CertiK Audit Report For FrenchICO

Request Date: 2019-05-09
Revision Date: 2019-06-04
Platform Name: Ethereum

Contents

Disclaimer **1**

Exective Summary **2**

Vulnerability Classification **2**

Testing Summary **3**
Audit Score 3
Type of Issues 3
Vulnerability Details 4

Manual Review Notes **5**

Formal Verification Results **6**
How to read 6

Static Analysis Results **35**

Source Code with CertiK Labels **40**

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and FrenchICO(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by FrenchICO. This audit was conducted to discover issues and vulnerabilities in the source code of FrenchICO'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.*

Jun 04, 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.	4	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.	1	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

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **FrenchIcoGateway.sol** `0391e2dc208dd8b8e96957c7afaa8aa2356259ef61d925bd84f2dd89fb966688`
- **FrenchIco.sol** `29e568f3e62d94b6fde26f752fd2277e3ae21ddbf6d4d36ee588ab13b76a9ce9`
- **FrenchIcoCorporate.sol** `f527deee26797f4646dd719dfc06e3422a31a76964542cc09aa5bd8f7f45e533`
- **FrenchIcoCrowdsale.sol** `b3883d6b5e9920ad2ee1193ac37d08b298cedc50e7146891993fe75dd19b768b`
- **FrenchIcoToken.sol** `cc7520a510d0162a0799bfb6ccff7a2923e66b3d1c2b5d76953019edf27db1ed`
- **OwnablePayable.sol** `0ad4083c2c4562a1c159cc934acd39cc8fe594d681a14b1f761ceb244ae1b8c5`

Summary

The FrenchICO team asked CertiK to conduct a security audit of the design and implementation of its to-be-released MiniMe-based smart contracts. In this comprehensive audit, the source code analysis was conducted through a variety of methods and tools, such as CertiK Formal Verification as well as manual review by smart contract experts. CertiK directly interfaced with client-side engineers to fix critical loopholes and address recommended design changes throughout the audit process. The FrenchICO team provided timely enhancements to source code suggestions, as well as supportive feedback surrounding the business logics.

At the moment, the FrenchICO team did not have testing and documentation repositories available for reference. CertiK recommends additional unit test coverage, along with documentation, to more thoroughly simulate potential use cases and functionalities for token holders, especially with respect to super admin privileges that may impact the FrenchICO token's decentralized nature.

Overall, CertiK observed that the contract follows good practices, and the smart contract design is among one of the innovative projects in industry by providing ICO issuers much easier access to raise funds and issue tokens. With the final update of source code and delivery of the audit report, CertiK concludes that the contract is not vulnerable to the classically-known anti-patterns or security issues at this time. It should be noted that this audit report is not an absolute guarantee of correctness or trustworthiness, and CertiK always recommends seeking multiple opinions, increased test coverage, and live sandbox deployments before a mainnet release.

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 to
        ) {
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 }
```

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

FrenchIcoToken

04, Jun 2019
6.57 ms

Line 31-34 in File FrenchIcoToken.sol

```
31    /*@CTK FrenchIcoToken
32      @post __post.symbol == _symbol
33      @post __post.name == _name
34     */
```

Line 35-39 in File FrenchIcoToken.sol

```
35    constructor(string memory _symbol, string memory _name) public {
36        symbol = _symbol;
37        name = _name;
38        emit NewTokenFico(msg.sender, "Copyright FRENCHICO", name, symbol);
39    }
```

The code meets the specification

Formal Verification Request 2

toggleGeneralPause

04, Jun 2019
42.83 ms

Line 46-50 in File FrenchIcoCorporate.sol

```
46    /*@CTK toggleGeneralPause
47      @tag assume_completion
48      @post _owner == msg.sender
49      @post __post.pauseAllContracts == !pauseAllContracts
50     */
```

Line 51-53 in File FrenchIcoCorporate.sol

```
51    function toggleGeneralPause() public onlyOwner {
52        pauseAllContracts = !pauseAllContracts;
53    }
```

The code meets the specification

Formal Verification Request 3

setMaxAmount

04, Jun 2019
18.22 ms

Line 60-64 in File FrenchIcoCorporate.sol

```
60    /*@CTK setMaxAmount
61      @tag assume_completion
62      @post _owner == msg.sender
63      @post __post.maxAmount == _maxAmount
64     */
```

Line 65-67 in File FrenchIcoCorporate.sol

```
65    function setMaxAmount(uint _maxAmount) public onlyOwner {
66        maxAmount = _maxAmount;
67    }
```

The code meets the specification

Formal Verification Request 4

setRole

04, Jun 2019
31.99 ms

Line 77-83 in File FrenchIcoCorporate.sol

```
77    /*@CTK setRole
78      @tag assume_completion
79      @post _owner == msg.sender
80      @post __post.members[addr].role == _role
81      @post __post.members[addr].recordTime == now
82      @post __post.members[addr].comments == comments
83     */
```

Line 84-95 in File FrenchIcoCorporate.sol

```
84    function setRole(
85        address addr,
86        uint _role,
87        string memory comments
88    )
89        public onlyOwner
90    {
91        members[addr].role = _role;
92        members[addr].recordTime = now;
93        members[addr].comments = comments;
94        emit ListUpdated(addr, "role", _role);
95    }
```

The code meets the specification

Formal Verification Request 5

setCountryCode

04, Jun 2019
19.57 ms

Line 103-107 in File FrenchIcoCorporate.sol

```
/*@CTK setCountryCode
  @tag assume_completion
  @post _owner == msg.sender
  @post __post.members[addr].countryCode == countryCode
 */
```

Line 108-116 in File FrenchIcoCorporate.sol

```
function setCountryCode(
    address addr,
    uint countryCode
)
    public onlyOwner
{
    members[addr].countryCode = countryCode;
    emit ListUpdated(addr, "countryCode", countryCode);
}
```

✅ The code meets the specification

Formal Verification Request 6

setMemberMaxAmountAllowed

📅 04, Jun 2019
⏱ 18.59 ms

Line 124-128 in File FrenchIcoCorporate.sol

```
/*@CTK setMemberMaxAmountAllowed
  @tag assume_completion
  @post _owner == msg.sender
  @post __post.members[addr].maxAmountAllowed == maxAmountAllowed
 */
```

Line 129-137 in File FrenchIcoCorporate.sol

```
function setMemberMaxAmountAllowed(
    address addr,
    uint maxAmountAllowed
)
    public onlyOwner
{
    members[addr].maxAmountAllowed = maxAmountAllowed;
    emit ListUpdated(addr, "maxAmountAllowed", maxAmountAllowed);
}
```

✅ The code meets the specification

Formal Verification Request 7

addMember

📅 04, Jun 2019
⏱ 61.36 ms

Line 142-150 in File FrenchIcoCorporate.sol

```
142   /*@CTK addMember
143     @tag assume_completion
144     @post members[msg.sender].role == ROLE_NOT_REGISTERED
145     @post __post.members[msg.sender].role == ROLE_ANGEL
146     @post __post.members[msg.sender].countryCode == 0
147     @post __post.members[msg.sender].maxAmountAllowed == 0
148     @post __post.members[msg.sender].recordTime == now
149     @post __post.members[msg.sender].comments == "new member"
150    */
```

Line 151-161 in File FrenchIcoCorporate.sol

```
151   function addMember() public payable {
152       require (members[msg.sender].role == ROLE_NOT_REGISTERED, "user has to be new")
              ;
153       uint newMemberRole = ROLE_ANGEL;
154       members[msg.sender].role = newMemberRole;
155       members[msg.sender].countryCode = 0;
156       members[msg.sender].maxAmountAllowed = 0;
157       members[msg.sender].recordTime = now;
158       members[msg.sender].comments = "new member";
159       _owner.transfer(msg.value);
160       emit ListUpdated(msg.sender, "role", newMemberRole);
161   }
```

✅ The code meets the specification

Formal Verification Request 8

isGeneralPaused

📅 04, Jun 2019
⏱ 7.42 ms

Line 170-172 in File FrenchIcoCorporate.sol

```
170   /*@CTK isGeneralPaused
171     @post __return == pauseAllContracts
172    */
```

Line 173-175 in File FrenchIcoCorporate.sol

```
173   function isGeneralPaused() external view returns (bool) {
174       return pauseAllContracts;
175   }
```

✅ The code meets the specification

Formal Verification Request 9

getRole

📅 04, Jun 2019
⏱ 5.62 ms

Line 182-184 in File FrenchIcoCorporate.sol

```
182     /*@CTK getRole
183       @post __return == members[addr].role
184      */
```

Line 185-187 in File FrenchIcoCorporate.sol

```
185     function getRole(address addr) external view returns (uint) {
186         return members[addr].role;
187     }
```

✅ The code meets the specification

Formal Verification Request 10

getCountryCode

📅 04, Jun 2019
⏱ 5.32 ms

Line 194-196 in File FrenchIcoCorporate.sol

```
194     /*@CTK getCountryCode
195       @post __return == members[addr].countryCode
196      */
```

Line 197-199 in File FrenchIcoCorporate.sol

```
197     function getCountryCode(address addr) external view returns (uint) {
198         return members[addr].countryCode;
199     }
```

✅ The code meets the specification

Formal Verification Request 11

getMemberMaxAmountAllowed

📅 04, Jun 2019
⏱ 5.11 ms

Line 206-208 in File FrenchIcoCorporate.sol

```
206     /*@CTK getMemberMaxAmountAllowed
207       @post __return == members[addr].maxAmountAllowed
208      */
```

Line 209-211 in File FrenchIcoCorporate.sol

```
209     function getMemberMaxAmountAllowed(address addr) external view returns (uint) {
210         return members[addr].maxAmountAllowed;
211     }
```

✅ The code meets the specification

Formal Verification Request 12

getWalletFrenchICO

04, Jun 2019
4.75 ms

Line 218-220 in File FrenchIcoCorporate.sol

```
/*@CTK getWalletFrenchICO
  @post __return == _owner
 */
```

Line 221-223 in File FrenchIcoCorporate.sol

```
function getWalletFrenchICO() external view returns (address payable) {
    return _owner;
}
```

The code meets the specification

Formal Verification Request 13

getMaxAmount

04, Jun 2019
6.23 ms

Line 230-232 in File FrenchIcoCorporate.sol

```
/*@CTK getMaxAmount
  @post __return == maxAmount
 */
```

Line 233-235 in File FrenchIcoCorporate.sol

```
function getMaxAmount() external view returns (uint) {
    return maxAmount;
}
```

The code meets the specification

Formal Verification Request 14

FrenchIcoCrowdsale

04, Jun 2019
59.05 ms

Line 117-121 in File FrenchIcoCrowdsale.sol

```
/*@CTK FrenchIcoCrowdsale
  @tag assume_completion
  @post __post.icoNumber == 0
  @post __post.ICOs[__post.icoNumber].status == STATUS_READY_FOR_NEW_ICO
 */
```

Line 122-129 in File FrenchIcoCrowdsale.sol

```
    constructor(string memory _name, string memory _symbol, uint _minCap, uint
        _startTime, uint _endTime, uint _presalePeriodDuration, uint
        _firstPeriodDuration, uint _secondPeriodDuration) public payable {
        token = new FrenchIcoToken(_symbol, _name);

        icoNumber = 0;
        ICOs[icoNumber].status = STATUS_READY_FOR_NEW_ICO;
        // newICO(rate, _minCap, _startTime, _endTime, _presalePeriodDuration,
            _firstPeriodDuration, _secondPeriodDuration); // Create the new ICO
        emit Copyright("Copyright FRENCHICO");
    }
```

The code meets the specification

Formal Verification Request 15

setNewIcoPossible

04, Jun 2019
22.38 ms

Line 141-145 in File FrenchIcoCrowdsale.sol

```
    /*@CTK setNewIcoPossible
      @tag assume_completion
      @post msg.sender == 0x1
      @post __post.isNewIcoPossible
     */
```

Line 146-149 in File FrenchIcoCrowdsale.sol

```
    function setNewIcoPossible() public {
        require (msg.sender == fico.getWalletFrenchICO());
        isNewIcoPossible = true;
    }
```

The code meets the specification

Formal Verification Request 16

getCurrentBonus

04, Jun 2019
18.24 ms

Line 300-305 in File FrenchIcoCrowdsale.sol

```
    /*@CTK getCurrentBonus
      @post now <= endPresalePeriod -> __return == presaleBonus
      @post now > endPresalePeriod && now <= endFirstPeriod -> __return ==
          firstPeriodBonus
      @post now > endPresalePeriod && now > endFirstPeriod && now <= endSecondPeriod
          -> __return == secondPeriodBonus
      @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod ->
           __return == 0
     */
```

Line 306-319 in File FrenchIcoCrowdsale.sol

```
function getCurrentBonus() public view returns(uint) {
    uint bonus;

    if (now <= endPresalePeriod) {
        bonus = presaleBonus;
    } else if (now <= endFirstPeriod) {
        bonus = firstPeriodBonus;
    } else if (now <= endSecondPeriod) {
        bonus = secondPeriodBonus;
    } else {
        bonus = 0;
    }
    return bonus;
}
```

The code meets the specification

Formal Verification Request 17

isCapReached

04, Jun 2019
1026.81 ms

Line 355-362 in File FrenchIcoCrowdsale.sol

```
/*@CTK isCapReached
  @tag assume_completion
  @post now <= endPresalePeriod -> __return == ICOs[icoNumber].fundsDeposited + value > presalePeriodCap
  @post now > endPresalePeriod && now <= endFirstPeriod -> __return == ICOs[icoNumber].fundsDeposited + value > firstPeriodCap
  @post now > endPresalePeriod && now > endFirstPeriod && now <= endSecondPeriod -> __return == ICOs[icoNumber].fundsDeposited + value > secondPeriodCap
  @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod && now <= ICOs[icoNumber].endTime -> __return == ICOs[icoNumber].fundsDeposited > ICOs[icoNumber].maxCap
  @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod && now > ICOs[icoNumber].endTime -> !__return
 */
```

Line 363-378 in File FrenchIcoCrowdsale.sol

```
function isCapReached(uint value) public view returns(bool) {
    bool goal;

    if (now <= endPresalePeriod) {
        goal = ICOs[icoNumber].fundsDeposited.add(value) > presalePeriodCap;
    } else if (now <= endFirstPeriod) {
        goal = ICOs[icoNumber].fundsDeposited.add(value) > firstPeriodCap;
    } else if (now <= endSecondPeriod) {
        goal = ICOs[icoNumber].fundsDeposited.add(value) > secondPeriodCap;
    } else if (now <= ICOs[icoNumber].endTime) {
        goal = ICOs[icoNumber].fundsDeposited > ICOs[icoNumber].maxCap;
    } else {
        goal = false;
```

```
        }
        return goal;
    }
```

The code meets the specification

Formal Verification Request 18

isFinished

04, Jun 2019
9.45 ms

Line 385-389 in File FrenchIcoCrowdsale.sol

```
385    /*@CTK isFinished
386      @post __return == (ICOs[icoNumber].fundsDeposited >=
387                         ICOs[icoNumber].maxCap ||
388                         now > ICOs[icoNumber].endTime)
389     */
```

Line 390-392 in File FrenchIcoCrowdsale.sol

```
390    function isFinished() public view returns(bool) {
391        return (ICOs[icoNumber].fundsDeposited >= ICOs[icoNumber].maxCap || now > ICOs[
               icoNumber].endTime);
392    }
```

The code meets the specification

Formal Verification Request 19

OwnablePayable

04, Jun 2019
5.06 ms

Line 22-24 in File OwnablePayable.sol

```
22    /*@CTK OwnablePayable
23      @post __post._owner == msg.sender
24     */
```

Line 25-28 in File OwnablePayable.sol

```
25    constructor() internal {
26        _owner = msg.sender;
27        emit OwnershipTransferred(address(0), _owner);
28    }
```

The code meets the specification

Formal Verification Request 20

transferOwnership

04, Jun 2019
51.44 ms

Line 44-49 in File OwnablePayable.sol

```
44  /*@CTK transferOwnership
45    @tag assume_completion
46    @post _owner == msg.sender
47    @post newOwner != address(0)
48    @post __post._owner == newOwner
49   */
```

Line 50-52 in File OwnablePayable.sol

```
50  function transferOwnership(address payable newOwner) public onlyOwner {
51      _transferOwnership(newOwner);
52  }
```

The code meets the specification

Formal Verification Request 21

_transferOwnership

04, Jun 2019
1.3 ms

Line 59-63 in File OwnablePayable.sol

```
59  /*@CTK _transferOwnership
60    @tag assume_completion
61    @post newOwner != address(0)
62    @post __post._owner == newOwner
63   */
```

Line 64-68 in File OwnablePayable.sol

```
64  function _transferOwnership(address payable newOwner) internal {
65      require(newOwner != address(0));
66      emit OwnershipTransferred(_owner, newOwner);
67      _owner = newOwner;
68  }
```

The code meets the specification

Formal Verification Request 22

payOrder

04, Jun 2019
439.05 ms

Line 82-87 in File FrenchIcoGateway.sol

```
/*@CTK payOrder
  @tag assume_completion
  @post payment[orderId].buyer == address(0x0)
  @post __post.payment[orderId].buyer == sender
  @post __post.payment[orderId].amount == amount
 */
```

Line 88-102 in File FrenchIcoGateway.sol

```
function payOrder(
    address sender,
    uint amount,
    address tokenAddr,
    uint orderId
)
    internal
{
    require (payment[orderId].buyer == address(0x00));
    ERC20 token = ERC20(tokenAddr);
    token.transferFrom(sender, address(this), amount);
    token.transfer(_owner, amount);
    payment[orderId] = Order(sender, amount);
    emit PaidWithToken(sender, msg.sender, amount, orderId);
}
```

✅ The code meets the specification

Formal Verification Request 23

anchorHash

📅 04, Jun 2019
⏱ 197.38 ms

Line 112-119 in File FrenchIcoGateway.sol

```
/*@CTK anchorHash
  @tag assume_completion
  @post anchors[hash].anchorOwner == msg.sender ||
        anchors[hash].anchorOwner == address(0x0)
  @post __post.anchors[hash].anchorDate == now
  @post __post.anchors[hash].anchorOwner == anchorOwner
  @post __post.anchors[hash].amount == amount
 */
```

Line 120-134 in File FrenchIcoGateway.sol

```
function anchorHash(
    address anchorOwner,
    uint amount,
    address tokenAddr,
    string memory hash
)
    internal
{
    require ((anchors[hash].anchorOwner == msg.sender) || (anchors[hash].
        anchorOwner == address(0x00)));
    ERC20 token = ERC20(tokenAddr);
```

```
        token.transferFrom(anchorOwner, address(this), amount);
        token.transfer(_owner, amount);
        anchors[hash] = Anchor(now, anchorOwner, amount);
        emit AnchorExecuted(now, anchorOwner, amount, hash);
    }
```

The code meets the specification

Formal Verification Request 24

totalSupply

04, Jun 2019
5.24 ms

Line 25-27 in File ERC20.sol

```
25  /*@CTK totalSupply
26    @post __return == _totalSupply
27   */
```

Line 28-30 in File ERC20.sol

```
28  function totalSupply() public view returns (uint256) {
29    return _totalSupply;
30  }
```

The code meets the specification

Formal Verification Request 25

balanceOf

04, Jun 2019
5.33 ms

Line 37-39 in File ERC20.sol

```
37  /*@CTK balanceOf
38    @post __return == _balances[owner]
39   */
```

Line 40-42 in File ERC20.sol

```
40  function balanceOf(address owner) public view returns (uint256) {
41    return _balances[owner];
42  }
```

The code meets the specification

Formal Verification Request 26

allowance

04, Jun 2019
5.39 ms

Line 50-52 in File ERC20.sol

```
50 /*@CTK allowance
51   @post __return == _allowed[owner][spender]
52  */
```

Line 53-62 in File ERC20.sol

```
53 function allowance(
54   address owner,
55   address spender
56  )
57   public
58   view
59   returns (uint256)
60 {
61   return _allowed[owner][spender];
62 }
```

The code meets the specification

Formal Verification Request 27

transfer

04, Jun 2019
283.4 ms

Line 69-76 in File ERC20.sol

```
69 /*@CTK transfer
70   @tag assume_completion
71   @pre msg.sender != to
72   @post to != address(0)
73   @post value <= _balances[msg.sender]
74   @post __post._balances[to] == _balances[to] + value
75   @post __post._balances[msg.sender] == _balances[msg.sender] - value
76  */
```

Line 77-80 in File ERC20.sol

```
77 function transfer(address to, uint256 value) public returns (bool) {
78   _transfer(msg.sender, to, value);
79   return true;
80 }
```

The code meets the specification

Formal Verification Request 28

approve

04, Jun 2019
14.67 ms

Line 91-95 in File ERC20.sol

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

Line 96-102 in File ERC20.sol

```
function approve(address spender, uint256 value) public returns (bool) {
  require(spender != address(0));

  _allowed[msg.sender][spender] = value;
  emit Approval(msg.sender, spender, value);
  return true;
}
```

The code meets the specification

Formal Verification Request 29

transfer_from

04, Jun 2019
211.69 ms

Line 110-119 in File ERC20.sol

```
/*@CTK transfer_from
  @tag assume_completion
  @pre from != to
  @post to != address(0)
  @post value <= _allowed[from][msg.sender]
  @post __post._balances[from] == _balances[from] - value
  @post __post._balances[to] == _balances[to] + value
  @post __post._allowed[from][msg.sender] ==
    _allowed[from][msg.sender] - value
 */
```

Line 120-133 in File ERC20.sol

```
function transferFrom(
  address from,
  address to,
  uint256 value
)
  public
  returns (bool)
{
  require(value <= _allowed[from][msg.sender]);

  _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
  _transfer(from, to, value);
  return true;
}
```

The code meets the specification

Formal Verification Request 30

increaseAllowance

04, Jun 2019
39.84 ms

Line 144-149 in File ERC20.sol

```
144 /*@CTK increaseAllowance
145   @tag assume_completion
146   @post spender != address(0)
147   @post __post._allowed[msg.sender][spender] ==
148         _allowed[msg.sender][spender] + addedValue
149  */
```

Line 150-163 in File ERC20.sol

```
150 function increaseAllowance(
151   address spender,
152   uint256 addedValue
153 )
154   public
155   returns (bool)
156 {
157   require(spender != address(0));
158
159   _allowed[msg.sender][spender] = (
160     _allowed[msg.sender][spender].add(addedValue));
161   emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
162   return true;
163 }
```

The code meets the specification

Formal Verification Request 31

decreaseAllowance

04, Jun 2019
47.43 ms

Line 174-179 in File ERC20.sol

```
174 /*@CTK decreaseAllowance
175   @tag assume_completion
176   @post spender != address(0)
177   @post __post._allowed[msg.sender][spender] ==
178         _allowed[msg.sender][spender] - subtractedValue
179  */
```

Line 180-193 in File ERC20.sol

```
180 function decreaseAllowance(
181   address spender,
182   uint256 subtractedValue
183 )
184   public
185   returns (bool)
```

```
{
  require(spender != address(0));

  _allowed[msg.sender][spender] = (
    _allowed[msg.sender][spender].sub(subtractedValue));
  emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
  return true;
}
```

The code meets the specification

Formal Verification Request 32

_transfer

04, Jun 2019
100.24 ms

Line 201-208 in File ERC20.sol

```
/*@CTK _transfer
  @tag assume_completion
  @pre from != to
  @post to != address(0)
  @post value <= _balances[from]
  @post __post._balances[to] == _balances[to] + value
  @post __post._balances[from] == _balances[from] - value
 */
```

Line 209-216 in File ERC20.sol

```
function _transfer(address from, address to, uint256 value) internal {
  require(value <= _balances[from]);
  require(to != address(0));

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

The code meets the specification

Formal Verification Request 33

_mint

04, Jun 2019
73.11 ms

Line 225-230 in File ERC20.sol

```
/*@CTK _mint
  @tag assume_completion
  @post account != 0
  @post __post._totalSupply == _totalSupply + value
  @post __post._balances[account] == _balances[account] + value
 */
```

Line 231-236 in File ERC20.sol

```
function _mint(address account, uint256 value) internal {
  require(account != 0);
  _totalSupply = _totalSupply.add(value);
  _balances[account] = _balances[account].add(value);
  emit Transfer(address(0), account, value);
}
```

The code meets the specification

Formal Verification Request 34

_burn

04, Jun 2019
162.02 ms

Line 244-250 in File ERC20.sol

```
/*@CTK _burn
  @tag assume_completion
  @post account != 0
  @post value <= _balances[account]
  @post __post._totalSupply == _totalSupply - value
  @post __post._balances[account] == _balances[account] - value
 */
```

Line 251-258 in File ERC20.sol

```
function _burn(address account, uint256 value) internal {
  require(account != 0);
  require(value <= _balances[account]);

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

The code meets the specification

Formal Verification Request 35

_burnFrom

04, Jun 2019
270.23 ms

Line 267-273 in File ERC20.sol

```
/*@CTK _burnFrom
  @tag assume_completion
  @post value <= _allowed[account][msg.sender]
  @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
      value
  @post __post._totalSupply == _totalSupply - value
  @post __post._balances[account] == _balances[account] - value
 */
```

Line 274-282 in File ERC20.sol

```
274 function _burnFrom(address account, uint256 value) internal {
275   require(value <= _allowed[account][msg.sender]);
276
277   // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
278   // this function needs to emit an event with the updated approval.
279   _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
280     value);
281   _burn(account, value);
282 }
```

✅ The code meets the specification

Formal Verification Request 36

mint

📅 04, Jun 2019
⏱ 200.57 ms

Line 17-20 in File ERC20Mintable.sol

```
17 /*@CTK mint
18   @tag assume_completion
19   @post minters.bearer[msg.sender]
20  */
```

Line 21-31 in File ERC20Mintable.sol

```
21 function mint(
22   address to,
23   uint256 value
24 )
25   public
26   onlyMinter
27   returns (bool)
28 {
29   _mint(to, value);
30   return true;
31 }
```

✅ The code meets the specification

Formal Verification Request 37

ERC20Detailed

📅 04, Jun 2019
⏱ 9.53 ms

Line 16-20 in File ERC20Detailed.sol

```
16 /*@CTK ERC20Detailed
17   @post __post._name == name
18   @post __post._symbol == symbol
```

```
19    @post __post._decimals == decimals
20   */
```

Line 21-25 in File ERC20Detailed.sol

```
21  constructor(string name, string symbol, uint8 decimals) public {
22    _name = name;
23    _symbol = symbol;
24    _decimals = decimals;
25  }
```

✅ The code meets the specification

Formal Verification Request 38

name

📅 04, Jun 2019
⏱ 5.75 ms

Line 30-32 in File ERC20Detailed.sol

```
30  /*@CTK name
31    @post __post._name == _name
32   */
```

Line 33-35 in File ERC20Detailed.sol

```
33  function name() public view returns(string) {
34    return _name;
35  }
```

✅ The code meets the specification

Formal Verification Request 39

symbol

📅 04, Jun 2019
⏱ 5.68 ms

Line 40-42 in File ERC20Detailed.sol

```
40  /*@CTK symbol
41    @post __return == _symbol
42   */
```

Line 43-45 in File ERC20Detailed.sol

```
43  function symbol() public view returns(string) {
44    return _symbol;
45  }
```

✅ The code meets the specification

Formal Verification Request 40

decimals

04, Jun 2019
4.97 ms

Line 50-52 in File ERC20Detailed.sol

```
50 /*@CTK decimals
51   @post __return == _decimals
52  */
```

Line 53-55 in File ERC20Detailed.sol

```
53 function decimals() public view returns(uint8) {
54   return _decimals;
55 }
```

The code meets the specification

Formal Verification Request 41

ERC20Capped

04, Jun 2019
13.21 ms

Line 13-17 in File ERC20Capped.sol

```
13 /*@CTK ERC20Capped
14   @tag assume_completion
15   @post cap > 0
16   @post __post._cap == cap
17  */
```

Line 18-23 in File ERC20Capped.sol

```
18 constructor(uint256 cap)
19   public
20 {
21   require(cap > 0);
22   _cap = cap;
23 }
```

The code meets the specification

Formal Verification Request 42

cap

04, Jun 2019
5.19 ms

Line 28-30 in File ERC20Capped.sol

```
28 /*@CTK cap
29   @post __return == _cap
30  */
```

Line 31-33 in File ERC20Capped.sol

```
31 function cap() public view returns(uint256) {
32   return _cap;
33 }
```

The code meets the specification

Formal Verification Request 43

_mint

04, Jun 2019
471.79 ms

Line 35-40 in File ERC20Capped.sol

```
35 /*@CTK _mint
36   @tag assume_completion
37   @post __post._totalSupply == _totalSupply + value
38   @post __post._totalSupply <= _cap
39   @post __post._balances[account] == _balances[account] + value
40  */
```

Line 41-44 in File ERC20Capped.sol

```
41 function _mint(address account, uint256 value) internal {
42   require(totalSupply().add(value) <= _cap);
43   super._mint(account, value);
44 }
```

The code meets the specification

Formal Verification Request 44

burn

04, Jun 2019
209.81 ms

Line 15-19 in File ERC20Burnable.sol

```
15 /*@CTK burn
16   @tag assume_completion
17   @post __post._totalSupply == _totalSupply - value
18   @post __post._balances[msg.sender] == _balances[msg.sender] - value
19  */
```

Line 20-22 in File ERC20Burnable.sol

```
20 function burn(uint256 value) public {
21   _burn(msg.sender, value);
22 }
```

The code meets the specification

Formal Verification Request 45

burnFrom

04, Jun 2019
437.97 ms

Line 29-33 in File ERC20Burnable.sol

```
29 /*@CTK burnFrom
30   @tag assume_completion
31   @post __post._totalSupply == _totalSupply - value
32   @post __post._balances[from] == _balances[from] - value
33  */
```

Line 34-36 in File ERC20Burnable.sol

```
34 function burnFrom(address from, uint256 value) public {
35   _burnFrom(from, value);
36 }
```

The code meets the specification

Formal Verification Request 46

has

04, Jun 2019
13.79 ms

Line 48-52 in File Roles.sol

```
48 /*@CTK has
49   @tag assume_completion
50   @post account != address(0)
51   @post __return == role.bearer[account]
52  */
```

Line 53-60 in File Roles.sol

```
53 function has(Role storage role, address account)
54   internal
55   view
56   returns (bool)
57 {
58   require(account != address(0));
59   return role.bearer[account];
60 }
```

The code meets the specification

Formal Verification Request 47

isMinter

04, Jun 2019
49.34 ms

Line 22-25 in File MinterRole.sol

```
22 /*@CTK isMinter
23   @tag assume_completion
24   @post __return == minters.bearer[account]
25  */
```

Line 26-28 in File MinterRole.sol

```
26 function isMinter(address account) public view returns (bool) {
27   return minters.has(account);
28 }
```

✅ The code meets the specification

Formal Verification Request 48

ReentrancyGuard

📅 04, Jun 2019
⏱ 6.33 ms

Line 13-15 in File ReentrancyGuard.sol

```
13 /*@CTK ReentrancyGuard
14   @post __post._guardCounter == 1
15  */
```

Line 16-20 in File ReentrancyGuard.sol

```
16 constructor() internal {
17   // The counter starts at one to prevent changing it from zero to a non-zero
18   // value, which is a more expensive operation.
19   _guardCounter = 1;
20 }
```

✅ The code meets the specification

Formal Verification Request 49

SafeMath mul

📅 04, Jun 2019
⏱ 303.76 ms

Line 12-17 in File SafeMath.sol

```
12 /*@CTK "SafeMath mul"
13   @post (a > 0) && (((a * b) / a) != b) -> __reverted
14   @post __reverted -> (a > 0) && (((a * b) / a) != b)
15   @post !__reverted -> __return == a * b
16   @post !__reverted == !__has_overflow
17  */
```

Line 18-30 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 50

SafeMath div

04, Jun 2019
12.27 ms

Line 35-39 in File SafeMath.sol

```
35 /*@CTK "SafeMath div"
36   @post b != 0 -> !__reverted
37   @post !__reverted -> __return == a / b
38   @post !__reverted -> !__has_overflow
39  */
```

Line 40-46 in File SafeMath.sol

```
40 function div(uint256 a, uint256 b) internal pure returns (uint256) {
41   require(b > 0); // Solidity only automatically asserts when dividing by 0
42   uint256 c = a / b;
43   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
44
45   return c;
46 }
```

The code meets the specification

Formal Verification Request 51

SafeMath sub

04, Jun 2019
11.1 ms

Line 51-55 in File SafeMath.sol

```
51 /*@CTK "SafeMath sub"
52   @post (a < b) == __reverted
53   @post !__reverted -> __return == a - b
54   @post !__reverted -> !__has_overflow
55 */
```

Line 56-61 in File SafeMath.sol

```
56 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
57   require(b <= a);
58   uint256 c = a - b;
59
60   return c;
61 }
```

✅ The code meets the specification

Formal Verification Request 52

SafeMath add

📅 04, Jun 2019
⏱ 14.35 ms

Line 66-70 in File SafeMath.sol

```
66 /*@CTK "SafeMath add"
67   @post (a + b < a || a + b < b) == __reverted
68   @post !__reverted -> __return == a + b
69   @post !__reverted -> !__has_overflow
70  */
```

Line 71-76 in File SafeMath.sol

```
71 function add(uint256 a, uint256 b) internal pure returns (uint256) {
72   uint256 c = a + b;
73   require(c >= a);
74
75   return c;
76 }
```

✅ The code meets the specification

Formal Verification Request 53

SafeMath mod

📅 04, Jun 2019
⏱ 12.77 ms

Line 82-87 in File SafeMath.sol

```
82 /*@CTK "SafeMath mod"
83   @post (b == 0) == __reverted
84   @post !__reverted -> b != 0
85   @post !__reverted -> __return == a % b
86   @post !__reverted -> !__has_overflow
87  */
```

Line 88-91 in File SafeMath.sol

```
88 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
89   require(b != 0);
90   return a % b;
91 }
```

The code meets the specification

Formal Verification Request 54

Ownable

04, Jun 2019
5.65 ms

Line 20-22 in File Ownable.sol

```
20 /*@CTK Ownable
21   @post __post._owner == msg.sender
22  */
```

Line 23-26 in File Ownable.sol

```
23 constructor() internal {
24   _owner = msg.sender;
25   emit OwnershipTransferred(address(0), _owner);
26 }
```

The code meets the specification

Formal Verification Request 55

owner

04, Jun 2019
5.96 ms

Line 31-33 in File Ownable.sol

```
31 /*@CTK owner
32   @post __return == _owner
33  */
```

Line 34-36 in File Ownable.sol

```
34 function owner() public view returns(address) {
35   return _owner;
36 }
```

The code meets the specification

Formal Verification Request 56

isOwner

04, Jun 2019
7.45 ms

Line 49-51 in File Ownable.sol

```
49 /*@CTK isOwner
50   @post __return == (msg.sender == _owner)
51  */
```

Line 52-54 in File Ownable.sol

```
52 function isOwner() public view returns(bool) {
53   return msg.sender == _owner;
54 }
```

The code meets the specification

Formal Verification Request 57

renounceOwnership

04, Jun 2019
27.87 ms

Line 62-66 in File Ownable.sol

```
62 /*@CTK renounceOwnership
63   @tag assume_completion
64   @post _owner == msg.sender
65   @post __post._owner == address(0)
66  */
```

Line 67-70 in File Ownable.sol

```
67 function renounceOwnership() public onlyOwner {
68   emit OwnershipTransferred(_owner, address(0));
69   _owner = address(0);
70 }
```

The code meets the specification

Formal Verification Request 58

transferOwnership

04, Jun 2019
58.16 ms

Line 76-79 in File Ownable.sol

```
76 /*@CTK transferOwnership
77   @tag assume_completion
78   @post _owner == msg.sender
79  */
```

Line 80-82 in File Ownable.sol

```
80 function transferOwnership(address newOwner) public onlyOwner {
81   _transferOwnership(newOwner);
82 }
```

The code meets the specification

Formal Verification Request 59

_transferOwnership

04, Jun 2019
1.32 ms

Line 88-92 in File Ownable.sol

```
88  /*@CTK _transferOwnership
89    @tag assume_completion
90    @post newOwner != address(0)
91    @post __post._owner == newOwner
92   */
```

Line 93-97 in File Ownable.sol

```
93  function _transferOwnership(address newOwner) internal {
94    require(newOwner != address(0));
95    emit OwnershipTransferred(_owner, newOwner);
96    _owner = newOwner;
97  }
```

The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 7 in File FrenchIcoToken.sol

```
7 pragma solidity 0.5.8;
```

No compiler version found

INSECURE_COMPILER_VERSION

Line 7 in File FrenchIcoCorporate.sol

```
7 pragma solidity 0.5.8;
```

No compiler version found

TIMESTAMP_DEPENDENCY

Line 92 in File FrenchIcoCorporate.sol

```
92        members[addr].recordTime = now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 157 in File FrenchIcoCorporate.sol

```
157        members[msg.sender].recordTime = now;
```

"now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 7 in File FrenchIcoCrowdsale.sol

```
7 pragma solidity ^0.5.5;
```

Only these compiler versions are safe to compile your code: 0.5.6

TIMESTAMP_DEPENDENCY

Line 171 in File FrenchIcoCrowdsale.sol

```
171        require (_startTime > now, "too early");
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 209 in File FrenchIcoCrowdsale.sol

```
209        require (now >= ICOs[icoNumber].startTime && now <= ICOs[icoNumber].endTime, "
               ICO not running");
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 209 in File FrenchIcoCrowdsale.sol

```
209        require (now >= ICOs[icoNumber].startTime && now <= ICOs[icoNumber].endTime, "
               ICO not running");
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 309 in File FrenchIcoCrowdsale.sol

```
309        if (now <= endPresalePeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 311 in File FrenchIcoCrowdsale.sol

```
311        } else if (now <= endFirstPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 313 in File FrenchIcoCrowdsale.sol

```
313        } else if (now <= endSecondPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 333 in File FrenchIcoCrowdsale.sol

```
333        if (now >= endPresalePeriod && senderRole == ROLE_ANGEL && investors[msg.sender
               ].fundsDeposited.add(value) <= fico.getMaxAmount()) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 335 in File FrenchIcoCrowdsale.sol

```
335        } else if (now <= endPresalePeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 337 in File FrenchIcoCrowdsale.sol

```
337        } else if (now <= endFirstPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 339 in File FrenchIcoCrowdsale.sol

```
339        } else if (now <= endSecondPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 341 in File FrenchIcoCrowdsale.sol

```
341        } else if (now <= ICOs[icoNumber].endTime) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 366 in File FrenchIcoCrowdsale.sol

```
366        if (now <= endPresalePeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 368 in File FrenchIcoCrowdsale.sol

```
368        } else if (now <= endFirstPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 370 in File FrenchIcoCrowdsale.sol

```
370        } else if (now <= endSecondPeriod) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 372 in File FrenchIcoCrowdsale.sol

```
372        } else if (now <= ICOs[icoNumber].endTime) {
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 391 in File FrenchIcoCrowdsale.sol

```
391        return (ICOs[icoNumber].fundsDeposited >= ICOs[icoNumber].maxCap || now > ICOs[
               icoNumber].endTime);
```

! "now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 6 in File OwnablePayable.sol

```
6 pragma solidity 0.5.8;
```

! No compiler version found

INSECURE_COMPILER_VERSION

Line 8 in File FrenchIcoGateway.sol

```
8 pragma solidity ^0.5.5;
```

Only these compiler versions are safe to compile your code: 0.5.6

TIMESTAMP_DEPENDENCY

Line 132 in File FrenchIcoGateway.sol

```
132         anchors[hash] = Anchor(now, anchorOwner, amount);
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 133 in File FrenchIcoGateway.sol

```
133         emit AnchorExecuted(now, anchorOwner, amount, hash);
```

"now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File ERC20.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Mintable.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Detailed.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Capped.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Burnable.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File Roles.sol

```
pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File MinterRole.sol

```
pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File ReentrancyGuard.sol

```
pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

Source Code with CertiK Labels

File FrenchIcoToken.sol

```
/**
 * Author : FRENCH-ICO.com
 * Website : www.french-ico.com
 * Version : 9.7
 */

pragma solidity 0.5.8;

import "openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol";

interface FrenchIcoGateway {
    function orderFromToken(address, uint, address, uint, uint[] calldata, string
        calldata, address) external returns (bool);
}

/**
 * @title FRENCHICO TOKEN
 */
// contract FrenchIcoToken is ERC20Mintable {
contract FrenchIcoToken {

    event NewTokenFico(address owner, string copyright, string name, string symbol);

    uint8 public constant decimals = 18;
    string public name;
    string public symbol;

    /**
     * @param _symbol Token symbol
     * @param _name Token name
     */
    /*@CTK FrenchIcoToken
      @post __post.symbol == _symbol
      @post __post.name == _name
     */
    constructor(string memory _symbol, string memory _name) public {
        symbol = _symbol;
        name = _name;
        emit NewTokenFico(msg.sender, "Copyright FRENCHICO", name, symbol);
    }

    /**
     * Send tokens through a gateway
     *
     * @param gatewayAddr Gateway address
     * @param amount Tokens amount
     * @param orderId Order ID
     * @param instruction Instructions to send to smart contract
     * @param message message to send to smart contract
     * @param addr ETH address
     */
    function sendToGateway(
        address gatewayAddr,
        uint amount,
```

```
        uint orderId,
        uint[] calldata instruction,
        string calldata message,
        address addr
    )
        external
    {
        approve(address(gatewayAddr), amount);
        FrenchIcoGateway gateway = FrenchIcoGateway(address(gatewayAddr));
        gateway.orderFromToken(
            msg.sender,
            amount,
            address(this),
            orderId,
            instruction,
            message,
            addr
        );
    }
}
```

File FrenchIcoCorporate.sol

```
/**
 * Author : FRENCH-ICO.com
 * Website : www.french-ico.com
 * Version : 9.8
 */

pragma solidity 0.5.8;

import "./OwnablePayable.sol";

/**
 * @title FRENCH-ICO Corporate contract
 */
contract FrenchIcoCorporate is OwnablePayable {

    bool pauseAllContracts = false; // Used to pause all ICOs
    uint maxAmount; // Max ETH contribution for the role 1 (ANGEL)

    // Roles
    uint constant ROLE_NOT_REGISTERED = 0;
    uint constant ROLE_ANGEL = 1;
    uint constant ROLE_ANGEL_PREMIUM = 2;
    uint constant ROLE_ANGEL_PREMIUM_PRO = 3;

    // White list member
    struct Member {
        uint role; // Role
        uint countryCode; // Country Code
        uint maxAmountAllowed; // Max contribution allowed to this user
        uint recordTime; // Creation / modification time
        string comments; // Comments
    }
    mapping(address => Member) public members; // White list of members

    event ListUpdated(address addr, string key, uint value);

```

```
    constructor() public {
        maxAmount = 10 ether; // Initial max amount
        // Assign the role 3 (Angel Premium Pro) to the contract owner
        setRole(address(msg.sender), ROLE_ANGEL_PREMIUM_PRO, "admin");
    }

    /**
     * Pause/unpause all the ICOs
     */
    /*@CTK toggleGeneralPause
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.pauseAllContracts == !pauseAllContracts
     */
    function toggleGeneralPause() public onlyOwner {
        pauseAllContracts = !pauseAllContracts;
    }

    /**
     * Setup the max amount for role 1 (ANGEL)
     *
     * @param _maxAmount Max amount
     */
    /*@CTK setMaxAmount
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.maxAmount == _maxAmount
     */
    function setMaxAmount(uint _maxAmount) public onlyOwner {
        maxAmount = _maxAmount;
    }


    /**
     * Setup a member's role
     *
     * @param addr Member's address
     * @param _role Role
     * @param comments Free comments
     */
    /*@CTK setRole
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.members[addr].role == _role
      @post __post.members[addr].recordTime == now
      @post __post.members[addr].comments == comments
     */
    function setRole(
        address addr,
        uint _role,
        string memory comments
    )
        public onlyOwner
    {
        members[addr].role = _role;
        members[addr].recordTime = now;
        members[addr].comments = comments;
        emit ListUpdated(addr, "role", _role);
```

```
    }

    /**
     * Setup a member's country code
     *
     * @param addr Member's address
     * @param countryCode Country code
     */
    /*@CTK setCountryCode
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.members[addr].countryCode == countryCode
     */
    function setCountryCode(
        address addr,
        uint countryCode
    )
        public onlyOwner
    {
        members[addr].countryCode = countryCode;
        emit ListUpdated(addr, "countryCode", countryCode);
    }

    /**
     * Setup a member's max amount value
     *
     * @param addr Member's address
     * @param maxAmountAllowed Max amount allowed for the member
     */
    /*@CTK setMemberMaxAmountAllowed
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.members[addr].maxAmountAllowed == maxAmountAllowed
     */
    function setMemberMaxAmountAllowed(
        address addr,
        uint maxAmountAllowed
    )
        public onlyOwner
    {
        members[addr].maxAmountAllowed = maxAmountAllowed;
        emit ListUpdated(addr, "maxAmountAllowed", maxAmountAllowed);
    }

    /**
     * Add a member on the white list
     */
    /*@CTK addMember
      @tag assume_completion
      @post members[msg.sender].role == ROLE_NOT_REGISTERED
      @post __post.members[msg.sender].role == ROLE_ANGEL
      @post __post.members[msg.sender].countryCode == 0
      @post __post.members[msg.sender].maxAmountAllowed == 0
      @post __post.members[msg.sender].recordTime == now
      @post __post.members[msg.sender].comments == "new member"
     */
    function addMember() public payable {
        require (members[msg.sender].role == ROLE_NOT_REGISTERED, "user has to be new")
```

```
            ;
        uint newMemberRole = ROLE_ANGEL;
        members[msg.sender].role = newMemberRole;
        members[msg.sender].countryCode = 0;
        members[msg.sender].maxAmountAllowed = 0;
        members[msg.sender].recordTime = now;
        members[msg.sender].comments = "new member";
        _owner.transfer(msg.value);
        emit ListUpdated(msg.sender, "role", newMemberRole);
    }

    /* USABLE BY EXTERNAL CONTRACTS */

    /**
     * Are the ICO's contracts paused?
     *
     * @return bool
     */
    /*@CTK isGeneralPaused
      @post __return == pauseAllContracts
     */
    function isGeneralPaused() external view returns (bool) {
        return pauseAllContracts;
    }

    /**
     * Get a member's role
     *
     * @return Role
     */
    /*@CTK getRole
      @post __return == members[addr].role
     */
    function getRole(address addr) external view returns (uint) {
        return members[addr].role;
    }

    /**
     * Get a member's country code
     *
     * @return Country Code
     */
    /*@CTK getCountryCode
      @post __return == members[addr].countryCode
     */
    function getCountryCode(address addr) external view returns (uint) {
        return members[addr].countryCode;
    }

    /**
     * Get a member's maximum amount allowed
     *
     * @return Max Amount Allowed
     */
    /*@CTK getMemberMaxAmountAllowed
      @post __return == members[addr].maxAmountAllowed
     */
    function getMemberMaxAmountAllowed(address addr) external view returns (uint) {
```

```
        return members[addr].maxAmountAllowed;
    }

    /**
     * Get the FRENCHICO wallet ETH address
     *
     * @return FRENCHICO wallet ETH address
     */
    /*@CTK getWalletFrenchICO
      @post __return == _owner
     */
    function getWalletFrenchICO() external view returns (address payable) {
        return _owner;
    }

    /**
     * Get the max amount
     *
     * @return Max amount
     */
    /*@CTK getMaxAmount
      @post __return == maxAmount
     */
    function getMaxAmount() external view returns (uint) {
        return maxAmount;
    }
}
```

File FrenchIcoCrowdsale.sol

```
/**
 * Author : FRENCH-ICO.com
 * Website : www.french-ico.com
 * Version : 9.7
 */

pragma solidity ^0.5.5;

import "./OwnablePayable.sol";
import "./FrenchIco.sol";
import "./FrenchIcoToken.sol";
import "./SafeMath.sol";

/**
 * @title FRENCHICO Crowdsale contract
 */
contract FrenchIcoCrowdsale is OwnablePayable, FrenchIco {

    using SafeMath for uint256;
    FrenchIcoToken public token;

    /*
     * Parameters NOT adjustable by Project Owner
     * rate is conversion rate for 1 ETH = 1000 Tokens
     * maxCap is maximum founds in wei can be collected during the ICO
     * endPresalePeriod is the ICO's end presale period timestamp
     * endFirstPeriod is the ICO's end first period timestamp
     * endSecondPeriod is the ICO's end scond period timestamp
     * presaleBonus is purcentage of aditionnals tokens delivered during preseale
```

```
        period
     * firstPeriodBonusBonus is purcentage of aditionnals tokens delivered during first period
     * secondPeriodBonus is purcentage of aditionnals tokens delivered during second period
     * weiRaised is total founds collected during ICO in progress
     * presalPeriodCap is the max founds can be colleced during presale period in purcentage of maxCap
     * firstPeriodCap is the max founds can be colleced during fist period in purcentage of maxCap
     * secondPeriodCap is the max founds can be colleced during second period in purcentage of maxCap
     * tokenBooked is the quatity of tokens booked by the investors but not relaised yet
     * symbol is the token symbol choosen by the project owner
     * isNewIcoPossible true if new ICO is accepted
     */
    uint public rate = 1000;
    uint public endPresalePeriod;
    uint public endFirstPeriod;
    uint public endSecondPeriod;
    uint public constant presaleBonus = 50; // Bonus in percent
    uint public constant firstPeriodBonus = 20; // Bonus in percent
    uint public constant secondPeriodBonus = 10; // Bonus in percent
    uint public presalePeriodCap;
    uint public firstPeriodCap;
    uint public secondPeriodCap;
    uint public icoNumber;
    bool public isNewIcoPossible = true;

    // Roles
    uint constant ROLE_NOT_REGISTERED = 0;
    uint constant ROLE_ANGEL = 1;
    uint constant ROLE_ANGEL_PREMIUM = 2;
    uint constant ROLE_ANGEL_PREMIUM_PRO = 3;

    // ICOs statuses
    uint8 constant STATUS_IN_PROGRESS = 0;
    uint8 constant STATUS_GOAL_REACHED = 1;
    uint8 constant STATUS_GOAL_NOT_REACHED = 2;
    uint8 constant STATUS_READY_FOR_NEW_ICO = 3;

     // Investors
    struct Investor {
        uint fundsDeposited; // wei deposited by the Investor
        uint tokensBooked; // amount of tokens booked by the Investor but not released yet
    }
    mapping(address => Investor) public investors;

    // ICOs histories
    struct ICO {
        uint8 status; // ICO_Status 0 in progress / 1 goal reached / 2 goal not reached / 3 ready for an New ICO
        uint rate; // conversion rate for 1 ETH = 1000 Tokens
        uint minCap; // ICO min cap
        uint maxCap; // ICO max cap
        uint fundsDeposited; // Total funds deposited on the contract
```

```
        uint fundsRefund; // Total refunds
        uint startTime; // ICO start time
        uint endTime; // ICO end time
        uint tokensBooked; // Total tokens booked
        uint tokensReleased; // Total tokens released
    }
    mapping(uint => ICO) public ICOs;

    /*
     * Events
     * TokensBooked is the event published each time tokens are booked by investor
     * TokensReleased is the event published each time tokens are released
     * DepositRefund is the event published each time deposit are refud to a investor
     * MarketplaceDeployed is the event published when the Marketplace is deployed
     */

    event TokensBooked(address beneficiary, uint amount, uint deposit);
    event TokensReleased(address beneficiary, uint amount);
    event DepositRefund(address beneficiary, uint amount);
    event NewIco(uint _rate, uint _minCap, uint _startTime, uint _endTime);
    event Copyright(string copyright);

    /**
     * The function is launched only when the contract is deployed on the blockchain
     *
     * create a new token's smart contrat. Contract's address is recorded in token
     * launch the ICO according _minCap, _startTime, _endTime choosen by the project
       owner
     * transfer commission to walletFrenchICO address
     *
     * @param _name Token name
     * @param _symbol Token symbol
     * @param _minCap ICO min cap (in ETH)
     * @param _startTime ICO start time (timestamp)
     * @param _endTime ICO end time (timestamp)
     * @param _presalePeriodDuration Presale period duration (in seconds)
     * @param _firstPeriodDuration First period duration (in seconds)
     * @param _secondPeriodDuration Second period duration (in seconds)
     */
    /*@CTK FrenchIcoCrowdsale
      @tag assume_completion
      @post __post.icoNumber == 0
      @post __post.ICOs[__post.icoNumber].status == STATUS_READY_FOR_NEW_ICO
     */
    constructor(string memory _name, string memory _symbol, uint _minCap, uint
        _startTime, uint _endTime, uint _presalePeriodDuration, uint
        _firstPeriodDuration, uint _secondPeriodDuration) public payable {
        token = new FrenchIcoToken(_symbol, _name);

        icoNumber = 0;
        ICOs[icoNumber].status = STATUS_READY_FOR_NEW_ICO;
        // newICO(rate, _minCap, _startTime, _endTime, _presalePeriodDuration,
            _firstPeriodDuration, _secondPeriodDuration); // Create the new ICO
        emit Copyright("Copyright FRENCHICO");
    }

    /**
     * Fallback function executed if ETH are sent directly from a wallet to this
```

```
        contract
     */
    function() external payable {
        bookTokens();
    }

    /**
     * FRENCH-ICO.com accepts new ICO
     */
    /*@CTK setNewIcoPossible
      @tag assume_completion
      @post msg.sender == 0x1
      @post __post.isNewIcoPossible
     */
    function setNewIcoPossible() public {
        require (msg.sender == fico.getWalletFrenchICO());
        isNewIcoPossible = true;
    }

    /**
     * This function launch a new ICO
     * This function is payable
     * It's possible to launch a new ICO only if all balance = 0 ie all investors have been refunded or all funds have been collected
     * PeriodCap are percentage of maxCap. It's the funds able to collect during the period
     *
     * @param _rate New rate number of Tokens for 1 ETH
     * @param _minCap Minimum amount in ether wished by the project onwer. A minimum of 10 ethers are required to avoid small projects
     * @param _startTime ICO's start timestamp choosen by the project owner
     * @param _endTime ICO's end timestamp choosen by the project owner
     * @param _presalePeriodDuration Presale period duration (in seconds)
     * @param _firstPeriodDuration First period duration (in seconds)
     * @param _secondPeriodDuration Second period duration (in seconds)
     */
    function newICO(uint _rate, uint _minCap, uint _startTime, uint _endTime, uint _presalePeriodDuration, uint _firstPeriodDuration, uint _secondPeriodDuration) public payable isNotStoppedByFrenchIco onlyOwner {

        require (fico.getRole(msg.sender) >= ROLE_ANGEL_PREMIUM, "owner has to be KYC validated");
        require (isNewIcoPossible, "new ICO not accepted");
        require (ICOs[icoNumber].status == STATUS_READY_FOR_NEW_ICO, "previous ICO is not finished or funds collected not released");
        require (_minCap > 0, "error min Cap");
        require (_startTime > now, "too early");

        fico.getWalletFrenchICO().transfer(msg.value);

        rate = _rate;
        presalePeriodCap = 10; // in percent of the maxCap
        firstPeriodCap = 10; // + in percent of the maxCap
        secondPeriodCap = 20; // + in percent of the maxCap

        require (_endTime > (_startTime.add(_presalePeriodDuration).add(_firstPeriodDuration).add(_secondPeriodDuration)), "too early");

```

```
        icoNumber++;
        ICOs[icoNumber].status = STATUS_IN_PROGRESS;
        ICOs[icoNumber].rate = _rate;
        ICOs[icoNumber].minCap = _minCap.mul(1 ether);
        ICOs[icoNumber].maxCap = ICOs[icoNumber].minCap.mul(2);
        ICOs[icoNumber].startTime = _startTime;
        ICOs[icoNumber].endTime = _endTime;
        endPresalePeriod = ICOs[icoNumber].startTime.add(_presalePeriodDuration);
        endFirstPeriod = endPresalePeriod.add(_firstPeriodDuration);
        endSecondPeriod = endFirstPeriod.add(_secondPeriodDuration);
        presalePeriodCap = ICOs[icoNumber].maxCap.mul(presalePeriodCap).div(100);
        firstPeriodCap = (ICOs[icoNumber].maxCap.mul(firstPeriodCap).div(100)).add(
            presalePeriodCap);
        secondPeriodCap = (ICOs[icoNumber].maxCap.mul(secondPeriodCap).div(100)).add(
            firstPeriodCap);

        isNewIcoPossible = false;

        emit NewIco(_rate, _minCap, _startTime, _endTime);
    }

    /**
     * Book tokens by an investor
     * According the period in progress, allow investor to deposit ether to book
         tokens + bonus
     * To be able to depositi funds, investors must have a Valid Access according the
         KYC result
     */
    function bookTokens() public payable isNotStoppedByFrenchIco {
        require (msg.value > 0, "empty");
        require (isValidAccess(msg.value), "Control Access Denied");
        require (now >= ICOs[icoNumber].startTime && now <= ICOs[icoNumber].endTime, "
            ICO not running");
        require (!isFinished(), "ICO is finished" );
        require (!isCapReached(msg.value), "Cap Reached");
        require (ICOs[icoNumber].status == STATUS_IN_PROGRESS, "ICO is not in progress"
            );

        uint tokensAmount = msg.value.mul(rate); // Tokens amount (in wei)
        uint bonus = getCurrentBonus(); // Calculate bonus (in percentage)

        // Add the bonus
        tokensAmount = tokensAmount.add(tokensAmount.mul(bonus).div(100));

        // Store the funds and booked tokens by investor
        investors[msg.sender].fundsDeposited = investors[msg.sender].fundsDeposited.add
            (msg.value);
        investors[msg.sender].tokensBooked = investors[msg.sender].tokensBooked.add(
            tokensAmount);

        // Update the total funds and booked tokens
        ICOs[icoNumber].fundsDeposited = ICOs[icoNumber].fundsDeposited.add(msg.value);
        ICOs[icoNumber].tokensBooked = ICOs[icoNumber].tokensBooked.add(tokensAmount);

        emit TokensBooked(msg.sender, tokensAmount, msg.value);
    }

    /**
```

```
     * Refund the investor's deposits on its wallet
     * Works only if the ICO's goal is NOT reached on time
     * Investor tokens are destroyed
     *
     * @param beneficiary Investor's ETH address
     */
    function refundDeposit(address payable beneficiary) public isNotStoppedByFrenchIco {
        require (isFinished(), "ICO is not finished");
        require (ICOs[icoNumber].fundsDeposited < ICOs[icoNumber].minCap, "minCap is reached");
        require (investors[beneficiary].fundsDeposited > 0 || beneficiary == _owner, "none funds deposited");

        ICOs[icoNumber].status = STATUS_GOAL_NOT_REACHED; // ICO failed

        beneficiary.transfer(investors[beneficiary].fundsDeposited);
        ICOs[icoNumber].fundsRefund = ICOs[icoNumber].fundsRefund.add(investors[beneficiary].fundsDeposited);
        emit DepositRefund(beneficiary, investors[beneficiary].fundsDeposited);

        // Reset the investor funds and booked tokens
        investors[beneficiary].fundsDeposited = 0;
        investors[beneficiary].tokensBooked = 0;

        // If all funds are refunded, the ICO is ready for a new ICO
        if (ICOs[icoNumber].fundsRefund == ICOs[icoNumber].fundsDeposited) {
            ICOs[icoNumber].status = STATUS_READY_FOR_NEW_ICO;
        }
    }

    /**
     * Release tokens to the investor
     * Works only if the ICO's goal is reached on time
     * Transfer funds collected to the project owner's ETH address
     * Transfer commission to FRENCH-ICO.com
     * Tokens are minted only when the tokens are released
     *
     * @param beneficiary Investor's ETH address
     */
    function releaseTokens(address beneficiary) public isNotStoppedByFrenchIco {
        require (isFinished(), "ICO is not finished");
        require (ICOs[icoNumber].fundsDeposited >= ICOs[icoNumber].minCap, "minCap is not reached");
        require (investors[beneficiary].tokensBooked > 0 || beneficiary == _owner);

        ICOs[icoNumber].status = STATUS_GOAL_REACHED; // ICO successful

        token.mint(beneficiary, investors[beneficiary].tokensBooked);
        ICOs[icoNumber].tokensReleased = ICOs[icoNumber].tokensReleased.add(investors[beneficiary].tokensBooked);
        emit TokensReleased(beneficiary, investors[beneficiary].tokensBooked);

        // Reset the investor funds and booked tokens
        investors[beneficiary].fundsDeposited = 0;
        investors[beneficiary].tokensBooked = 0;

        // Transfer 95% of the funds to the ICO owner
```

```
        _owner.transfer(address(this).balance.mul(95).div(100));

        // Transfer the balance (5%) to FRENCH-ICO.com
        fico.getWalletFrenchICO().transfer(msg.value);

        // If all tokens are released, the ICO is ready for a new ICO
        if (ICOs[icoNumber].tokensReleased == ICOs[icoNumber].tokensBooked) {
            ICOs[icoNumber].status = STATUS_READY_FOR_NEW_ICO;
        }
    }

    /**
     * Calculate the current bonus according to the period in progress
     *
     * @return Bonus (in percentage)
     */
    /*@CTK getCurrentBonus
      @post now <= endPresalePeriod -> __return == presaleBonus
      @post now > endPresalePeriod && now <= endFirstPeriod -> __return ==
          firstPeriodBonus
      @post now > endPresalePeriod && now > endFirstPeriod && now <= endSecondPeriod
          -> __return == secondPeriodBonus
      @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod ->
           __return == 0
     */
    function getCurrentBonus() public view returns(uint) {
        uint bonus;

        if (now <= endPresalePeriod) {
            bonus = presaleBonus;
        } else if (now <= endFirstPeriod) {
            bonus = firstPeriodBonus;
        } else if (now <= endSecondPeriod) {
            bonus = secondPeriodBonus;
        } else {
            bonus = 0;
        }
        return bonus;
    }

    /**
     * Check if the ETH address used is recorded in the whitelist
     * KYC are not required if total of deposit is < maxAmount ether by the same
       address
     * nevertheless member has to be registered
     *
     * @param value ETH amount the investor wishes to transfer (in wei)
     * @return Is the access valid?
     */
    function isValidAccess(uint value) public view returns(bool) {
        bool access;

        uint senderRole = fico.getRole(msg.sender);
        if (now >= endPresalePeriod && senderRole == ROLE_ANGEL && investors[msg.sender
            ].fundsDeposited.add(value) <= fico.getMaxAmount()) {
            access = true;
        } else if (now <= endPresalePeriod) {
            access = (senderRole == ROLE_ANGEL_PREMIUM_PRO) && (investors[msg.sender].
```

```
                fundsDeposited.add(value) <= fico.getMemberMaxAmountAllowed(msg.sender)
                );
        } else if (now <= endFirstPeriod) {
            access = senderRole >= ROLE_ANGEL_PREMIUM;
        } else if (now <= endSecondPeriod) {
            access = senderRole >= ROLE_ANGEL_PREMIUM;
        } else if (now <= ICOs[icoNumber].endTime) {
            access = senderRole >= ROLE_ANGEL_PREMIUM;
        } else {
            access = false;
        }
        return access;
    }

    /**
     * Check if maxCap is reached according to the current period
     *
     * @param value ETH amount the investor wishes to transfer (in wei)
     * @return MaxCap reached?
     */
    /*@CTK isCapReached
      @tag assume_completion
      @post now <= endPresalePeriod -> __return == ICOs[icoNumber].fundsDeposited +
          value > presalePeriodCap
      @post now > endPresalePeriod && now <= endFirstPeriod -> __return == ICOs[
          icoNumber].fundsDeposited + value > firstPeriodCap
      @post now > endPresalePeriod && now > endFirstPeriod && now <= endSecondPeriod
          -> __return == ICOs[icoNumber].fundsDeposited + value > secondPeriodCap
      @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod &&
           now <= ICOs[icoNumber].endTime -> __return == ICOs[icoNumber].fundsDeposited
           > ICOs[icoNumber].maxCap
      @post now > endPresalePeriod && now > endFirstPeriod && now > endSecondPeriod &&
           now > ICOs[icoNumber].endTime -> !__return
     */
    function isCapReached(uint value) public view returns(bool) {
        bool goal;

        if (now <= endPresalePeriod) {
            goal = ICOs[icoNumber].fundsDeposited.add(value) > presalePeriodCap;
        } else if (now <= endFirstPeriod) {
            goal = ICOs[icoNumber].fundsDeposited.add(value) > firstPeriodCap;
        } else if (now <= endSecondPeriod) {
            goal = ICOs[icoNumber].fundsDeposited.add(value) > secondPeriodCap;
        } else if (now <= ICOs[icoNumber].endTime) {
            goal = ICOs[icoNumber].fundsDeposited > ICOs[icoNumber].maxCap;
        } else {
            goal = false;
        }
        return goal;
     }

    /**
     * Check if the ICO is finished
     *
     * @return ICO finished?
     */
    /*@CTK isFinished
      @post __return == (ICOs[icoNumber].fundsDeposited >=
```

```
                        ICOs[icoNumber].maxCap ||
                        now > ICOs[icoNumber].endTime)
     */
    function isFinished() public view returns(bool) {
        return (ICOs[icoNumber].fundsDeposited >= ICOs[icoNumber].maxCap || now > ICOs[
            icoNumber].endTime);
    }
}
```

File OwnablePayable.sol

```
/**
 * Author : FRENCH-ICO.com
 * Website : www.french-ico.com
 */

pragma solidity 0.5.8;

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

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

    /**
     * The OwnablePayable constructor sets the original `owner` of the contract
     * to the sender account
     */
    /*@CTK OwnablePayable
      @post __post._owner == msg.sender
     */
    constructor() internal {
        _owner = msg.sender;
        emit OwnershipTransferred(address(0), _owner);
    }


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

    /**
     * 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 payable newOwner) public onlyOwner {
       _transferOwnership(newOwner);
   }

   /**
    * 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 payable newOwner) internal {
       require(newOwner != address(0));
       emit OwnershipTransferred(_owner, newOwner);
       _owner = newOwner;
   }
}
```

File FrenchIcoGateway.sol

```
/**
 * Contract: FrenchIcoGateway
 * Author : FRENCH-ICO.com
 * Website : www.french-ico.com
 * Version : 9.7
 */

pragma solidity ^0.5.5;

import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";
import "./OwnablePayable.sol";

/**
 * @title FRENCH-ICO Gateway contract
 */
contract FrenchIcoGateway is OwnablePayable, ERC20 {

   using SafeMath for uint256;

   struct Order {
       address buyer;
       uint amount;
   }
   struct Anchor {
       uint anchorDate;
       address anchorOwner;
       uint amount;
   }
   mapping (string => Anchor) public anchors;
   mapping (uint => Order) public payment;

   event AnchorExecuted(uint anchorDate, address anchorOwner, uint amount, string
       hash);
```

```
    event PaidWithToken(address buyer, address token, uint amount, uint orderId);
    event OrderReceipt(address sender, uint amount, address tokenAddr, uint orderId,
        uint[] instruction, string message);

    /**
     * Transfer founds from another smart contract
     *
     * @param sender Sender
     * @param amount Amount
     * @param tokenAddr Token Address
     * @param orderId Order ID
     * @param instruction Order instruction
     * @param message Message
     * @param addr Address
     * @return result Result
     */
    /*CTK orderFromToken_1_too_long
      @tag assume_completion
      @pre instruction[0] == 1
      @post __post.payment[orderId].buyer == sender
      @post __post.payment[orderId].amount == amount
      @post result
     */
    /*CTK orderFromToken_2_too_long
      @tag assume_completion
      @pre instruction[0] == 2
      @post __post.anchors[message].anchorDate == now
      @post __post.anchors[message].anchorOwner == sender
      @post __post.anchors[message].amount == amount
      @post result
     */
    function orderFromToken(address sender, uint amount, address tokenAddr, uint
        orderId, uint[] calldata instruction, string calldata message, address addr)
        external returns (bool result) {
        if (instruction[0] == 1) {
            payOrder(sender, amount, tokenAddr, orderId);
        }
        if (instruction[0] == 2) {
            anchorHash(sender, amount, tokenAddr, message);
        }
        result = true;
        emit OrderReceipt(sender, amount, tokenAddr, orderId, instruction, message);
    }

    /**
     * Pay an order
     *
     * @param sender Sender
     * @param amount Amount
     * @param tokenAddr Token Address
     * @param orderId Order ID
     */
    /*@CTK payOrder
      @tag assume_completion
      @post payment[orderId].buyer == address(0x0)
      @post __post.payment[orderId].buyer == sender
      @post __post.payment[orderId].amount == amount
     */
```

```
    function payOrder(
        address sender,
        uint amount,
        address tokenAddr,
        uint orderId
    )
        internal
    {
        require (payment[orderId].buyer == address(0x00));
        ERC20 token = ERC20(tokenAddr);
        token.transferFrom(sender, address(this), amount);
        token.transfer(_owner, amount);
        payment[orderId] = Order(sender, amount);
        emit PaidWithToken(sender, msg.sender, amount, orderId);
    }

    /**
     * Anchor a hash on the blockchain
     *
     * @param anchorOwner Owner
     * @param amount Amount
     * @param tokenAddr Token Address
     * @param hash Hash to anchor
     */
    /*@CTK anchorHash
      @tag assume_completion
      @post anchors[hash].anchorOwner == msg.sender ||
            anchors[hash].anchorOwner == address(0x0)
      @post __post.anchors[hash].anchorDate == now
      @post __post.anchors[hash].anchorOwner == anchorOwner
      @post __post.anchors[hash].amount == amount
     */
    function anchorHash(
        address anchorOwner,
        uint amount,
        address tokenAddr,
        string memory hash
    )
        internal
    {
        require ((anchors[hash].anchorOwner == msg.sender) || (anchors[hash].
            anchorOwner == address(0x00)));
        ERC20 token = ERC20(tokenAddr);
        token.transferFrom(anchorOwner, address(this), amount);
        token.transfer(_owner, amount);
        anchors[hash] = Anchor(now, anchorOwner, amount);
        emit AnchorExecuted(now, anchorOwner, amount, hash);
    }

}
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20.sol

```
pragma solidity ^0.4.24;

import "./IERC20.sol";
import "../../math/SafeMath.sol";

/**
```

```
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
 * Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/
     master/smart_contract/FirstBloodToken.sol
 */
contract ERC20 is IERC20 {
  using SafeMath for uint256;

  mapping (address => uint256) private _balances;

  mapping (address => mapping (address => uint256)) private _allowed;

  uint256 private _totalSupply;

  /**
  * @dev 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 to != address(0)
  @post value <= _balances[msg.sender]
  @post __post._balances[to] == _balances[to] + value
  @post __post._balances[msg.sender] == _balances[msg.sender] - value
 */
function transfer(address to, uint256 value) public returns (bool) {
  _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.
 */
/*@CTK approve
  @tag assume_completion
  @post spender != address(0)
  @post __post._allowed[msg.sender][spender] == value
 */
function approve(address spender, uint256 value) public returns (bool) {
  require(spender != address(0));

  _allowed[msg.sender][spender] = value;
  emit Approval(msg.sender, spender, value);
  return true;
}

/**
 * @dev Transfer tokens from one address to another
 * @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 transfer_from
  @tag assume_completion
  @pre from != to
  @post to != address(0)
  @post value <= _allowed[from][msg.sender]
  @post __post._balances[from] == _balances[from] - value
  @post __post._balances[to] == _balances[to] + value
```

```
    @post __post._allowed[from][msg.sender] ==
      _allowed[from][msg.sender] - value
  */
  function transferFrom(
    address from,
    address to,
    uint256 value
  )
    public
    returns (bool)
  {
    require(value <= _allowed[from][msg.sender]);

    _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
    _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
   * @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 spender != address(0)
    @post __post._allowed[msg.sender][spender] ==
          _allowed[msg.sender][spender] + addedValue
   */
  function increaseAllowance(
    address spender,
    uint256 addedValue
  )
    public
    returns (bool)
  {
    require(spender != address(0));

    _allowed[msg.sender][spender] = (
      _allowed[msg.sender][spender].add(addedValue));
    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
    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
   * @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 spender != address(0)
  @post __post._allowed[msg.sender][spender] ==
        _allowed[msg.sender][spender] - subtractedValue
 */
function decreaseAllowance(
  address spender,
  uint256 subtractedValue
)
  public
  returns (bool)
{
  require(spender != address(0));

  _allowed[msg.sender][spender] = (
    _allowed[msg.sender][spender].sub(subtractedValue));
  emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
  return true;
}

/**
* @dev Transfer token for a specified addresses
* @param from The address to transfer from.
* @param to The address to transfer to.
* @param value The amount to be transferred.
*/
/*@CTK _transfer
  @tag assume_completion
  @pre from != to
  @post to != address(0)
  @post value <= _balances[from]
  @post __post._balances[to] == _balances[to] + value
  @post __post._balances[from] == _balances[from] - value
 */
function _transfer(address from, address to, uint256 value) internal {
  require(value <= _balances[from]);
  require(to != address(0));

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

/**
 * @dev Internal 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.
 */
/*@CTK _mint
  @tag assume_completion
  @post account != 0
  @post __post._totalSupply == _totalSupply + value
  @post __post._balances[account] == _balances[account] + value
 */
function _mint(address account, uint256 value) internal {
  require(account != 0);
```

```
    _totalSupply = _totalSupply.add(value);
    _balances[account] = _balances[account].add(value);
    emit Transfer(address(0), account, value);
  }

  /**
   * @dev Internal function that burns an amount of the token of a given
   * account.
   * @param account The account whose tokens will be burnt.
   * @param value The amount that will be burnt.
   */
  /*@CTK _burn
    @tag assume_completion
    @post account != 0
    @post value <= _balances[account]
    @post __post._totalSupply == _totalSupply - value
    @post __post._balances[account] == _balances[account] - value
   */
  function _burn(address account, uint256 value) internal {
    require(account != 0);
    require(value <= _balances[account]);

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

  /**
   * @dev Internal 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.
   * @param account The account whose tokens will be burnt.
   * @param value The amount that will be burnt.
   */
  /*@CTK _burnFrom
    @tag assume_completion
    @post value <= _allowed[account][msg.sender]
    @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
        value
    @post __post._totalSupply == _totalSupply - value
    @post __post._balances[account] == _balances[account] - value
   */
  function _burnFrom(address account, uint256 value) internal {
    require(value <= _allowed[account][msg.sender]);

    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
        ,
    // this function needs to emit an event with the updated approval.
    _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
      value);
    _burn(account, value);
  }
}
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol

```
pragma solidity ^0.4.24;

import "./ERC20.sol";
```

```
import "../../access/roles/MinterRole.sol";

/**
 * @title ERC20Mintable
 * @dev ERC20 minting logic
 */
contract ERC20Mintable is ERC20, MinterRole {
  /**
   * @dev Function to mint tokens
   * @param to The address that will receive the minted tokens.
   * @param value The amount of tokens to mint.
   * @return A boolean that indicates if the operation was successful.
   */
  /*@CTK mint
    @tag assume_completion
    @post minters.bearer[msg.sender]
   */
  function mint(
    address to,
    uint256 value
  )
    public
    onlyMinter
    returns (bool)
  {
    _mint(to, value);
    return true;
  }
}
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol

```
pragma solidity ^0.4.24;

import "./IERC20.sol";

/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract ERC20Detailed is IERC20 {
  string private _name;
  string private _symbol;
  uint8 private _decimals;

  /*@CTK ERC20Detailed
    @post __post._name == name
    @post __post._symbol == symbol
    @post __post._decimals == decimals
   */
  constructor(string name, string symbol, uint8 decimals) public {
    _name = name;
    _symbol = symbol;
    _decimals = decimals;
  }

  /**
```

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

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

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

File openzeppelin-solidity/contracts/token/ERC20/ERC20Capped.sol

```
pragma solidity ^0.4.24;

import "./ERC20Mintable.sol";

/**
 * @title Capped token
 * @dev Mintable token with a token cap.
 */
contract ERC20Capped is ERC20Mintable {

  uint256 private _cap;

  /*@CTK ERC20Capped
    @tag assume_completion
    @post cap > 0
    @post __post._cap == cap
   */
  constructor(uint256 cap)
    public
  {
    require(cap > 0);
    _cap = cap;
  }

  /**
   * @return the cap for the token minting.
   */
```

```
  /*@CTK cap
    @post __return == _cap
   */
  function cap() public view returns(uint256) {
    return _cap;
  }

  /*@CTK _mint
    @tag assume_completion
    @post __post._totalSupply == _totalSupply + value
    @post __post._totalSupply <= _cap
    @post __post._balances[account] == _balances[account] + value
   */
  function _mint(address account, uint256 value) internal {
    require(totalSupply().add(value) <= _cap);
    super._mint(account, value);
  }
}
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Burnable.sol

```
pragma solidity ^0.4.24;

import "./ERC20.sol";

/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destroyed).
 */
contract ERC20Burnable is ERC20 {

  /**
   * @dev Burns a specific amount of tokens.
   * @param value The amount of token to be burned.
   */
  /*@CTK burn
    @tag assume_completion
    @post __post._totalSupply == _totalSupply - value
    @post __post._balances[msg.sender] == _balances[msg.sender] - value
   */
  function burn(uint256 value) public {
    _burn(msg.sender, value);
  }

  /**
   * @dev Burns a specific amount of tokens from the target address and decrements
       allowance
   * @param from address The address which you want to send tokens from
   * @param value uint256 The amount of token to be burned
   */
  /*@CTK burnFrom
    @tag assume_completion
    @post __post._totalSupply == _totalSupply - value
    @post __post._balances[from] == _balances[from] - value
   */
  function burnFrom(address from, uint256 value) public {
    _burnFrom(from, value);
  }
}
```

File openzeppelin-solidity/contracts/access/Roles.sol

```
pragma solidity ^0.4.24;

/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
library Roles {
  struct Role {
    mapping (address => bool) bearer;
  }

  /**
   * @dev give an account access to this role
   */
  /*CTK add
    @tag assume_completion
    @post account != address(0)
    @post !role.bearer[account]
    @post __post.role.bearer[account]
   */
  function add(Role storage role, address account) internal {
    require(account != address(0));
    require(!has(role, account));

    role.bearer[account] = true;
  }

  /**
   * @dev remove an account's access to this role
   */
  /*CTK remove
    @tag assume_completion
    @post account != address(0)
    @post role.bearer[account]
    @post !__post.role.bearer[account]
   */
  function remove(Role storage role, address account) internal {
    require(account != address(0));
    require(has(role, account));

    role.bearer[account] = false;
  }

  /**
   * @dev check if an account has this role
   * @return bool
   */
  /*@CTK has
    @tag assume_completion
    @post account != address(0)
    @post __return == role.bearer[account]
   */
  function has(Role storage role, address account)
    internal
    view
    returns (bool)
  {
```

```
    require(account != address(0));
    return role.bearer[account];
  }
}
```

File openzeppelin-solidity/contracts/access/roles/MinterRole.sol

```
pragma solidity ^0.4.24;

import "../Roles.sol";

contract MinterRole {
  using Roles for Roles.Role;

  event MinterAdded(address indexed account);
  event MinterRemoved(address indexed account);

  Roles.Role private minters;

  constructor() internal {
    _addMinter(msg.sender);
  }

  modifier onlyMinter() {
    require(isMinter(msg.sender));
    _;
  }

  /*@CTK isMinter
    @tag assume_completion
    @post __return == minters.bearer[account]
   */
  function isMinter(address account) public view returns (bool) {
    return minters.has(account);
  }

  function addMinter(address account) public onlyMinter {
    _addMinter(account);
  }

  function renounceMinter() public {
    _removeMinter(msg.sender);
  }

  function _addMinter(address account) internal {
    minters.add(account);
    emit MinterAdded(account);
  }

  function _removeMinter(address account) internal {
    minters.remove(account);
    emit MinterRemoved(account);
  }
}
```

File openzeppelin-solidity/contracts/utils/ReentrancyGuard.sol

```
pragma solidity ^0.4.24;

/**
```

```
 * @title Helps contracts guard against reentrancy attacks.
 * @dev If you mark a function ‘nonReentrant‘, you should also
 * mark it ‘external‘.
 */
contract ReentrancyGuard {

  /// @dev counter to allow mutex lock with only one SSTORE operation
  uint256 private _guardCounter;

  /*@CTK ReentrancyGuard
    @post __post._guardCounter == 1
   */
  constructor() internal {
    // The counter starts at one to prevent changing it from zero to a non-zero
    // value, which is a more expensive operation.
    _guardCounter = 1;
  }

  /**
   * @dev Prevents a contract from calling itself, directly or indirectly.
   * Calling a ‘nonReentrant‘ function from another ‘nonReentrant‘
   * function is not supported. It is possible to prevent this from happening
   * by making the ‘nonReentrant‘ function external, and make it call a
   * ‘private‘ function that does the actual work.
   */
  modifier nonReentrant() {
    _guardCounter += 1;
    uint256 localCounter = _guardCounter;
    _;
    require(localCounter == _guardCounter);
  }

}
```

File openzeppelin-solidity/contracts/math/SafeMath.sol

```
pragma solidity ^0.4.24;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {

  /**
  * @dev Multiplies two numbers, reverts on overflow.
  */
  /*@CTK "SafeMath mul"
    @post (a > 0) && (((a * b) / a) != b) -> __reverted
    @post __reverted -> (a > 0) && (((a * b) / a) != b)
    @post !__reverted -> __return == a * b
    @post !__reverted == !__has_overflow
   */
  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 numbers 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) {
  require(b > 0); // Solidity only automatically asserts when dividing by 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 numbers, 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 numbers, reverts on overflow.
*/
/*@CTK "SafeMath add"
  @post (a + b < a || a + b < b) == __reverted
  @post !__reverted -> __return == a + b
  @post !__reverted -> !__has_overflow
 */
function add(uint256 a, uint256 b) internal pure returns (uint256) {
  uint256 c = a + b;
  require(c >= a);

  return c;
}

/**
* @dev Divides two numbers and returns the remainder (unsigned integer modulo),
```

```
  * reverts when dividing by zero.
  */
  /*@CTK "SafeMath mod"
    @post (b == 0) == __reverted
    @post !__reverted -> b != 0
    @post !__reverted -> __return == a % b
    @post !__reverted -> !__has_overflow
   */
  function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b != 0);
    return a % b;
  }
}
```

File openzeppelin-solidity/contracts/ownership/Ownable.sol

```
pragma solidity ^0.4.24;

/**
 * @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 private _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() internal {
    _owner = msg.sender;
    emit OwnershipTransferred(address(0), _owner);
  }

  /**
   * @return the address of the owner.
   */
  /*@CTK owner
    @post __return == _owner
   */
  function owner() public view returns(address) {
    return _owner;
  }

  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
```

```
    require(isOwner());
    _;
  }

  /**
   * @return true if `msg.sender` is the owner of the contract.
   */
  /*@CTK isOwner
    @post __return == (msg.sender == _owner)
   */
  function isOwner() public view returns(bool) {
    return 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 _owner == msg.sender
    @post __post._owner == address(0)
   */
  function renounceOwnership() public onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
  }

  /**
   * @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
   */
  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));
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
  }
}
```