CertiK Audit Report For KStarLive

Request Date: 2019-07-29
Revision Date: 2019-07-31
Platform Name: Luniverse

Contents

Disclaimer

About CertiK

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

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

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

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

Exective Summary

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

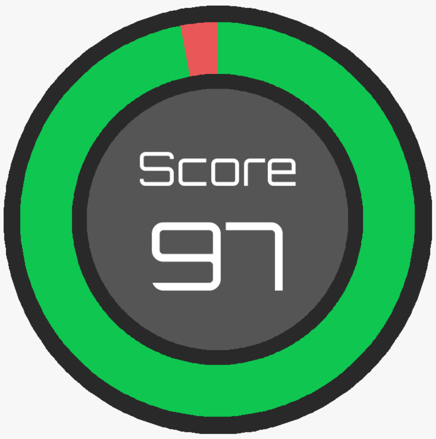
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

Testing Summary

PASS

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

Jul 31, 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

- **ERC20Token.sol**
  `b89d2a3fd7d45c7f90e1949dbc0cb67cb9d62ae74bb67accc5924e6b3de8b19b`
- **LinearMintableToken.sol**
  `9641d0b8896dc78a524e99fc655cc16a74719f230144c6a49f802a258623ddde`
- **MainToken.sol**
  `f8221358c9820da03e7acf08c2f18fe7f023971b13bac367286b3bbec0d32af2`
- **Ownable.sol**
  `5db8b7271c1c283dca4be7b5fb81ebafd207f0cd1b377d041feadbb66862aeed`
- **SafeMath.sol**
  `30465ec63e5f02d79d09da2ddbf3f963ac128c130322dd123d41fbcfc6642de5`
- **TokenRecipient.sol**
  `a5af13920e2a00dd888d4c75000227fd20ad2e929596c164ae05a29655956fab`

Summary

CertiK was chosen by KStarLive to audit the design and implementation of its MainToken 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.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes. The entries are labeled $\boxed{\text{CRITICAL}}$, $\boxed{\text{MAJOR}}$, $\boxed{\text{MINOR}}$, $\boxed{\text{INFO}}$, $\boxed{\text{DISCUSSION}}$(in decreasing significance).

MainToken.sol

- $\boxed{\text{INFO}}$ Recommend providing error messages to `require` statements.
- $\boxed{\text{INFO}}$ Recommend locking the compiler version.

- $\boxed{\text{INFO}}$ Recommend providing a getter method for `isLocked` for ease of use.

LinearMintableToken.sol

- $\boxed{\text{MINOR}}$ `calculateMintAmount()`: Recommend using `SafeMath`.
- $\boxed{\text{INFO}}$ Recommend providing error messages to `require` statements.
- $\boxed{\text{INFO}}$ `mintInternal()`: Recommend changing `if ( mintingAmount == 0 )` to `require()` and supplement error messages.
- $\boxed{\text{INFO}}$ Recommend locking the compiler version.

ERC20Token.sol

- $\boxed{\text{INFO}}$ Recommend providing error messages to `require` statements.
- $\boxed{\text{INFO}}$ Recommend locking the compiler version.

Ownable.sol

- $\boxed{\text{INFO}}$ Recommend providing error messages to `require` statements.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File LinearMintableToken.sol

```
pragma solidity ^0.4.24;
```

Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

TIMESTAMP_DEPENDENCY

Line 54 in File LinearMintableToken.sol

```
    createdTimestamp = block.timestamp;
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 58 in File LinearMintableToken.sol

```
    mintInternal(block.timestamp);
```

"block.timestamp" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
pragma solidity ^0.4.24;
```

Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Token.sol

```
pragma solidity ^0.4.24;
```

Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File MainToken.sol

```
pragma solidity ^0.4.24;
```

Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
pragma solidity ^0.4.24;
```

Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*: Line 30-34 in File howtoread.sol

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

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

52              }
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

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 156.82 ms

Line 20 in File LinearMintableToken.sol

```
//@CTK NO_OVERFLOW
```

Line 38-55 in File LinearMintableToken.sol

```
function registerLinearMint(
  uint256 _mintingSupply,
  uint256 _mintAmountPerPeriod,
  uint256 _intervalPeriodInDays
) external onlyOwner() {
  require(!mintingStatus);
  require(_mintingSupply > 0);
  require(totalSupply.add(_mintingSupply) <= maxSupply);
  require(_mintAmountPerPeriod > 0 );
  require(_intervalPeriodInDays > 0 );

  mintingStatus = true;

  mintingSupply = _mintingSupply;
  mintAmountPerPeriod = _mintAmountPerPeriod;
  intervalPeriodInDays = _intervalPeriodInDays;
  createdTimestamp = block.timestamp;
}
```

✓ The code meets the specification.

Formal Verification Request 2

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

📅 31, Jul 2019
⏱ 76.84 ms

Line 21 in File LinearMintableToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 38-55 in File LinearMintableToken.sol

```
function registerLinearMint(
  uint256 _mintingSupply,
  uint256 _mintAmountPerPeriod,
  uint256 _intervalPeriodInDays
) external onlyOwner() {
  require(!mintingStatus);
  require(_mintingSupply > 0);
  require(totalSupply.add(_mintingSupply) <= maxSupply);
  require(_mintAmountPerPeriod > 0 );
  require(_intervalPeriodInDays > 0 );

  mintingStatus = true;
```

```

    mintingSupply = _mintingSupply;
    mintAmountPerPeriod = _mintAmountPerPeriod;
    intervalPeriodInDays = _intervalPeriodInDays;
    createdTimestamp = block.timestamp;
  }
```

✓ The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

31, Jul 2019
43.46 ms

Line 22 in File LinearMintableToken.sol

```
  //@CTK NO_ASF
```

Line 38-55 in File LinearMintableToken.sol

```
  function registerLinearMint(
    uint256 _mintingSupply,
    uint256 _mintAmountPerPeriod,
    uint256 _intervalPeriodInDays
  ) external onlyOwner() {
    require(!mintingStatus);
    require(_mintingSupply > 0);
    require(totalSupply.add(_mintingSupply) <= maxSupply);
    require(_mintAmountPerPeriod > 0 );
    require(_intervalPeriodInDays > 0 );

    mintingStatus = true;

    mintingSupply = _mintingSupply;
    mintAmountPerPeriod = _mintAmountPerPeriod;
    intervalPeriodInDays = _intervalPeriodInDays;
    createdTimestamp = block.timestamp;
  }
```

✓ The code meets the specification.

Formal Verification Request 4

registerLinearMint correctness

31, Jul 2019
182.87 ms

Line 23-37 in File LinearMintableToken.sol

```
  /*@CTK "registerLinearMint correctness"
    @tag assume_completion
    @post mintingStatus == false
    @post totalSupply + _mintingSupply <= maxSupply
```

```
27     @post _owner == msg.sender
28     @post __post.mintingStatus == true
29     @post __post.mintingSupply == _mintingSupply
30     @post __post.mintingSupply > 0
31     @post __post.mintingSupply == _mintingSupply
32     @post __post.mintAmountPerPeriod == _mintAmountPerPeriod
33     @post __post.mintAmountPerPeriod > 0
34     @post __post.intervalPeriodInDays == _intervalPeriodInDays
35     @post __post.intervalPeriodInDays > 0
36     @post __post.createdTimestamp == block.timestamp
37   */
```

Line 38-55 in File LinearMintableToken.sol

```
38   function registerLinearMint(
39     uint256 _mintingSupply,
40     uint256 _mintAmountPerPeriod,
41     uint256 _intervalPeriodInDays
42   ) external onlyOwner() {
43     require(!mintingStatus);
44     require(_mintingSupply > 0);
45     require(totalSupply.add(_mintingSupply) <= maxSupply);
46     require(_mintAmountPerPeriod > 0 );
47     require(_intervalPeriodInDays > 0 );
48
49     mintingStatus = true;
50
51     mintingSupply = _mintingSupply;
52     mintAmountPerPeriod = _mintAmountPerPeriod;
53     intervalPeriodInDays = _intervalPeriodInDays;
54     createdTimestamp = block.timestamp;
55   }
```

✓ The code meets the specification.

Formal Verification Request 5

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 11075.7 ms

Line 61 in File LinearMintableToken.sol

```
61   //@CTK FAIL NO_OVERFLOW
```

Line 70-92 in File LinearMintableToken.sol

```
70   function mintInternal(uint256 blockTimestamp) internal {
71     require(mintingStatus);
72
73     address tokenOwner = owner();
74     uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75     mintingAmount = mintingAmount.sub(mintedAmount);
76
77     if ( mintingAmount == 0 )
78       return;
79
```

```
    if ( mintingAmount >= mintingSupply ) {
      mintingAmount = mintingSupply.sub(mintedAmount);
      mintingStatus = false;
    }

    totalSupply = totalSupply.add(mintingAmount);

    _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);

    mintedAmount = mintedAmount.add(mintingAmount);

    emit Minted(tokenOwner, mintingAmount, mintedAmount);
  }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        blockTimestamp = 32
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "LinearMintableToken",
          "balance": 0,
          "contract": {
            "SECONDS_IN_A_DAY": 1,
            "mintingStatus": true,
            "mintingSupply": 8,
            "intervalPeriodInDays": 1,
            "intervalCount": 0,
            "mintAmountPerPeriod": 129,
            "createdTimestamp": 15,
            "mintedAmount": 18,
            "_owner": 0,
            "name": "",
            "symbol": "",
            "totalSupply": 20,
            "maxSupply": 0,
            "decimals": 0,
            "_balances": [
              {
                "key": 8,
                "value": 2
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 33,
                "value": 1
              },
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 129,
                "value": 0
              },
              {
                "key": 128,
                "value": 4
              },
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
            ],
            "_allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 1
                  },
                  {
                    "key": 4,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 64
                  }
                ]
              },
              {
                "key": 32,
                "value": [
                  {
                    "key": 0,
                    "value": 32
                  },
                  {
                   "key": "ALL_OTHERS",
                   "value": 64
                 }
               ]
             },
             {
               "key": "ALL_OTHERS",
               "value": [
                 {
                   "key": "ALL_OTHERS",
                   "value": 16
                 }
               ]
             }
           ]
         }
       }
     },
     {
       "key": "ALL_OTHERS",
       "value": "EmptyAddress"
     }
   ]

After Execution:
   Input = {
       blockTimestamp = 32
   }
   This = 0
   Internal = {
       __has_assertion_failure = false
       __has_buf_overflow = false
       __has_overflow = true
       __has_returned = true
       __reverted = false
       msg = {
         "gas": 0,
         "sender": 0,
         "value": 0
       }
   }
   Other = {
       block = {
         "number": 0,
         "timestamp": 0
       }
       mintingAmount = 0
       tokenOwner = 0
   }
   Address_Map = [
     {
       "key": 0,
       "value": {
         "contract_name": "LinearMintableToken",
         "balance": 0,
         "contract": {
           "SECONDS_IN_A_DAY": 1,
           "mintingStatus": true,
            "mintingSupply": 8,
            "intervalPeriodInDays": 1,
            "intervalCount": 0,
            "mintAmountPerPeriod": 129,
            "createdTimestamp": 15,
            "mintedAmount": 18,
            "_owner": 0,
            "name": "",
            "symbol": "",
            "totalSupply": 20,
            "maxSupply": 0,
            "decimals": 0,
            "_balances": [
              {
                "key": 8,
                "value": 2
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 33,
                "value": 1
              },
              {
                "key": 0,
                "value": 0
              },
              {
                "key": 129,
                "value": 0
              },
              {
                "key": 128,
                "value": 4
              },
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
            ],
            "_allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 1
                  },
                  {
                    "key": 4,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 64
                  }
```

```
       ]
      },
      {
       "key": 32,
       "value": [
        {
         "key": 0,
         "value": 32
        },
        {
         "key": "ALL_OTHERS",
         "value": 64
        }
       ]
      },
      {
       "key": "ALL_OTHERS",
       "value": [
        {
         "key": "ALL_OTHERS",
         "value": 16
        }
       ]
      }
     ]
    }
   }
  },
  {
   "key": "ALL_OTHERS",
   "value": "EmptyAddress"
  }
 ]
```

Formal Verification Request 6

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

31, Jul 2019
191.55 ms

Line 62 in File LinearMintableToken.sol

```
62  //@CTK NO_BUF_OVERFLOW
```

Line 70-92 in File LinearMintableToken.sol

```
70  function mintInternal(uint256 blockTimestamp) internal {
71    require(mintingStatus);
72
73    address tokenOwner = owner();
74    uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75    mintingAmount = mintingAmount.sub(mintedAmount);
76
77    if ( mintingAmount == 0 )
78      return;
79
80    if ( mintingAmount >= mintingSupply ) {
      mintingAmount = mintingSupply.sub(mintedAmount);
      mintingStatus = false;
    }

    totalSupply = totalSupply.add(mintingAmount);

    _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);

    mintedAmount = mintedAmount.add(mintingAmount);

    emit Minted(tokenOwner, mintingAmount, mintedAmount);
  }
```

✓ The code meets the specification.

Formal Verification Request 7

mintInternal NO_ASF and correctness

31, Jul 2019
205.09 ms

Line 63-69 in File LinearMintableToken.sol

```
  /*@CTK "mintInternal NO_ASF and correctness"
    @tag assume_completion
    @pre intervalPeriodInDays > 0
    @pre SECONDS_IN_A_DAY > 0
    @post mintingStatus == true
    @post !(__has_assertion_failure)
  */
```

Line 70-92 in File LinearMintableToken.sol

```
  function mintInternal(uint256 blockTimestamp) internal {
    require(mintingStatus);

    address tokenOwner = owner();
    uint256 mintingAmount = calculateMintAmount(blockTimestamp);
    mintingAmount = mintingAmount.sub(mintedAmount);

    if ( mintingAmount == 0 )
      return;

    if ( mintingAmount >= mintingSupply ) {
      mintingAmount = mintingSupply.sub(mintedAmount);
      mintingStatus = false;
    }

    totalSupply = totalSupply.add(mintingAmount);

    _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);

    mintedAmount = mintedAmount.add(mintingAmount);

    emit Minted(tokenOwner, mintingAmount, mintedAmount);
  }
```

✓ The code meets the specification.

Formal Verification Request 8

If method completes, integer overflow would not happen.

31, Jul 2019
21.77 ms

Line 94 in File LinearMintableToken.sol

```
//@CTK FAIL NO_OVERFLOW
```

Line 102-107 in File LinearMintableToken.sol

```
function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
    mintAmount) {
  uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
  uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;

  return pastIntervals * mintAmountPerPeriod;
}
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        blockTimestamp = 128
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        mintAmount = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "LinearMintableToken",
          "balance": 0,
          "contract": {
            "SECONDS_IN_A_DAY": 40,
            "mintingStatus": false,
            "mintingSupply": 0,
            "intervalPeriodInDays": 1,
            "intervalCount": 0,
            "mintAmountPerPeriod": 129,
            "createdTimestamp": 64,
            "mintedAmount": 0,
            "_owner": 0,
            "name": "",
            "symbol": "",
            "totalSupply": 0,
            "maxSupply": 0,
            "decimals": 0,
            "_balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 4
              }
            ],
            "_allowed": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": 32,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 4
                  }
                ]
              }
            ]
          }
        }
      }
    ]

After Execution:
    Input = {
        blockTimestamp = 128
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = true
        __has_returned = true
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        mintAmount = 2
        pastDays = 1
        pastIntervals = 2
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": {
          "contract_name": "LinearMintableToken",
          "balance": 0,
          "contract": {
            "SECONDS_IN_A_DAY": 40,
            "mintingStatus": false,
            "mintingSupply": 0,
            "intervalPeriodInDays": 1,
            "intervalCount": 0,
            "mintAmountPerPeriod": 129,
            "createdTimestamp": 64,
            "mintedAmount": 0,
            "_owner": 0,
            "name": "",
            "symbol": "",
            "totalSupply": 0,
            "maxSupply": 0,
            "decimals": 0,
            "_balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 4
              }
            ],
            "_allowed": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": 32,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 4
                  }
                ]
              }
            ]
          }
      }
    }
  ]
```

Formal Verification Request 9

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

31, Jul 2019
0.84 ms

Line 95 in File LinearMintableToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 102-107 in File LinearMintableToken.sol

```
function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
    mintAmount) {
  uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
  uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;

  return pastIntervals * mintAmountPerPeriod;
}
```

✓ The code meets the specification.

Formal Verification Request 10

calculateMintAmount NO_ASF

31, Jul 2019
4.65 ms

Line 96-101 in File LinearMintableToken.sol

```
/*@CTK "calculateMintAmount NO_ASF"
  @tag assume_completion
  @pre intervalPeriodInDays > 0
  @pre SECONDS_IN_A_DAY > 0
  @post !(__has_assertion_failure)
*/
```

Line 102-107 in File LinearMintableToken.sol

```
function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
    mintAmount) {
  uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
  uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;

  return pastIntervals * mintAmountPerPeriod;
}
```

✓ The code meets the specification.

Formal Verification Request 11

Ownable

📅 31, Jul 2019
⏱ 5.63 ms

Line 17-19 in File Ownable.sol

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

Line 20-23 in File Ownable.sol

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

✓ The code meets the specification.

Formal Verification Request 12

owner

📅 31, Jul 2019
⏱ 5.17 ms

Line 28-30 in File Ownable.sol

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

Line 31-33 in File Ownable.sol

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

✓ The code meets the specification.

Formal Verification Request 13

isOwner

📅 31, Jul 2019
⏱ 6.2 ms

Line 46-48 in File Ownable.sol

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

Line 49-51 in File Ownable.sol

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

✓ The code meets the specification.

Formal Verification Request 14

transferOwnership

📅 31, Jul 2019
⏱ 53.14 ms

Line 69-72 in File Ownable.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @post _owner == msg.sender
 */
```

Line 73-75 in File Ownable.sol

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

✓ The code meets the specification.

Formal Verification Request 15

_transferOwnership

📅 31, Jul 2019
⏱ 1.07 ms

Line 81-85 in File Ownable.sol

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

Line 86-90 in File Ownable.sol

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

✓ The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 28.09 ms

Line 23 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 32-42 in File ERC20Token.sol

```
constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
    uint256 _maxSupply) public {
  require(_maxSupply >= _initialSupply);

  name = _name;
  symbol = _symbol;
  decimals = _decimals;

  _balances[msg.sender] = _initialSupply;
  totalSupply = _initialSupply;
  maxSupply = _maxSupply;
}
```

✓ The code meets the specification.

Formal Verification Request 17

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

📅 31, Jul 2019
⏱ 0.49 ms

Line 24 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 32-42 in File ERC20Token.sol

```
constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
    uint256 _maxSupply) public {
  require(_maxSupply >= _initialSupply);

  name = _name;
  symbol = _symbol;
  decimals = _decimals;

  _balances[msg.sender] = _initialSupply;
  totalSupply = _initialSupply;
  maxSupply = _maxSupply;
}
```

✓ The code meets the specification.

Formal Verification Request 18

Method will not encounter an assertion failure.

📅 31, Jul 2019
⏱ 0.48 ms

Line 25 in File ERC20Token.sol

```
25  //@CTK NO_ASF
```

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
        uint256 _maxSupply) public {
33    require(_maxSupply >= _initialSupply);
34
35    name = _name;
36    symbol = _symbol;
37    decimals = _decimals;
38
39    _balances[msg.sender] = _initialSupply;
40    totalSupply = _initialSupply;
41    maxSupply = _maxSupply;
42  }
```

✓ The code meets the specification.

Formal Verification Request 19

constructor correctness

📅 31, Jul 2019
⏱ 4.65 ms

Line 26-31 in File ERC20Token.sol

```
26  /*@CTK "constructor correctness"
27    @tag assume_completion
28    @post __post._balances[msg.sender] == __post.totalSupply
29    @post __post._balances[msg.sender] == _initialSupply
30    @post __post.maxSupply == _maxSupply
31  */
```

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
        uint256 _maxSupply) public {
33    require(_maxSupply >= _initialSupply);
34
35    name = _name;
36    symbol = _symbol;
37    decimals = _decimals;
38
39    _balances[msg.sender] = _initialSupply;
40    totalSupply = _initialSupply;
41    maxSupply = _maxSupply;
42  }
```

✓ The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

31, Jul 2019
5.71 ms

Line 44 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 50-52 in File ERC20Token.sol

```
function balanceOf(address _owner) public view returns (uint balance) {
  return _balances[_owner];
}
```

✓ The code meets the specification.

Formal Verification Request 21

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

31, Jul 2019
0.35 ms

Line 45 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 50-52 in File ERC20Token.sol

```
function balanceOf(address _owner) public view returns (uint balance) {
  return _balances[_owner];
}
```

✓ The code meets the specification.

Formal Verification Request 22

Method will not encounter an assertion failure.

31, Jul 2019
0.33 ms

Line 46 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 50-52 in File ERC20Token.sol

```
function balanceOf(address _owner) public view returns (uint balance) {
  return _balances[_owner];
}
```

✓ The code meets the specification.

Formal Verification Request 23

balanceOf correctness

📅 31, Jul 2019
⏱ 0.34 ms

Line 47-49 in File ERC20Token.sol

```
47 /*@CTK "balanceOf correctness"
48   @post balance == _balances[_owner]
49  */
```

Line 50-52 in File ERC20Token.sol

```
50 function balanceOf(address _owner) public view returns (uint balance) {
51   return _balances[_owner];
52 }
```

✓ The code meets the specification.

Formal Verification Request 24

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 6.21 ms

Line 54 in File ERC20Token.sol

```
54 //@CTK NO_OVERFLOW
```

Line 60-62 in File ERC20Token.sol

```
60 function allowance(address _owner, address _spender) public view returns (uint256
       remaining) {
61   return _allowed[_owner][_spender];
62 }
```

✓ The code meets the specification.

Formal Verification Request 25

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

📅 31, Jul 2019
⏱ 0.33 ms

Line 55 in File ERC20Token.sol

```
55 //@CTK NO_BUF_OVERFLOW
```

Line 60-62 in File ERC20Token.sol

```
60 function allowance(address _owner, address _spender) public view returns (uint256
       remaining) {
61   return _allowed[_owner][_spender];
62 }
```

✓ The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

📅 31, Jul 2019
⏱ 0.33 ms

Line 56 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 60-62 in File ERC20Token.sol

```
function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
  return _allowed[_owner][_spender];
}
```

✅ The code meets the specification.

Formal Verification Request 27

allowance correctness

📅 31, Jul 2019
⏱ 0.34 ms

Line 57-59 in File ERC20Token.sol

```
/*@CTK "allowance correctness"
  @post remaining == _allowed[_owner][_spender]
 */
```

Line 60-62 in File ERC20Token.sol

```
function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
  return _allowed[_owner][_spender];
}
```

✅ The code meets the specification.

Formal Verification Request 28

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 182.53 ms

Line 69 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 81-84 in File ERC20Token.sol

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

✓ The code meets the specification.

Formal Verification Request 29

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

31, Jul 2019
11.26 ms

Line 70 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 81-84 in File ERC20Token.sol

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

✓ The code meets the specification.

Formal Verification Request 30

Method will not encounter an assertion failure.

31, Jul 2019
12.48 ms

Line 71 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 81-84 in File ERC20Token.sol

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

✓ The code meets the specification.

Formal Verification Request 31

transfer correctness

31, Jul 2019
81.95 ms

Line 72-80 in File ERC20Token.sol

```
/*@CTK "transfer correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _value <= _balances[msg.sender]
  @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
      _value
  @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
  @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
  @post success == true
 */
```

Line 81-84 in File ERC20Token.sol

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

✓ The code meets the specification.

Formal Verification Request 32

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 165.57 ms

Line 94 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 107-111 in File ERC20Token.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
  _transfer(_from, _to, _value);
  _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 33

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

📅 31, Jul 2019
⏱ 21.16 ms

Line 95 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 107-111 in File ERC20Token.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
  _transfer(_from, _to, _value);
  _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 34

Method will not encounter an assertion failure.

📅 31, Jul 2019
⏱ 22.03 ms

Line 96 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 107-111 in File ERC20Token.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
  _transfer(_from, _to, _value);
  _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
  return true;
}
```

✅ The code meets the specification.

Formal Verification Request 35

transferFrom correctness

📅 31, Jul 2019
⏱ 450.99 ms

Line 97-106 in File ERC20Token.sol

```
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
  @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]
  @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
  @post success == true
 */
```

Line 107-111 in File ERC20Token.sol

```
function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
  _transfer(_from, _to, _value);
  _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
  return true;
}
```

✅ The code meets the specification.

Formal Verification Request 36

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 34.44 ms

Line 122 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 129-132 in File ERC20Token.sol

```
function approve(address _spender, uint256 _value) public returns (bool success) {
  _approve(msg.sender, _spender, _value);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 37

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

31, Jul 2019
0.61 ms

Line 123 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 129-132 in File ERC20Token.sol

```
function approve(address _spender, uint256 _value) public returns (bool success) {
  _approve(msg.sender, _spender, _value);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 38

Method will not encounter an assertion failure.

31, Jul 2019
0.59 ms

Line 124 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 129-132 in File ERC20Token.sol

```
function approve(address _spender, uint256 _value) public returns (bool success) {
  _approve(msg.sender, _spender, _value);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 39

approve correctness

📅 31, Jul 2019
⏱ 6.92 ms

Line 125-128 in File ERC20Token.sol

```
/*@CTK "approve correctness"
  @tag assume_completion
  @post __post._allowed[msg.sender][_spender] == _value
 */
```

Line 129-132 in File ERC20Token.sol

```
function approve(address _spender, uint256 _value) public returns (bool success) {
  _approve(msg.sender, _spender, _value);
  return true;
}
```

✅ The code meets the specification.

Formal Verification Request 40

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 62.55 ms

Line 144 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 151-154 in File ERC20Token.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 41

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

📅 31, Jul 2019
⏱ 0.91 ms

Line 145 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 151-154 in File ERC20Token.sol

```
151  function increaseAllowance(address _spender, uint256 _addedValue) public returns (
         bool) {
152    _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153    return true;
154  }
```

The code meets the specification.

Formal Verification Request 42

Method will not encounter an assertion failure.

31, Jul 2019
0.86 ms

Line 146 in File ERC20Token.sol

```
146  //@CTK NO_ASF
```

Line 151-154 in File ERC20Token.sol

```
151  function increaseAllowance(address _spender, uint256 _addedValue) public returns (
         bool) {
152    _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153    return true;
154  }
```

The code meets the specification.

Formal Verification Request 43

increaseApproval correctness

31, Jul 2019
7.5 ms

Line 147-150 in File ERC20Token.sol

```
147  /*@CTK "increaseApproval correctness"
148    @tag assume_completion
149    @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
           _addedValue
150   */
```

Line 151-154 in File ERC20Token.sol

```
151  function increaseAllowance(address _spender, uint256 _addedValue) public returns (
         bool) {
152    _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153    return true;
154  }
```

The code meets the specification.

Formal Verification Request 44

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 52.33 ms

Line 166 in File ERC20Token.sol

```
//@CTK NO_OVERFLOW
```

Line 180-183 in File ERC20Token.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 45

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

📅 31, Jul 2019
⏱ 0.99 ms

Line 167 in File ERC20Token.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 180-183 in File ERC20Token.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 46

Method will not encounter an assertion failure.

📅 31, Jul 2019
⏱ 0.92 ms

Line 168 in File ERC20Token.sol

```
//@CTK NO_ASF
```

Line 180-183 in File ERC20Token.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 47

decreaseApproval0

31, Jul 2019
19.62 ms

Line 169-173 in File ERC20Token.sol

```
/*@CTK decreaseApproval0
  @pre __return == true
  @pre _allowed[msg.sender][_spender] <= _subtractedValue
  @post __post._allowed[msg.sender][_spender] == 0
*/
```

Line 180-183 in File ERC20Token.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 48

decreaseApproval

31, Jul 2019
7.52 ms

Line 174-179 in File ERC20Token.sol

```
/*@CTK decreaseApproval
  @pre __return == true
  @pre _allowed[msg.sender][_spender] > _subtractedValue
  @post __post._allowed[msg.sender][_spender] ==
        _allowed[msg.sender][_spender] - _subtractedValue
*/
```

Line 180-183 in File ERC20Token.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 49

If method completes, integer overflow would not happen.

31, Jul 2019
206.98 ms

Line 28 in File MainToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 39-51 in File MainToken.sol

```
  function mint(uint256 _amount) onlyOwner() external {
    require(!mintingStatus);
    uint newTotalSupply = totalSupply.add(_amount);
    address tokenOwner = owner();

    require( newTotalSupply <= maxSupply );

    _balances[tokenOwner] = _balances[tokenOwner].add(_amount);

    totalSupply = newTotalSupply;

    emit Minted(tokenOwner, _amount, _amount);
  }
```

✓ The code meets the specification.

Formal Verification Request 50

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

31, Jul 2019
27.15 ms

Line 29 in File MainToken.sol

```
  //@CTK NO_BUF_OVERFLOW
```

Line 39-51 in File MainToken.sol

```
  function mint(uint256 _amount) onlyOwner() external {
    require(!mintingStatus);
    uint newTotalSupply = totalSupply.add(_amount);
    address tokenOwner = owner();

    require( newTotalSupply <= maxSupply );

    _balances[tokenOwner] = _balances[tokenOwner].add(_amount);

    totalSupply = newTotalSupply;

    emit Minted(tokenOwner, _amount, _amount);
  }
```

✓ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

31, Jul 2019
26.23 ms

Line 30 in File MainToken.sol

```
  //@CTK NO_ASF
```

Line 39-51 in File MainToken.sol

```
  function mint(uint256 _amount) onlyOwner() external {
    require(!mintingStatus);
    uint newTotalSupply = totalSupply.add(_amount);
    address tokenOwner = owner();

    require( newTotalSupply <= maxSupply );

    _balances[tokenOwner] = _balances[tokenOwner].add(_amount);

    totalSupply = newTotalSupply;

    emit Minted(tokenOwner, _amount, _amount);
  }
```

✓ The code meets the specification.

Formal Verification Request 52

mint

31, Jul 2019
275.43 ms

Line 31-38 in File MainToken.sol

```
  /*@CTK "mint"
    @tag assume_completion
    @post mintingStatus == false
    @post _owner == msg.sender
    @post __post.totalSupply == totalSupply + _amount
    @post __post.totalSupply <= maxSupply
    @post __post._balances[msg.sender] == _balances[msg.sender] + _amount
   */
```

Line 39-51 in File MainToken.sol

```
function mint(uint256 _amount) onlyOwner() external {
  require(!mintingStatus);
  uint newTotalSupply = totalSupply.add(_amount);
  address tokenOwner = owner();

  require( newTotalSupply <= maxSupply );

  _balances[tokenOwner] = _balances[tokenOwner].add(_amount);

  totalSupply = newTotalSupply;

  emit Minted(tokenOwner, _amount, _amount);
}
```

✓ The code meets the specification.

Formal Verification Request 53

lockAccount correctness

31, Jul 2019
32.22 ms

Line 53-58 in File MainToken.sol

```
/*@CTK "lockAccount correctness"
  @tag assume_completion
  @post isLocked[_account] == false
  @post _owner == msg.sender
  @post __post.isLocked[_account] == true
 */
```

Line 59-63 in File MainToken.sol

```
function lockAccount(address _account) onlyOwner() external {
  require(!isLocked[_account]);

  isLocked[_account] = true;
}
```

✓ The code meets the specification.

Formal Verification Request 54

unlockAccount correctness

31, Jul 2019
31.55 ms

Line 65-70 in File MainToken.sol

```
/*@CTK "unlockAccount correctness"
  @tag assume_completion
  @post isLocked[_account] == true
  @post _owner == msg.sender
  @post __post.isLocked[_account] == false
 */
```

Line 71-75 in File MainToken.sol

```
function unlockAccount(address _account) onlyOwner() external {
  require(isLocked[_account]);

  isLocked[_account] = false;
}
```

✓ The code meets the specification.

Formal Verification Request 55

paused correctness

📅 31, Jul 2019
⏱ 4.42 ms

Line 80-83 in File MainToken.sol

```
/*@CTK "paused correctness"
  @tag assume_completion
  @post __return == _paused
 */
```

Line 84-86 in File MainToken.sol

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

✓ The code meets the specification.

Formal Verification Request 56

pause correctness

📅 31, Jul 2019
⏱ 37.96 ms

Line 107-112 in File MainToken.sol

```
/*@CTK "pause correctness"
  @tag assume_completion
  @post _paused == false
  @post _owner == msg.sender
  @post __post._paused == true
 */
```

Line 113-116 in File MainToken.sol

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

✓ The code meets the specification.

Formal Verification Request 57

unpause correctness

📅 31, Jul 2019
⏱ 35.62 ms

Line 121-126 in File MainToken.sol

```
/*@CTK "unpause correctness"
  @tag assume_completion
  @post _paused == true
  @post _owner == msg.sender
  @post __post._paused == false
 */
```

Line 127-130 in File MainToken.sol

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

✓ The code meets the specification.

Formal Verification Request 58

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 274.03 ms

Line 132 in File MainToken.sol

```
//@CTK NO_OVERFLOW
```

Line 146-148 in File MainToken.sol

```
function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
  return super.transfer(_to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 59

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

📅 31, Jul 2019
⏱ 8.76 ms

Line 133 in File MainToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 146-148 in File MainToken.sol

```
function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
  return super.transfer(_to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 60

Method will not encounter an assertion failure.

31, Jul 2019
10.02 ms

Line 134 in File MainToken.sol

```
//@CTK NO_ASF
```

Line 146-148 in File MainToken.sol

```
function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
  return super.transfer(_to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 61

transfer correctness

31, Jul 2019
275.36 ms

Line 135-145 in File MainToken.sol

```
/*@CTK "transfer correctness"
  @tag assume_completion
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post _to != 0x0
  @post _value <= _balances[msg.sender]
  @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
       _value
  @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
  @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
  @post __return == true
 */
```

Line 146-148 in File MainToken.sol

```
function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
  return super.transfer(_to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 62

If method completes, integer overflow would not happen.

📅 31, Jul 2019
⏱ 333.82 ms

Line 150 in File MainToken.sol

```
//@CTK NO_OVERFLOW
```

Line 165-169 in File MainToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.transferFrom(_from, _to, _value);
}
```

✅ The code meets the specification.

Formal Verification Request 63

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

📅 31, Jul 2019
⏱ 41.78 ms

Line 151 in File MainToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 165-169 in File MainToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.transferFrom(_from, _to, _value);
}
```

✅ The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.

📅 31, Jul 2019
⏱ 44.79 ms

Line 152 in File MainToken.sol

```
//@CTK NO_ASF
```

Line 165-169 in File MainToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.transferFrom(_from, _to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 65

transferFrom correctness

31, Jul 2019
614.57 ms

Line 153-164 in File MainToken.sol

```
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
  @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]
  @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
  @post __return == true
 */
```

Line 165-169 in File MainToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.transferFrom(_from, _to, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 66

If method completes, integer overflow would not happen.

31, Jul 2019
98.66 ms

Line 171 in File MainToken.sol

```
//@CTK NO_OVERFLOW
```

Line 180-183 in File MainToken.sol

```
function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
(bool) {
  return super.approve(_spender, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 67

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

31, Jul 2019
5.07 ms

Line 172 in File MainToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 180-183 in File MainToken.sol

```
function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
(bool) {
  return super.approve(_spender, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 68

Method will not encounter an assertion failure.

31, Jul 2019
5.24 ms

Line 173 in File MainToken.sol

```
//@CTK NO_ASF
```

Line 180-183 in File MainToken.sol

```
function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
(bool) {
  return super.approve(_spender, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 69

approve correctness

31, Jul 2019
10.47 ms

Line 174-179 in File MainToken.sol

```
/*@CTK "approve correctness"
  @post _paused == false
  @post isLocked[msg.sender] == false
  @tag assume_completion
  @post __post._allowed[msg.sender][_spender] == _value
 */
```

Line 180-183 in File MainToken.sol

```
function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
     returns
(bool) {
  return super.approve(_spender, _value);
}
```

✓ The code meets the specification.

Formal Verification Request 70

If method completes, integer overflow would not happen.

31, Jul 2019
146.82 ms

Line 185 in File MainToken.sol

```
//@CTK NO_OVERFLOW
```

Line 194-197 in File MainToken.sol

```
function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
returns (bool) {
  return super.increaseAllowance(_spender, _addedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 71

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

31, Jul 2019
6.57 ms

Line 186 in File MainToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 194-197 in File MainToken.sol

```
function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
returns (bool) {
  return super.increaseAllowance(_spender, _addedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 72

Method will not encounter an assertion failure.

31, Jul 2019
6.47 ms

Line 187 in File MainToken.sol

```
//@CTK NO_ASF
```

Line 194-197 in File MainToken.sol

```
function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
returns (bool) {
  return super.increaseAllowance(_spender, _addedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 73

increaseApproval correctness

31, Jul 2019
60.81 ms

Line 188-193 in File MainToken.sol

```
/*@CTK "increaseApproval correctness"
  @tag assume_completion
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
      _addedValue
 */
```

Line 194-197 in File MainToken.sol

```
function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
returns (bool) {
  return super.increaseAllowance(_spender, _addedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 74

If method completes, integer overflow would not happen.

31, Jul 2019
129.68 ms

Line 199 in File MainToken.sol

```
//@CTK NO_OVERFLOW
```

Line 217-221 in File MainToken.sol

```
function decreaseAllowance(address _spender, uint _subtractedValue) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.decreaseAllowance(_spender, _subtractedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 75

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

31, Jul 2019
6.88 ms

Line 200 in File MainToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 217-221 in File MainToken.sol

```
function decreaseAllowance(address _spender, uint _subtractedValue) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.decreaseAllowance(_spender, _subtractedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 76

Method will not encounter an assertion failure.

31, Jul 2019
5.85 ms

Line 201 in File MainToken.sol

```
//@CTK NO_ASF
```

Line 217-221 in File MainToken.sol

```
function decreaseAllowance(address _spender, uint _subtractedValue) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.decreaseAllowance(_spender, _subtractedValue);
}
```

✓ The code meets the specification.

Formal Verification Request 77

decreaseApproval0

📅 31, Jul 2019
⏱ 32.6 ms

Line 202-208 in File MainToken.sol

```
/*@CTK decreaseApproval0
  @pre __return == true
  @pre _allowed[msg.sender][_spender] <= _subtractedValue
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post __post._allowed[msg.sender][_spender] == 0
*/
```

Line 217-221 in File MainToken.sol

```
function decreaseAllowance(address _spender, uint _subtractedValue) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.decreaseAllowance(_spender, _subtractedValue);
}
```

✅ The code meets the specification.

Formal Verification Request 78

decreaseApproval

📅 31, Jul 2019
⏱ 36.53 ms

Line 209-216 in File MainToken.sol

```
/*@CTK decreaseApproval
  @pre __return == true
  @pre _allowed[msg.sender][_spender] > _subtractedValue
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post __post._allowed[msg.sender][_spender] ==
        _allowed[msg.sender][_spender] - _subtractedValue
*/
```

Line 217-221 in File MainToken.sol

```
function decreaseAllowance(address _spender, uint _subtractedValue) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.decreaseAllowance(_spender, _subtractedValue);
}
```

✅ The code meets the specification.

Formal Verification Request 79

SafeMath mul

📅 31, Jul 2019
⏱ 316.77 ms

Line 11-17 in File SafeMath.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 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 80

SafeMath div

📅 31, Jul 2019
⏱ 12.44 ms

Line 35-41 in File SafeMath.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 42-49 in File SafeMath.sol

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

```

    return c;
  }
```

✓ The code meets the specification.

Formal Verification Request 81

SafeMath sub

📅 31, Jul 2019
⏱ 10.91 ms

Line 54-60 in File SafeMath.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 61-66 in File SafeMath.sol

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

    return c;
  }
```

✓ The code meets the specification.

Formal Verification Request 82

SafeMath add

📅 31, Jul 2019
⏱ 13.86 ms

Line 71-77 in File SafeMath.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 78-83 in File SafeMath.sol

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

    return c;
  }
```

✓ The code meets the specification.

Formal Verification Request 83

SafeMath div

31, Jul 2019
10.43 ms

Line 89-95 in File SafeMath.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 96-99 in File SafeMath.sol

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

✓ The code meets the specification.

Source Code with CertiK Labels

File LinearMintableToken.sol

```
pragma solidity ^0.4.24;

import "./ERC20Token.sol";
import "./Ownable.sol";

contract LinearMintableToken is ERC20Token, Ownable {
  event Minted(address recipient, uint256 mintingAmount, uint256 mintedAmount);

  uint256 constant SECONDS_IN_A_DAY = 86400;

  bool public mintingStatus = false;

  uint256 public mintingSupply;
  uint256 public intervalPeriodInDays;
  uint256 public intervalCount;
  uint256 public mintAmountPerPeriod;
  uint256 public createdTimestamp;
  uint256 public mintedAmount;

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "registerLinearMint correctness"
    @tag assume_completion
    @post mintingStatus == false
    @post totalSupply + _mintingSupply <= maxSupply
    @post _owner == msg.sender
    @post __post.mintingStatus == true
    @post __post.mintingSupply == _mintingSupply
    @post __post.mintingSupply > 0
    @post __post.mintingSupply == _mintingSupply
    @post __post.mintAmountPerPeriod == _mintAmountPerPeriod
    @post __post.mintAmountPerPeriod > 0
    @post __post.intervalPeriodInDays == _intervalPeriodInDays
    @post __post.intervalPeriodInDays > 0
    @post __post.createdTimestamp == block.timestamp
  */
  function registerLinearMint(
    uint256 _mintingSupply,
    uint256 _mintAmountPerPeriod,
    uint256 _intervalPeriodInDays
  ) external onlyOwner() {
    require(!mintingStatus);
    require(_mintingSupply > 0);
    require(totalSupply.add(_mintingSupply) <= maxSupply);
    require(_mintAmountPerPeriod > 0 );
    require(_intervalPeriodInDays > 0 );

    mintingStatus = true;

    mintingSupply = _mintingSupply;
    mintAmountPerPeriod = _mintAmountPerPeriod;
    intervalPeriodInDays = _intervalPeriodInDays;
    createdTimestamp = block.timestamp;
  }

  function linearMint() external {
    mintInternal(block.timestamp);
  }

  //@CTK FAIL NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  /*@CTK "mintInternal NO_ASF and correctness"
    @tag assume_completion
    @pre intervalPeriodInDays > 0
    @pre SECONDS_IN_A_DAY > 0
    @post mintingStatus == true
    @post !(__has_assertion_failure)
  */
  function mintInternal(uint256 blockTimestamp) internal {
    require(mintingStatus);

    address tokenOwner = owner();
    uint256 mintingAmount = calculateMintAmount(blockTimestamp);
    mintingAmount = mintingAmount.sub(mintedAmount);

    if ( mintingAmount == 0 )
      return;

    if ( mintingAmount >= mintingSupply ) {
      mintingAmount = mintingSupply.sub(mintedAmount);
      mintingStatus = false;
    }

    totalSupply = totalSupply.add(mintingAmount);

    _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);

    mintedAmount = mintedAmount.add(mintingAmount);

    emit Minted(tokenOwner, mintingAmount, mintedAmount);
  }

  //@CTK FAIL NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  /*@CTK "calculateMintAmount NO_ASF"
    @tag assume_completion
    @pre intervalPeriodInDays > 0
    @pre SECONDS_IN_A_DAY > 0
    @post !(__has_assertion_failure)
  */
  function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
      mintAmount) {
    uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
    uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;

    return pastIntervals * mintAmountPerPeriod;
  }
}
```

File 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.
   * It will not be possible to call the functions with the ‘onlyOwner‘
   * modifier anymore.
   * @notice Renouncing ownership will leave the contract without an owner,
   * thereby removing any functionality that is only available to the owner.
   */
//  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;
  }
}
```

File ERC20Token.sol

```
pragma solidity ^0.4.24;

import "./SafeMath.sol";
import "./TokenRecipient.sol";

contract ERC20Token {
  using SafeMath for uint;

  string public name;
  string public symbol;
  uint256 public totalSupply;
  uint256 public maxSupply;
  uint8 public decimals;

  mapping(address => uint256) _balances;

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

  event Transfer(address from, address to, uint value);
  event Approval(address from, address to, uint value);
  event ApproveAndCall(address spender, uint value, bytes extraData);

```

```
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "constructor correctness"
  @tag assume_completion
  @post __post._balances[msg.sender] == __post.totalSupply
  @post __post._balances[msg.sender] == _initialSupply
  @post __post.maxSupply == _maxSupply
*/
constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
    uint256 _maxSupply) public {
  require(_maxSupply >= _initialSupply);

  name = _name;
  symbol = _symbol;
  decimals = _decimals;

  _balances[msg.sender] = _initialSupply;
  totalSupply = _initialSupply;
  maxSupply = _maxSupply;
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "balanceOf correctness"
  @post balance == _balances[_owner]
 */
function balanceOf(address _owner) public view returns (uint balance) {
  return _balances[_owner];
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "allowance correctness"
  @post remaining == _allowed[_owner][_spender]
 */
function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
  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
/*@CTK "transfer correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _value <= _balances[msg.sender]
  @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
       _value
  @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
  @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
  @post success == true
 */
function transfer(address _to, uint256 _value) public returns (bool success) {
  _transfer(msg.sender, _to, _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
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
  @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]
  @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
  @post success == true
 */
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
  _transfer(_from, _to, _value);
  _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_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
/*@CTK "approve correctness"
  @tag assume_completion
  @post __post._allowed[msg.sender][_spender] == _value
 */
function approve(address _spender, uint256 _value) public returns (bool success) {
  _approve(msg.sender, _spender, _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
/*@CTK "increaseApproval correctness"
  @tag assume_completion
  @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
      _addedValue
 */
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
/*@CTK decreaseApproval0
  @pre __return == true
  @pre _allowed[msg.sender][_spender] <= _subtractedValue
  @post __post._allowed[msg.sender][_spender] == 0
*/
/*@CTK decreaseApproval
  @pre __return == true
  @pre _allowed[msg.sender][_spender] > _subtractedValue
  @post __post._allowed[msg.sender][_spender] ==
        _allowed[msg.sender][_spender] - _subtractedValue
*/
function decreaseAllowance(address _spender, uint256 _subtractedValue) public
    returns (bool) {
  _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
      ));
  return true;
}

function approveAndCall(address _spender, uint256 _value, bytes _extraData) public
    returns (bool success) {
  require(_spender != address(0));

  TokenRecipient spender = TokenRecipient(_spender);
  if (approve(_spender, _value)) {
    spender.receiveApproval(msg.sender, _value, this, _extraData);
    emit ApproveAndCall(_spender, _value, _extraData );
    return true;
  }

  return false;
}

/**
 * @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.
 */
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 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.
 */
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);
}
}
```

File MainToken.sol

```
pragma solidity ^0.4.24;

import "./LinearMintableToken.sol";

contract MainToken is LinearMintableToken {
  event Paused(address account);
  event Unpaused(address account);

  mapping (address => bool) public isLocked;
  bool private _paused;

  constructor(
    string _name,
    string _symbol,
    uint8 _decimals,
    uint256 _initialSupply,
    uint256 _maxSupply
  ) ERC20Token(_name, _symbol, _decimals, _initialSupply, _maxSupply)
  public {
    _paused = false;
  }

  modifier onlyUnlocked() {
    require(!isLocked[msg.sender], "msg.sender is locked");
    _;
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "mint"
    @tag assume_completion
    @post mintingStatus == false
    @post _owner == msg.sender
    @post __post.totalSupply == totalSupply + _amount
    @post __post.totalSupply <= maxSupply
    @post __post._balances[msg.sender] == _balances[msg.sender] + _amount
   */
  function mint(uint256 _amount) onlyOwner() external {
    require(!mintingStatus);
    uint newTotalSupply = totalSupply.add(_amount);
    address tokenOwner = owner();

    require( newTotalSupply <= maxSupply );

    _balances[tokenOwner] = _balances[tokenOwner].add(_amount);

    totalSupply = newTotalSupply;

    emit Minted(tokenOwner, _amount, _amount);
  }

  /*@CTK "lockAccount correctness"
    @tag assume_completion
    @post isLocked[_account] == false
    @post _owner == msg.sender
    @post __post.isLocked[_account] == true
   */
  function lockAccount(address _account) onlyOwner() external {
    require(!isLocked[_account]);

    isLocked[_account] = true;
  }

  /*@CTK "unlockAccount correctness"
    @tag assume_completion
    @post isLocked[_account] == true
    @post _owner == msg.sender
    @post __post.isLocked[_account] == false
   */
  function unlockAccount(address _account) onlyOwner() external {
    require(isLocked[_account]);

    isLocked[_account] = false;
  }

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

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

  /**
   * @dev Modifier to make a function callable only when the contract is paused.
   */
  modifier whenPaused() {
    require(_paused, "Pausable: not paused");
    _;
  }

  /**
   * @dev Called by a owner to pause, triggers stopped state.
   */
  /*@CTK "pause correctness"
    @tag assume_completion
    @post _paused == false
    @post _owner == msg.sender
    @post __post._paused == true
   */
  function pause() public onlyOwner whenNotPaused {
    _paused = true;
    emit Paused(msg.sender);
  }

  /**
   * @dev Called by a owner to unpause, returns to normal state.
   */
  /*@CTK "unpause correctness"
    @tag assume_completion
    @post _paused == true
    @post _owner == msg.sender
    @post __post._paused == false
   */
  function unpause() public onlyOwner whenPaused {
    _paused = false;
    emit Unpaused(msg.sender);
```

```
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "transfer correctness"
  @tag assume_completion
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post _to != 0x0
  @post _value <= _balances[msg.sender]
  @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
      _value
  @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
  @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
  @post __return == true
 */
function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
  return super.transfer(_to, _value);
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _paused == false
  @post isLocked[msg.sender] == false
  @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
  @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]
  @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
  @post __return == true
 */
function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
whenNotPaused returns
(bool) {
  return super.transferFrom(_from, _to, _value);
}

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "approve correctness"
  @post _paused == false
  @post isLocked[msg.sender] == false
  @tag assume_completion
  @post __post._allowed[msg.sender][_spender] == _value
 */
function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
     returns
(bool) {
  return super.approve(_spender, _value);
}
```

```

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "increaseApproval correctness"
    @tag assume_completion
    @post _paused == false
    @post isLocked[msg.sender] == false
    @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
        _addedValue
   */
  function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
      whenNotPaused
  returns (bool) {
    return super.increaseAllowance(_spender, _addedValue);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK decreaseApproval0
    @pre __return == true
    @pre _allowed[msg.sender][_spender] <= _subtractedValue
    @post _paused == false
    @post isLocked[msg.sender] == false
    @post __post._allowed[msg.sender][_spender] == 0
  */
  /*@CTK decreaseApproval
    @pre __return == true
    @pre _allowed[msg.sender][_spender] > _subtractedValue
    @post _paused == false
    @post isLocked[msg.sender] == false
    @post __post._allowed[msg.sender][_spender] ==
          _allowed[msg.sender][_spender] - _subtractedValue
  */
  function decreaseAllowance(address _spender, uint _subtractedValue) public
      onlyUnlocked
  whenNotPaused returns
  (bool) {
    return super.decreaseAllowance(_spender, _subtractedValue);
  }

  function approveAndCall(address _spender, uint256 _value, bytes _extraData)
      onlyUnlocked whenNotPaused
  public returns
  (bool success) {
    return super.approveAndCall(_spender, _value, _extraData);
  }
}
```

File SafeMath.sol

```
pragma solidity ^0.4.24;

/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error.
 */
library SafeMath {
```

```
/**
 * @dev Multiplies two unsigned integers, reverts on overflow.
 */
/*@CTK "SafeMath mul"
  @post (((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;
  }
}
```