CertiK Audit Report For TauschBloc

TAUSCH BLOC
Shopping mall blockchain

Request Date: 2019-09-05
Revision Date: 2019-09-12
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

About CertiK

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

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

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

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

Executive Summary

This report has been prepared as the product of the Smart Contract Audit request by TauschBloc. This audit was conducted to discover issues and vulnerabilities in the source code of TauschBloc'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 issue 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 conditions, or it could affect the security standard by lost of access control.

Low

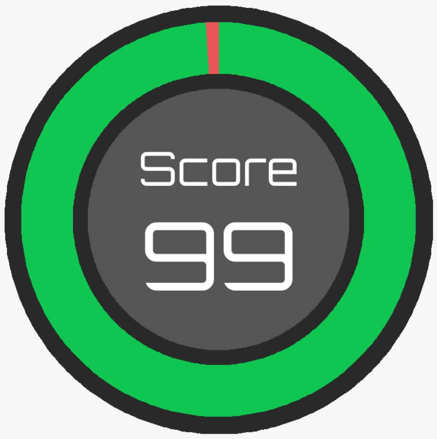
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

Testing Summary

PASS

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

Sep 06, 2019



Type of Issues

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

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

- **tauschbloc.sol**
  `9d4a11a9cd90413d0524751196a18c6f3d56eb63b457756b6b9968b00e203f98`

Summary

CertiK was chosen by TauschBloc to audit the design and implementation of its soon to be released smart contract. To ensure comprehensive protection, the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Static Analysis Results

INSECURE_COMPILER_VERSION

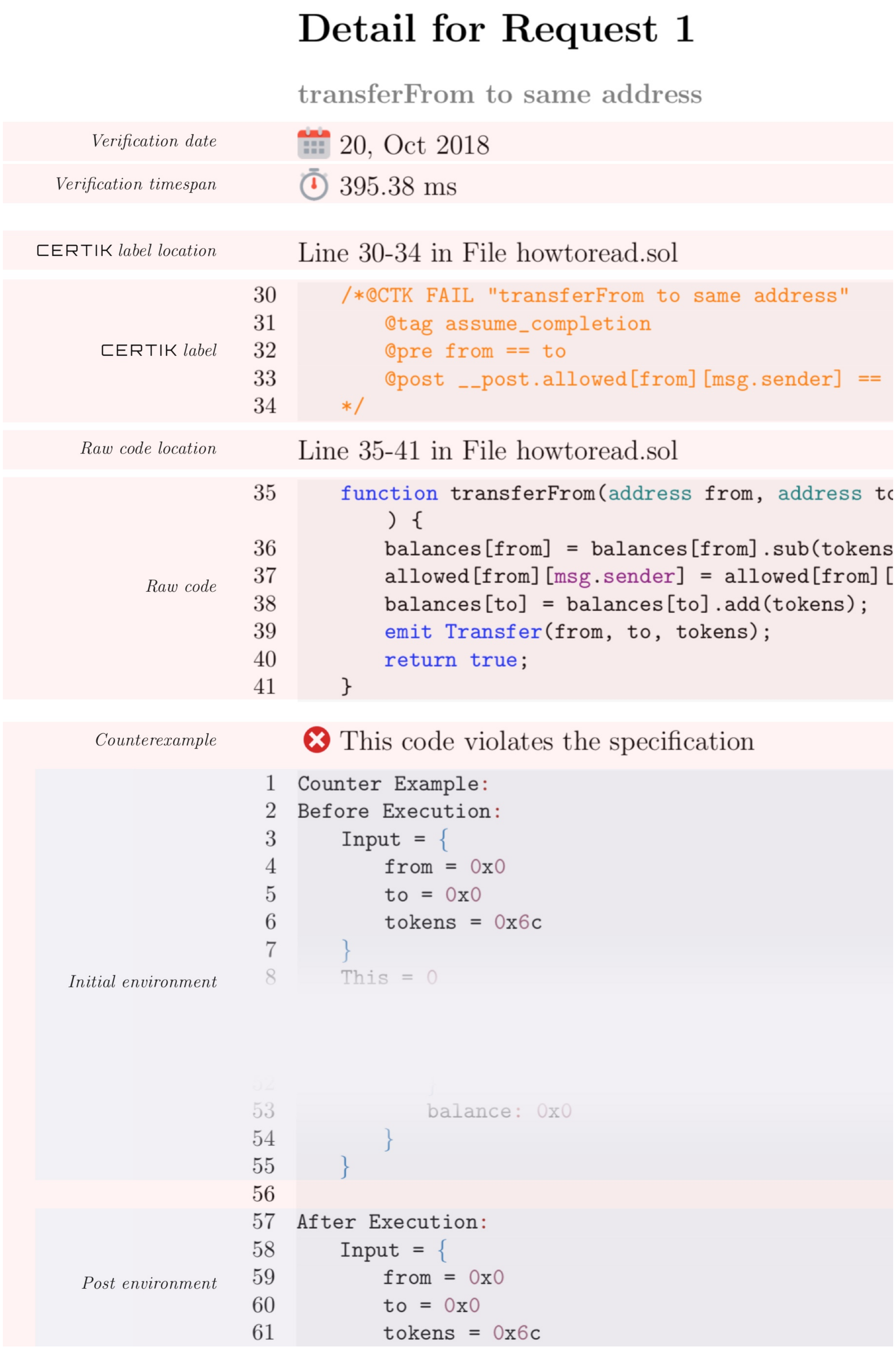
Line 5 in File tauschbloc.sol

```
pragma solidity ^0.5.2;
```

Only these compiler versions are safe to compile your code: 0.5.10

Formal Verification Results

How to read



Detail for Request 1

transferFrom to same address

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

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

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

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

Post environment

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

Formal Verification Request 1

SafeMath mul

📅 06, Sep 2019
⏱ 418.55 ms

Line 34-40 in File tauschbloc.sol

```
    /*@CTK "SafeMath mul"
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
     */
```

Line 41-53 in File tauschbloc.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 2

SafeMath div

📅 06, Sep 2019
⏱ 16.98 ms

Line 58-64 in File tauschbloc.sol

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

Line 65-72 in File tauschbloc.sol

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

```

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 3

SafeMath sub

06, Sep 2019
14.76 ms

Line 77-83 in File tauschbloc.sol

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

Line 84-89 in File tauschbloc.sol

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

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 4

SafeMath add

06, Sep 2019
19.94 ms

Line 94-100 in File tauschbloc.sol

```
     /*@CTK "SafeMath add"
       @post (a + b < a || a + b < b) == __reverted
       @post !__reverted -> __return == a + b
       @post !__reverted -> !__has_overflow
       @post !(__has_buf_overflow)
       @post !(__has_assertion_failure)
     */
```

Line 101-106 in File tauschbloc.sol

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

        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 5

SafeMath div

06, Sep 2019
15.0 ms

Line 112-118 in File tauschbloc.sol

```
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
    */
```

Line 119-122 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 6

If method completes, integer overflow would not happen.

06, Sep 2019
5.79 ms

Line 138 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 144-146 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 7

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.4 ms

Line 139 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 144-146 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 8

Method will not encounter an assertion failure.

06, Sep 2019
0.41 ms

Line 140 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 144-146 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 9

totalSupply correctness

06, Sep 2019
0.41 ms

Line 141-143 in File tauschbloc.sol

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

Line 144-146 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

06, Sep 2019
5.8 ms

Line 153 in File tauschbloc.sol

```
153      //@CTK NO_OVERFLOW
```

Line 159-161 in File tauschbloc.sol

```
159     function balanceOf(address owner) public view returns (uint256) {
160         return _balances[owner];
161     }
```

The code meets the specification.

Formal Verification Request 11

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.42 ms

Line 154 in File tauschbloc.sol

```
154      //@CTK NO_BUF_OVERFLOW
```

Line 159-161 in File tauschbloc.sol

```
159     function balanceOf(address owner) public view returns (uint256) {
160         return _balances[owner];
161     }
```

The code meets the specification.

Formal Verification Request 12

Method will not encounter an assertion failure.

06, Sep 2019
0.47 ms

Line 155 in File tauschbloc.sol

```
155      //@CTK NO_ASF
```

Line 159-161 in File tauschbloc.sol

```
159     function balanceOf(address owner) public view returns (uint256) {
160         return _balances[owner];
161     }
```

The code meets the specification.

Formal Verification Request 13

balanceOf correctness

06, Sep 2019
0.64 ms

Line 156-158 in File tauschbloc.sol

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

Line 159-161 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 14

If method completes, integer overflow would not happen.

06, Sep 2019
8.29 ms

Line 169 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 175-177 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 15

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.36 ms

Line 170 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 175-177 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 16

Method will not encounter an assertion failure.

06, Sep 2019
0.46 ms

Line 171 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 175-177 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 17

allowance correctness

06, Sep 2019
0.53 ms

Line 172-174 in File tauschbloc.sol

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

Line 175-177 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 18

If method completes, integer overflow would not happen.

06, Sep 2019
162.2 ms

Line 184 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 187-190 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 19

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
4.31 ms

Line 185 in File tauschbloc.sol

```
185     //@CTK NO_BUF_OVERFLOW
```

Line 187-190 in File tauschbloc.sol

```
187    function transfer(address to, uint256 value) public returns (bool) {
188        _transfer(msg.sender, to, value);
189        return true;
190    }
```

✓ The code meets the specification.

Formal Verification Request 20

Method will not encounter an assertion failure.

06, Sep 2019
3.78 ms

Line 186 in File tauschbloc.sol

```
186     //@CTK NO_ASF
```

Line 187-190 in File tauschbloc.sol

```
187    function transfer(address to, uint256 value) public returns (bool) {
188        _transfer(msg.sender, to, value);
189        return true;
190    }
```

✓ The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

06, Sep 2019
63.93 ms

Line 201 in File tauschbloc.sol

```
201     //@CTK NO_OVERFLOW
```

Line 204-207 in File tauschbloc.sol

```
204    function approve(address spender, uint256 value) public returns (bool) {
205        _approve(msg.sender, spender, value);
206        return true;
207    }
```

✓ The code meets the specification.

Formal Verification Request 22

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.71 ms

Line 202 in File tauschbloc.sol

```
202    //@CTK NO_BUF_OVERFLOW
```

Line 204-207 in File tauschbloc.sol

```
204    function approve(address spender, uint256 value) public returns (bool) {
205        _approve(msg.sender, spender, value);
206        return true;
207    }
```

✓ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

06, Sep 2019
0.66 ms

Line 203 in File tauschbloc.sol

```
203    //@CTK NO_ASF
```

Line 204-207 in File tauschbloc.sol

```
204    function approve(address spender, uint256 value) public returns (bool) {
205        _approve(msg.sender, spender, value);
206        return true;
207    }
```

✓ The code meets the specification.

Formal Verification Request 24

If method completes, integer overflow would not happen.

06, Sep 2019
152.64 ms

Line 217 in File tauschbloc.sol

```
217    //@CTK NO_OVERFLOW
```

Line 220-224 in File tauschbloc.sol

```
220    function transferFrom(address from, address to, uint256 value) public returns (
           bool) {
221        _transfer(from, to, value);
222        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
223        return true;
224    }
```

✓ The code meets the specification.

Formal Verification Request 25

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
7.41 ms

Line 218 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 220-224 in File tauschbloc.sol

```
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

06, Sep 2019
7.57 ms

Line 219 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 220-224 in File tauschbloc.sol

```
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 27

If method completes, integer overflow would not happen.

06, Sep 2019
73.82 ms

Line 236 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 239-242 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 28

Buffer overflow / array index out of bound would never happen.

📅 06, Sep 2019
⏱ 0.97 ms

Line 237 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 239-242 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 29

Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 1.01 ms

Line 238 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 239-242 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 30

If method completes, integer overflow would not happen.

📅 06, Sep 2019
⏱ 67.58 ms

Line 254 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 257-260 in File tauschbloc.sol

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

✅ The code meets the specification.

Formal Verification Request 31

Buffer overflow / array index out of bound would never happen.

📅 06, Sep 2019
⏱ 1.04 ms

Line 255 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 257-260 in File tauschbloc.sol

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

✅ The code meets the specification.

Formal Verification Request 32

Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 1.01 ms

Line 256 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 257-260 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

06, Sep 2019
8.33 ms

Line 268 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 278-284 in File tauschbloc.sol

```
function _transfer(address from, address to, uint256 value) internal {
    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 34

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
3.83 ms

Line 269 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 278-284 in File tauschbloc.sol

```
function _transfer(address from, address to, uint256 value) internal {
    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 35

Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 3.54 ms

Line 270 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 278-284 in File tauschbloc.sol

```
   function _transfer(address from, address to, uint256 value) internal {
       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 36

_transfer correctness

📅 06, Sep 2019
⏱ 81.61 ms

Line 271-277 in File tauschbloc.sol

```
    /*@CTK "_transfer correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != from -> __post._balances[from] == _balances[from] - value
      @post to != from -> __post._balances[to] == _balances[to] + value
      @post to == from -> __post._balances[from] == _balances[from]
     */
```

Line 278-284 in File tauschbloc.sol

```
   function _transfer(address from, address to, uint256 value) internal {
       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 37

If method completes, integer overflow would not happen.

📅 06, Sep 2019
⏱ 80.4 ms

Line 293 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 302-308 in File tauschbloc.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(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 38

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
4.05 ms

Line 294 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 302-308 in File tauschbloc.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(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 39

Method will not encounter an assertion failure.

06, Sep 2019
3.46 ms

Line 295 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 302-308 in File tauschbloc.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(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 40

_mint correctness

06, Sep 2019
23.39 ms

Line 296-301 in File tauschbloc.sol

```
    /*@CTK "_mint correctness"
      @tag assume_completion
      @post account != 0x0
      @post __post._balances[account] == _balances[account] + value
      @post __post._totalSupply == _totalSupply + value
     */
```

Line 302-308 in File tauschbloc.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(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 41

If method completes, integer overflow would not happen.

06, Sep 2019
71.11 ms

Line 316 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 325-331 in File tauschbloc.sol

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

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

✓ The code meets the specification.

Formal Verification Request 42

Buffer overflow / array index out of bound would never happen.

📅 06, Sep 2019
⏱ 3.45 ms

Line 317 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 325-331 in File tauschbloc.sol

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

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

✓ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 3.42 ms

Line 318 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 325-331 in File tauschbloc.sol

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

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

✓ The code meets the specification.

Formal Verification Request 44

_burn correctness

📅 06, Sep 2019
⏱ 45.16 ms

Line 319-324 in File tauschbloc.sol

```
    /*@CTK "_burn correctness"
      @tag assume_completion
      @post account != 0x0
      @post __post._balances[account] == _balances[account] - value
      @post __post._totalSupply == _totalSupply - value
     */
```

Line 325-331 in File tauschbloc.sol

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

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

✓ The code meets the specification.

Formal Verification Request 45

_approve

06, Sep 2019
2.76 ms

Line 339-344 in File tauschbloc.sol

```
  /*@CTK _approve
    @tag assume_completion
    @post spender != 0x0
    @post owner != 0x0
    @post __post._allowed[owner][spender] == value
  */
```

Line 345-351 in File tauschbloc.sol

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

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

✓ The code meets the specification.

Formal Verification Request 46

If method completes, integer overflow would not happen.

06, Sep 2019
142.23 ms

Line 361 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 364-367 in File tauschbloc.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✓ The code meets the specification.

Formal Verification Request 47

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
7.15 ms

Line 362 in File tauschbloc.sol

```
     //@CTK NO_BUF_OVERFLOW
```

Line 364-367 in File tauschbloc.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✓ The code meets the specification.

Formal Verification Request 48

Method will not encounter an assertion failure.

06, Sep 2019
6.2 ms

Line 363 in File tauschbloc.sol

```
     //@CTK NO_ASF
```

Line 364-367 in File tauschbloc.sol

```
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
```

✓ The code meets the specification.

Formal Verification Request 49

Roles has correctness

06, Sep 2019
15.61 ms

Line 405-408 in File tauschbloc.sol

```
    /*@CTK "Roles has correctness"
      @post account == 0x0 -> __reverted
      @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
     */
```

Line 409-412 in File tauschbloc.sol

```
    function has(Role storage role, address account) internal view returns (bool) {
        require(account != address(0));
        return role.bearer[account];
    }
```

✓ The code meets the specification.

Formal Verification Request 50

If method completes, integer overflow would not happen.

06, Sep 2019
19.3 ms

Line 423 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 426-428 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 51

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.52 ms

Line 424 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 426-428 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 52

Method will not encounter an assertion failure.

06, Sep 2019
0.45 ms

Line 425 in File tauschbloc.sol

```
425     //@CTK NO_ASF
```

Line 426-428 in File tauschbloc.sol

```
426     constructor () internal {
427         _addMinter(msg.sender);
428     }
```

The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

06, Sep 2019
34.77 ms

Line 434 in File tauschbloc.sol

```
434     //@CTK NO_OVERFLOW
```

Line 441-443 in File tauschbloc.sol

```
441     function isMinter(address account) public view returns (bool) {
442         return _minters.has(account);
443     }
```

The code meets the specification.

Formal Verification Request 54

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.62 ms

Line 435 in File tauschbloc.sol

```
435     //@CTK NO_BUF_OVERFLOW
```

Line 441-443 in File tauschbloc.sol

```
441     function isMinter(address account) public view returns (bool) {
442         return _minters.has(account);
443     }
```

The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

06, Sep 2019
0.63 ms

Line 436 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 441-443 in File tauschbloc.sol

```
    function isMinter(address account) public view returns (bool) {
        return _minters.has(account);
    }
```

✓ The code meets the specification.

Formal Verification Request 56

isMinter correctness

06, Sep 2019
1.61 ms

Line 437-440 in File tauschbloc.sol

```
    /*@CTK "isMinter correctness"
      @post account == 0x0 -> __reverted
      @post account != 0x0 -> !__reverted && __return == _minters.bearer[account]
     */
```

Line 441-443 in File tauschbloc.sol

```
    function isMinter(address account) public view returns (bool) {
        return _minters.has(account);
    }
```

✓ The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

06, Sep 2019
66.4 ms

Line 444 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 447-449 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 58

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.93 ms

Line 445 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 447-449 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.

06, Sep 2019
0.9 ms

Line 446 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 447-449 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 60

If method completes, integer overflow would not happen.

06, Sep 2019
18.81 ms

Line 450 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 453-455 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 61

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.49 ms

Line 451 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 453-455 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 62

Method will not encounter an assertion failure.

06, Sep 2019
0.47 ms

Line 452 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 453-455 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 63

If method completes, integer overflow would not happen.

06, Sep 2019
257.43 ms

Line 480 in File tauschbloc.sol

```
     //@CTK NO_OVERFLOW
```

Line 489-492 in File tauschbloc.sol

```
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 64

Buffer overflow / array index out of bound would never happen.

📅 06, Sep 2019
⏱ 5.63 ms

Line 481 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 489-492 in File tauschbloc.sol

```
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 65

Method will not encounter an assertion failure.

📅 06, Sep 2019
⏱ 5.43 ms

Line 482 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 489-492 in File tauschbloc.sol

```
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 66

mint

📅 06, Sep 2019
⏱ 174.98 ms

Line 483-488 in File tauschbloc.sol

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

Line 489-492 in File tauschbloc.sol

```
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 67

If method completes, integer overflow would not happen.

06, Sep 2019
149.63 ms

Line 501 in File tauschbloc.sol

```
501     //@CTK NO_OVERFLOW
```

Line 504-506 in File tauschbloc.sol

```
504     function burn(uint256 value) public {
505         _burn(msg.sender, value);
506     }
```

✓ The code meets the specification.

Formal Verification Request 68

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
3.88 ms

Line 502 in File tauschbloc.sol

```
502     //@CTK NO_BUF_OVERFLOW
```

Line 504-506 in File tauschbloc.sol

```
504     function burn(uint256 value) public {
505         _burn(msg.sender, value);
506     }
```

✓ The code meets the specification.

Formal Verification Request 69

Method will not encounter an assertion failure.

06, Sep 2019
3.87 ms

Line 503 in File tauschbloc.sol

```
503     //@CTK NO_ASF
```

Line 504-506 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 70

If method completes, integer overflow would not happen.

06, Sep 2019
311.59 ms

Line 513 in File tauschbloc.sol

```
     //@CTK NO_OVERFLOW
```

Line 516-518 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 71

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
7.26 ms

Line 514 in File tauschbloc.sol

```
     //@CTK NO_BUF_OVERFLOW
```

Line 516-518 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 72

Method will not encounter an assertion failure.

06, Sep 2019
6.61 ms

Line 515 in File tauschbloc.sol

```
     //@CTK NO_ASF
```

Line 516-518 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 73

If method completes, integer overflow would not happen.

06, Sep 2019
10.36 ms

Line 526 in File tauschbloc.sol

```
//@CTK NO_OVERFLOW
```

Line 534-538 in File tauschbloc.sol

```
constructor (string memory name, string memory symbol, uint8 decimals) public {
    _name = name;
    _symbol = symbol;
    _decimals = decimals;
}
```

✓ The code meets the specification.

Formal Verification Request 74

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.41 ms

Line 527 in File tauschbloc.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 534-538 in File tauschbloc.sol

```
constructor (string memory name, string memory symbol, uint8 decimals) public {
    _name = name;
    _symbol = symbol;
    _decimals = decimals;
}
```

✓ The code meets the specification.

Formal Verification Request 75

Method will not encounter an assertion failure.

06, Sep 2019
0.41 ms

Line 528 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 534-538 in File tauschbloc.sol

```
    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }
```

✓ The code meets the specification.

Formal Verification Request 76

ERC20Detailed constructor correctness

📅 06, Sep 2019
⏱ 0.82 ms

Line 529-533 in File tauschbloc.sol

```
    /*@CTK "ERC20Detailed constructor correctness"
      @post __post._name == name
      @post __post._symbol == symbol
      @post __post._decimals == decimals
     */
```

Line 534-538 in File tauschbloc.sol

```
    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }
```

✓ The code meets the specification.

Formal Verification Request 77

If method completes, integer overflow would not happen.

📅 06, Sep 2019
⏱ 7.03 ms

Line 543 in File tauschbloc.sol

```
     //@CTK NO_OVERFLOW
```

Line 549-551 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 78

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.45 ms

Line 544 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 549-551 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 79

Method will not encounter an assertion failure.

06, Sep 2019
0.41 ms

Line 545 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 549-551 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 80

ERC20Detailed name correctness

06, Sep 2019
0.45 ms

Line 546-548 in File tauschbloc.sol

```
    /*@CTK "ERC20Detailed name correctness"
      @post __return == _name
     */
```

Line 549-551 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 81

If method completes, integer overflow would not happen.

06, Sep 2019
6.45 ms

Line 556 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 562-564 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 82

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.48 ms

Line 557 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 562-564 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 83

Method will not encounter an assertion failure.

06, Sep 2019
0.44 ms

Line 558 in File tauschbloc.sol

```
    //@CTK NO_ASF
```

Line 562-564 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 84

ERC20Detailed symbol correctness

06, Sep 2019
0.51 ms

Line 559-561 in File tauschbloc.sol

```
    /*@CTK "ERC20Detailed symbol correctness"
      @post __return == _symbol
     */
```

Line 562-564 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 85

If method completes, integer overflow would not happen.

06, Sep 2019
5.75 ms

Line 569 in File tauschbloc.sol

```
    //@CTK NO_OVERFLOW
```

Line 575-577 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 86

Buffer overflow / array index out of bound would never happen.

06, Sep 2019
0.42 ms

Line 570 in File tauschbloc.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 575-577 in File tauschbloc.sol

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

✓ The code meets the specification.

Formal Verification Request 87

Method will not encounter an assertion failure.

06, Sep 2019
0.4 ms

Line 571 in File tauschbloc.sol

```
571     //@CTK NO_ASF
```

Line 575-577 in File tauschbloc.sol

```
575    function decimals() public view returns (uint8) {
576        return _decimals;
577    }
```

✓ The code meets the specification.

Formal Verification Request 88

ERC20Detailed decimals correctness

06, Sep 2019
0.4 ms

Line 572-574 in File tauschbloc.sol

```
572     /*@CTK "ERC20Detailed decimals correctness"
573       @post __return == _decimals
574      */
```

Line 575-577 in File tauschbloc.sol

```
575    function decimals() public view returns (uint8) {
576        return _decimals;
577    }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File tauschbloc.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-05-27
*/

pragma solidity ^0.5.2;

/**
 * @title ERC20 interface
 * @dev see https://eips.ethereum.org/EIPS/eip-20
 */
interface IERC20 {
    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(address from, address to, uint256 value) external returns (
        bool);

    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender) external view returns (uint256)
        ;

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}


library SafeMath {
    /**
     * @dev Multiplies two unsigned integers, reverts on overflow.
     */
     /*@CTK "SafeMath mul"
       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
       @post !__reverted -> __return == a * b
       @post !__reverted == !__has_overflow
       @post !(__has_buf_overflow)
       @post !(__has_assertion_failure)
      */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

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

        return c;
    }

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

        return c;
    }

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

        return c;
    }

    /**
     * @dev Adds two unsigned integers, reverts on overflow.
     */
     /*@CTK "SafeMath add"
       @post (a + b < a || a + b < b) == __reverted
       @post !__reverted -> __return == a + b
       @post !__reverted -> !__has_overflow
       @post !(__has_buf_overflow)
       @post !(__has_assertion_failure)
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
```

```
     * @dev Divides two unsigned integers and returns the remainder (unsigned integer
         modulo),
     * reverts when dividing by zero.
     */
     /*@CTK "SafeMath div"
       @post b != 0 -> !__reverted
       @post !__reverted -> __return == a % b
       @post !__reverted -> !__has_overflow
       @post !(__has_buf_overflow)
       @post !(__has_assertion_failure)
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}


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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "totalSupply correctness"
       @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 A uint256 representing the amount owned by the passed address.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "balanceOf correctness"
       @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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "allowance correctness"
       @post __return == _allowed[owner][spender]
      */
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token to a specified address
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another.
     * Note that while this function emits an Approval event, this is not required as
        per the specification,
     * and other compliant implementations may not emit the event.
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        _transfer(from, to, value);
        _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
        return true;
    }

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

    /**
     * @dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
     * @param spender The address which will spend the funds.
     * @param subtractedValue The amount of tokens to decrease the allowance by.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue
            ));
        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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "_transfer correctness"
```

```
      @tag assume_completion
      @post to != 0x0
      @post to != from -> __post._balances[from] == _balances[from] - value
      @post to != from -> __post._balances[to] == _balances[to] + value
      @post to == from -> __post._balances[from] == _balances[from]
     */
    function _transfer(address from, address to, uint256 value) internal {
        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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "_mint correctness"
      @tag assume_completion
      @post account != 0x0
      @post __post._balances[account] == _balances[account] + value
      @post __post._totalSupply == _totalSupply + value
     */
    function _mint(address account, uint256 value) internal {
        require(account != address(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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "_burn correctness"
      @tag assume_completion
      @post account != 0x0
      @post __post._balances[account] == _balances[account] - value
      @post __post._totalSupply == _totalSupply - value
     */
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

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

    /**
     * @dev Approve an address to spend another addresses' tokens.
     * @param owner The address that owns the tokens.
     * @param spender The address that will spend the tokens.
     * @param value The number of tokens that can be spent.
     */
    /*@CTK _approve
      @tag assume_completion
      @post spender != 0x0
      @post owner != 0x0
      @post __post._allowed[owner][spender] == value
    */
    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

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

    /**
     * @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.
     * Emits an Approval event (reflecting the reduced allowance).
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
}


library Roles {
    struct Role {
        mapping (address => bool) bearer;
    }

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


        role.bearer[account] = true;
    }

    /**
```

```
     * @dev remove an account’s access to this role
     */
    function remove(Role storage role, address account) internal {
        require(account != address(0));
        require(role.bearer[account]);

        role.bearer[account] = false;
    }

    /**
     * @dev check if an account has this role
     * @return bool
     */
     /*@CTK "Roles has correctness"
       @post account == 0x0 -> __reverted
       @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
      */
    function has(Role storage role, address account) internal view returns (bool) {
        require(account != address(0));
        return role.bearer[account];
    }
}


contract MinterRole {
    using Roles for Roles.Role;

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

    Roles.Role private _minters;
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    constructor () internal {
        _addMinter(msg.sender);
    }

    modifier onlyMinter() {
        require(isMinter(msg.sender));
        _;
    }
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "isMinter correctness"
      @post account == 0x0 -> __reverted
      @post account != 0x0 -> !__reverted && __return == _minters.bearer[account]
     */
    function isMinter(address account) public view returns (bool) {
        return _minters.has(account);
    }
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    function addMinter(address account) public onlyMinter {
        _addMinter(account);
    }
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    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);
    }
}


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 NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK mint
       @tag assume_completion
       @post to != 0
       @post __post._totalSupply == _totalSupply + value
       @post __post._balances[to] == _balances[to] + value
      */
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }
}


contract ERC20Burnable is ERC20 {
    /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of token to be burned.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    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 account whose tokens will be burned.
     * @param value uint256 The amount of token to be burned.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
    function burnFrom(address from, uint256 value) public {
        _burnFrom(from, value);
    }
}


contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "ERC20Detailed constructor correctness"
      @post __post._name == name
      @post __post._symbol == symbol
      @post __post._decimals == decimals
     */
    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }

    /**
     * @return the name of the token.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "ERC20Detailed name correctness"
       @post __return == _name
      */
    function name() public view returns (string memory) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "ERC20Detailed symbol correctness"
       @post __return == _symbol
      */
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
     //@CTK NO_OVERFLOW
     //@CTK NO_BUF_OVERFLOW
     //@CTK NO_ASF
     /*@CTK "ERC20Detailed decimals correctness"
       @post __return == _decimals
      */
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}

contract TauschToken is ERC20Mintable, ERC20Burnable, ERC20Detailed {

    string private _name = "TauschToken";
    string private _symbol = "TUC";
    uint8 private _decimals = 10;

    uint256 public constant INITIAL_SUPPLY = 50 * (10 ** 18);

    address account = msg.sender;

    constructor()
        ERC20Detailed(_name, _symbol, _decimals)
        ERC20Burnable()
        ERC20Mintable()
        public {
            _mint(account, INITIAL_SUPPLY);
        }
}
```