client.c.

Here is the call graph for this function:

#### 6.23.1.9 save\_neighbours()

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

Save neighbours list in .neighbours/neighbours.

Definition at line 74 of file client.c.

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

### 6.23.1.10 set\_neighbour()

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

Sets a neighbour in the client.neighbours section.

#### Returns

0 if success, -1 otherwise if full

Definition at line 19 of file client.c.

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

## 6.23.2 Variable Documentation

### 6.23.2.1 client\_connections

```
client_connection* client_connections = NULL
```

Definition at line 4 of file client.c.

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

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

### Macros

- #define [GENESIS\\_RSA\\_PUB\\_1](#) "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAwcXVgJ6Hy9nry↵  
AmSFGpRYxLtPJ1Vcl9XTbV34hniNMztMGpwSTG\nCQ28WIWID43qjmHxvY4Y26mLYXPjJ2HiwneSo↵  
ZcLtY+gJfObGcclpl1DSA0vE72\neTBbDz8enRbJqFWenwopKDoBjvf7nwc/fqRwD0ptLC7xwIPccRiLGd↵  
OvP/lusLY0\nLCP6A9R50H7tGsbaAQfGoHYezY8p05K6XRankb7I8wsLFdU6Ew6OghX1tq02liP4\nns5↵  
DrloSsxi1mJtW7d+vln0D/a7t2bz4jl+OMtD5M5jldGMyQpzq3D8ZJokMyh6K2\nNLwrAiqDKZiIHJTw8FZid↵  
A9/yuzlRpxNHQIBAw==\n-----END RSA PUBLIC KEY-----\n"
- #define [GENESIS\\_RSA\\_PUB\\_2](#) "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAwrHwAjOQzmoguF↵  
CpWTEe/3x/T6KSr7dF1zYFnCq3V3v6OQFAcyt7\nQG0q138XFasRM70Hc0k589s5nKfPYSz5MCA6iDD1↵  
IKo1qrGSyF9CPfW87DwZuLXW\nnhShifhsLu+VfkbjYx5h/SGmC5WSedro3cTrex7V1BbZkCeKqRMYCgtT↵  
PucyYE4pP\nnqEnFQtMVAasyaDckjpWzpwun9wsoZ3qkqAAREwNecR7i2ojyUBJ8L5ZUryqmx4F\ngwvF↵  
LnlhAeoraWqk40L3bSdnGH1u/YV59f4/MSyVmTezI6DhFx2E3PlD/Kar5PnF\nnrJSEQjtjwg+OvDGrT46S↵  
Kq8JQQlgFVZzwIBAw==\n-----END RSA PUBLIC KEY-----\n"

## Functions

- `Block * get_genesis_block ()`
- `ChunkBlockchain * load_blockchain (size_t nb_chunk)`  
*Loads a blockchain object with a padding of 'nb\_chunk'.*
- `ChunkBlockchain * load_last_blockchain ()`  
*Load the last local blockchain chunk.*
- `size_t get_last_block_height ()`  
*Get the last block height.*
- `void write_block_file (Block block)`  
*Writes a block struct in a file.*
- `void convert_data_to_blockdata (BlockData *blockdata, int fd)`
- `void convert_data_to_block (Block *block, int fd)`
- `Block * get_block (size_t block_height)`  
*Get a block object.*
- `void free_block (Block *block)`  
*Free a block struct.*
- `Block * get_next_block (Block *block)`  
*For a block of height  $h$ , returns the block of height  $h+1$*
- `Block * get_prev_block (Block *block)`  
*For a block of height  $h$ , return the block of height  $h-1$*
- `char * get_blockdata_data (Block *block, size_t *size)`  
*Get the blockdata data object.*
- `void write_blockdata (BlockData blockdata, int fd)`  
*Writes blockdata in a file.*
- `void write_block (Block block, int fd)`  
*Writes a block in a file.*

### 6.24.1 Macro Definition Documentation

#### 6.24.1.1 GENESIS\_RSA\_PUB\_1

```
#define GENESIS_RSA_PUB_1 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAwcXVgJ6Hy9nryAmSFGpR↵  
YxLtPJlVcI9XTbV34hniNMtztMGpwSTG\nCQ28WIWID43qjmHxvY4Y26mLYXPjlJ2HiwneSoZcLtY+gJfObGcc1pI1D↵  
SA0vE72\neTBbDz8enRbJqFWenwopKDoBjvf7nwc/fqRwD0ptLC7xw1PccRiLGdOvP/IusLY0\nnLCP6A9R50H7tGsba↵  
AQfGoHYezY8p05K6XRankb7I8wsLFdU6Ew6OghX1tq02liP4\nns5Dr1oSsxilmJtW7d+vln0D/a7t2bz4jI+OMtD5↵  
M5jldGMyQpzq3D8ZJokMyh6K2\nnNLwrAiqDKZiIHJT8FZidA9/yuzlRpxNHQIBAw==\n-----END RSA PUBLIC K↵  
EY-----\n"
```

Definition at line 3 of file block.c.

### 6.24.1.2 GENESIS\_RSA\_PUB\_2

```
#define GENESIS_RSA_PUB_2 "-----BEGIN RSA PUBLIC KEY-----\nMIIBCAKCAQEAsrHwAjoQzmoguFCpWT↵
Ee/3x/T6KSr7dF1zYFnCq3V3v6OQFAcyt7\nQG0q138XFasRM70Hc0k589s5nKfPYSz5Mca6iDD1IKo1qrGSyF9CPf↵
W87DwZuLXW\nhShifhsLu+VfkbjYx5h/SGmC5WSedro3cTrex7V1BbZkCeKqRMYCgtTPucyYE4pP\nqEnFQtMVAssya↵
DckjpWzpwun9wsoZ3qkqAAREwNecR7i2ojyUBJ8L5ZUryqmx14F\nngwvFLnlhAeoraWqk40L3bSdnGH1u/YV59f4/M↵
SyVmTezI6DhFx2E3PlD/Kar5PnF\nnrJSEQjtjwg+OVdGnrT46SKq8JQqlgFVZzwIBAw==\n-----END RSA PUBLIC K↵
EY-----\n"
```

Definition at line 33 of file block.c.

## 6.24.2 Function Documentation

### 6.24.2.1 convert\_data\_to\_block()

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

Definition at line 172 of file block.c.

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

### 6.24.2.2 convert\_data\_to\_blockdata()

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

Definition at line 139 of file block.c.

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

### 6.24.2.3 free\_block()

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

Free a block struct.

#### Parameters

<i>block</i>	The block to free
--------------	-------------------

Definition at line 202 of file block.c.

Here is the caller graph for this function:

#### 6.24.2.4 get\_block()

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

Get a block object.

##### Parameters

<i>block_height</i>	The height of the block
---------------------	-------------------------

##### Returns

Block\*

Definition at line 180 of file block.c.

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

#### 6.24.2.5 get\_blockdata\_data()

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

Get the blockdata data object.

##### Parameters

<i>block</i>	The block
<i>size</i>	The size of the block

##### Returns

char\*

Definition at line 241 of file block.c.

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

#### 6.24.2.6 get\_genesis\_block()

```
Block* get_genesis_block ( )
```

Definition at line 35 of file block.c.

Here is the call graph for this function:

#### 6.24.2.7 `get_last_block_height()`

```
size_t get_last_block_height ( )
```

Get the last block height.

##### Returns

`size_t`

Definition at line 115 of file `block.c`.

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

#### 6.24.2.8 `get_next_block()`

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

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

##### Parameters

<code>block</code>	The base block
--------------------	----------------

##### Returns

The next Block\*

Definition at line 221 of file `block.c`.

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

#### 6.24.2.9 `get_prev_block()`

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

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

##### Parameters

<code>block</code>	The base block
--------------------	----------------

##### Returns

The next Block\*

Definition at line 231 of file `block.c`.

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

#### 6.24.2.10 load\_blockchain()

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

Loads a blockchain object with a padding of 'nb\_chunk'.

##### Parameters

<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 69 of file block.c.

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

#### 6.24.2.11 load\_last\_blockchain()

```
ChunkBlockchain* load_last_blockchain ( )
```

Load the last local blockchain chunk.

##### Parameters

<i>nb_chunk</i>	
-----------------	--

##### Returns

ChunkBlockchain\*

Definition at line 110 of file block.c.

Here is the call graph for this function:

#### 6.24.2.12 write\_block()

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

Writes a block in a file.

##### Parameters

<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 309 of file block.c.

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

#### 6.24.2.13 write\_block\_file()

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

Writes a block struct in a file.

##### Parameters

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

Definition at line 121 of file block.c.

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

#### 6.24.2.14 write\_blockdata()

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

Writes blockdata in a file.

##### Parameters

<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 277 of file block.c.

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

## 6.25 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/blockchain\_header.c File Reference

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

### Functions

- void [write\\_block\\_header](#) (FILE \*fd, [Block](#) \*block, size\_t height)
- void [gen\\_blockchain\\_header](#) ([infos\\_st](#) \*infos)



## 6.25.1 Function Documentation

### 6.25.1.1 gen\_blockchain\_header()

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

Definition at line 9 of file blockchain\_header.c.

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

### 6.25.1.2 write\_block\_header()

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

Definition at line 3 of file blockchain\_header.c.

Here is the caller graph for this function:

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

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

## Functions

- void [write\\_transactiondata](#) ([TransactionData](#) \*transaction, int fd)
- void [write\\_transaction](#) ([Transaction](#) \*transaction, int fd)
- void [get\\_transaction\\_data](#) ([Transaction](#) \*trans, char \*\*buff, size\_t \*index)  
*Convert transactions to char \* buffer.*
- void [convert\\_data\\_to\\_transactiondata](#) ([TransactionData](#) \*transactiondata, int fd)
- void [load\\_transaction](#) ([Transaction](#) \*transaction, int fd)
- [Transaction](#) \* [load\\_pending\\_transaction](#) (time\_t timestamp)
- void [add\\_pending\\_transaction](#) ([Transaction](#) \*transaction)

## 6.26.1 Function Documentation

#### 6.26.1.1 add\_pending\_transaction()

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

Definition at line 158 of file transaction.c.

Here is the call graph for this function:

#### 6.26.1.2 convert\_data\_to\_transactiondata()

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

Definition at line 99 of file transaction.c.

Here is the caller graph for this function:

#### 6.26.1.3 get\_transaction\_data()

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

Convert transactions to char \* buffer.

##### Parameters

<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 48 of file transaction.c.

Here is the caller graph for this function:

#### 6.26.1.4 load\_pending\_transaction()

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

Definition at line 145 of file transaction.c.

Here is the call graph for this function:

#### 6.26.1.5 load\_transaction()

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

Definition at line 135 of file transaction.c.

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

#### 6.26.1.6 write\_transaction()

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

Definition at line 42 of file transaction.c.

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

#### 6.26.1.7 write\_transactiondata()

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

Definition at line 3 of file transaction.c.

Here is the caller graph for this function:

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

```
#include <time.h>
#include "blockchain/wallet.h"
#include "cryptosystem/rsa.h"
#include "blockchain/transaction.h"
Include dependency graph for wallet.c:
```

### Functions

- `Wallet * get_my_wallet ()`  
*Get my wallet object.*
- `int create_account ()`  
*Creates an account in local and broadcasts the creation to the network.*

## 6.27.1 Function Documentation

### 6.27.1.1 create\_account()

```
int create_account ( )
```

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

#### Returns

0 if the broadcast succeeds, otherwise 1

Definition at line 19 of file wallet.c.

Here is the call graph for this function:

### 6.27.1.2 get\_my\_wallet()

```
Wallet* get_my_wallet ( )
```

Get my wallet object.

#### Returns

[Wallet](#)

Definition at line 7 of file wallet.c.

Here is the caller graph for this function:

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

```
#include <openssl/sha.h>
#include "cryptosystem/hash.h"
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
Include dependency graph for hash.c:
```

### Functions

- char \* [sha384\\_data](#) (void \*data, size\_t len\_data)  
*Apply the SHA384 algorithm on a 'data' of size 'len\_data'.*
- char \* [hash\\_block\\_transactions](#) ([Block](#) \*block)  
*Apply the SHA384 to all block transactions.*

## 6.28.1 Function Documentation

### 6.28.1.1 hash\_block\_transactions()

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

Apply the SHA384 to all block transactions.

**Parameters**

<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:

**6.28.1.2 sha384\_data()**

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

Apply the SHA384 algorithm on a 'data' of size 'len\_data'.

**Parameters**

<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:

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

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

```
#include <errno.h>
#include <openssl/bn.h>
#include <openssl/crypto.h>
#include <string.h>
Include dependency graph for rsa.c:
```

## Macros

- #define [RSA\\_NUM\\_E](#) 3

## Functions

- void [get\\_keys](#) ([\\_\\_attribute\\_\\_](#)((unused)) char \*password)

## 6.29.1 Macro Definition Documentation

### 6.29.1.1 [RSA\\_NUM\\_E](#)

```
#define RSA_NUM_E 3
```

Definition at line 16 of file `rsa.c`.

## 6.29.2 Function Documentation

### 6.29.2.1 [get\\_keys\(\)](#)

```
void get_keys (
    \_\_attribute\_\_((unused)) char * password )
```

Definition at line 21 of file `rsa.c`.

Here is the call graph for this function:

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

```
#include "blockchain/block.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/hash.h"
#include <openssl/bio.h>
#include <openssl/rsa.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
Include dependency graph for signature.c:
```

## Functions

- char \* [sign\\_message](#) (char \*data, size\_t len\_data, void \*buffer)  
*buffer <- encrypt(SHA284(msg,len\_data),wallet\_priv\_key)*
- char \* [sign\\_message\\_with\\_key](#) (char \*data, size\_t len\_data, RSA \*key, void \*buffer)  
*encrypt(SHA284(msg,len\_data),key) buffer <- encrypt(SHA284(msg,len\_data),key)*
- int [verify\\_signature](#) (void \*data, size\_t data\_len, char \*signature, RSA \*pub\_key)  
*Verifies if SHA384(data) == decrypt(signature,pub\_key)*
- int [verify\\_block\\_signature](#) (Block block)  
*Verifies if a block signature is valid.*
- int [verify\\_transaction\\_signature](#) (Transaction transaction)  
*Verifies if a transaction signature is valid.*
- void [sign\\_block](#) (Block \*block)  
*Signs a block.*
- void [sign\\_block\\_with\\_key](#) (Block \*block, RSA \*key)
- void [sign\\_transaction](#) (Transaction \*transaction)
- void [sign\\_transaction\\_with\\_key](#) (Transaction \*transaction, RSA \*key)  
*Sign a transaction.*
- void [sign\\_block\\_transactions](#) (Block \*block)  
*Signs transactions of a block.*

### 6.30.1 Function Documentation

#### 6.30.1.1 sign\_block()

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

Signs a block.

##### Parameters

<i>block</i>	The block to sign
--------------	-------------------

Definition at line 102 of file signature.c.

Here is the call graph for this function:

#### 6.30.1.2 sign\_block\_transactions()

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

Signs transactions of a block.



## Parameters

<i>block</i>	The block to sign
--------------	-------------------

Definition at line 132 of file signature.c.

Here is the call graph for this function:

**6.30.1.3 sign\_block\_with\_key()**

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

Definition at line 109 of file signature.c.

Here is the call graph for this function:

**6.30.1.4 sign\_message()**

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

buffer <- encrypt(SHA284(msg,len\_data),wallet\_priv\_key)

If buffer == NULL, return a new allocated buffer

## Parameters

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>buffer</i>	The buffer to put signature into

## Returns

char\*

Definition at line 10 of file signature.c.

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

**6.30.1.5 sign\_message\_with\_key()**

```
char* sign_message_with_key (
    char * data,
    size_t len_data,
```

```
RSA * key,  
void * buffer )
```

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

If `buffer == NULL`, return a new allocated buffer

**Parameters**

<i>data</i>	The data to sign
<i>len_data</i>	The length of the data
<i>key</i>	The key to use for the signature
<i>buffer</i>	The buffer to put signature into

**Returns**

char\*

Definition at line 34 of file signature.c.

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

**6.30.1.6 sign\_transaction()**

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

Definition at line 116 of file signature.c.

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

**6.30.1.7 sign\_transaction\_with\_key()**

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

Sign a transaction.

**Parameters**

<i>transaction</i>	The transaction to sign
--------------------	-------------------------

Definition at line 124 of file signature.c.

Here is the call graph for this function:

**6.30.1.8 verify\_block\_signature()**

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

Verifies if a block signature is valid.

**Parameters**

<i>block</i>	The block to verify
--------------	---------------------

**Returns**

1 if valid, 0 otherwise

Definition at line 77 of file signature.c.

Here is the call graph for this function:

**6.30.1.9 verify\_signature()**

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

Verifies if SHA384(data) == decrypt(signature, pub\_key)

**Parameters**

<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>pub_key</i>	The RSA public key used for the decryption

**Returns**

int

Definition at line 57 of file signature.c.

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

**6.30.1.10 verify\_transaction\_signature()**

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

Verifies if a transaction signature is valid.

**Parameters**

<i>transaction</i>	The transaction to verify
--------------------	---------------------------

## Returns

1 if valid, 0 otherwise

Definition at line 89 of file signature.c.

Here is the call graph for this function:

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

```
#include "misc/files.h"
#include <dirent.h>
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for files.c:

## Macros

- `#define \_GNU\_SOURCE`

## Functions

- `char * last\_file\_in\_folder (char folder_path[ ])`  
*Return the last file (reverse alphabetical order) of a folder path.*

### 6.31.1 Macro Definition Documentation

#### 6.31.1.1 `_GNU_SOURCE`

```
#define _GNU_SOURCE
```

Definition at line 1 of file files.c.

### 6.31.2 Function Documentation

#### 6.31.2.1 `last_file_in_folder()`

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

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

## Parameters

<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:

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

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

### Functions

- int [safe\\_write](#) (int fd, const void \*buf, ssize\_t count)  
*Writes safely to a file descriptor.*
- int [safe\\_send](#) (int fd, const void \*buf, ssize\_t count)  
*Send safely to a file descriptor.*
- ssize\_t [safe\\_read](#) (int fd, const void \*\*buf, size\_t \*bufsize)  
*Reads safely in a file descriptor until '\r\n\r\n'.*
- ssize\_t [safe\\_fread](#) (void \*buffer, const size\_t size, const size\_t n, FILE \*file)  
*Calls 'fread' but safely !*

### 6.32.1 Function Documentation

#### 6.32.1.1 [safe\\_fread\(\)](#)

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

Calls 'fread' but safely !

## Parameters

<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 58 of file safe.c.

Here is the caller graph for this function:

### 6.32.1.2 safe\_read()

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

Reads safely in a file descriptor until '\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 31 of file safe.c.

Here is the caller graph for this function:

### 6.32.1.3 safe\_send()

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

Send safely to a file descriptor.

## Parameters

<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 17 of file safe.c.

Here is the caller graph for this function:

**6.32.1.4 safe\_write()**

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

Writes safely to a file descriptor.

**Parameters**

<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 3 of file safe.c.

Here is the caller graph for this function:

## 6.33 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/get\_data.c File Reference

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

**Functions**

- [size\\_t process\\_header](#) (char \*header, int sockfd, [infos\\_st](#) \*infos)
- [int fetch\\_client\\_list](#) (char who, int fd)
 

*Merges my neighbours list with the one sent by 'neighbour\_id'.*
- [size\\_t read\\_header](#) (int sockfd, [infos\\_st](#) \*infos)
 

*Waits a header in 'sockfd', reads it and processes it.*
- [int read\\_get\\_blocks](#) (int fd, [infos\\_st](#) \*infos)
- [size\\_t read\\_actual\\_height](#) (int fd)
- [int read\\_send\\_block](#) (int fd)
- [int read\\_vote](#) ([\\_\\_attribute\\_\\_\(\(unused\)\)](#) int fd)
- [int read\\_epoch\\_block](#) (int fd)
- [int read\\_pending\\_transaction\\_list](#) ([\\_\\_attribute\\_\\_\(\(unused\)\)](#) int fd)



## 6.33.1 Function Documentation

### 6.33.1.1 fetch\_client\_list()

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

Merges my neighbours list with the one sent by 'neighbour\_id'.

#### Parameters

<i>sockfd</i>	The sockfd to read
---------------	--------------------

#### Returns

0 if success, -1 otherwise

Definition at line 95 of file get\_data.c.

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

### 6.33.1.2 process\_header()

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

Definition at line 3 of file get\_data.c.

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

### 6.33.1.3 read\_actual\_height()

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

Definition at line 172 of file get\_data.c.

Here is the caller graph for this function:

### 6.33.1.4 read\_epoch\_block()

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

Definition at line 222 of file get\_data.c.

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

**6.33.1.5 read\_get\_blocks()**

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

Definition at line 144 of file get\_data.c.

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

**6.33.1.6 read\_header()**

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

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

**Parameters**

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

**Returns**

0 if success, -1 otherwise

Definition at line 125 of file get\_data.c.

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

**6.33.1.7 read\_pending\_transaction\_list()**

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

Definition at line 229 of file get\_data.c.

Here is the caller graph for this function:

**6.33.1.8 read\_send\_block()**

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

Definition at line 178 of file get\_data.c.

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

### 6.33.1.9 read\_vote()

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

Definition at line 217 of file get\_data.c.

Here is the caller graph for this function:

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

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

### Variables

- const [Neighbour](#) `HARD_CODED_ADDR` [ ]

### 6.34.1 Variable Documentation

#### 6.34.1.1 HARD\_CODED\_ADDR

```
const Neighbour HARD_CODED_ADDR[ ]
```

Initial value:

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

Definition at line 5 of file network.c.

## 6.35 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/send\_data.c File Reference

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

## Functions

- int [send\\_client\\_list](#) (char who, int sockfd, char \*sockip)  
*Sends my client list to a node via 'sockfd'.*
- void [send\\_get\\_blocks](#) ([client\\_connection](#) \*cc)  
*Sends get blocks.*
- void [send\\_actual\\_height](#) (int fd, [infos\\_st](#) \*infos)
- void [send\\_reject\\_demand](#) (int fd)
- void [send\\_send\\_block](#) (int fd, size\_t height)
- void [send\\_pending\\_transaction\\_list](#) ([\\_\\_attribute\\_\\_\(\(unused\)\)](#) int fd)

### 6.35.1 Function Documentation

#### 6.35.1.1 [send\\_actual\\_height\(\)](#)

```
void send_actual_height (
    int fd,
    infos\_st * infos )
```

Definition at line 58 of file [send\\_data.c](#).

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

#### 6.35.1.2 [send\\_client\\_list\(\)](#)

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

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

##### Parameters

<a href="#">sockfd</a>	The sock FD
------------------------	-------------

##### Returns

0 if success, -1 otherwise

Definition at line 3 of file [send\\_data.c](#).

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

#### 6.35.1.3 send\_get\_blocks()

```
void send_get_blocks (
    client_connection * cc )
```

Sends get blocks.

Definition at line 52 of file send\_data.c.

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

#### 6.35.1.4 send\_pending\_transaction\_list()

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

Definition at line 104 of file send\_data.c.

#### 6.35.1.5 send\_reject\_demand()

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

Definition at line 65 of file send\_data.c.

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

#### 6.35.1.6 send\_send\_block()

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

Definition at line 71 of file send\_data.c.

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

## 6.36 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/server.c File Reference

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

## Functions

- void \* [accept\\_connection](#) (void \*args)
- void \* [redirect\\_connection](#) (void \*arg)
- void \* [init\\_server](#) (void \*args)

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

### 6.36.1 Function Documentation

#### 6.36.1.1 [accept\\_connection\(\)](#)

```
void* accept_connection (
    void * args )
```

Definition at line 3 of file server.c.

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

#### 6.36.1.2 [init\\_server\(\)](#)

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

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

##### Parameters

<i>type</i>	Type of the server
-------------	--------------------

##### Returns

0 if success, -1 otherwise

Definition at line 77 of file server.c.

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

#### 6.36.1.3 [redirect\\_connection\(\)](#)

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

Definition at line 43 of file server.c.

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

## 6.37 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/server.c File Reference

```
#include "network/server.h"
#include "network/client.h"
#include "cryptosystem/signature.h"
#include "blockchain/block.h"
#include <time.h>
Include dependency graph for server.c:
```

### Functions

- int [main](#) ()

#### 6.37.1 Function Documentation

##### 6.37.1.1 main()

```
int main ( )
```

Definition at line 7 of file server.c.

Here is the call graph for this function:

## 6.38 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/ui/ui.c File Reference

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

### Functions

- void \* [setup](#) (void \*args)  
*Setups the gtk widgets for the GUI.*
- void [change\\_label\\_text](#) (GtkLabel \*label, char \*text)
- void [add\\_new\\_blockinfo](#) (size\_t height, size\_t transaction)
- void [update\\_sync](#) (size\_t actual, size\_t final)
- gboolean [on\\_main\\_window\\_delete](#) (GtkWidget \*widget, [\\_\\_attribute\\_\\_\(\(unused\)\)](#) gpointer data)  
*Destroys the window when it is closed.*
- void [on\\_main\\_window\\_destroy](#) ([\\_\\_attribute\\_\\_\(\(unused\)\)](#) GtkWidget \*widget, [\\_\\_attribute\\_\\_\(\(unused\)\)](#) gpointer data)

- gboolean `on_transaction_button_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- void `add_transaction_with_pkey` (double amount, char \*public\_key, char \*date)
- void `add_transaction_with_contact` (double amount, char \*public\_key, char \*date)
- void `add_transaction_from_file` (double amount, char \*public\_key, char \*date)
- void `load_transactions_from_file` ()
- gboolean `on_pkey_button_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_invest_button1_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_invest_button2_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_recover_button1_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_recover_button2_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_add_contact_button1_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `add_contact` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- void `add_contact_to_combobox` (char \*name)
- void `add_contacts_from_file` (char \*name, char \*public\_key)
- void `load_contacts_from_file` ()
- char \* `get_public_key_from_contacts` (const char \*name)
- gboolean `on_create_key_but1_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_create_key_but2_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- gboolean `on_connect_but_press` (`__attribute__((unused))` GtkWidget \*widget, `__attribute__((unused))` GdkEventKey \*event, `__attribute__((unused))` gpointer user\_data)
- void `update_labels` ()

## Variables

- GtkWidget \* `balance_1`
- GtkWidget \* `balance_2`
- GtkWidget \* `private_key_label`
- GtkWidget \* `stake_label1`
- GtkWidget \* `stake_label2`
- GtkWidget \* `stake_label3`
- GtkWidget \* `password_error_label`
- GtkWidget \* `latest_block_name1`
- GtkWidget \* `latest_block_name2`
- GtkWidget \* `latest_block_name3`
- GtkWidget \* `transa_amount`
- GtkWidget \* `recipient_key`
- GtkWidget \* `invest_entry`
- GtkWidget \* `recover_entry`
- GtkWidget \* `name_entry_con`
- GtkWidget \* `public_key_entry_con`
- GtkWidget \* `password_entry1`
- GtkWidget \* `password_entry2`
- GtkWidget \* `key_entry`
- GtkWidget \* `tv_con`



- GtkTreeStore \* [ts\\_con](#)
- GtkTreeViewColumn \* [cx1\\_con](#)
- GtkTreeViewColumn \* [cx2\\_con](#)
- GtkCellRenderer \* [cr1\\_con](#)
- GtkCellRenderer \* [cr2\\_con](#)
- GtkTreeView \* [tv\\_th](#)
- GtkTreeStore \* [ts\\_th](#)
- GtkTreeViewColumn \* [cx1\\_th](#)
- GtkTreeViewColumn \* [cx2\\_th](#)
- GtkTreeViewColumn \* [cx3\\_th](#)
- GtkCellRenderer \* [cr1\\_th](#)
- GtkCellRenderer \* [cr2\\_th](#)
- GtkCellRenderer \* [cr3\\_th](#)
- GtkComboBox \* [contacts\\_combo](#)
- GtkListStore \* [ls\\_combo](#)
- GtkCellRenderer \* [cr1\\_combo](#)
- GtkProgressBar \* [progress\\_bar\\_blockchain](#)

## 6.38.1 Function Documentation

### 6.38.1.1 `add_contact()`

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

Definition at line 437 of file ui.c.

Here is the call graph for this function:

### 6.38.1.2 `add_contact_to_combobox()`

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

Definition at line 466 of file ui.c.

Here is the caller graph for this function:

### 6.38.1.3 `add_contacts_from_file()`

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

Definition at line 474 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.4 add\_new\_blockinfo()

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

Definition at line 199 of file ui.c.

#### 6.38.1.5 add\_transaction\_from\_file()

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

Definition at line 324 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.6 add\_transaction\_with\_contact()

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

Definition at line 304 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.7 add\_transaction\_with\_pkey()

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

Definition at line 283 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.8 change\_label\_text()

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

Definition at line 194 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.9 get\_public\_key\_from\_contacts()

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

Definition at line 511 of file ui.c.

Here is the caller graph for this function:

#### 6.38.1.10 load\_contacts\_from\_file()

```
void load_contacts_from_file ( )
```

Definition at line 483 of file ui.c.

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

#### 6.38.1.11 load\_transactions\_from\_file()

```
void load_transactions_from_file ( )
```

Definition at line 334 of file ui.c.

Here is the call graph for this function:

#### 6.38.1.12 on\_add\_contact\_button1\_press()

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

Definition at line 428 of file ui.c.

#### 6.38.1.13 on\_connect\_but\_press()

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

Definition at line 608 of file ui.c.

Here is the call graph for this function:

#### 6.38.1.14 on\_create\_key\_but1\_press()

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

Definition at line 548 of file ui.c.

#### 6.38.1.15 on\_create\_key\_but2\_press()

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

Definition at line 563 of file ui.c.

Here is the call graph for this function:

#### 6.38.1.16 on\_invest\_button1\_press()

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

Definition at line 389 of file ui.c.

#### 6.38.1.17 on\_invest\_button2\_press()

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

Definition at line 398 of file ui.c.

#### 6.38.1.18 on\_main\_window\_delete()

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

Destroys the window when it is closed.

#### Parameters

<i>widget</i>	The main window of the GUI
---------------	----------------------------

#### Returns

gboolean Error code

Definition at line 233 of file ui.c.

#### 6.38.1.19 on\_main\_window\_destroy()

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

Definition at line 242 of file ui.c.

#### 6.38.1.20 on\_pkey\_button\_press()

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

Definition at line 371 of file ui.c.

#### 6.38.1.21 on\_recover\_button1\_press()

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

Definition at line 408 of file ui.c.

#### 6.38.1.22 on\_recover\_button2\_press()

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

Definition at line 417 of file ui.c.

### 6.38.1.23 on\_transaction\_button\_press()

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

Definition at line 249 of file ui.c.

Here is the call graph for this function:

### 6.38.1.24 setup()

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

Setups the gtk widgets for the GUI.

#### Returns

int Returns 1 if there is an error, 0 otherwise

Definition at line 63 of file ui.c.

Here is the caller graph for this function:

### 6.38.1.25 update\_labels()

```
void update_labels ( )
```

Definition at line 658 of file ui.c.

Here is the call graph for this function:

### 6.38.1.26 update\_sync()

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

Definition at line 214 of file ui.c.

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

## 6.38.2 Variable Documentation

#### 6.38.2.1 balance\_1

```
GtkLabel* balance_1
```

Definition at line 23 of file ui.c.

#### 6.38.2.2 balance\_2

```
GtkLabel* balance_2
```

Definition at line 24 of file ui.c.

#### 6.38.2.3 contacts\_combo

```
GtkComboBox* contacts_combo
```

Definition at line 56 of file ui.c.

#### 6.38.2.4 cr1\_combo

```
GtkCellRenderer* cr1_combo
```

Definition at line 58 of file ui.c.

#### 6.38.2.5 cr1\_con

```
GtkCellRenderer* cr1_con
```

Definition at line 46 of file ui.c.

#### 6.38.2.6 cr1\_th

```
GtkCellRenderer* cr1_th
```

Definition at line 53 of file ui.c.

**6.38.2.7 cr2\_con**

```
GtkCellRenderer* cr2_con
```

Definition at line 47 of file ui.c.

**6.38.2.8 cr2\_th**

```
GtkCellRenderer* cr2_th
```

Definition at line 54 of file ui.c.

**6.38.2.9 cr3\_th**

```
GtkCellRenderer* cr3_th
```

Definition at line 55 of file ui.c.

**6.38.2.10 cx1\_con**

```
GtkTreeViewColumn* cx1_con
```

Definition at line 44 of file ui.c.

**6.38.2.11 cx1\_th**

```
GtkTreeViewColumn* cx1_th
```

Definition at line 50 of file ui.c.

**6.38.2.12 cx2\_con**

```
GtkTreeViewColumn* cx2_con
```

Definition at line 45 of file ui.c.



#### 6.38.2.13 cx2\_th

GtkTreeViewColumn\* cx2\_th

Definition at line 51 of file ui.c.

#### 6.38.2.14 cx3\_th

GtkTreeViewColumn\* cx3\_th

Definition at line 52 of file ui.c.

#### 6.38.2.15 invest\_entry

GtkEntry\* invest\_entry

Definition at line 35 of file ui.c.

#### 6.38.2.16 key\_entry

GtkEntry\* key\_entry

Definition at line 41 of file ui.c.

#### 6.38.2.17 latest\_block\_name1

GtkLabel\* latest\_block\_name1

Definition at line 30 of file ui.c.

#### 6.38.2.18 latest\_block\_name2

GtkLabel\* latest\_block\_name2

Definition at line 31 of file ui.c.

**6.38.2.19 latest\_block\_name3**

```
GtkLabel* latest_block_name3
```

Definition at line 32 of file ui.c.

**6.38.2.20 ls\_combo**

```
GtkListStore* ls_combo
```

Definition at line 57 of file ui.c.

**6.38.2.21 name\_entry\_con**

```
GtkEntry* name_entry_con
```

Definition at line 37 of file ui.c.

**6.38.2.22 password\_entry1**

```
GtkEntry* password_entry1
```

Definition at line 39 of file ui.c.

**6.38.2.23 password\_entry2**

```
GtkEntry* password_entry2
```

Definition at line 40 of file ui.c.

**6.38.2.24 password\_error\_label**

```
GtkLabel* password_error_label
```

Definition at line 29 of file ui.c.

#### 6.38.2.25 private\_key\_label

```
GtkLabel* private_key_label
```

Definition at line 25 of file ui.c.

#### 6.38.2.26 progress\_bar\_blockchain

```
GtkProgressBar* progress_bar_blockchain
```

Definition at line 59 of file ui.c.

#### 6.38.2.27 public\_key\_entry\_con

```
GtkEntry* public_key_entry_con
```

Definition at line 38 of file ui.c.

#### 6.38.2.28 recipient\_key

```
GtkEntry* recipient_key
```

Definition at line 34 of file ui.c.

#### 6.38.2.29 recover\_entry

```
GtkEntry* recover_entry
```

Definition at line 36 of file ui.c.

#### 6.38.2.30 stake\_label1

```
GtkLabel* stake_label1
```

Definition at line 26 of file ui.c.

**6.38.2.31 stake\_label2**

```
GtkLabel* stake_label2
```

Definition at line 27 of file ui.c.

**6.38.2.32 stake\_label3**

```
GtkLabel* stake_label3
```

Definition at line 28 of file ui.c.

**6.38.2.33 transa\_amount**

```
GtkEntry* transa_amount
```

Definition at line 33 of file ui.c.

**6.38.2.34 ts\_con**

```
GtkTreeStore* ts_con
```

Definition at line 43 of file ui.c.

**6.38.2.35 ts\_th**

```
GtkTreeStore* ts_th
```

Definition at line 49 of file ui.c.

**6.38.2.36 tv\_con**

```
GtkTreeView* tv_con
```

Definition at line 42 of file ui.c.

**6.38.2.37 tv\_th**

```
GtkTreeView* tv_th
```

Definition at line 48 of file ui.c.

## 6.39 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/epoch\_man.c File Reference

```
#include "validation/epoch_man.h"
```

Include dependency graph for epoch\_man.c:

**Functions**

- RSA \* [get\\_epoch\\_man\\_pkey](#) (BlockData \*block\_data)  
*Give the pkey of the creator of a block.*

**6.39.1 Function Documentation****6.39.1.1 get\_epoch\_man\_pkey()**

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

Give the pkey of the creator of a block.

**Parameters**

<i>block_data</i>	The created block data
-------------------	------------------------

**Returns**

RSA\*

Definition at line 3 of file epoch\_man.c.

Here is the caller graph for this function:

## 6.40 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/validation/validation\_engine.c File Reference

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

```
#include "validation/validators.h"
#include "misc/math.h"
Include dependency graph for validators.c:
```

### Macros

- #define [NB\\_RSA\\_CHUNK](#) 2048/64

### Functions

- uint16\_t [define\\_nb\\_validators](#) (size\_t n)
- char \* [hash\\_block\\_transactions\\_epoch](#) (Block \*block)
- RSA \*\* [get\\_comittee](#) (size\_t block\_height, size\_t \*nb\_validators)  
*Get the a comittee RSA public keys on a specific epoch.*
- RSA \*\* [get\\_next\\_comittee](#) (size\_t \*nb\_validators)  
*Get the a comittee RSA public keys on a specific epoch.*
- ssize\_t [get\\_validators\\_states\\_total\\_stake](#) ()  
*Get the total stake of the network (parse 'validators.state')*
- ssize\_t [get\\_validators\\_states\\_nb\\_validators](#) ()  
*Get the number of validators of the network (parse 'validators.state')*
- ssize\_t [get\\_validators\\_states\\_block\\_height\\_validity](#) ()  
*Get the validators states block height validity (parse 'validators.state')*
- ssize\_t [get\\_validator\\_stake](#) (size\_t validator\_id)  
*Get a validator total stake (parse 'validators.state')*
- ssize\_t [get\\_validator\\_power](#) (size\_t validator\_id)  
*Get a validator power (parse 'validators.state')*
- RSA \* [get\\_validator\\_pkey](#) (size\_t validator\_id)  
*Get the validator pkey as RSA\* (parse 'validators.state')*
- ssize\_t [get\\_validator\\_id](#) (char pkey[])  
*Get the validator id in 'validators.state'.*

### 6.41.1 Macro Definition Documentation

### 6.41.1.1 NB\_RSA\_CHUNK

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

Definition at line 4 of file validators.c.

## 6.41.2 Function Documentation

### 6.41.2.1 define\_nb\_validators()

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

Definition at line 6 of file validators.c.

Here is the caller graph for this function:

### 6.41.2.2 get\_comittee()

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

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

#### Parameters

<i>block_height</i>	The height of the block you want a comitte from
<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 32 of file validators.c.

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

### 6.41.2.3 get\_next\_comittee()

```
RSA** get_next_comittee (
    size_t * nb_validators )
```

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

**Parameters**

<i>block_height</i>	The height of the block you want a comitte from
<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 124 of file validators.c.

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

**6.41.2.4 get\_validator\_id()**

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

Get the validator id in 'validators.state'.

**Parameters**

<i>pkey</i>	The string public key (without ---BEGIN RSA KEY--- & ---END RSA KEY---)
-------------	---

**Returns**

ssize\_t, the validator index

Definition at line 209 of file validators.c.

Here is the call graph for this function:

**6.41.2.5 get\_validator\_pkey()**

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

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

**Parameters**

<i>validator_id</i>	The id of the validator in 'validators.state'
---------------------	---



**Returns**

RSA\*

Definition at line 188 of file validators.c.

Here is the call graph for this function:

**6.41.2.6 get\_validator\_power()**

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

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

**Parameters**

<i>validator</i> ↔ _id	The id of the validator in 'validators.state'
---------------------------	---

**Returns**

ssize\_t

Definition at line 176 of file validators.c.

Here is the call graph for this function:

**6.41.2.7 get\_validator\_stake()**

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

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

**Parameters**

<i>validator</i> ↔ _id	The id of the validator in 'validators.state'
---------------------------	---

**Returns**

ssize\_t

Definition at line 164 of file validators.c.

Here is the call graph for this function:

#### 6.41.2.8 `get_validators_states_block_height_validity()`

```
ssize_t get_validators_states_block_height_validity ( )
```

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

##### Returns

`ssize_t`

Definition at line 152 of file validators.c.

Here is the call graph for this function:

#### 6.41.2.9 `get_validators_states_nb_validators()`

```
ssize_t get_validators_states_nb_validators ( )
```

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

##### Returns

`ssize_t`

Definition at line 141 of file validators.c.

Here is the call graph for this function:

#### 6.41.2.10 `get_validators_states_total_stake()`

```
ssize_t get_validators_states_total_stake ( )
```

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

##### Returns

`ssize_t`

Definition at line 129 of file validators.c.

Here is the call graph for this function:

#### 6.41.2.11 `hash_block_transactions_epoch()`

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

Definition at line 20 of file validators.c.

Here is the call graph for this function:

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

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

### Functions

- int [main](#) (int argc, char \*\*argv)

#### 6.42.1 Function Documentation

##### 6.42.1.1 main()

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

Definition at line 3 of file gui.c.

Here is the call graph for this function:

## 6.43 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/serverdoor.c File Reference

```
#include "network/server.h"
#include "network/client.h"
#include "cryptosystem/signature.h"
#include "blockchain/block.h"
#include <time.h>
Include dependency graph for serverdoor.c:
```

### Functions

- int [main](#) ()

#### 6.43.1 Function Documentation

#### 6.43.1.1 main()

```
int main ( )
```

Definition at line 10 of file serverdoor.c.

Here is the call graph for this function:

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

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

Include dependency graph for sign.c:

#### Functions

- int [main](#) ()

#### 6.44.1 Function Documentation

##### 6.44.1.1 main()

```
int main ( )
```

Definition at line 10 of file sign.c.

Here is the call graph for this function:

### 6.45 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN\_blockchain\_files.c File Reference

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

Include dependency graph for GEN\_blockchain\_files.c: This graph shows which files directly or indirectly include this file:

## Macros

- #define [GEN\\_BLC\\_F\\_C](#)

## Functions

- void [rand\\_data](#) (size\_t size, char \*buffer)
- void [gen\\_blockchain](#) (size\_t nb\_blocks)

### 6.45.1 Macro Definition Documentation

#### 6.45.1.1 GEN\_BLC\_F\_C

```
#define GEN_BLC_F_C
```

Definition at line 2 of file GEN\_blockchain\_files.c.

### 6.45.2 Function Documentation

#### 6.45.2.1 gen\_blockchain()

```
void gen_blockchain (  
    size_t nb_blocks )
```

Definition at line 22 of file GEN\_blockchain\_files.c.

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

#### 6.45.2.2 rand\_data()

```
void rand_data (  
    size_t size,  
    char * buffer )
```

Definition at line 8 of file GEN\_blockchain\_files.c.

Here is the caller graph for this function:

## 6.46 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/gen/GEN\_validators\_file.c File Reference

```
#include <stdio.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include "tests_macros.h"
#include "cryptosystem/rsa.h"
```

Include dependency graph for GEN\_validators\_file.c: This graph shows which files directly or indirectly include this file:

### Macros

- #define [GEN\\_VALIDATORS\\_FILE\\_H](#)
- #define [NB\\_FAKE\\_VALIDATORS](#) 10
- #define [str\(x\)](#) #x

### Functions

- void [gen\\_validators\\_file](#) (char path[])  
*Generate a mock validators states file.*

## 6.46.1 Macro Definition Documentation

### 6.46.1.1 GEN\_VALIDATORS\_FILE\_H

```
#define GEN_VALIDATORS_FILE_H
```

Definition at line 2 of file GEN\_validators\_file.c.

### 6.46.1.2 NB\_FAKE\_VALIDATORS

```
#define NB_FAKE_VALIDATORS 10
```

Definition at line 14 of file GEN\_validators\_file.c.

### 6.46.1.3 str

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

Definition at line 15 of file GEN\_validators\_file.c.

## 6.46.2 Function Documentation

### 6.46.2.1 gen\_validators\_file()

```
void gen_validators_file (  
    char path[] )
```

Generate a mock validators states file.

#### Parameters

<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 31 of file GEN\_validators\_file.c.

Here is the caller graph for this function:

## 6.47 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/tests/headers/blockchain/block\_test.h File Reference

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

### Functions

- void [block\\_test](#) (void)

## 6.47.1 Function Documentation

### 6.47.1.1 block\_test()

```
void block_test (
    void )
```

Definition at line 13 of file block\_test.c.

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

## 6.48 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/cryptosystem/rsa\_test.h File Reference

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

### Functions

- void [get\\_keys\\_test](#) ()
- void [get\\_keys\\_equality\\_test](#) ()

## 6.48.1 Function Documentation

### 6.48.1.1 get\_keys\_equality\_test()

```
void get_keys_equality_test ( )
```

Definition at line 32 of file rsa\_test.c.

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

### 6.48.1.2 get\_keys\_test()

```
void get_keys_test ( )
```

Definition at line 18 of file rsa\_test.c.

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



## 6.49 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/tests/headers/cryptosystem/signature\_test.h File Reference

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

### Functions

- void [verify\\_sign\\_test](#) ()

#### 6.49.1 Function Documentation

##### 6.49.1.1 [verify\\_sign\\_test](#)()

```
void verify_sign_test ( )
```

Definition at line 4 of file signature\_test.c.

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

## 6.50 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/tests/headers/network/client\_test.h File Reference

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

### Functions

- void [network\\_test](#) ()

#### 6.50.1 Function Documentation

##### 6.50.1.1 [network\\_test](#)()

```
void network_test ( )
```

Definition at line 15 of file client\_test.c.

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

## 6.51 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/headers/validation/validations\_test.h File Reference

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

### Functions

- void [validations\\_test](#) ()

### 6.51.1 Function Documentation

#### 6.51.1.1 validations\_test()

```
void validations_test ( )
```

Definition at line 6 of file `validations_test.c`.

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

## 6.52 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/main\_test.c File Reference

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

Include dependency graph for `main_test.c`:

### Macros

- #define [MAIN\\_TEST\\_C](#)

### Functions

- int [main](#) ()

### 6.52.1 Macro Definition Documentation

### 6.52.1.1 MAIN\_TEST\_C

```
#define MAIN_TEST_C
```

Definition at line 2 of file main\_test.c.

## 6.52.2 Function Documentation

### 6.52.2.1 main()

```
int main ( )
```

Definition at line 5 of file main\_test.c.

Here is the call graph for this function:

## 6.53 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/blockchain/block\_test.c File Reference

```
#include "tests_macros.h"
#include "blockchain/block.h"
#include "blockchain/transaction.h"
#include "gen/GEN_blockchain_files.c"
Include dependency graph for block_test.c:
```

### Macros

- #define [BLOCK\\_TEST\\_C](#)
- #define [NB\\_BLOCK\\_PER\\_CHUNK](#) 10
- #define [NB MOCK\\_BLOCKS](#) 13

### Functions

- void [block\\_test](#) (void)

### 6.53.1 Macro Definition Documentation

### 6.53.1.1 BLOCK\_TEST\_C

```
#define BLOCK_TEST_C
```

Definition at line 2 of file block\_test.c.

### 6.53.1.2 NB\_BLOCK\_PER\_CHUNK

```
#define NB_BLOCK_PER_CHUNK 10
```

Definition at line 9 of file block\_test.c.

### 6.53.1.3 NB MOCK\_BLOCKS

```
#define NB MOCK_BLOCKS 13
```

Definition at line 11 of file block\_test.c.

## 6.53.2 Function Documentation

### 6.53.2.1 block\_test()

```
void block_test (
    void )
```

Definition at line 13 of file block\_test.c.

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

## 6.54 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/cryptosystem/rsa\_test.c File Reference

```
#include "tests_macros.h"
#include "cryptosystem/signature.h"
#include "cryptosystem/rsa.h"
#include "blockchain/wallet.h"
#include "misc/math.h"
#include <stdio.h>
#include <unistd.h>
#include <openssl/sha.h>
#include "misc/safe.h"
#include <fcntl.h>
#include <sys/stat.h>
```

Include dependency graph for rsa\_test.c:

## Macros

- #define [RSA\\_SIZE\\_C](#)

## Functions

- void [get\\_keys\\_test](#) ()
- void [get\\_keys\\_equality\\_test](#) ()

### 6.54.1 Macro Definition Documentation

#### 6.54.1.1 RSA\_SIZE\_C

```
#define RSA_SIZE_C
```

Definition at line 2 of file `rsa_test.c`.

### 6.54.2 Function Documentation

#### 6.54.2.1 [get\\_keys\\_equality\\_test\(\)](#)

```
void get_keys_equality_test ( )
```

Definition at line 32 of file `rsa_test.c`.

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

#### 6.54.2.2 [get\\_keys\\_test\(\)](#)

```
void get_keys_test ( )
```

Definition at line 18 of file `rsa_test.c`.

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

## 6.55 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/tests/src/cryptosystem/signature\_test.c File Reference

```
#include "tests_macros.h"  
#include "cryptosystem/signature.h"  
Include dependency graph for signature_test.c:
```

## Functions

- void [verify\\_sign\\_test](#) ()

### 6.55.1 Function Documentation

#### 6.55.1.1 [verify\\_sign\\_test](#)()

```
void verify_sign_test ( )
```

Definition at line 4 of file `signature_test.c`.

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

## 6.56 [/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/src/network/client\\_test.c](#) File Reference

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

## Macros

- #define [CLIENT\\_TEST\\_C](#)

## Functions

- void [network\\_test](#) ()

## Variables

- [client\\_connection](#) \* [client\\_connections](#)

### 6.56.1 Macro Definition Documentation

### 6.56.1.1 CLIENT\_TEST\_C

```
#define CLIENT_TEST_C
```

Definition at line 2 of file client\_test.c.

## 6.56.2 Function Documentation

### 6.56.2.1 network\_test()

```
void network_test ( )
```

Definition at line 15 of file client\_test.c.

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

## 6.56.3 Variable Documentation

### 6.56.3.1 client\_connections

```
client_connection* client_connections
```

Definition at line 4 of file client.c.

## 6.57 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS- Cryptocurrency/tests/src/validation/validations\_test.c File Reference

```
#include "gen/GEN_validators_file.c"  
#include "validation/validators.h"  
#include "tests_macros.h"  
Include dependency graph for validations_test.c:
```

## Functions

- void [validations\\_test](#) ()

### 6.57.1 Function Documentation

### 6.57.1.1 validations\_test()

```
void validations_test ( )
```

Definition at line 6 of file validations\_test.c.

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

## 6.58 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/tests\_macros.h File Reference

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

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

### Macros

- #define [DEBUG](#)(function)
- #define [LOG](#)(str...)
- #define [TEST\\_PASSED](#)(name...)
- #define [TEST\\_FAILED](#)(name, reason...)
- #define [TEST\\_WARNING](#)(name, reason...)

### 6.58.1 Macro Definition Documentation

#### 6.58.1.1 DEBUG

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

##### Value:

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

Definition at line 5 of file tests\_macros.h.

#### 6.58.1.2 LOG

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

##### Value:

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

Definition at line 9 of file tests\_macros.h.



### 6.58.1.3 TEST\_FAILED

```
#define TEST_FAILED(  
    name,  
    reason... )
```

**Value:**

```
printf("\033[0;31m[X] TEST '%s' failed\n\t-> REASON : ", name); \  
printf(reason); \  
printf("\033[0m\n"); \  
exit(1)
```

Definition at line 19 of file tests\_macros.h.

### 6.58.1.4 TEST\_PASSED

```
#define TEST_PASSED(  
    name... )
```

**Value:**

```
printf("\033[0;32m[OK] TEST -> '"); \  
printf(name); \  
printf("' success\033[0m\n")
```

Definition at line 14 of file tests\_macros.h.

### 6.58.1.5 TEST\_WARNING

```
#define TEST_WARNING(  
    name,  
    reason... )
```

**Value:**

```
printf("\033[0;33m[!] WARNING '%s'\n\t-> BECAUSE : ", name); \  
printf(reason); \  
printf("\033[0m\n")
```

Definition at line 25 of file tests\_macros.h.

## 6.59 /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit\_testing.c File Reference

```
#include "tests_macros.h"  
#include "cryptosystem/rsa_test.h"  
#include "cryptosystem/signature_test.h"  
#include "network/client_test.h"  
#include "blockchain/block_test.h"  
#include "validation/validations_test.h"
```

Include dependency graph for unit\_testing.c:

## Functions

- int `main` ()

### 6.59.1 Function Documentation

#### 6.59.1.1 `main()`

```
int main ( )
```

Definition at line 8 of file `unit_testing.c`.

Here is the call graph for this function:

# Index

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/P2P\_Protocol.md, [82](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/README.md, [82](#)

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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/blockchain\_header.h, [25](#)

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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/blockchain/wallet.h, [30](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/hash.h, [32](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/cryptosystem/rsa.h, [33](#)

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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/misc/files.h, [42](#)

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

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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/client.h, [46](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/get\_data.h, [49](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/network.h, [52](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/send\_data.h, [61](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/network/server.h, [63](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/ui/ui.h, [64](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/epoch\_man.h, [73](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validation\_engine.h, [75](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/headers/validation/validators.h, [77](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/client.c, [82](#)

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

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/blockchain\_header.c, [94](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/transaction.c, [95](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/blockchain/wallet.c, [97](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/hash.c, [98](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/rsa.c, [100](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/cryptosystem/signature.c, [101](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/files.c, [107](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/misc/safe.c, [108](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/client.c, [84](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/get\_data.c, [110](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/network.c, [113](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/send\_data.c, [113](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/network/server.c, [115](#)

/home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/src/core/ui/ui.c, [117](#)

- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-Cryptocurrency/tests/unit\_testing.c, 151
- Cryptocurrency/src/core/validation/epoch\_man.c, GNU\_SOURCE 131
- files.c, 107
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-\_\_attribute\_\_
- Cryptocurrency/src/core/validation/validation\_engine.network.h, 60
- 132
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-accept\_connection
- Cryptocurrency/src/core/validation/validators.c, server.c, 116
- 132
- actual\_client\_height
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-client\_connection, 15
- Cryptocurrency/src/gui.c, 137
- actual\_height
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-infos\_st, 16
- Cryptocurrency/src/server.c, 117
- add\_contact
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-ui.c, 119
- Cryptocurrency/src/serverdoor.c, 137
- ui.h, 65
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_contact\_to\_combobox
- Cryptocurrency/src/sign.c, 138
- ui.c, 119
- ui.h, 65
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_contacts\_from\_file
- Cryptocurrency/tests/gen/GEN\_blockchain\_files.c, ui.c, 119
- 138
- ui.h, 65
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_new\_blockinfo
- Cryptocurrency/tests/gen/GEN\_validators\_file.c, ui.c, 119
- 140
- ui.h, 66
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_pending\_transaction
- Cryptocurrency/tests/headers/blockchain/block\_test.h, transaction.c, 95
- 141
- transaction.h, 28
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_transaction\_from\_file
- Cryptocurrency/tests/headers/cryptosystem/rsa\_test.h, ui.c, 120
- 142
- ui.h, 66
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_transaction\_with\_contact
- Cryptocurrency/tests/headers/cryptosystem/signature\_test.h, ui.c, 120
- 143
- ui.h, 66
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-add\_transaction\_with\_pkey
- Cryptocurrency/tests/headers/network/client\_test.h, ui.c, 120
- 143
- ui.h, 66
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-amount
- Cryptocurrency/tests/headers/validation/validations\_test.h, TransactionData, 21
- 144
- Wallet, 23
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-asset
- Cryptocurrency/tests/main\_test.c, 144
- TransactionData, 21
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-balance\_1
- Cryptocurrency/tests/src/blockchain/block\_test.c, ui.c, 124
- 145
- balance\_2
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-ui.c, 125
- Cryptocurrency/tests/src/cryptosystem/rsa\_test.c, 146
- Block, 9
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-block\_data, 9
- Cryptocurrency/tests/src/cryptosystem/signature\_test.c, block\_signature, 9
- 147
- chunk\_id, 10
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-validators\_votes, 10
- Cryptocurrency/tests/src/network/client\_test.c, vote\_signature, 10
- 148
- block.c
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-convert\_data\_to\_block, 90
- Cryptocurrency/tests/src/validation/validations\_test.c, convert\_data\_to\_blockdata, 90
- 149
- free\_block, 90
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-GENESIS\_RSA\_PUB\_1, 89
- Cryptocurrency/tests/tests\_macros.h, 150
- GENESIS\_RSA\_PUB\_2, 89
- /home/runner/work/PEPITAS-Cryptocurrency/PEPITAS-get\_block, 90

- get\_blockdata\_data, 91
- get\_genesis\_block, 91
- get\_last\_block\_height, 91
- get\_next\_block, 92
- get\_prev\_block, 92
- load\_blockchain, 92
- load\_last\_blockchain, 93
- write\_block, 93
- write\_block\_file, 94
- write\_blockdata, 94
- block\_amount\_label
  - ui.h, 72
- block\_data
  - Block, 9
- block\_signature
  - Block, 9
- block\_test
  - block\_test.c, 146
  - block\_test.h, 142
- block\_test.c
  - block\_test, 146
  - BLOCK\_TEST\_C, 145
  - NB\_BLOCK\_PER\_CHUNK, 146
  - NB MOCK\_BLOCKS, 146
- block\_test.h
  - block\_test, 142
- BLOCK\_TEST\_C
  - block\_test.c, 145
- block\_timestamp
  - BlockData, 11
- blockchain\_header.c
  - gen\_blockchain\_header, 95
  - write\_block\_header, 95
- blockchain\_header.h
  - gen\_blockchain\_header, 26
- BlockData, 10
  - block\_timestamp, 11
  - epoch\_id, 11
  - height, 11
  - is\_prev\_block\_valid, 11
  - magic, 11
  - nb\_transactions, 11
  - nb\_validators, 12
  - prev\_validators\_votes, 12
  - previous\_block\_hash, 12
  - transactions, 12
  - validators\_public\_keys, 12
- blockinfo, 13
  - height, 13
  - transactions, 13
- blocksinfo
  - ui.h, 72
- cause
  - TransactionData, 21
- change\_label\_text
  - ui.c, 120
  - ui.h, 66
- chunk
  - ChunkBlockchain, 14
- chunk\_id
  - Block, 10
- chunk\_nb
  - ChunkBlockchain, 14
- ChunkBlockchain, 13
  - chunk, 14
  - chunk\_nb, 14
  - nb\_blocks, 14
- client.c
  - client\_connections, 84, 88
  - client\_thread, 85
  - connection\_to\_others, 83
  - find\_empty\_connection, 85
  - get\_my\_node, 85
  - join\_network\_door, 83
  - listen\_to, 86
  - load\_neighbours, 86
  - main, 83
  - number\_neighbours, 86
  - print\_neighbours, 87
  - remove\_neighbour, 87
  - save\_neighbours, 87
  - set\_neighbour, 87
  - update\_blockchain, 84
  - update\_blockchain\_height, 84
- client.h
  - client\_thread, 47
  - find\_empty\_connection, 47
  - get\_my\_node, 47
  - listen\_to, 47
  - load\_neighbours, 48
  - number\_neighbours, 48
  - print\_neighbours, 48
  - remove\_neighbour, 48
  - save\_neighbours, 48
  - set\_neighbour, 49
- client\_con
  - th\_arg, 19
- client\_connection, 14
  - actual\_client\_height, 15
  - clientfd, 15
  - demand, 15
  - lock, 15
  - network.h, 59
  - Payload, 15
  - Payloadsize, 16
  - thread, 16
- client\_connections
  - client.c, 84, 88
  - client\_test.c, 149
- client\_test.c
  - client\_connections, 149
  - CLIENT\_TEST\_C, 148
  - network\_test, 149
- client\_test.h
  - network\_test, 143
- CLIENT\_TEST\_C

- client\_test.c, 148
- client\_thread
  - client.c, 85
  - client.h, 47
- clientfd
  - client\_connection, 15
- CLIENTMSG
  - network.h, 54
- connection\_to\_others
  - client.c, 83
- connections\_label
  - ui.h, 72
- contacts\_combo
  - ui.c, 125
- convert\_data\_to\_block
  - block.c, 90
- convert\_data\_to\_blockdata
  - block.c, 90
- convert\_data\_to\_transactiondata
  - transaction.c, 96
  - transaction.h, 28
- cr1\_combo
  - ui.c, 125
- cr1\_con
  - ui.c, 125
- cr1\_th
  - ui.c, 125
- cr2\_con
  - ui.c, 125
- cr2\_th
  - ui.c, 126
- cr3\_th
  - ui.c, 126
- create\_account
  - wallet.c, 98
  - wallet.h, 31
- create\_epoch\_block
  - epoch\_man.h, 73
- cx1\_con
  - ui.c, 126
- cx1\_th
  - ui.c, 126
- cx2\_con
  - ui.c, 126
- cx2\_th
  - ui.c, 126
- cx3\_th
  - ui.c, 127
- DD\_GET\_BLOCKS
  - network.h, 54
- DD\_GET\_HEIGHT
  - network.h, 54
- DEBUG
  - tests\_macros.h, 150
- define\_nb\_validators
  - validators.c, 133
- demand
  - client\_connection, 15

- DOORSERVER
  - network.h, 54
- epoch\_id
  - BlockData, 11
- epoch\_man.c
  - get\_epoch\_man\_pkey, 131
- epoch\_man.h
  - create\_epoch\_block, 73
  - flush\_pending\_transactions, 74
  - get\_epoch\_man\_pkey, 74
  - give\_punishments\_and\_rewards, 75
- family
  - Neighbour, 17
- fetch\_client\_list
  - get\_data.c, 111
  - get\_data.h, 50
- files.c
  - \_GNU\_SOURCE, 107
  - last\_file\_in\_folder, 107
- files.h
  - last\_file\_in\_folder, 42
- find\_empty\_connection
  - client.c, 85
  - client.h, 47
- flush\_pending\_transactions
  - epoch\_man.h, 74
- free\_block
  - block.c, 90
- GEN\_BLC\_F\_C
  - GEN\_blockchain\_files.c, 139
- gen\_blockchain
  - GEN\_blockchain\_files.c, 139
- GEN\_blockchain\_files.c
  - GEN\_BLC\_F\_C, 139
  - gen\_blockchain, 139
  - rand\_data, 139
- gen\_blockchain\_header
  - blockchain\_header.c, 95
  - blockchain\_header.h, 26
- gen\_validators\_file
  - GEN\_validators\_file.c, 141
- GEN\_validators\_file.c
  - gen\_validators\_file, 141
  - GEN\_VALIDATORS\_FILE\_H, 140
  - NB\_FAKE\_VALIDATORS, 140
  - str, 140
- GEN\_VALIDATORS\_FILE\_H
  - GEN\_validators\_file.c, 140
- GENESIS\_RSA\_PUB\_1
  - block.c, 89
- GENESIS\_RSA\_PUB\_2
  - block.c, 89
- get\_block
  - block.c, 90
- get\_blockdata\_data
  - block.c, 91

- get\_blocks\_t
  - network.h, 60
- get\_comittee
  - validators.c, 133
  - validators.h, 78
- get\_data.c
  - fetch\_client\_list, 111
  - process\_header, 111
  - read\_actual\_height, 111
  - read\_epoch\_block, 111
  - read\_get\_blocks, 111
  - read\_header, 112
  - read\_pending\_transaction\_list, 112
  - read\_send\_block, 112
  - read\_vote, 112
- get\_data.h
  - fetch\_client\_list, 50
  - read\_actual\_height, 50
  - read\_epoch\_block, 50
  - read\_get\_blocks, 50
  - read\_header, 50
  - read\_pending\_transaction\_list, 52
  - read\_send\_block, 52
  - read\_vote, 52
- get\_epoch\_man\_pkey
  - epoch\_man.c, 131
  - epoch\_man.h, 74
- get\_genesis\_block
  - block.c, 91
- get\_keys
  - rsa.c, 101
  - rsa.h, 34
- get\_keys\_equality\_test
  - rsa\_test.c, 147
  - rsa\_test.h, 142
- get\_keys\_test
  - rsa\_test.c, 147
  - rsa\_test.h, 142
- get\_last\_block\_height
  - block.c, 91
- get\_my\_node
  - client.c, 85
  - client.h, 47
- get\_my\_wallet
  - wallet.c, 98
  - wallet.h, 31
- get\_next\_block
  - block.c, 92
- get\_next\_comittee
  - validators.c, 133
  - validators.h, 78
- get\_prev\_block
  - block.c, 92
- get\_public\_key\_from\_contacts
  - ui.c, 120
  - ui.h, 67
- get\_transaction\_data
  - signature.h, 35
  - transaction.c, 96
  - transaction.h, 29
- get\_validator\_id
  - validators.c, 134
  - validators.h, 79
- get\_validator\_pkey
  - validators.c, 134
  - validators.h, 79
- get\_validator\_power
  - validators.c, 135
  - validators.h, 80
- get\_validator\_stake
  - validators.c, 135
  - validators.h, 80
- get\_validators\_states\_block\_height\_validity
  - validators.c, 135
  - validators.h, 80
- get\_validators\_states\_nb\_validators
  - validators.c, 136
  - validators.h, 81
- get\_validators\_states\_total\_stake
  - validators.c, 136
  - validators.h, 81
- give\_punishments\_and\_rewards
  - epoch\_man.h, 75
- gui.c
  - main, 137
- HARD\_CODED\_ADDR
  - network.c, 113
  - network.h, 60
- hash.c
  - hash\_block\_transactions, 99
  - sha384\_data, 100
- hash.h
  - hash\_block\_transactions, 32
  - sha384\_data, 32
- hash\_block\_transactions
  - hash.c, 99
  - hash.h, 32
- hash\_block\_transactions\_epoch
  - validators.c, 136
- HD\_ACTUAL\_HEIGHT
  - network.h, 54
- HD\_CONNECTION\_TO\_NETWORK
  - network.h, 54
- HD\_CONNECTION\_TO\_NODE
  - network.h, 55
- HD\_GET\_BLOCKS
  - network.h, 55
- HD\_GET\_CLIENT\_LIST
  - network.h, 55
- HD\_GET\_PENDING\_TRANSACTION\_LIST
  - network.h, 55
- HD\_REJECT\_DEMAND
  - network.h, 55
- HD\_SEND\_BLOCK
  - network.h, 55
- HD\_SEND\_CLIENT\_LIST

- network.h, 56
- HD\_SEND\_EPOCH\_BLOCK
  - network.h, 56
- HD\_SEND\_PENDING\_TRANSACTION
  - network.h, 56
- HD\_SEND\_VOTE
  - network.h, 56
- height
  - BlockData, 11
  - blockinfo, 13
- hostname
  - Neighbour, 18
- IM\_CLIENT
  - network.h, 56
- IM\_SERVER
  - network.h, 56
- infos
  - th\_arg, 19
- infos\_st, 16
  - actual\_height, 16
  - is\_synchronize, 17
  - network.h, 59
  - serv\_type, 17
- init\_server
  - server.c, 116
  - server.h, 63
- invest\_entry
  - ui.c, 127
- is\_prev\_block\_valid
  - BlockData, 11
- is\_synchronize
  - infos\_st, 17
- is\_validator
  - Wallet, 23
- join\_network\_door
  - client.c, 83
- key\_entry
  - ui.c, 127
- last\_file\_in\_folder
  - files.c, 107
  - files.h, 42
- latest\_block\_name1
  - ui.c, 127
- latest\_block\_name2
  - ui.c, 127
- latest\_block\_name3
  - ui.c, 127
- listen\_to
  - client.c, 86
  - client.h, 47
- load\_blockchain
  - block.c, 92
- load\_contacts\_from\_file
  - ui.c, 121
  - ui.h, 67
- load\_last\_blockchain
  - block.c, 93
- load\_neighbours
  - client.c, 86
  - client.h, 48
- load\_pending\_transaction
  - transaction.c, 96
  - transaction.h, 29
- load\_transaction
  - transaction.c, 96
  - transaction.h, 29
- load\_transaction\_from\_file
  - ui.h, 67
- load\_transactions\_from\_file
  - ui.c, 121
- lock
  - client\_connection, 15
- LOG
  - tests\_macros.h, 150
- ls\_combo
  - ui.c, 128
- magic
  - BlockData, 11
  - TransactionData, 21
- main
  - client.c, 83
  - gui.c, 137
  - main\_test.c, 145
  - server.c, 117
  - serverdoor.c, 137
  - sign.c, 138
  - unit\_testing.c, 152
- main\_test.c
  - main, 145
  - MAIN\_TEST\_C, 144
- MAIN\_TEST\_C
  - main\_test.c, 144
- MANAGERMSG
  - network.h, 57
- math.h
  - MAX, 43
  - MIN, 43
- MAX
  - math.h, 43
- MAX\_CONNECTION
  - network.h, 57
- MAX\_NEIGHBOURS
  - network.h, 57
- MAX\_SERVER
  - network.h, 57
- MAX\_VALIDATORS\_PER\_BLOCK
  - network.h, 57
  - validators.h, 78
- mempool\_label
  - ui.h, 73
- MIN
  - math.h, 43



- name\_entry\_con
  - ui.c, 128
- NB\_BLOCK\_PER\_CHUNK
  - block\_test.c, 146
- nb\_blocks
  - ChunkBlockchain, 14
- NB\_FAKE\_VALIDATORS
  - GEN\_validators\_file.c, 140
- NB\_HARD\_CODED\_ADDR
  - network.h, 57
- NB MOCK\_BLOCKS
  - block\_test.c, 146
- NB\_RSA\_CHUNK
  - validators.c, 132
- nb\_transactions
  - BlockData, 11
- nb\_validators
  - BlockData, 12
- Neighbour, 17
  - family, 17
  - hostname, 18
  - network.h, 59
- neighbours
  - Node, 18
- network.c
  - HARD\_CODED\_ADDR, 113
- network.h
  - \_\_attribute\_\_, 60
  - client\_connection, 59
  - CLIENTMSG, 54
  - DD\_GET\_BLOCKS, 54
  - DD\_GET\_HEIGHT, 54
  - DOORSERVER, 54
  - get\_blocks\_t, 60
  - HARD\_CODED\_ADDR, 60
  - HD\_ACTUAL\_HEIGHT, 54
  - HD\_CONNECTION\_TO\_NETWORK, 54
  - HD\_CONNECTION\_TO\_NODE, 55
  - HD\_GET\_BLOCKS, 55
  - HD\_GET\_CLIENT\_LIST, 55
  - HD\_GET\_PENDING\_TRANSACTION\_LIST, 55
  - HD\_REJECT\_DEMAND, 55
  - HD\_SEND\_BLOCK, 55
  - HD\_SEND\_CLIENT\_LIST, 56
  - HD\_SEND\_EPOCH\_BLOCK, 56
  - HD\_SEND\_PENDING\_TRANSACTION, 56
  - HD\_SEND\_VOTE, 56
  - IM\_CLIENT, 56
  - IM\_SERVER, 56
  - infos\_st, 59
  - MANAGERMSG, 57
  - MAX\_CONNECTION, 57
  - MAX\_NEIGHBOURS, 57
  - MAX\_SERVER, 57
  - MAX\_VALIDATORS\_PER\_BLOCK, 57
  - NB\_HARD\_CODED\_ADDR, 57
  - Neighbour, 59
  - Node, 59
  - NODESERVER, 58
  - P\_VERSION, 58
  - SERVERMSG, 58
  - SIZE\_OF\_HOSTNAME, 58
  - SOL\_TCP, 58
  - STATIC\_PORT, 58
  - TCP\_USER\_TIMEOUT, 59
  - th\_arg, 60
  - WARNINGMSG, 59
- network\_test
  - client\_test.c, 149
  - client\_test.h, 143
- Node, 18
  - neighbours, 18
  - network.h, 59
- NODESERVER
  - network.h, 58
- number\_neighbours
  - client.c, 86
  - client.h, 48
- on\_add\_contact\_button1\_press
  - ui.c, 121
  - ui.h, 67
- on\_connect\_but\_press
  - ui.c, 121
  - ui.h, 68
- on\_create\_key\_but1\_press
  - ui.c, 121
  - ui.h, 68
- on\_create\_key\_but2\_press
  - ui.c, 122
  - ui.h, 68
- on\_invest\_button1\_press
  - ui.c, 122
  - ui.h, 68
- on\_invest\_button2\_press
  - ui.c, 122
  - ui.h, 69
- on\_main\_window\_delete
  - ui.c, 122
  - ui.h, 69
- on\_main\_window\_destroy
  - ui.c, 123
  - ui.h, 69
- on\_pkey\_button\_press
  - ui.c, 123
  - ui.h, 70
- on\_recover\_button1\_press
  - ui.c, 123
  - ui.h, 70
- on\_recover\_button2\_press
  - ui.c, 123
  - ui.h, 70
- on\_transaction\_button\_press
  - ui.c, 123
  - ui.h, 71
- organisation\_public\_key
  - TransactionData, 21

- P\_VERSION
  - network.h, [58](#)
- password\_entry1
  - ui.c, [128](#)
- password\_entry2
  - ui.c, [128](#)
- password\_error\_label
  - ui.c, [128](#)
- Payload
  - client\_connection, [15](#)
- Payloadsize
  - client\_connection, [16](#)
- pop\_stake
  - validators.h, [81](#)
- prev\_validators\_votes
  - BlockData, [12](#)
- previous\_block\_hash
  - BlockData, [12](#)
- print\_neighbours
  - client.c, [87](#)
  - client.h, [48](#)
- priv\_key
  - Wallet, [23](#)
- private\_key\_label
  - ui.c, [128](#)
- process\_header
  - get\_data.c, [111](#)
- progress\_bar\_blockchain
  - ui.c, [129](#)
- pub\_key
  - Wallet, [23](#)
- public\_key\_entry\_con
  - ui.c, [129](#)
- push\_stake
  - validators.h, [82](#)
- rand\_data
  - GEN\_blockchain\_files.c, [139](#)
- read\_actual\_height
  - get\_data.c, [111](#)
  - get\_data.h, [50](#)
- read\_epoch\_block
  - get\_data.c, [111](#)
  - get\_data.h, [50](#)
- read\_get\_blocks
  - get\_data.c, [111](#)
  - get\_data.h, [50](#)
- read\_header
  - get\_data.c, [112](#)
  - get\_data.h, [50](#)
- read\_pending\_transaction\_list
  - get\_data.c, [112](#)
  - get\_data.h, [52](#)
- read\_send\_block
  - get\_data.c, [112](#)
  - get\_data.h, [52](#)
- read\_vote
  - get\_data.c, [112](#)
  - get\_data.h, [52](#)
- receiver\_public\_key
  - TransactionData, [21](#)
- receiver\_remaining\_money
  - TransactionData, [22](#)
- recipient\_key
  - ui.c, [129](#)
- recover\_entry
  - ui.c, [129](#)
- redirect\_connection
  - server.c, [116](#)
- remove\_neighbour
  - client.c, [87](#)
  - client.h, [48](#)
- rsa.c
  - get\_keys, [101](#)
  - RSA\_NUM\_E, [101](#)
- rsa.h
  - get\_keys, [34](#)
  - RSA\_BEGIN\_SIZE, [33](#)
  - RSA\_END\_SIZE, [33](#)
  - RSA\_FILE\_TOTAL\_SIZE, [34](#)
  - RSA\_KEY\_SIZE, [34](#)
- RSA\_BEGIN\_SIZE
  - rsa.h, [33](#)
- RSA\_END\_SIZE
  - rsa.h, [33](#)
- RSA\_FILE\_TOTAL\_SIZE
  - rsa.h, [34](#)
- RSA\_KEY\_SIZE
  - rsa.h, [34](#)
- RSA\_NUM\_E
  - rsa.c, [101](#)
- RSA\_SIZE\_C
  - rsa\_test.c, [147](#)
- rsa\_test.c
  - get\_keys\_equality\_test, [147](#)
  - get\_keys\_test, [147](#)
  - RSA\_SIZE\_C, [147](#)
- rsa\_test.h
  - get\_keys\_equality\_test, [142](#)
  - get\_keys\_test, [142](#)
- safe.c
  - safe\_fread, [108](#)
  - safe\_read, [109](#)
  - safe\_send, [109](#)
  - safe\_write, [110](#)
- safe.h
  - safe\_fread, [44](#)
  - safe\_read, [44](#)
  - safe\_send, [45](#)
  - safe\_write, [45](#)
- safe\_fread
  - safe.c, [108](#)
  - safe.h, [44](#)
- safe\_read
  - safe.c, [109](#)
  - safe.h, [44](#)
- safe\_send

- safe.c, 109
- safe.h, 45
- safe\_write
  - safe.c, 110
  - safe.h, 45
- save\_neighbours
  - client.c, 87
  - client.h, 48
- send\_actual\_height
  - send\_data.c, 114
  - send\_data.h, 61
- send\_client\_list
  - send\_data.c, 114
  - send\_data.h, 61
- send\_data.c
  - send\_actual\_height, 114
  - send\_client\_list, 114
  - send\_get\_blocks, 114
  - send\_pending\_transaction\_list, 115
  - send\_reject\_demand, 115
  - send\_send\_block, 115
- send\_data.h
  - send\_actual\_height, 61
  - send\_client\_list, 61
  - send\_get\_blocks, 62
  - send\_pending\_transaction\_list, 62
  - send\_reject\_demand, 62
  - send\_send\_block, 62
- send\_get\_blocks
  - send\_data.c, 114
  - send\_data.h, 62
- send\_money
  - transaction.h, 29
- send\_pending\_transaction\_list
  - send\_data.c, 115
  - send\_data.h, 62
- send\_reject\_demand
  - send\_data.c, 115
  - send\_data.h, 62
- send\_send\_block
  - send\_data.c, 115
  - send\_data.h, 62
- send\_verdict
  - validation\_engine.h, 75
- sender\_public\_key
  - TransactionData, 22
- sender\_remaining\_money
  - TransactionData, 22
- serv\_type
  - infos\_st, 17
- server.c
  - accept\_connection, 116
  - init\_server, 116
  - main, 117
  - redirect\_connection, 116
- server.h
  - init\_server, 63
- serverdoor.c
  - main, 137
- SERVERMSG
  - network.h, 58
- set\_neighbour
  - client.c, 87
  - client.h, 49
- setup
  - ui.c, 124
  - ui.h, 71
- sha384\_data
  - hash.c, 100
  - hash.h, 32
- sign.c
  - main, 138
- sign\_block
  - signature.c, 102
  - signature.h, 36
- sign\_block\_transactions
  - signature.c, 102
  - signature.h, 36
- sign\_block\_with\_key
  - signature.c, 103
  - signature.h, 36
- sign\_message
  - signature.c, 103
  - signature.h, 36
- sign\_message\_with\_key
  - signature.c, 103
  - signature.h, 38
- sign\_transaction
  - signature.c, 105
  - signature.h, 38
- sign\_transaction\_with\_key
  - signature.c, 105
  - signature.h, 38
- signature.c
  - sign\_block, 102
  - sign\_block\_transactions, 102
  - sign\_block\_with\_key, 103
  - sign\_message, 103
  - sign\_message\_with\_key, 103
  - sign\_transaction, 105
  - sign\_transaction\_with\_key, 105
  - verify\_block\_signature, 105
  - verify\_signature, 106
  - verify\_transaction\_signature, 106
- signature.h
  - get\_transaction\_data, 35
  - sign\_block, 36
  - sign\_block\_transactions, 36
  - sign\_block\_with\_key, 36
  - sign\_message, 36
  - sign\_message\_with\_key, 38
  - sign\_transaction, 38
  - sign\_transaction\_with\_key, 38
  - verify\_block\_signature, 39
  - verify\_signature, 39
  - verify\_transaction\_signature, 40

- write\_block, 40
  - write\_blockdata, 40
- signature\_test.c
  - verify\_sign\_test, 148
- signature\_test.h
  - verify\_sign\_test, 143
- SIZE\_OF\_HOSTNAME
  - network.h, 58
- SOL\_TCP
  - network.h, 58
- stake\_label1
  - ui.c, 129
- stake\_label2
  - ui.c, 129
- stake\_label3
  - ui.c, 130
- STATIC\_PORT
  - network.h, 58
- str
  - GEN\_validators\_file.c, 140
- synchro\_label
  - ui.h, 73
- T\_TYPE\_ADD\_STAKE
  - transaction.h, 27
- T\_TYPE\_DEFAULT
  - transaction.h, 27
- T\_TYPE\_STAKE\_TO\_STAKE
  - transaction.h, 27
- T\_TYPE\_WITHDRAW\_STAKE
  - transaction.h, 27
- TCP\_USER\_TIMEOUT
  - network.h, 59
- TEST\_FAILED
  - tests\_macros.h, 150
- TEST\_PASSED
  - tests\_macros.h, 151
- TEST\_WARNING
  - tests\_macros.h, 151
- tests\_macros.h
  - DEBUG, 150
  - LOG, 150
  - TEST\_FAILED, 150
  - TEST\_PASSED, 151
  - TEST\_WARNING, 151
- th\_arg, 19
  - client\_con, 19
  - infos, 19
  - network.h, 60
- thread
  - client\_connection, 16
- transa\_amount
  - ui.c, 130
- Transaction, 19
  - transaction.h, 28
  - transaction\_data, 20
  - transaction\_signature, 20
- transaction.c
  - add\_pending\_transaction, 95
  - convert\_data\_to\_transactiondata, 96
  - get\_transaction\_data, 96
  - load\_pending\_transaction, 96
  - load\_transaction, 96
  - write\_transaction, 97
  - write\_transactiondata, 97
- transaction.h
  - add\_pending\_transaction, 28
  - convert\_data\_to\_transactiondata, 28
  - get\_transaction\_data, 29
  - load\_pending\_transaction, 29
  - load\_transaction, 29
  - send\_money, 29
  - T\_TYPE\_ADD\_STAKE, 27
  - T\_TYPE\_DEFAULT, 27
  - T\_TYPE\_STAKE\_TO\_STAKE, 27
  - T\_TYPE\_WITHDRAW\_STAKE, 27
  - Transaction, 28
  - TRANSACTION\_DATA\_SIZE, 27
  - TRANSACTION\_SIZE, 28
  - TransactionData, 28
  - write\_transaction, 30
  - write\_transactiondata, 30
- transaction\_data
  - Transaction, 20
- TRANSACTION\_DATA\_SIZE
  - transaction.h, 27
- transaction\_signature
  - Transaction, 20
- TRANSACTION\_SIZE
  - transaction.h, 28
- transaction\_timestamp
  - TransactionData, 22
- TransactionData, 20
  - amount, 21
  - asset, 21
  - cause, 21
  - magic, 21
  - organisation\_public\_key, 21
  - receiver\_public\_key, 21
  - receiver\_remaining\_money, 22
  - sender\_public\_key, 22
  - sender\_remaining\_money, 22
  - transaction.h, 28
  - transaction\_timestamp, 22
  - type, 22
- transactions
  - BlockData, 12
  - blockinfo, 13
- ts\_con
  - ui.c, 130
- ts\_th
  - ui.c, 130
- tv\_con
  - ui.c, 130
- tv\_th
  - ui.c, 130
- type

- TransactionData, 22
- ui.c
  - add\_contact, 119
  - add\_contact\_to\_combobox, 119
  - add\_contacts\_from\_file, 119
  - add\_new\_blockinfo, 119
  - add\_transaction\_from\_file, 120
  - add\_transaction\_with\_contact, 120
  - add\_transaction\_with\_pkey, 120
  - balance\_1, 124
  - balance\_2, 125
  - change\_label\_text, 120
  - contacts\_combo, 125
  - cr1\_combo, 125
  - cr1\_con, 125
  - cr1\_th, 125
  - cr2\_con, 125
  - cr2\_th, 126
  - cr3\_th, 126
  - cx1\_con, 126
  - cx1\_th, 126
  - cx2\_con, 126
  - cx2\_th, 126
  - cx3\_th, 127
  - get\_public\_key\_from\_contacts, 120
  - invest\_entry, 127
  - key\_entry, 127
  - latest\_block\_name1, 127
  - latest\_block\_name2, 127
  - latest\_block\_name3, 127
  - load\_contacts\_from\_file, 121
  - load\_transactions\_from\_file, 121
  - ls\_combo, 128
  - name\_entry\_con, 128
  - on\_add\_contact\_button1\_press, 121
  - on\_connect\_but\_press, 121
  - on\_create\_key\_but1\_press, 121
  - on\_create\_key\_but2\_press, 122
  - on\_invest\_button1\_press, 122
  - on\_invest\_button2\_press, 122
  - on\_main\_window\_delete, 122
  - on\_main\_window\_destroy, 123
  - on\_pkey\_button\_press, 123
  - on\_recover\_button1\_press, 123
  - on\_recover\_button2\_press, 123
  - on\_transaction\_button\_press, 123
  - password\_entry1, 128
  - password\_entry2, 128
  - password\_error\_label, 128
  - private\_key\_label, 128
  - progress\_bar\_blockchain, 129
  - public\_key\_entry\_con, 129
  - recipient\_key, 129
  - recover\_entry, 129
  - setup, 124
  - stake\_label1, 129
  - stake\_label2, 129
  - stake\_label3, 130
- transa\_amount, 130
- ts\_con, 130
- ts\_th, 130
- tv\_con, 130
- tv\_th, 130
- update\_labels, 124
- update\_sync, 124
- ui.h
  - add\_contact, 65
  - add\_contact\_to\_combobox, 65
  - add\_contacts\_from\_file, 65
  - add\_new\_blockinfo, 66
  - add\_transaction\_from\_file, 66
  - add\_transaction\_with\_contact, 66
  - add\_transaction\_with\_pkey, 66
  - block\_amount\_label, 72
  - blocksinfo, 72
  - change\_label\_text, 66
  - connections\_label, 72
  - get\_public\_key\_from\_contacts, 67
  - load\_contacts\_from\_file, 67
  - load\_transaction\_from\_file, 67
  - mempool\_label, 73
  - on\_add\_contact\_button1\_press, 67
  - on\_connect\_but\_press, 68
  - on\_create\_key\_but1\_press, 68
  - on\_create\_key\_but2\_press, 68
  - on\_invest\_button1\_press, 68
  - on\_invest\_button2\_press, 69
  - on\_main\_window\_delete, 69
  - on\_main\_window\_destroy, 69
  - on\_pkey\_button\_press, 70
  - on\_recover\_button1\_press, 70
  - on\_recover\_button2\_press, 70
  - on\_transaction\_button\_press, 71
  - setup, 71
  - synchro\_label, 73
  - update\_labels, 72
  - update\_sync, 72
- unit\_testing.c
  - main, 152
- update\_blockchain
  - client.c, 84
- update\_blockchain\_height
  - client.c, 84
- update\_labels
  - ui.c, 124
  - ui.h, 72
- update\_sync
  - ui.c, 124
  - ui.h, 72
- validate\_transactions
  - validation\_engine.h, 76
- validation\_engine.h
  - send\_verdict, 75
  - validate\_transactions, 76
- validations\_test
  - validations\_test.c, 149

- validations\_test.h, 144
- validations\_test.c
  - validations\_test, 149
- validations\_test.h
  - validations\_test, 144
- validators.c
  - define\_nb\_validators, 133
  - get\_comittee, 133
  - get\_next\_comittee, 133
  - get\_validator\_id, 134
  - get\_validator\_pkey, 134
  - get\_validator\_power, 135
  - get\_validator\_stake, 135
  - get\_validators\_states\_block\_height\_validity, 135
  - get\_validators\_states\_nb\_validators, 136
  - get\_validators\_states\_total\_stake, 136
  - hash\_block\_transactions\_epoch, 136
  - NB\_RSA\_CHUNK, 132
- validators.h
  - get\_comittee, 78
  - get\_next\_comittee, 78
  - get\_validator\_id, 79
  - get\_validator\_pkey, 79
  - get\_validator\_power, 80
  - get\_validator\_stake, 80
  - get\_validators\_states\_block\_height\_validity, 80
  - get\_validators\_states\_nb\_validators, 81
  - get\_validators\_states\_total\_stake, 81
  - MAX\_VALIDATORS\_PER\_BLOCK, 78
  - pop\_stake, 81
  - push\_stake, 82
- validators\_public\_keys
  - BlockData, 12
- validators\_votes
  - Block, 10
- verify\_block\_signature
  - signature.c, 105
  - signature.h, 39
- verify\_sign\_test
  - signature\_test.c, 148
  - signature\_test.h, 143
- verify\_signature
  - signature.c, 106
  - signature.h, 39
- verify\_transaction\_signature
  - signature.c, 106
  - signature.h, 40
- vote\_signature
  - Block, 10
- Wallet, 23
  - amount, 23
  - is\_validator, 23
  - priv\_key, 23
  - pub\_key, 23
  - wallet.h, 31
- wallet.c
  - create\_account, 98
  - get\_my\_wallet, 98
- wallet.h
  - create\_account, 31
  - get\_my\_wallet, 31
  - Wallet, 31
- WARNINGMSG
  - network.h, 59
- write\_block
  - block.c, 93
  - signature.h, 40
- write\_block\_file
  - block.c, 94
- write\_block\_header
  - blockchain\_header.c, 95
- write\_blockdata
  - block.c, 94
  - signature.h, 40
- write\_transaction
  - transaction.c, 97
  - transaction.h, 30
- write\_transactiondata
  - transaction.c, 97
  - transaction.h, 30