CertiK Audit Report For Rupiah Token (rupiahtoken.com) - IDRT

Request Date: 2019-03-27
Revision Date: 2019-07-03

Contents

Disclaimer

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

For more information: https://certik.org/

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by Rupiah Token (rupiahtoken.com) - IDRT. This audit was conducted to discover issues and vulnerabilities in the source code of Rupiah Token (rupiahtoken.com) - IDRT's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.

Low

The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Jul 03, 2019

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

“tx.origin” for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 51 in File ERC20RupiahTokenV1.sol

```
pragma solidity ^0.4.25;
```

! Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

INSECURE_COMPILER_VERSION

Line 25 in File Pausable.sol

```
pragma solidity ^0.4.25;
```

! Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

INSECURE_COMPILER_VERSION

Line 24 in File SafeMath.sol

```
pragma solidity ^0.4.25;
```

! Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

INSECURE_COMPILER_VERSION

Line 24 in File Ownable.sol

```
pragma solidity ^0.4.25;
```

! Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

INSECURE_COMPILER_VERSION

Line 60 in File Blacklistable.sol

```
pragma solidity ^0.4.25;
```

! Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIV2

INSECURE_COMPILER_VERSION

Line 60 in File IDRTWalletV1.sol

```
pragma solidity ^0.4.25;
```

❗ Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

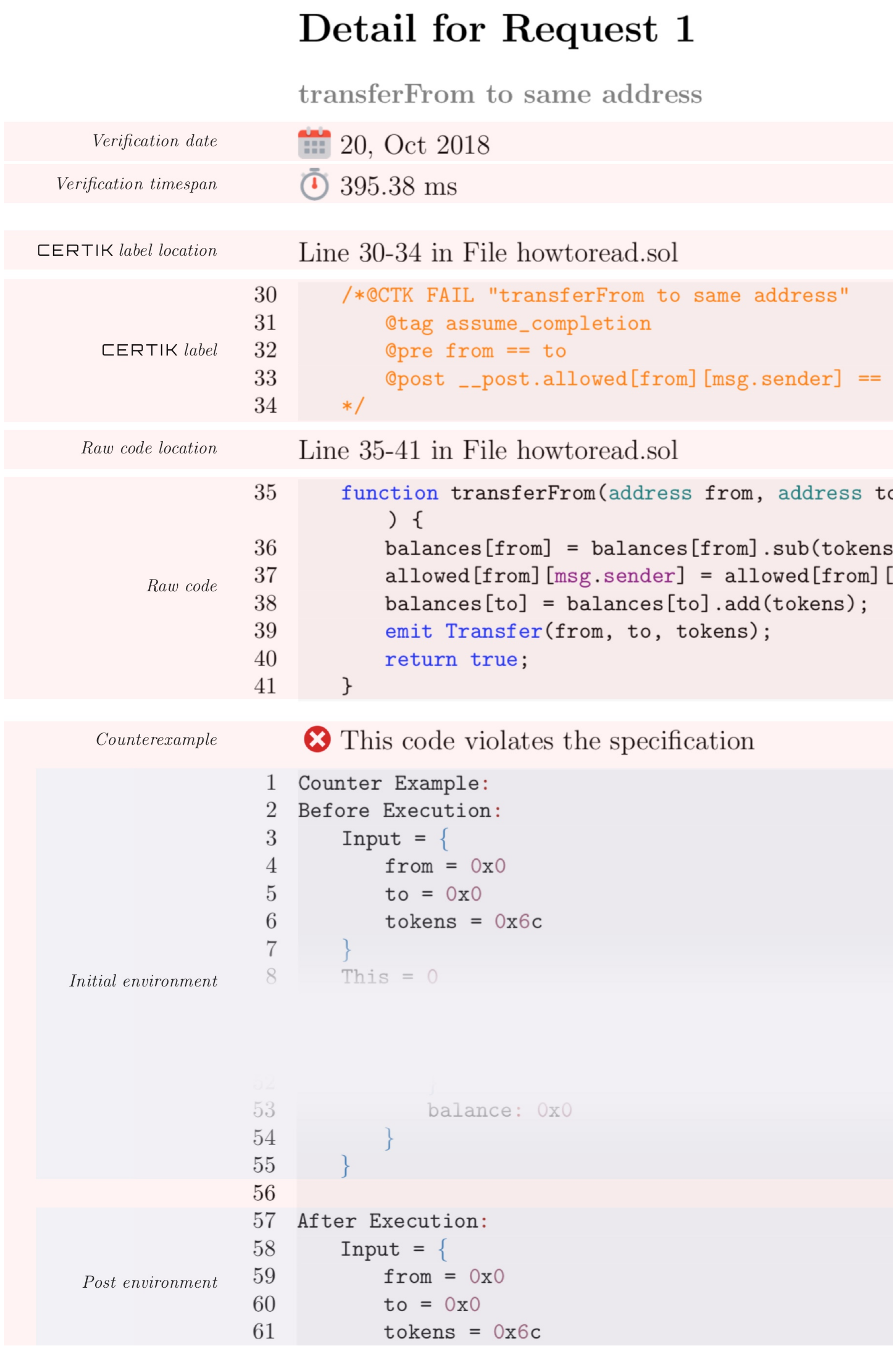
Line 44 in File MultiSigWallet.sol

```
pragma solidity ^0.4.25;
```

❗ Version to compile has the following bug: 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

Formal Verification Results

How to read



Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms
CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address tc
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample This code violates the specification

Initial environment

```
 1  Counter Example:
 2  Before Execution:
 3      Input = {
 4          from = 0x0
 5          to = 0x0
 6          tokens = 0x6c
 7      }
 8      This = 0

53              balance: 0x0
54          }
55      }
56
```

Post environment

```
57  After Execution:
58      Input = {
59          from = 0x0
60          to = 0x0
61          tokens = 0x6c
```

Formal Verification Request 1

initialize

📅 03, Jul 2019
⏱ 68.79 ms

Line 81-87 in File ERC20RupiahTokenV1.sol

```
    /*@CTK initialize
      @post __post.owner == msg.sender
      @post __post._name == name
      @post __post._symbol == symbol
      @post __post._currency == currency
      @post __post._decimals == decimals
     */
```

Line 88-94 in File ERC20RupiahTokenV1.sol

```
    function initialize(string name, string symbol, string currency, uint8 decimals)
        initializer public {
  owner = msg.sender;
        _name = name;
        _symbol = symbol;
        _currency = currency;
        _decimals = decimals;
    }
```

✅ The code meets the specification.

Formal Verification Request 2

name

📅 03, Jul 2019
⏱ 4.67 ms

Line 99-101 in File ERC20RupiahTokenV1.sol

```
    /*@CTK name
      @post __return == _name
     */
```

Line 102-104 in File ERC20RupiahTokenV1.sol

```
    function name() public view returns (string memory) {
        return _name;
    }
```

✅ The code meets the specification.

Formal Verification Request 3

symbol

📅 03, Jul 2019
⏱ 4.58 ms

Line 109-111 in File ERC20RupiahTokenV1.sol

```
/*@CTK symbol
  @post __return == _symbol
 */
```

Line 112-114 in File ERC20RupiahTokenV1.sol

```
function symbol() public view returns (string memory) {
    return _symbol;
}
```

✓ The code meets the specification.

Formal Verification Request 4

currency

03, Jul 2019
4.43 ms

Line 119-121 in File ERC20RupiahTokenV1.sol

```
/*@CTK currency
  @post __return == _currency
 */
```

Line 122-124 in File ERC20RupiahTokenV1.sol

```
function currency() public view returns (string memory) {
    return _currency;
}
```

✓ The code meets the specification.

Formal Verification Request 5

decimals

03, Jul 2019
4.1 ms

Line 129-131 in File ERC20RupiahTokenV1.sol

```
/*@CTK decimals
  @post __return == _decimals
 */
```

Line 132-134 in File ERC20RupiahTokenV1.sol

```
function decimals() public view returns (uint8) {
    return _decimals;
}
```

✓ The code meets the specification.

Formal Verification Request 6

totalSupply

03, Jul 2019
4.5 ms

Line 139-141 in File ERC20RupiahTokenV1.sol

```
    /*@CTK totalSupply
      @post __return == _totalSupply
     */
```

Line 142-144 in File ERC20RupiahTokenV1.sol

```
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }
```

✓ The code meets the specification.

Formal Verification Request 7

balanceOf

03, Jul 2019
5.13 ms

Line 151-153 in File ERC20RupiahTokenV1.sol

```
    /*@CTK balanceOf
      @post __return == _balances[owner]
     */
```

Line 154-156 in File ERC20RupiahTokenV1.sol

```
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }
```

✓ The code meets the specification.

Formal Verification Request 8

allowance

03, Jul 2019
4.58 ms

Line 164-166 in File ERC20RupiahTokenV1.sol

```
    /*@CTK allowance
      @post __return == _allowed[owner][spender]
     */
```

Line 167-169 in File ERC20RupiahTokenV1.sol

```
function allowance(address owner, address spender) public view returns (uint256) {
    return _allowed[owner][spender];
}
```

✓ The code meets the specification.

Formal Verification Request 9

transfer

📅 03, Jul 2019
⏱ 286.25 ms

Line 176-185 in File ERC20RupiahTokenV1.sol

```
/*@CTK transfer
  @tag assume_completion
  @pre msg.sender != to
  @post _paused == false
  @post blacklisted[msg.sender] == false
  @post blacklisted[to] == false
  @post to != address(0)
  @post __post._balances[msg.sender] == _balances[msg.sender] - value
  @post __post._balances[to] == _balances[to] + value
 */
```

Line 186-194 in File ERC20RupiahTokenV1.sol

```
function transfer(address to, uint256 value) public whenNotPaused notBlacklisted(
    msg.sender) notBlacklisted(to) returns (bool) {
    require(to != address(0));

    _balances[msg.sender] = _balances[msg.sender].sub(value);
    _balances[to] = _balances[to].add(value);
    emit Transfer(msg.sender, to, value);

    return true;
}
```

✓ The code meets the specification.

Formal Verification Request 10

transferFrom

📅 03, Jul 2019
⏱ 518.88 ms

Line 218-228 in File ERC20RupiahTokenV1.sol

```
/*@CTK transferFrom
  @tag assume_completion
  @pre from != to
  @post _paused == false
  @post blacklisted[msg.sender] == false
  @post blacklisted[from] == false
```

```
      @post blacklisted[to] == false
      @post to != address(0)
      @post __post._balances[from] == _balances[from] - value
      @post __post._balances[to] == _balances[to] + value
     */
```

Line 229-238 in File ERC20RupiahTokenV1.sol

```
    function transferFrom(address from, address to, uint256 value) public
        whenNotPaused notBlacklisted(msg.sender) notBlacklisted(from) notBlacklisted(
        to) returns (bool) {
        require(to != address(0));

        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 11

increaseAllowance

📅 03, Jul 2019
⏱ 159.94 ms

Line 250-256 in File ERC20RupiahTokenV1.sol

```
    /*@CTK increaseAllowance
      @tag assume_completion
      @post _paused == false
      @post !blacklisted[msg.sender]
      @post !blacklisted[spender]
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
     */
```

Line 257-260 in File ERC20RupiahTokenV1.sol

```
    function increaseAllowance(address spender, uint256 addedValue) public
        whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
        ) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 12

decreaseAllowance

📅 03, Jul 2019
⏱ 136.66 ms

Line 272-278 in File ERC20RupiahTokenV1.sol

```
/*@CTK decreaseAllowance
  @tag assume_completion
  @post _paused == false
  @post !blacklisted[msg.sender]
  @post !blacklisted[spender]
  @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
      subtractedValue
 */
```

Line 279-282 in File ERC20RupiahTokenV1.sol

```
function decreaseAllowance(address spender, uint256 subtractedValue) public
    whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
    ) {
    _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
        ));
    return true;
}
```

✓ The code meets the specification.

Formal Verification Request 13

_approve

03, Jul 2019
4.74 ms

Line 337-342 in File ERC20RupiahTokenV1.sol

```
/*@CTK _approve
  @tag assume_completion
  @post spender != address(0)
  @post owner != address(0)
  @post __post._allowed[owner][spender] == value
 */
```

Line 343-349 in File ERC20RupiahTokenV1.sol

```
function _approve(address owner, address spender, uint256 value) internal {
    require(spender != address(0));
    require(owner != address(0));

    _allowed[owner][spender] = value;
    emit Approval(owner, spender, value);
}
```

✓ The code meets the specification.

Formal Verification Request 14

paused

03, Jul 2019
5.74 ms

Line 42-44 in File Pausable.sol

```
    /*@CTK paused
      @post __return == _paused
     */
```

Line 45-47 in File Pausable.sol

```
    function paused() public view returns (bool) {
        return _paused;
    }
```

✓ The code meets the specification.

Formal Verification Request 15

pause

03, Jul 2019
12.28 ms

Line 68-72 in File Pausable.sol

```
    /*@CTK pause
      @tag assume_completion
      @post owner == msg.sender
      @post __post._paused == true
     */
```

Line 73-76 in File Pausable.sol

```
    function pause() public onlyOwner {
        _paused = true;
        emit Paused(msg.sender);
    }
```

✓ The code meets the specification.

Formal Verification Request 16

unpause

03, Jul 2019
13.61 ms

Line 81-85 in File Pausable.sol

```
    /*@CTK unpause
      @tag assume_completion
      @post owner == msg.sender
      @post __post._paused == false
     */
```

Line 86-89 in File Pausable.sol

```
    function unpause() public onlyOwner {
        _paused = false;
        emit Unpaused(msg.sender);
    }
```

✓ The code meets the specification.

Formal Verification Request 17

SafeMath_mul

📅 03, Jul 2019
⏱ 311.59 ms

Line 34-41 in File SafeMath.sol

```
    /*@CTK SafeMath_mul
      @post __reverted == __has_overflow
      @post __reverted == false -> __return == a * b
      @post a == 0 -> __return == 0
      @post msg == msg__post
      @post (a > 0 && (a * b / a != b)) == __reverted
      @post __addr_map == __addr_map__post
    */
```

Line 42-54 in File SafeMath.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 18

SafeMath div

📅 03, Jul 2019
⏱ 11.64 ms

Line 59-63 in File SafeMath.sol

```
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
    */
```

Line 64-71 in File SafeMath.sol

```
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // Solidity only automatically asserts when dividing by 0
    require(b > 0);
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold

    return c;
}
```

✓ The code meets the specification.

Formal Verification Request 19

SafeMath sub

📅 03, Jul 2019
⏱ 10.48 ms

Line 76-80 in File SafeMath.sol

```
/*@CTK "SafeMath sub"
  @post (a < b) == __reverted
  @post !__reverted -> __return == a - b
  @post !__reverted -> !__has_overflow
*/
```

Line 81-86 in File SafeMath.sol

```
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b <= a);
    uint256 c = a - b;

    return c;
}
```

✓ The code meets the specification.

Formal Verification Request 20

SafeMath_add

📅 03, Jul 2019
⏱ 12.84 ms

Line 91-97 in File SafeMath.sol

```
/*@CTK SafeMath_add
  @post __reverted == __has_overflow
  @post __reverted == false -> __return == a + b
  @post msg == msg__post
  @post (a + b < a) == __has_overflow
  @post __addr_map == __addr_map__post
 */
```

Line 98-103 in File SafeMath.sol

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a);

    return c;
}
```

✓ The code meets the specification.

Formal Verification Request 21

SafeMath_mod

03, Jul 2019
10.74 ms

Line 109-112 in File SafeMath.sol

```
/*@CTK SafeMath_mod
  @tag assume_completion
  @post __return == a % b
 */
```

Line 113-116 in File SafeMath.sol

```
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b != 0);
    return a % b;
}
```

✓ The code meets the specification.

Formal Verification Request 22

Ownable

03, Jul 2019
4.83 ms

Line 43-45 in File Ownable.sol

```
/*@CTK Ownable
  @post __post.owner == msg.sender
 */
```

Line 46-48 in File Ownable.sol

```
constructor() public {
  owner = msg.sender;
}
```

✓ The code meets the specification.

Formal Verification Request 23

renounceOwnership

📅 03, Jul 2019
⏱ 13.9 ms

Line 64-68 in File Ownable.sol

```
/*@CTK renounceOwnership
  @tag assume_completion
  @post __post.owner == address(0)
  @post owner == msg.sender
 */
```

Line 69-72 in File Ownable.sol

```
function renounceOwnership() public onlyOwner {
  owner = address(0);
  emit OwnershipTransferred(msg.sender, owner);
}
```

✅ The code meets the specification.

Formal Verification Request 24

transferOwnership

📅 03, Jul 2019
⏱ 42.94 ms

Line 78-83 in File Ownable.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @post owner == msg.sender
  @post _newOwner != address(0)
  @post __post.owner == _newOwner
 */
```

Line 84-86 in File Ownable.sol

```
function transferOwnership(address _newOwner) public onlyOwner {
  _transferOwnership(_newOwner);
}
```

✅ The code meets the specification.

Formal Verification Request 25

_transferOwnership

📅 03, Jul 2019
⏱ 1.42 ms

Line 92-96 in File Ownable.sol

```
/*@CTK _transferOwnership
  @tag assume_completion
  @post _newOwner != address(0)
  @post __post.owner == _newOwner
 */
```

Line 97-101 in File Ownable.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  owner = _newOwner;
  emit OwnershipTransferred(owner, _newOwner);
}
```

✓ The code meets the specification.

Formal Verification Request 26

isBlacklisted

03, Jul 2019
4.68 ms

Line 87-89 in File Blacklistable.sol

```
  /*@CTK isBlacklisted
    @post __return == blacklisted[_account]
   */
```

Line 90-92 in File Blacklistable.sol

```
  function isBlacklisted(address _account) public view returns (bool) {
      return blacklisted[_account];
  }
```

✓ The code meets the specification.

Formal Verification Request 27

blacklist

03, Jul 2019
20.93 ms

Line 98-103 in File Blacklistable.sol

```
  /*@CTK blacklist
    @tag assume_completion
    @post owner == msg.sender
    @post _paused == false
    @post __post.blacklisted[_account]
   */
```

Line 104-107 in File Blacklistable.sol

```
  function blacklist(address _account) public onlyOwner whenNotPaused {
      blacklisted[_account] = true;
      emit Blacklisted(_account);
  }
```

✓ The code meets the specification.

Formal Verification Request 28

unblacklist

03, Jul 2019
23.3 ms

Line 113-118 in File Blacklistable.sol

```
/*@CTK unblacklist
  @tag assume_completion
  @post owner == msg.sender
  @post _paused == false
  @post __post.blacklisted[_account] == false
 */
```

Line 119-122 in File Blacklistable.sol

```
function unblacklist(address _account) public onlyOwner whenNotPaused {
    blacklisted[_account] = false;
    emit Unblacklisted(_account);
}
```

✓ The code meets the specification.

Formal Verification Request 29

setPrintLimit

03, Jul 2019
14.42 ms

Line 200-204 in File IDRTWalletV1.sol

```
/*@CTK setPrintLimit
  @tag assume_completion
  @post msg.sender == _superOwner
  @post __post._printLimit == newLimit
 */
```

Line 205-211 in File IDRTWalletV1.sol

```
function setPrintLimit(uint256 newLimit)
    public
    onlySuperOwner()
{
    emit PrintLimitChanged(_printLimit, newLimit);
    _printLimit = newLimit;
}
```

✓ The code meets the specification.

Formal Verification Request 30

transferOwnership

📅 03, Jul 2019
⏱ 21.11 ms

Line 217-221 in File IDRTWalletV1.sol

```
    /*@CTK transferOwnership
      @tag assume_completion
      @post msg.sender == _superOwner
      @post newAddress != address(0)
     */
```

Line 222-230 in File IDRTWalletV1.sol

```
    function transferOwnership(address newAddress)
        public
        onlySuperOwner()
    {
        require(newAddress != address(0));

        _superOwner = newAddress;
        emit OwnershipTransferred(msg.sender, newAddress);
    }
```

✓ The code meets the specification.

Formal Verification Request 31

superOwner

📅 03, Jul 2019
⏱ 4.84 ms

Line 235-237 in File IDRTWalletV1.sol

```
    /*@CTK superOwner
      @post __return == _superOwner
     */
```

Line 238-243 in File IDRTWalletV1.sol

```
    function superOwner()
        public view
        returns (address)
    {
        return _superOwner;
    }
```

✓ The code meets the specification.

Formal Verification Request 32

requireFinalization

📅 03, Jul 2019
⏱ 4.57 ms

Line 249-251 in File IDRTWalletV1.sol

```
/*@CTK requireFinalization
  @post __return == _requireFinalization[transactionId]
 */
```

Line 252-257 in File IDRTWalletV1.sol

```
function requireFinalization(uint transactionId)
    public view
    returns (bool)
{
    return _requireFinalization[transactionId];
}
```

✓ The code meets the specification.

Formal Verification Request 33

addOwner

📅 03, Jul 2019
⏱ 56.72 ms

Line 167-174 in File MultiSigWallet.sol

```
/*@CTK addOwner
  @tag assume_completion
  @post msg.sender == address(this)
  @post !isOwner[owner]
  @post __post.isOwner[owner]
  @post owner != 0
  @post __post.owners[owners.length] == owner
 */
```

Line 175-185 in File MultiSigWallet.sol

```
function addOwner(address owner)
    public
    onlyWallet
    ownerDoesNotExist(owner)
    notNull(owner)
    validRequirement(owners.length + 1, required)
{
    isOwner[owner] = true;
    owners.push(owner);
    emit OwnerAddition(owner);
}
```

✓ The code meets the specification.

Formal Verification Request 34

changeRequirement

📅 03, Jul 2019
⏱ 26.33 ms

Line 236-241 in File MultiSigWallet.sol

```
    /*@CTK changeRequirement
      @tag assume_completion
      @post msg.sender == address(this)
      @post owners.length >= _required
      @post __post.required == _required
     */
```

Line 242-249 in File MultiSigWallet.sol

```
    function changeRequirement(uint _required)
        public
        onlyWallet
        validRequirement(owners.length, _required)
    {
        required = _required;
        emit RequirementChange(_required);
    }
```

✓ The code meets the specification.

Formal Verification Request 35

revokeConfirmation

03, Jul 2019
42.91 ms

Line 286-292 in File MultiSigWallet.sol

```
    /*@CTK revokeConfirmation
      @tag assume_completion
      @post isOwner[msg.sender]
      @post confirmations[transactionId][msg.sender]
      @post !transactions[transactionId].executed
      @post __post.confirmations[transactionId][msg.sender] == false
     */
```

Line 293-301 in File MultiSigWallet.sol

```
    function revokeConfirmation(uint transactionId)
        public
        ownerExists(msg.sender)
        confirmed(transactionId, msg.sender)
        notExecuted(transactionId)
    {
        confirmations[transactionId][msg.sender] = false;
        emit Revocation(msg.sender, transactionId);
    }
```

✓ The code meets the specification.

Formal Verification Request 36

getOwners

03, Jul 2019
4.65 ms

Line 437-439 in File MultiSigWallet.sol

```
    /*@CTK getOwners
      @post __return == owners
     */
```

Line 440-446 in File MultiSigWallet.sol

```
    function getOwners()
        public
        constant
        returns (address[])
    {
        return owners;
    }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File token/ERC20RupiahTokenV1.sol

```
/**
 * Rupiah Token Smart Contract
 * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered byt the following copyright and
 * permission notice:
 *
 *    OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
 *    Copyright (c) 2016 Smart Contract Solutions, Inc.
 *    Modified for Rupiah Token by FengkieJ 2019.
 *
 *    centre-tokens <https://github.com/centrehq/centre-tokens>
 *    Copyright CENTRE SECZ 2018.
 *    Modified for Rupiah Token by FengkieJ 2019.
 *
 *    ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
 *    Copyright (c) 2018 ZeppelinOS Global Limited.
 *
 *    The MIT License (MIT)
 *
 *    Permission is hereby granted, free of charge, to any person obtaining a copy
 *    of this software and associated documentation files (the "Software"), to deal
 *    in the Software without restriction, including without limitation the rights
 *    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 *    copies of the Software, and to permit persons to whom the Software is furnished
 to
 *    do so, subject to the following conditions:
 *
 *    The above copyright notice and this permission notice shall be included in all
 *    copies or substantial portions of the Software.
 *
 *    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 *    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 *    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 LIABILITY,
 *    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 *    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

```

```
import "./IERC20.sol";
import "../math/SafeMath.sol";
import "../governance/Blacklistable.sol";
import "../zos/Initializable.sol";

/**
 * @title ERC20RupiahToken
 * @dev ERC20 compliant fiat token that is backed by Indonesian Rupiah 1:1
 */
contract ERC20RupiahToken is IERC20, Blacklistable, Initializable {
    using SafeMath for uint256;

    string internal _name;
    string internal _symbol;
    string internal _currency;
    uint8 internal _decimals;

    mapping (address => uint256) internal _balances;
    mapping (address => mapping (address => uint256)) internal _allowed;
    uint256 internal _totalSupply;

    /**
     * @dev Initialize the smart contract to work with ZeppelinOS, can only be called
         once.
     * @param name describes the name of the token.
     * @param symbol describes the symbol of the token.
     * @param currency describes the currency of the token.
     * @param decimals describes the number of decimals of the token.
     */
    /*@CTK initialize
      @post __post.owner == msg.sender
      @post __post._name == name
      @post __post._symbol == symbol
      @post __post._currency == currency
      @post __post._decimals == decimals
     */
    function initialize(string name, string symbol, string currency, uint8 decimals)
        initializer public {
  owner = msg.sender;
        _name = name;
        _symbol = symbol;
        _currency = currency;
        _decimals = decimals;
    }

    /**
     * @return the name of the token.
     */
    /*@CTK name
      @post __return == _name
     */
    function name() public view returns (string memory) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    /*@CTK symbol
      @post __return == _symbol
     */
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    /**
     * @return the currency of the token.
     */
    /*@CTK currency
      @post __return == _currency
     */
    function currency() public view returns (string memory) {
        return _currency;
    }

    /**
     * @return the number of decimals of the token.
     */
    /*@CTK decimals
      @post __return == _decimals
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }

    /**
     * @return the total number of tokens in existence
     */
    /*@CTK totalSupply
      @post __return == _totalSupply
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
    * @dev Gets the balance of the specified address.
    * @param owner The address to query the balance of.
    * @return An uint256 representing the amount owned by the passed address.
    */
    /*@CTK balanceOf
      @post __return == _balances[owner]
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }

    /**
     * @dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param owner address The address which owns the funds.
     * @param spender address The address which will spend the funds.
     * @return A uint256 specifying the amount of tokens still available for the
         spender.
     */
    /*@CTK allowance
      @post __return == _allowed[owner][spender]
     */
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }

    /**
    * @dev Transfer token for a specified address
    * @param to The address to transfer to.
    * @param value The amount to be transferred.
    */
    /*@CTK transfer
      @tag assume_completion
      @pre msg.sender != to
      @post _paused == false
      @post blacklisted[msg.sender] == false
      @post blacklisted[to] == false
      @post to != address(0)
      @post __post._balances[msg.sender] == _balances[msg.sender] - value
      @post __post._balances[to] == _balances[to] + value
     */
    function transfer(address to, uint256 value) public whenNotPaused notBlacklisted(
        msg.sender) notBlacklisted(to) returns (bool) {
        require(to != address(0));

        _balances[msg.sender] = _balances[msg.sender].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(msg.sender, to, value);

        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on
         behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
         may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible
         solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
         desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param spender The address which will spend the funds.
     * @param value The amount of tokens to be spent.
     */
    function approve(address spender, uint256 value) public whenNotPaused
        notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another.
     * Note that while this function emits an Approval event, this is not required as
         per the specification,
     * and other compliant implementations may not emit the event.
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
    /*@CTK transferFrom
      @tag assume_completion
      @pre from != to
      @post _paused == false
      @post blacklisted[msg.sender] == false
      @post blacklisted[from] == false
      @post blacklisted[to] == false
      @post to != address(0)
      @post __post._balances[from] == _balances[from] - value
      @post __post._balances[to] == _balances[to] + value
     */
    function transferFrom(address from, address to, uint256 value) public
        whenNotPaused notBlacklisted(msg.sender) notBlacklisted(from) notBlacklisted(
        to) returns (bool) {
        require(to != address(0));

        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param addedValue The amount of tokens to increase the allowance by.
     */
    /*@CTK increaseAllowance
      @tag assume_completion
      @post _paused == false
      @post !blacklisted[msg.sender]
      @post !blacklisted[spender]
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
     */
    function increaseAllowance(address spender, uint256 addedValue) public
        whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
        ) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
        return true;
    }

    /**
     * @dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param subtractedValue The amount of tokens to decrease the allowance by.
     */
    /*@CTK decreaseAllowance
      @tag assume_completion
      @post _paused == false
      @post !blacklisted[msg.sender]
      @post !blacklisted[spender]
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
          subtractedValue
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        whenNotPaused notBlacklisted(msg.sender) notBlacklisted(spender) returns (bool
        ) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        return true;
    }

    /**
     * @dev Function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
    function mint(address account, uint256 value) public whenNotPaused notBlacklisted(
        account) onlyOwner {
        require(account != address(0));

        value = value.mul(10**_decimals);
        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }

    /**
     * @dev Function that burns an amount of the token.
     * @param value The amount that will be burnt.
     */
    function burn(uint256 value) public whenNotPaused onlyOwner {
        value = value.mul(10**_decimals);

        _totalSupply = _totalSupply.sub(value);
        _balances[msg.sender] = _balances[msg.sender].sub(value);
        emit Transfer(msg.sender, address(0), value);
    }

    /**
     * @dev Function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * Emits an Approval event (reflecting the reduced allowance).
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function burnFrom(address account, uint256 value) public whenNotPaused
        notBlacklisted(account) onlyOwner {
        require(account != address(0));

        value = value.mul(10**_decimals);
        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);

        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }

    /**
     * @dev Approve an address to spend another addresses' tokens.
     * @param owner The address that owns the tokens.
     * @param spender The address that will spend the tokens.
     * @param value The number of tokens that can be spent.
     */
    /*@CTK _approve
      @tag assume_completion
      @post spender != address(0)
      @post owner != address(0)
      @post __post._allowed[owner][spender] == value
     */
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }
}
```

File lifecycle/Pausable.sol

```
/**
 * The MIT License (MIT)
 *
 * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
 * Copyright (c) 2016 Smart Contract Solutions, Inc.
 * Modified for Rupiah Token by FengkieJ 2019.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is furnished to
 * do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

import "../ownership/Ownable.sol";
```

```

/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
contract Pausable is Ownable {
    event Paused(address account);
    event Unpaused(address account);

    bool private _paused;

    /**
     * @return true if the contract is paused, false otherwise.
     */
    /*@CTK paused
      @post __return == _paused
     */
    function paused() public view returns (bool) {
        return _paused;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     */
    modifier whenNotPaused() {
        require(!_paused);
        _;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is paused.
     */
    modifier whenPaused() {
        require(_paused);
        _;
    }

    /**
     * @dev called by the owner to pause, triggers stopped state
     */
    /*@CTK pause
      @tag assume_completion
      @post owner == msg.sender
      @post __post._paused == true
     */
    function pause() public onlyOwner {
        _paused = true;
        emit Paused(msg.sender);
    }

    /**
     * @dev called by the owner to unpause, returns to normal state
     */
    /*@CTK unpause
      @tag assume_completion
      @post owner == msg.sender
      @post __post._paused == false
     */
    function unpause() public onlyOwner {
        _paused = false;
        emit Unpaused(msg.sender);
    }
}
```

File math/SafeMath.sol

```
/**
 * The MIT License (MIT)
 *
 * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
 * Copyright (c) 2016 Smart Contract Solutions, Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is furnished to
 * do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error
 */
library SafeMath {
    /**
    * @dev Multiplies two unsigned integers, reverts on overflow.
    */
    /*@CTK SafeMath_mul
      @post __reverted == __has_overflow
      @post __reverted == false -> __return == a * b
      @post a == 0 -> __return == 0
      @post msg == msg__post
      @post (a > 0 && (a * b / a != b)) == __reverted
      @post __addr_map == __addr_map__post
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
    * @dev Integer division of two unsigned integers truncating the quotient, reverts
        on division by zero.
    */
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
    * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is
         greater than minuend).
    */
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
    * @dev Adds two unsigned integers, reverts on overflow.
    */
    /*@CTK SafeMath_add
      @post __reverted == __has_overflow
      @post __reverted == false -> __return == a + b
      @post msg == msg__post
      @post (a + b < a) == __has_overflow
      @post __addr_map == __addr_map__post
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
    * @dev Divides two unsigned integers and returns the remainder (unsigned integer
        modulo),
    * reverts when dividing by zero.
    */
    /*@CTK SafeMath_mod
      @tag assume_completion
      @post __return == a % b
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
```

File ownership/Ownable.sol

```
/**
 * The MIT License (MIT)
 *
 * OpenZeppelin <https://github.com/OpenZeppelin/openzeppelin-solidity/>
 * Copyright (c) 2016 Smart Contract Solutions, Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is furnished to
 * do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
    control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;

  event OwnershipTransferred(
    address indexed previousOwner,
    address indexed newOwner
  );

  /**
   * @dev The Ownable constructor sets the original ‘owner‘ of the contract to the
      sender
   * account.
   */
  /*@CTK Ownable
```

```
    @post __post.owner == msg.sender
   */
  constructor() public {
    owner = msg.sender;
  }

  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /**
   * @dev Allows the current owner to relinquish control of the contract.
   * @notice Renouncing to ownership will leave the contract without an owner.
   * It will not be possible to call the functions with the `onlyOwner`
   * modifier anymore.
   */
  /*@CTK renounceOwnership
    @tag assume_completion
    @post __post.owner == address(0)
    @post owner == msg.sender
   */
  function renounceOwnership() public onlyOwner {
    owner = address(0);
    emit OwnershipTransferred(msg.sender, owner);
  }

  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  /*@CTK transferOwnership
    @tag assume_completion
    @post owner == msg.sender
    @post _newOwner != address(0)
    @post __post.owner == _newOwner
   */
  function transferOwnership(address _newOwner) public onlyOwner {
    _transferOwnership(_newOwner);
  }

  /**
   * @dev Transfers control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  /*@CTK _transferOwnership
    @tag assume_completion
    @post _newOwner != address(0)
    @post __post.owner == _newOwner
   */
  function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    owner = _newOwner;
    emit OwnershipTransferred(owner, _newOwner);
  }
```

```
}
```

File governance/Blacklistable.sol

```
/**
 * Rupiah Token Smart Contract
 * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered by the following copyright and
 * permission notice:
 *
 *    Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
 *    Copyright (c) 2016 Gnosis Ltd.
 *    Modified for Rupiah Token by FengkieJ 2019.
 *
 *    This program is free software: you can redistribute it and/or modify
 *    it under the terms of the GNU Lesser General Public License as published by
 *    the Free Software Foundation, either version 3 of the License, or
 *    (at your option) any later version.
 *
 *    This program is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *    GNU Lesser General Public License for more details.
 *
 *    You should have received a copy of the GNU Lesser General Public License
 *    along with this program. If not, see <http://www.gnu.org/licenses/>.
 *-----------------------------------------------------------------------------------------

 *    ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
 *    Copyright (c) 2018 ZeppelinOS Global Limited.
 *
 *    The MIT License (MIT)
 *
 *    Permission is hereby granted, free of charge, to any person obtaining a copy
 *    of this software and associated documentation files (the "Software"), to deal
 *    in the Software without restriction, including without limitation the rights
 *    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 *    copies of the Software, and to permit persons to whom the Software is furnished
      to
 *    do so, subject to the following conditions:
 *
 *    The above copyright notice and this permission notice shall be included in all
 *    copies or substantial portions of the Software.
 *
 *    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 *    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 *    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     LIABILITY,
 *    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 *    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

import "../lifecycle/Pausable.sol";

/**
 * @title Blacklistable
 * @dev Allows accounts to be blacklisted by a "blacklister" role
*/
contract Blacklistable is Pausable {
    mapping(address => bool) internal blacklisted;

    event Blacklisted(address indexed _account);
    event Unblacklisted(address indexed _account);

    /**
     * @dev Throws if argument account is blacklisted
     * @param _account The address to check
    */
    modifier notBlacklisted(address _account) {
        require(blacklisted[_account] == false);
        _;
    }

    /**
     * @dev Checks if account is blacklisted
     * @param _account The address to check
    */
    /*@CTK isBlacklisted
      @post __return == blacklisted[_account]
     */
    function isBlacklisted(address _account) public view returns (bool) {
        return blacklisted[_account];
    }

    /**
     * @dev Adds account to blacklist
     * @param _account The address to blacklist
    */
    /*@CTK blacklist
      @tag assume_completion
      @post owner == msg.sender
      @post _paused == false
      @post __post.blacklisted[_account]
     */
    function blacklist(address _account) public onlyOwner whenNotPaused {
        blacklisted[_account] = true;
        emit Blacklisted(_account);
    }

    /**
     * @dev Removes account from blacklist
     * @param _account The address to remove from the blacklist
    */
    /*@CTK unblacklist
      @tag assume_completion
      @post owner == msg.sender
      @post _paused == false
      @post __post.blacklisted[_account] == false
     */
    function unblacklist(address _account) public onlyOwner whenNotPaused {
        blacklisted[_account] = false;
        emit Unblacklisted(_account);
    }
}
```

File governance/wallet/IDRTWalletV1.sol

```
/**
 * Rupiah Token Smart Contract
 * Copyright (C) 2019 PT. Rupiah Token Indonesia <https://www.rupiahtoken.com/>.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered by the following copyright and
 * permission notice:
 *
 *     Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
 *     Copyright (c) 2016 Gnosis Ltd.
 *     Modified for Rupiah Token by FengkieJ 2019.
 *
 *     This program is free software: you can redistribute it and/or modify
 *     it under the terms of the GNU Lesser General Public License as published by
 *     the Free Software Foundation, either version 3 of the License, or
 *     (at your option) any later version.
 *
 *     This program is distributed in the hope that it will be useful,
 *     but WITHOUT ANY WARRANTY; without even the implied warranty of
 *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *     GNU Lesser General Public License for more details.
 *
 *     You should have received a copy of the GNU Lesser General Public License
 *     along with this program. If not, see <http://www.gnu.org/licenses/>.
 *------------------------------------------------------------------------------------------

 *     ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
 *     Copyright (c) 2018 ZeppelinOS Global Limited.
 *
 *     The MIT License (MIT)
 *
 *    Permission is hereby granted, free of charge, to any person obtaining a copy
 *    of this software and associated documentation files (the "Software"), to deal
 *    in the Software without restriction, including without limitation the rights
 *    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 *    copies of the Software, and to permit persons to whom the Software is furnished
      to
 *    do so, subject to the following conditions:
 *
 *    The above copyright notice and this permission notice shall be included in all
 *    copies or substantial portions of the Software.
 *
 *    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 *    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 *    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY,
 *    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 *    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
pragma solidity ^0.4.25;

import "./MultiSigWallet.sol";

contract IDRTWallet is MultiSigWallet {
    uint256 internal _printLimit;
    mapping (uint => bool) internal _requireFinalization;
    address internal _superOwner;

    event OwnershipTransferred(
        address indexed previousOwner,
        address indexed newOwner
    );

    event PrintLimitChanged(
        uint256 indexed oldValue,
        uint256 indexed newValue
    );

    event RequireFinalization(uint indexed transactionId);

    event Finalized(uint indexed transactionId);

    /**
     * @dev Throws if called by any account other than _superOwner.
     */
    modifier onlySuperOwner() {
        require(msg.sender == _superOwner);
        _;
    }

    /**
     * @dev Initialize the smart contract to work with ZeppelinOS, can only be called
         once.
     * @param admins list of the multisig contract admins.
     * @param required number of required confirmations to execute a transaction.
     * @param printLimit maximum amount of minting limit before _superOwner need to
         finalize.
     */
    function initialize(address[] admins, uint256 required, uint256 printLimit) public
         initializer {
        MultiSigWallet.initialize(admins, required);
        _superOwner = msg.sender;
        _printLimit = printLimit;
    }

    /**
     * @dev Get the function signature from call data.
     * @param data the call data in bytes.
     * @return function signature in bytes4.
     */
    function getFunctionSignature(bytes memory data) internal pure returns (bytes4 out
        ) {
        assembly {
            out := mload(add(data, 0x20))
        }
    }

    /**
     * @dev Get the value to mint from call data.
     * @param data the call data in bytes.
     * @return value to mint in uint256.
     */
    function getValueToMint(bytes memory data) internal pure returns (uint256 value) {
        bytes32 x;
        assembly {
            x := mload(add(data, 0x44))
        }
        value = uint256(x);
    }

    /**
     * @dev Allows an owner to submit and confirm a transaction.
     * @param destination Transaction target address.
     * @param value Transaction ether value.
     * @param data Transaction data payload.
     * @return the transaction ID.
     */
    function submitTransaction(address destination, uint value, bytes data)
        public
        returns (uint transactionId)
    {
        transactionId = addTransaction(destination, value, data);
        bytes4 functionSignature = getFunctionSignature(data);
    if(
            (functionSignature == 0x99a88ec4) || //ZeppelinOS ProxyAdmin.sol's upgrade
                function
            (functionSignature == 0x9623609d) || //ZeppelinOS ProxyAdmin.sol's
                upgradeAndCall function
            (functionSignature == 0xe20056e6) || //MultiSigWallet.sol's replaceOwner
                function
            (functionSignature == 0x7065cb48) || //MultiSigWallet.sol's addOwner
                function
            (functionSignature == 0x173825d9) || //MultiSigWallet.sol's removeOwner
                function
            (functionSignature == 0x715018a6) || //ERC20 Ownable's renounceOwnership
                function
            (functionSignature == 0xf2fde38b) || //ERC20 Ownable's transferOwnership
                function
            ((functionSignature == 0x40c10f19) && (getValueToMint(data) > _printLimit))
                 //Calls mint function and value exceeds _printLimit
        ) {
            _requireFinalization[transactionId] = true;
            emit RequireFinalization(transactionId);
        }
        confirmTransaction(transactionId);
    }

    /**
     * @dev Allows anyone to execute a confirmed transaction.
     * @param transactionId Transaction ID.
     */
    function executeTransaction(uint transactionId)
        public
        ownerExists(msg.sender)
        confirmed(transactionId, msg.sender)
        notExecuted(transactionId)
    {
        if(!_requireFinalization[transactionId]) {
            super.executeTransaction(transactionId);
        } else {
            emit RequireFinalization(transactionId);
        }
    }

    /**
     * @dev Finalize tx by _superOwner.
     * @param transactionId Transaction ID.
     */
    function finalizeTransaction(uint transactionId)
        public
        onlySuperOwner()
        notExecuted(transactionId)
    {
        require(_requireFinalization[transactionId]);
        require(isConfirmed(transactionId));

        Transaction storage txn = transactions[transactionId];
        txn.executed = true;
        if (external_call(txn.destination, txn.value, txn.data.length, txn.data)) {
            emit Execution(transactionId);
            emit Finalized(transactionId);
        } else {
            emit ExecutionFailure(transactionId);
            txn.executed = false;
        }
    }

    /**
     * @dev Set new printLimit before _superOwner need to finalize.
     * @param newLimit of print limit amount.
     */
    /*@CTK setPrintLimit
      @tag assume_completion
      @post msg.sender == _superOwner
      @post __post._printLimit == newLimit
     */
    function setPrintLimit(uint256 newLimit)
        public
        onlySuperOwner()
    {
        emit PrintLimitChanged(_printLimit, newLimit);
        _printLimit = newLimit;
    }

    /**
     * @dev Set new _superOwner address.
     * @param newAddress new address for _superOwner
     */
    /*@CTK transferOwnership
      @tag assume_completion
      @post msg.sender == _superOwner
      @post newAddress != address(0)
     */
    function transferOwnership(address newAddress)
        public
        onlySuperOwner()
    {
        require(newAddress != address(0));

        _superOwner = newAddress;
        emit OwnershipTransferred(msg.sender, newAddress);
    }

    /**
     * @dev Get current _superOwner address.
     */
    /*@CTK superOwner
      @post __return == _superOwner
     */
    function superOwner()
        public view
        returns (address)
    {
        return _superOwner;
    }


    /**
     * @dev Get whether a transaction require finalization or not.
     */
    /*@CTK requireFinalization
      @post __return == _requireFinalization[transactionId]
     */
    function requireFinalization(uint transactionId)
        public view
        returns (bool)
    {
        return _requireFinalization[transactionId];
    }
}
```

File governance/wallet/MultiSigWallet.sol

```
/**
 * Ethereum Multisignature Wallet <https://github.com/gnosis/MultiSigWallet>
 * Copyright (c) 2016 Gnosis Ltd.
 * Modified for Rupiah Token by FengkieJ 2019.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered by the following copyright and
 * permission notice:
 *    ZeppelinOS (zos) <https://github.com/zeppelinos/zos>
 *    Copyright (c) 2018 ZeppelinOS Global Limited.
 *
 *    The MIT License (MIT)
 *
 *    Permission is hereby granted, free of charge, to any person obtaining a copy
 *    of this software and associated documentation files (the "Software"), to deal
 *    in the Software without restriction, including without limitation the rights
 *    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 *    copies of the Software, and to permit persons to whom the Software is furnished
      to
 *    do so, subject to the following conditions:
 *
 *    The above copyright notice and this permission notice shall be included in all
 *    copies or substantial portions of the Software.
 *
 *    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 *    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 *    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
     LIABILITY,
 *    WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 *    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

pragma solidity ^0.4.25;

import "../../zos/Initializable.sol";

/// @title Multisignature wallet - Allows multiple parties to agree on transactions
    before execution.
/// @author Stefan George - <stefan.george@consensys.net>
/// Modified for Rupiah Token by FengkieJ 2019

contract MultiSigWallet is Initializable {
    /*
     *  Events
     */
    event Confirmation(address indexed sender, uint indexed transactionId);
    event Revocation(address indexed sender, uint indexed transactionId);
    event Submission(uint indexed transactionId);
    event Execution(uint indexed transactionId);
    event ExecutionFailure(uint indexed transactionId);
    event Deposit(address indexed sender, uint value);
    event OwnerAddition(address indexed owner);
    event OwnerRemoval(address indexed owner);
    event RequirementChange(uint required);

    /*
     *  Constants
     */
    uint constant public MAX_OWNER_COUNT = 50;

    /*
     *  Storage
     */
    mapping (uint => Transaction) public transactions;
    mapping (uint => mapping (address => bool)) public confirmations;
    mapping (address => bool) public isOwner;
    address[] public owners;
    uint public required;
    uint public transactionCount;

    struct Transaction {
        address destination;
        uint value;
        bytes data;
        bool executed;
    }

    /*
     *  Modifiers
     */
    modifier onlyWallet() {
        require(msg.sender == address(this));
        _;
    }

    modifier ownerDoesNotExist(address owner) {
        require(!isOwner[owner]);
        _;
    }

    modifier ownerExists(address owner) {
        require(isOwner[owner]);
        _;
    }

    modifier transactionExists(uint transactionId) {
        require(transactions[transactionId].destination != 0);
        _;
    }

    modifier confirmed(uint transactionId, address owner) {
        require(confirmations[transactionId][owner]);
        _;
    }

    modifier notConfirmed(uint transactionId, address owner) {
        require(!confirmations[transactionId][owner]);
        _;
    }

    modifier notExecuted(uint transactionId) {
        require(!transactions[transactionId].executed);
        _;
    }

    modifier notNull(address _address) {
        require(_address != 0);
        _;
    }

    modifier validRequirement(uint ownerCount, uint _required) {
        require(ownerCount <= MAX_OWNER_COUNT
            && _required <= ownerCount
            && _required != 0
            && ownerCount != 0);
        _;
    }

    /// @dev Fallback function allows to deposit ether.
    function()
        payable
    {
        if (msg.value > 0)
            emit Deposit(msg.sender, msg.value);
    }

    /*
     * Public functions
     */
    /// @dev Initializer sets initial owners and required number of confirmations.
    /// @param _owners List of initial owners.
    /// @param _required Number of required confirmations.
    function initialize(address[] _owners, uint _required)
        public
        validRequirement(_owners.length, _required) initializer
    {
  for (uint i=0; i<_owners.length; i++) {
            require(!isOwner[_owners[i]] && _owners[i] != 0);
            isOwner[_owners[i]] = true;
        }
        owners = _owners;
        required = _required;
    }

    /// @dev Allows to add a new owner. Transaction has to be sent by wallet.
    /// @param owner Address of new owner.
    /*@CTK addOwner
      @tag assume_completion
      @post msg.sender == address(this)
      @post !isOwner[owner]
      @post __post.isOwner[owner]
      @post owner != 0
      @post __post.owners[owners.length] == owner
     */
    function addOwner(address owner)
        public
        onlyWallet
        ownerDoesNotExist(owner)
        notNull(owner)
        validRequirement(owners.length + 1, required)
    {
        isOwner[owner] = true;
        owners.push(owner);
        emit OwnerAddition(owner);
    }

    /// @dev Allows to remove an owner. Transaction has to be sent by wallet.
    /// @param owner Address of owner.
    function removeOwner(address owner)
        public
        onlyWallet
        ownerExists(owner)
    {
        isOwner[owner] = false;
        /*CTK find_owner_index
          @inv this == this__pre
          @inv owners == owners__pre
          @inv i <= owners.length - 1
          @inv !__should_return
          @inv i > 0 -> owners[i - 1] != owner
          @post i == owners.length - 1 || owners[i] == owner
         */
        for (uint i=0; i<owners.length - 1; i++)
            if (owners[i] == owner) {
                owners[i] = owners[owners.length - 1];
                break;
            }
        owners.length -= 1;
        if (required > owners.length)
            changeRequirement(owners.length);
        emit OwnerRemoval(owner);
    }

    /// @dev Allows to replace an owner with a new owner. Transaction has to be sent
        by wallet.
    /// @param owner Address of owner to be replaced.
    /// @param newOwner Address of new owner.
    function replaceOwner(address owner, address newOwner)
        public
        onlyWallet
        ownerExists(owner)
        ownerDoesNotExist(newOwner)
    {
        for (uint i=0; i<owners.length; i++)
            if (owners[i] == owner) {
                owners[i] = newOwner;
                break;
            }
        isOwner[owner] = false;
        isOwner[newOwner] = true;
        emit OwnerRemoval(owner);
        emit OwnerAddition(newOwner);
    }

    /// @dev Allows to change the number of required confirmations. Transaction has to
        be sent by wallet.
    /// @param _required Number of required confirmations.
    /*@CTK changeRequirement
      @tag assume_completion
      @post msg.sender == address(this)
      @post owners.length >= _required
      @post __post.required == _required
     */
    function changeRequirement(uint _required)
        public
        onlyWallet
        validRequirement(owners.length, _required)
    {
        required = _required;
        emit RequirementChange(_required);
    }

    /// @dev Allows an owner to submit and confirm a transaction.
    /// @param destination Transaction target address.
    /// @param value Transaction ether value.
    /// @param data Transaction data payload.
    /// @return Returns transaction ID.
    function submitTransaction(address destination, uint value, bytes data)
        public
        returns (uint transactionId)
    {
        transactionId = addTransaction(destination, value, data);
        confirmTransaction(transactionId);
    }

    /// @dev Allows an owner to confirm a transaction.
    /// @param transactionId Transaction ID.
    /*CTK confirmTransaction
      @tag assume_completion
      @post isOwner[msg.sender]
      @post !confirmations[transactionId][msg.sender]
      @post transactions[transactionId].destination != 0
      @post __post.confirmations[transactionId][msg.sender]
     */
    function confirmTransaction(uint transactionId)
        public
        ownerExists(msg.sender)
        transactionExists(transactionId)
        notConfirmed(transactionId, msg.sender)
    {
        confirmations[transactionId][msg.sender] = true;
        emit Confirmation(msg.sender, transactionId);
        executeTransaction(transactionId);
    }

    /// @dev Allows an owner to revoke a confirmation for a transaction.
    /// @param transactionId Transaction ID.
    /*@CTK revokeConfirmation
      @tag assume_completion
      @post isOwner[msg.sender]
      @post confirmations[transactionId][msg.sender]
      @post !transactions[transactionId].executed
      @post __post.confirmations[transactionId][msg.sender] == false
     */
    function revokeConfirmation(uint transactionId)
        public
        ownerExists(msg.sender)
        confirmed(transactionId, msg.sender)
        notExecuted(transactionId)
    {
        confirmations[transactionId][msg.sender] = false;
        emit Revocation(msg.sender, transactionId);
    }

    /// @dev Allows anyone to execute a confirmed transaction.
    /// @param transactionId Transaction ID.
    function executeTransaction(uint transactionId)
        public
        ownerExists(msg.sender)
        confirmed(transactionId, msg.sender)
        notExecuted(transactionId)
    {
        if (isConfirmed(transactionId)) {
            Transaction storage txn = transactions[transactionId];
            txn.executed = true;
            if (external_call(txn.destination, txn.value, txn.data.length, txn.data))
                emit Execution(transactionId);
            else {
                emit ExecutionFailure(transactionId);
                txn.executed = false;
            }
        }
    }

    // call has been separated into its own function in order to take advantage
    // of the Solidity's code generator to produce a loop that copies tx.data into
       memory.
    function external_call(address destination, uint value, uint dataLength, bytes
       data) internal returns (bool) {
        bool result;
        assembly {
            let x := mload(0x40) // "Allocate" memory for output (0x40 is where "free
               memory" pointer is stored by convention)
            let d := add(data, 32) // First 32 bytes are the padded length of data, so
               exclude that
            result := call(
                sub(gas, 34710), // 34710 is the value that solidity is currently
                   emitting
                                 // It includes callGas (700) + callVeryLow (3, to pay
                                    for SUB) + callValueTransferGas (9000) +
                                 // callNewAccountGas (25000, in case the destination
                                    address does not exist and needs creating)
                destination,
                value,
                d,
                dataLength,       // Size of the input (in bytes) - this is what fixes
                    the padding problem
                x,
                0                 // Output is ignored, therefore the output size is
                    zero
            )
        }
        return result;
    }

    /// @dev Returns the confirmation status of a transaction.
    /// @param transactionId Transaction ID.
    /// @return Confirmation status.
    function isConfirmed(uint transactionId)
        public
        constant
        returns (bool)
    {
        uint count = 0;
        for (uint i=0; i<owners.length; i++) {
            if (confirmations[transactionId][owners[i]])
                count += 1;
            if (count == required)
                return true;
        }
    }

    /*
     * Internal functions
     */
    /// @dev Adds a new transaction to the transaction mapping, if transaction does
        not exist yet.
    /// @param destination Transaction target address.
    /// @param value Transaction ether value.
    /// @param data Transaction data payload.
    /// @return Returns transaction ID.
    /*CTK addTransaction
      @tag assume_completion
      @post destination != address(0)
      @post __post.transactions[transactionCount].destination == destination
      @post __post.transactions[transactionCount].value == value
      @post __post.transactions[transactionCount].data == data
      @post __post.transactions[transactionCount].executed == false
      @post __post.transactionCount == transactionCount + 1
     */
    function addTransaction(address destination, uint value, bytes data)
        internal
        notNull(destination)
        returns (uint transactionId)
    {
        transactionId = transactionCount;
        transactions[transactionId].destination = destination;
        transactions[transactionId].value = value;
        transactions[transactionId].data = data;
        transactions[transactionId].executed = false;
        // transactions[transactionId] = Transaction({
        //     destination: destination,
        //     value: value,
        //    data: data,
        //    executed: false
        // });
        transactionCount += 1;
        emit Submission(transactionId);
    }

    /*
     * Web3 call functions
     */
    /// @dev Returns number of confirmations of a transaction.
    /// @param transactionId Transaction ID.
    /// @return Number of confirmations.
    function getConfirmationCount(uint transactionId)
        public
        constant
        returns (uint count)
    {
        for (uint i=0; i<owners.length; i++)
            if (confirmations[transactionId][owners[i]])
                count += 1;
    }

    /// @dev Returns total number of transactions after filers are applied.
    /// @param pending Include pending transactions.
    /// @param executed Include executed transactions.
    /// @return Total number of transactions after filters are applied.
    //*CTK only two states are allowed. It is more readable to separate this
    // function into multiple small ones. For example,
    // 1. getTotalTransaction = 2 + 3
    // 2. getPendingTransaction
    // 3. getExecutedTransaction
    function getTransactionCount(bool pending, bool executed)
        public
        constant
        returns (uint count)
    {
        for (uint i=0; i<transactionCount; i++)
            if (   pending && !transactions[i].executed
                || executed && transactions[i].executed)
                count += 1;
    }

    /// @dev Returns list of owners.
    /// @return List of owner addresses.
    /*@CTK getOwners
      @post __return == owners
     */
    function getOwners()
        public
        constant
        returns (address[])
    {
        return owners;
    }

    /// @dev Returns array with owner addresses, which confirmed transaction.
    /// @param transactionId Transaction ID.
    /// @return Returns array of owner addresses.
    function getConfirmations(uint transactionId)
        public
        constant
        returns (address[] _confirmations)
    {
        address[] memory confirmationsTemp = new address[](owners.length);
        uint count = 0;
        uint i;
        for (i=0; i<owners.length; i++)
            if (confirmations[transactionId][owners[i]]) {
                confirmationsTemp[count] = owners[i];
                count += 1;
            }
        _confirmations = new address[](count);
        for (i=0; i<count; i++)
            _confirmations[i] = confirmationsTemp[i];
    }

    /// @dev Returns list of transaction IDs in defined range.
    /// @param from Index start position of transaction array.
    /// @param to Index end position of transaction array.
    /// @param pending Include pending transactions.
    /// @param executed Include executed transactions.
    /// @return Returns array of transaction IDs.
    //*CTK no checks for to > from.
    // can be used by getTransactionCount
    function getTransactionIds(uint from, uint to, bool pending, bool executed)
        public
        constant
        returns (uint[] _transactionIds)
    {
        uint[] memory transactionIdsTemp = new uint[](transactionCount);
        uint count = 0;
        uint i;
        for (i=0; i<transactionCount; i++)
            if (   pending && !transactions[i].executed
                || executed && transactions[i].executed)
            {
                transactionIdsTemp[count] = i;
                count += 1;
            }
        _transactionIds = new uint[](to - from);
        for (i=from; i<to; i++)
            _transactionIds[i - from] = transactionIdsTemp[i];
    }
}
```