# CertiK Audit Report For Nestree

Request Date: 2019-05-30
Revision Date: 2019-06-08
Platform Name: Ethereum

CERTIK

# Contents

## Disclaimer

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

## About CertiK

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

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

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

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

## Exective Summary

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

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

## Vulnerability Classification

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

**Critical**

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

**Medium**

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

**Low**

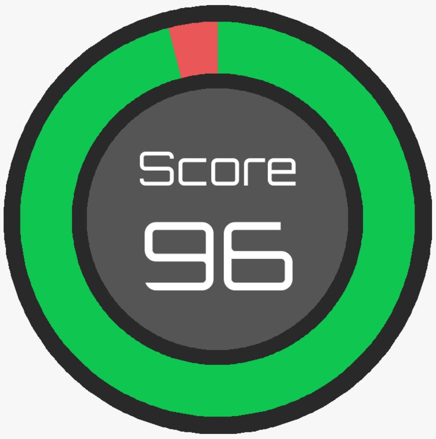
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

## Testing Summary

**PASS**

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

*Jun 08, 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. | 1 | SWC-116 |
| Insecure Compiler Version | Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used. | 0 | SWC-102 SWC-103 |
| Insecure Randomness | Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree. | 0 | SWC-120 |

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

## Vulnerability Details

### Critical

No issue found.

### Medium

No issue found.

### Low

No issue found.

# Manual Review Notes

## Scope of Work

Nestree invited CertiK to audit their soon-to-be released token based smart contracts. The goal of this audit is to review Nestree's solidity implementation on top of its business logic, detect potential security vulnerabilities, understand its general design and architecture, and uncover bugs that could compromise the system in production environment.

Above the manually reviewing process of vulnerability detection, CertiK employed our proprietary Formal Verification platform, together with static analysis tools to mathematically ensure that the entire code logic works as intended.

The audited source code **SHA-256 Checksum**:

- **BaseToken.sol** `a8f8d336541344b3aa46812abf34491550f02041ebb902780d0d9790abcb9978`
- **Nestree.sol** `767c23f430c0f9cf2dc3c3d13335eb37cddb2df543aedf3f9c1cce713c0283cc`
- **Ownable.sol** `55951b9d849def4bfc9b35957b919ddf7c1298f283ffe7b7100322949352b5d7`
- **SafeMath.sol** `388ca15d1e57c2466e4078f511d82ebd0682517667986cdb354f7002b1053db0`

### Recommendations

The items below are notes from the CertiK team in accordance to our audit. These suggestions are optional, and may have low impact to the overall aspects of the Nestree smart contracts. As such, these are optional edits for Nestree to consider for enhancement.

*Ownable.sol*

1. Our client uses `address[] public _allowed` to store allowed addresses. However the use of array data structure introduces unnecessary complexity when dealing with add/remove of items. We recommend using `mapping(address => bool)public _allowed`, which reduces the time complexity of each add/delete operation to $\mathcal{O}(1)$.
   - **[Status]**: Resolved by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`. The storage of `allowed` is switched from the array data structure to the mapping data structure.
2. Suggest to emit event at the end of each logic (after all the assignment).
   - **[Status]**: Event messages are added by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`.
3. The pull model (as compared to the push model) is a more secure practice for critical actions such as `transferOwnership`. Recommend adding a pending role such as `proposedOwner` to store new owner and letting the `proposedOwner` to claim the ownership itself later.
   - **[Status]**: Resolved by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`. The transfer of ownership is switched to the pull model.

*BaseToken.sol*

1. Regarding the `unlock()` function, all the locks belonging to address `who` will be removed by `owner`. Depending on the business logic, consider removing partial i.e. `unlock(address, idx)`.

   - **[Status]**: Partially resolved by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`.

2. Suggest keeping an `unlockAll()` function to facilitate resetting all locks of a wallet address.

   - **[Status]**: Resolved by the Nestree Team in commit `c81dab3604fc51e9b7e194821a946befcebc597a`.

3. Suggest changing the accessibility of `lock()` from `public` to `internal`/`private` as it should be invoked exclusively by `transferWithLock()` based on the business logic.

   - **[Status]**: Resolved by the Nestree Team in commit `c81dab3604fc51e9b7e194821a946befcebc597a`.

4. Recommend appending the index number of the new added Lock to the event message in `lock()`, as `unlock(address _who, uint256 _index)` uses index number to remove a specific lock. The event message should helps identifying the lock.

   - **[Status]**: Resolved by the Nestree Team in commit `c81dab3604fc51e9b7e194821a946befcebc597a`. Index number is now provided in the event message.

5. For `unlock(address _who, uint256 _index)`, the actual behavior of delete is to set the item value to default (in this case a lock struct with all fields set to 0). Nevertheless the length of the array remains the same. We suggest to use below code snippet to delete an item for a given index (swapping the item to be deleted with the last item, and truncate the length to `length-1`). Please have thorough tests to verify as the code snippet is for demo purposes only. `a[index] = a[a.length-1]; a.length--;`

   - **[Status]**: Suggestion adopted by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`.

6. Suggest adding an internal `_transfer()` to handle the transfer logic, which can be reused in `transferWithLock()`, `transfer()`.

   - **[Status]**: Suggestion adopted by the Nestree Team in commit `c81dab3604fc51e9b7e194821a946befcebc597a`. A new `_transfer()` is provided.

*NEST.sol → Nestree.sol*

1. Recommend using open source `SafeMath` library to avoid number overflow in `transferByMandate()` and `referralDrop()`.

- **[Status]**: Resolved by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`. The additions are now implemented using `add()` function from `SafeMath`.

2. From our understanding, when an address is marked as `mandated`, the owner of the contract could transfer arbitrary amount of tokens without any approval. This may lead to an over-privileged power of the owner if a user accidentally invoked the `updateMandate` function and set its`mandated` value to `true` (this introduced a time interval before the user issued another transaction to mark false). Suggest adding documentation to describe the business logics and use cases of the `mandated` functionality for better understanding.

   - **[Status]**: Resolved by the Nestree Team in commit `e4ea7f9101bd8c0a3b2e6f0e3a8af6af04740f0f`. The `mandated` functionality is removed at current stage.

3. Suggest checking if `msg.sender` can have same identity as `_to1` or `_to2` or `_sale` in function `referralDrop()`.

   - **[Status]**: Resolved by the Nestree Team in commit `4150c59a0e66f902c3fd29440f651fb7b3ba69fe`. Additional identity checks are added.

### Best practice

The checklist below helps to reflect the general quality of the solidity project.

#### Solidity Protocol

✓ Use latest stable solidity version

✓ Handle possible errors properly when making external calls

✓ Provide error message along with require()

✓ Use modifiers properly

✓ Use events to monitor contract activities

✓ Refer and use libraries properly

✓ No compiler warnings

#### Privilege Control

✓ Provide stop functionality for control and emergency handling

✓ Restrict access to sensitive functions

#### Documentation

– Provide project documentation and execution guidance

✓ Provide inline comment for function intention

✓ Provide instruction to initialize and execute the test files

**Testing**

– Provide migration scripts

– Provide test scripts and coverage for potential scenarios

With the final update of the source code and delivery of the audit report, CertiK is able to conclude that the Nestree contract is not vulnerable to any classically known anti-patterns or security issues.

While this CertiK review is a strong and positive indication, the audit report itself is not necessarily a guarantee of correctness or trustworthiness. CertiK always recommends seeking multiple opinions, test coverage, sandbox deployments before any mainnet release.

## Static Analysis Results

### INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
pragma solidity ^0.5.8;
```

Only these compiler versions are safe to compile your code: 0.5.9

### INSECURE_COMPILER_VERSION

Line 1 in File Nestree.sol

```
pragma solidity ^0.5.8;
```

Only these compiler versions are safe to compile your code: 0.5.9

### INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
pragma solidity ^0.5.8;
```

Only these compiler versions are safe to compile your code: 0.5.9

### INSECURE_COMPILER_VERSION

Line 1 in File BaseToken.sol

```
pragma solidity ^0.5.8;
```

Only these compiler versions are safe to compile your code: 0.5.9

### TIMESTAMP_DEPENDENCY

Line 89 in File BaseToken.sol

```
89                  if (now < locks[i].expiresAt)
```

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

### TIMESTAMP_DEPENDENCY

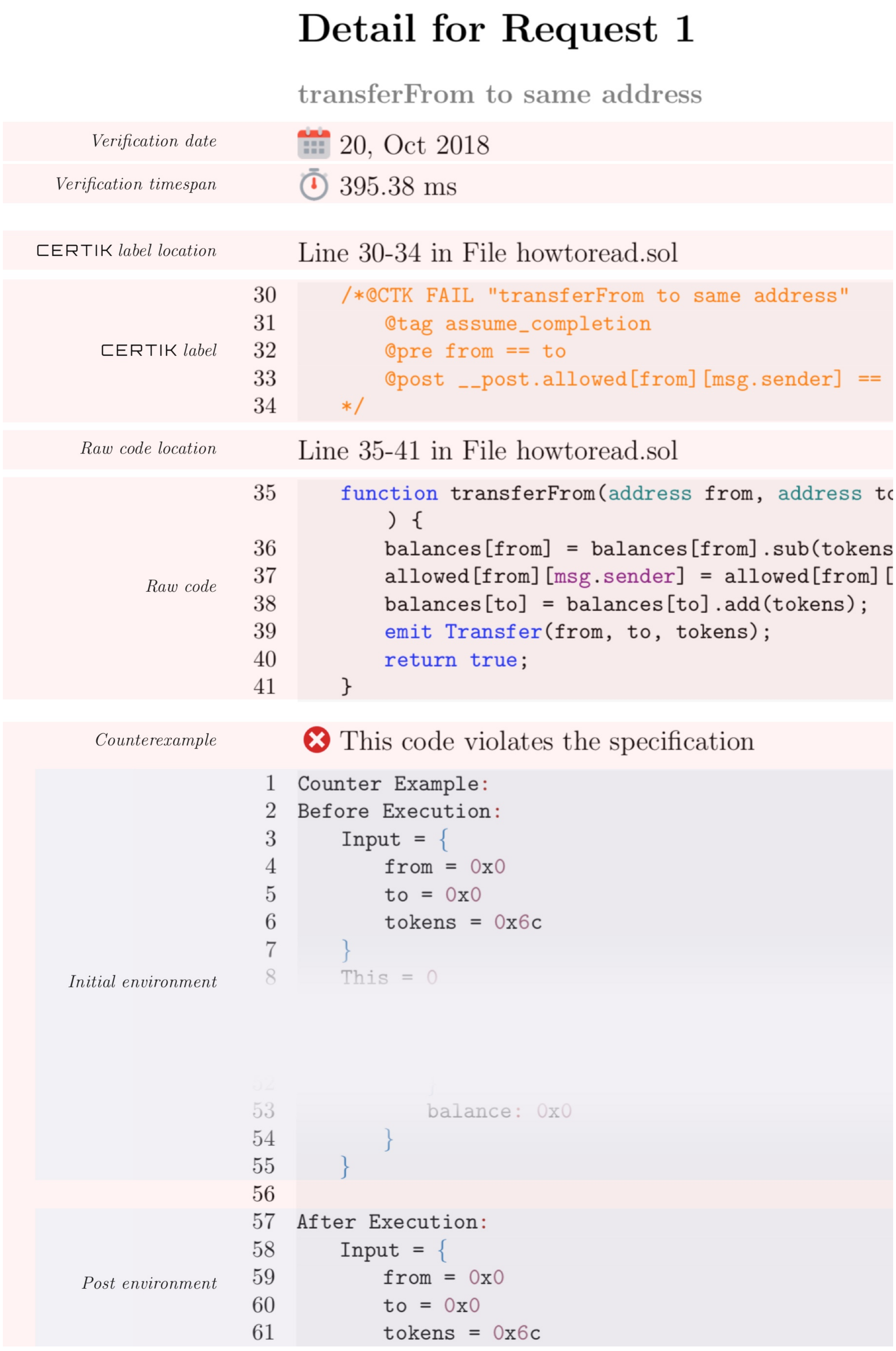
Line 177 in File BaseToken.sol

```
177         require(_time > now, ERROR_TIME_IS_PAST);
```

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

## Formal Verification Results

### How to read



**Detail for Request 1**

transferFrom to same address

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

CERTIK *label*

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

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

*Raw code*

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

*Counterexample* This code violates the specification

*Initial environment*

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            balance: 0x0
        }
    }
```

*Post environment*

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

## Formal Verification Request 1

### Ownable

📅 08, Jun 2019
⏱ 42.94 ms

Line 22-25 in File Ownable.sol

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

Line 26-31 in File Ownable.sol

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

✅ The code meets the specification

## Formal Verification Request 2

### owner

📅 08, Jun 2019
⏱ 5.67 ms

Line 33-35 in File Ownable.sol

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

Line 36-39 in File Ownable.sol

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

✅ The code meets the specification

## Formal Verification Request 3

### isOwner

📅 08, Jun 2019
⏱ 5.36 ms

Line 59-61 in File Ownable.sol

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

Line 62-65 in File Ownable.sol

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

✓ The code meets the specification

## Formal Verification Request 4

### isOwner

08, Jun 2019
5.6 ms

Line 67-69 in File Ownable.sol

```
/*@CTK isOwner
  @post __return == _allowed[msg.sender]
 */
```

Line 70-73 in File Ownable.sol

```
function isAllowed() public view returns (bool)
{
    return _allowed[msg.sender];
}
```

✓ The code meets the specification

## Formal Verification Request 5

### allow

08, Jun 2019
33.76 ms

Line 75-79 in File Ownable.sol

```
/*@CTK allow
  @tag assume_completion
  @post msg.sender == _owner
  @post __post._allowed[_target]
 */
```

Line 80-85 in File Ownable.sol

```
function allow(address _target) external onlyOwner returns (bool)
{
    _allowed[_target] = true;
    emit Allowed(_target);
    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 6

### removeAllowed

08, Jun 2019
28.36 ms

Line 87-91 in File Ownable.sol

```
    /*@CTK removeAllowed
      @tag assume_completion
      @post msg.sender == _owner
      @post !(__post._allowed[_target])
     */
```

Line 92-97 in File Ownable.sol

```
    function removeAllowed(address _target) external onlyOwner returns (bool)
    {
        _allowed[_target] = false;
        emit RemoveAllowed(_target);
        return true;
    }
```

✓ The code meets the specification

## Formal Verification Request 7

### isStopped

08, Jun 2019
38.12 ms

Line 99-103 in File Ownable.sol

```
    /*@CTK isStopped
      @tag assume_completion
      @post (msg.sender == _owner) || _allowed[msg.sender] -> __return == false
      @post (msg.sender != _owner) && !(_allowed[msg.sender]) -> __return == stopped
     */
```

Line 104-114 in File Ownable.sol

```
    function isStopped() public view returns (bool)
    {
        if(isOwner() || isAllowed())
        {
            return false;
        }
        else
        {
            return stopped;
        }
    }
```

✓ The code meets the specification

## Formal Verification Request 8

stop

📅 08, Jun 2019
⏱ 53.47 ms

Line 116-120 in File Ownable.sol

```
    /*@CTK stop
      @tag assume_completion
      @post msg.sender == _owner
      @post __post.stopped
     */
```

Line 121-124 in File Ownable.sol

```
    function stop() public onlyOwner
    {
        _stop();
    }
```

✓ The code meets the specification

## Formal Verification Request 9

start

📅 08, Jun 2019
⏱ 51.84 ms

Line 126-130 in File Ownable.sol

```
    /*@CTK start
      @tag assume_completion
      @post msg.sender == _owner
      @post !__post.stopped
     */
```

Line 131-134 in File Ownable.sol

```
    function start() public onlyOwner
    {
        _start();
    }
```

✓ The code meets the specification

## Formal Verification Request 10

proposeOwner

📅 08, Jun 2019
⏱ 35.07 ms

Line 136-141 in File Ownable.sol

```
/*@CTK proposeOwner
  @tag assume_completion
  @post msg.sender == _owner
  @post msg.sender != _proposedOwner
  @post __post.proposedOwner == _proposedOwner
 */
```

Line 142-146 in File Ownable.sol

```
function proposeOwner(address _proposedOwner) public onlyOwner
{
    require(msg.sender != _proposedOwner, ERROR_CALLER_ALREADY_OWNER);
    proposedOwner = _proposedOwner;
}
```

✓ The code meets the specification

## Formal Verification Request 11

proposeOwner

08, Jun 2019
21.08 ms

Line 148-153 in File Ownable.sol

```
/*@CTK proposeOwner
  @tag assume_completion
  @post msg.sender == proposedOwner
  @post __post._owner == proposedOwner
  @post __post.proposedOwner == address(0)
 */
```

Line 154-162 in File Ownable.sol

```
function claimOwnership() public
{
    require(msg.sender == proposedOwner, ERROR_NOT_PROPOSED_OWNER);

    emit OwnershipTransferred(_owner, proposedOwner);

    _owner = proposedOwner;
    proposedOwner = address(0);
}
```

✓ The code meets the specification

## Formal Verification Request 12

stop

08, Jun 2019
0.81 ms

Line 164-166 in File Ownable.sol

```
/*@CTK stop
  @post __post.stopped
 */
```

Line 167-171 in File Ownable.sol

```
function _stop() internal
{
    emit Stopped();
    stopped = true;
}
```

✓ The code meets the specification

## Formal Verification Request 13

### _start

08, Jun 2019
0.77 ms

Line 173-175 in File Ownable.sol

```
/*@CTK _start
  @post !__post.stopped
 */
```

Line 176-180 in File Ownable.sol

```
function _start() internal
{
    emit Started();
    stopped = false;
}
```

✓ The code meets the specification

## Formal Verification Request 14

### referralDrop

08, Jun 2019
12283.48 ms

Line 29-38 in File Nestree.sol

```
/*@CTK referralDrop
  @tag assume_completion
  @post (msg.sender == _owner) || (_allowed[msg.sender]) || !(stopped)
  @post _to1 != address(0)
  @post _to2 != address(0)
  @post _sale != address(0)
  @post (msg.sender != _to1) && (msg.sender != _to2) && (msg.sender != _sale)
  @post balances[msg.sender] >= _value1 + _value2 + _fee
  @post __post.balances[msg.sender] == balances[msg.sender] - (_value1 + _value2 +
       _fee)
 */
```

Line 39-67 in File Nestree.sol

```
function referralDrop(address _to1, uint256 _value1, address _to2, uint256 _value2
    , address _sale, uint256 _fee) external onlyWhenNotStopped returns (bool)
{
    require(_to1 != address(0), ERROR_ADDRESS_NOT_VALID);
    require(_to2 != address(0), ERROR_ADDRESS_NOT_VALID);
    require(_sale != address(0), ERROR_ADDRESS_NOT_VALID);
    require(balances[msg.sender] >= _value1.add(_value2).add(_fee),
        ERROR_VALUE_NOT_VALID);
    require(!isLocked(msg.sender, _value1.add(_value2).add(_fee)), ERROR_LOCKED);
    require(msg.sender != _to1 && msg.sender != _to2 && msg.sender != _sale,
        ERROR_DUPLICATE_ADDRESS);

    balances[msg.sender] = balances[msg.sender].sub(_value1.add(_value2).add(_fee))
        ;

    if(_value1 > 0)
    {
        balances[_to1] = balances[_to1].add(_value1);
    }

    if(_value2 > 0)
    {
        balances[_to2] = balances[_to2].add(_value2);
    }

    if(_fee > 0)
    {
        balances[_sale] = balances[_sale].add(_fee);
    }

    emit ReferralDrop(msg.sender, _to1, _value1, _to2, _value2);
    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 15

### SafeMath mul

📅 08, Jun 2019
⏱ 302.42 ms

Line 11-16 in File SafeMath.sol

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

Line 17-29 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 16

### SafeMath div

08, Jun 2019
12.3 ms

Line 34-38 in File SafeMath.sol

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

Line 39-46 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 17

### SafeMath sub

08, Jun 2019
11.54 ms

Line 51-55 in File SafeMath.sol

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

Line 56-61 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 18

### SafeMath add

08, Jun 2019
14.44 ms

Line 66-70 in File SafeMath.sol

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

Line 71-76 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 19

### SafeMath mod

08, Jun 2019
12.42 ms

Line 82-87 in File SafeMath.sol

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

Line 88-91 in File SafeMath.sol

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

✓ The code meets the specification

## Formal Verification Request 20

### BaseToken

08, Jun 2019
12.26 ms

Line 46-48 in File BaseToken.sol

```
    /*@CTK BaseToken
      @post __post.balances[msg.sender] == totalSupply
     */
```

Line 49-52 in File BaseToken.sol

```
    constructor() public
    {
        balances[msg.sender] = totalSupply;
    }
```

✓ The code meets the specification

## Formal Verification Request 21

### balanceOf

08, Jun 2019
4.97 ms

Line 65-67 in File BaseToken.sol

```
    /*@CTK balanceOf
      @post __return == balances[_who]
     */
```

Line 68-71 in File BaseToken.sol

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

✓ The code meets the specification

## Formal Verification Request 22

### allowance

08, Jun 2019
5.71 ms

Line 99-101 in File BaseToken.sol

```
    /*@CTK allowance
      @post __return == approvals[_owner][_spender]
     */
```

Line 102-105 in File BaseToken.sol

```
    function allowance(address _owner, address _spender) view external returns (
        uint256)
    {
        return approvals[_owner][_spender];
    }
```

✓ The code meets the specification

## Formal Verification Request 23

**transfer**

08, Jun 2019
1503.27 ms

Line 125-133 in File BaseToken.sol

```
    /*@CTK transfer
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !(stopped)
      @post _to != address(0)
      @post _value > 0
      @post balances[msg.sender] >= _value
      @post (msg.sender != _to) -> __post.balances[msg.sender] == balances[msg.sender]
           - _value
      @post (msg.sender != _to) -> __post.balances[_to] == balances[_to] + _value
     */
```

Line 134-139 in File BaseToken.sol

```
    function transfer(address _to, uint256 _value) external onlyWhenNotStopped
        transferParamsValidation(msg.sender, _to, _value) returns (bool)
    {
        _transfer(msg.sender, _to, _value);

        return true;
    }
```

✓ The code meets the specification

## Formal Verification Request 24

**transferFrom**

08, Jun 2019
1756.37 ms

Line 141-152 in File BaseToken.sol

```
/*@CTK transferFrom
  @tag assume_completion
  @post (msg.sender == _owner) || (_allowed[msg.sender]) || !stopped
  @post _from != address(0)
  @post _to != address(0)
  @post _value > 0
  @post balances[_from] >= _value
  @post approvals[_from][msg.sender] >= _value
  @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
      _value
  @post (_from != _to) -> __post.balances[_from] == balances[_from] - _value
  @post (_from != _to) -> __post.balances[_to] == balances[_to] + _value
 */
```

Line 153-162 in File BaseToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) external
    onlyWhenNotStopped transferParamsValidation(_from, _to, _value) returns (bool)
{
    require(approvals[_from][msg.sender] >= _value,
        ERROR_APPROVED_BALANCE_NOT_ENOUGH);

    approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);

    _transfer(_from, _to, _value);

    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 25

### transferWithLock

08, Jun 2019
1284.17 ms

Line 164-174 in File BaseToken.sol

```
/*@CTK transferWithLock
  @tag assume_completion
  @post _owner == msg.sender
  @post _to != address(0)
  @post _value > 0
  @post balances[msg.sender] >= _value
  @post __post.lockup[_to][lockup[_to].length].amount == _value
  @post __post.lockup[_to][lockup[_to].length].expiresAt == _time
  @post (msg.sender != _to) -> __post.balances[msg.sender] == balances[msg.sender]
       - _value
  @post (msg.sender != _to) -> __post.balances[_to] == balances[_to] + _value
 */
```

Line 175-183 in File BaseToken.sol

```
function transferWithLock(address _to, uint256 _value, uint256 _time) onlyOwner
    transferParamsValidation(msg.sender, _to, _value) external returns (bool)
{
    require(_time > now, ERROR_TIME_IS_PAST);

    _lock(_to, _value, _time);
    _transfer(msg.sender, _to, _value);

    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 26

### approve

📅 08, Jun 2019
⏱ 121.26 ms

Line 186-193 in File BaseToken.sol

```
    /*@CTK approve
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !stopped
      @post _spender != address(0)
      @post balances[msg.sender] >= _value
      @post msg.sender != _spender
      @post __post.approvals[msg.sender][_spender] == _value
     */
```

Line 194-205 in File BaseToken.sol

```
    function approve(address _spender, uint256 _value) external onlyWhenNotStopped
        returns (bool)
    {
        require(_spender != address(0), ERROR_VALUE_NOT_VALID);
        require(balances[msg.sender] >= _value, ERROR_BALANCE_NOT_ENOUGH);
        require(msg.sender != _spender, ERROR_ADDRESS_IS_SAME);

        approvals[msg.sender][_spender] = _value;

        emit Approval(msg.sender, _spender, _value);

        return true;
    }
```

✓ The code meets the specification

## Formal Verification Request 27

### unlock

📅 08, Jun 2019
⏱ 148.56 ms

Line 209-216 in File BaseToken.sol

```
/*@CTK unlock
  @tag assume_completion
  @post _owner == msg.sender
  @post lockup[_who].length > _index
  @post __post.lockup[_who][_index].amount == lockup[_who][lockup[_who].length -
      1].amount
  @post __post.lockup[_who][_index].expiresAt == lockup[_who][lockup[_who].length
      - 1].expiresAt
  @post __post.lockup[_who].length == lockup[_who].length - 1
 */
```

Line 217-228 in File BaseToken.sol

```
function unlock(address _who, uint256 _index) onlyOwner external returns (bool)
{
    uint256 length = lockup[_who].length;
    require(length > _index, ERROR_OUT_OF_INDEX);

    lockup[_who][_index] = lockup[_who][length - 1];
    lockup[_who].length--;

    emit UnlockedIndex(_who, _index);

    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 28

### unlockAll

08, Jun 2019
39.49 ms

Line 230-235 in File BaseToken.sol

```
/*@CTK unlockAll
  @tag assume_completion
  @post _owner == msg.sender
  @post lockup[_who].length > 0
  @post __post.lockup[_who].length == 0
 */
```

Line 236-244 in File BaseToken.sol

```
function unlockAll(address _who) onlyOwner external returns (bool)
{
    require(lockup[_who].length > 0, ERROR_NO_LOCKUP);

    delete lockup[_who];
    emit UnlockedAll(_who);

    return true;
}
```

✓ The code meets the specification

## Formal Verification Request 29

### burn

📅 08, Jun 2019
⏱ 278.05 ms

Line 246-252 in File BaseToken.sol

```
    /*@CTK burn
      @tag assume_completion
      @post balances[msg.sender] >= _value
      @post _value > 0
      @post __post.balances[msg.sender] == balances[msg.sender] - _value
      @post __post.totalSupply == totalSupply - _value
     */
```

Line 253-263 in File BaseToken.sol

```
    function burn(uint256 _value) external
    {
        require(balances[msg.sender] >= _value, ERROR_BALANCE_NOT_ENOUGH);
        require(_value > 0, ERROR_VALUE_NOT_VALID);

        balances[msg.sender] = balances[msg.sender].sub(_value);

        totalSupply = totalSupply.sub(_value);

        emit Burn(msg.sender, _value);
    }
```

✅ The code meets the specification

## Formal Verification Request 30

### _transfer

📅 08, Jun 2019
⏱ 94.77 ms

Line 266-271 in File BaseToken.sol

```
    /*@CTK _transfer
      @tag assume_completion
      @post (_from == _to) -> __post.balances[_from] == balances[_from]
      @post (_from != _to) -> __post.balances[_to] == balances[_to] + _value
      @post (_from != _to) -> __post.balances[_from] == balances[_from] - _value
     */
```

Line 272-278 in File BaseToken.sol

```
    function _transfer(address _from, address _to, uint256 _value) internal
    {
        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 31

_lock

08, Jun 2019
4.01 ms

Line 280-286 in File BaseToken.sol

```
    /*@CTK _lock
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.lockup[_who][lockup[_who].length].amount == _value
      @post __post.lockup[_who][lockup[_who].length].expiresAt == _dateTime
      @post __post.lockup[_who].length == lockup[_who].length + 1
     */
```

Line 287-292 in File BaseToken.sol

```
    function _lock(address _who, uint256 _value, uint256 _dateTime) onlyOwner internal
    {
        lockup[_who].push(Lock(_value, _dateTime));

        emit Locked(_who, lockup[_who].length - 1);
    }
```

✓ The code meets the specification

## Formal Verification Request 32

lockedBalanceOf for__Generated

08, Jun 2019
41.55 ms

(Loop) Line 83-86 in File BaseToken.sol

```
            /*@CTK "lockedBalanceOf for"
              @inv lockedBalance >= 0
              @post !__should_return
             */
```

(Loop) Line 83-93 in File BaseToken.sol

```
            /*@CTK "lockedBalanceOf for"
              @inv lockedBalance >= 0
              @post !__should_return
             */
            for (uint i = 0; i < length; i++)
            {
                if (now < locks[i].expiresAt)
                {
                    lockedBalance = lockedBalance.add(locks[i].amount);
                }
            }
```

✓ The code meets the specification

## Source Code with CertiK Labels

File Ownable.sol

```
pragma solidity ^0.5.8;

contract Ownable
{
    string constant internal ERROR_NO_HAVE_PERMISSION = 'Reason: No have permission.';
    string constant internal ERROR_IS_STOPPED      = 'Reason: Is stopped.';
    string constant internal ERROR_ADDRESS_NOT_VALID = 'Reason: Address is not valid.'
        ;
    string constant internal ERROR_CALLER_ALREADY_OWNER = 'Reason: Caller already is
        owner';
    string constant internal ERROR_NOT_PROPOSED_OWNER = 'Reason: Not proposed owner';

    bool private stopped;
    address private _owner;
    address private proposedOwner;
    mapping(address => bool) private _allowed;

    event Stopped();
    event Started();
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
        );
    event Allowed(address indexed _address);
    event RemoveAllowed(address indexed _address);

    /*@CTK Ownable
      @post !__post.stopped
      @post __post._owner == msg.sender
     */
    constructor () internal
    {
        stopped = false;
        _owner = msg.sender;
        emit OwnershipTransferred(address(0), _owner);
    }

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

    modifier onlyOwner()
    {
        require(isOwner(), ERROR_NO_HAVE_PERMISSION);
        _;
    }

    modifier onlyAllowed()
    {
        require(isAllowed() || isOwner(), ERROR_NO_HAVE_PERMISSION);
        _;
    }

    modifier onlyWhenNotStopped()
    {
        require(!isStopped(), ERROR_IS_STOPPED);
        _;
    }

    /*@CTK isOwner
      @post __return == (msg.sender == _owner)
     */
    function isOwner() public view returns (bool)
    {
        return msg.sender == _owner;
    }

    /*@CTK isOwner
      @post __return == _allowed[msg.sender]
     */
    function isAllowed() public view returns (bool)
    {
        return _allowed[msg.sender];
    }

    /*@CTK allow
      @tag assume_completion
      @post msg.sender == _owner
      @post __post._allowed[_target]
     */
    function allow(address _target) external onlyOwner returns (bool)
    {
        _allowed[_target] = true;
        emit Allowed(_target);
        return true;
    }

    /*@CTK removeAllowed
      @tag assume_completion
      @post msg.sender == _owner
      @post !(__post._allowed[_target])
     */
    function removeAllowed(address _target) external onlyOwner returns (bool)
    {
        _allowed[_target] = false;
        emit RemoveAllowed(_target);
        return true;
    }

    /*@CTK isStopped
      @tag assume_completion
      @post (msg.sender == _owner) || _allowed[msg.sender] -> __return == false
      @post (msg.sender != _owner) && !(_allowed[msg.sender]) -> __return == stopped
     */
    function isStopped() public view returns (bool)
    {
        if(isOwner() || isAllowed())
        {
            return false;
        }
        else
        {
            return stopped;
        }
    }

    /*@CTK stop
      @tag assume_completion
      @post msg.sender == _owner
      @post __post.stopped
     */
    function stop() public onlyOwner
    {
        _stop();
    }

    /*@CTK start
      @tag assume_completion
      @post msg.sender == _owner
      @post !__post.stopped
     */
    function start() public onlyOwner
    {
        _start();
    }

    /*@CTK proposeOwner
      @tag assume_completion
      @post msg.sender == _owner
      @post msg.sender != _proposedOwner
      @post __post.proposedOwner == _proposedOwner
     */
    function proposeOwner(address _proposedOwner) public onlyOwner
    {
        require(msg.sender != _proposedOwner, ERROR_CALLER_ALREADY_OWNER);
        proposedOwner = _proposedOwner;
    }

    /*@CTK proposeOwner
      @tag assume_completion
      @post msg.sender == proposedOwner
      @post __post._owner == proposedOwner
      @post __post.proposedOwner == address(0)
     */
    function claimOwnership() public
    {
        require(msg.sender == proposedOwner, ERROR_NOT_PROPOSED_OWNER);

        emit OwnershipTransferred(_owner, proposedOwner);

        _owner = proposedOwner;
        proposedOwner = address(0);
    }

    /*@CTK stop
      @post __post.stopped
     */
    function _stop() internal
    {
        emit Stopped();
        stopped = true;
    }

    /*@CTK _start
      @post !__post.stopped
     */
    function _start() internal
    {
        emit Started();
        stopped = false;
    }
}
```

File Nestree.sol

```
pragma solidity ^0.5.8;

import "./SafeMath.sol";
import "./BaseToken.sol";

contract Nestree is BaseToken
{
    using SafeMath for uint256;

    string constant internal ERROR_DUPLICATE_ADDRESS = 'Reason: msg.sender and
        receivers can not be the same.';

    // MARK: token information.
    string constant public name = 'Nestree';
    string constant public symbol = 'EGG';
    string constant public version = '1.0.0';

    // MARK: events
    event ReferralDrop(address indexed from, address indexed to1, uint256 value1,
        address indexed to2, uint256 value2);

    /*CTK Nestree
      @post __post.balances[msg.sender] == __post.totalSupply
     */
    constructor() public
    {
        totalSupply = 3000000000 * E18;
        balances[msg.sender] = totalSupply;
    }

    /*@CTK referralDrop
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !(stopped)
      @post _to1 != address(0)
      @post _to2 != address(0)
      @post _sale != address(0)
      @post (msg.sender != _to1) && (msg.sender != _to2) && (msg.sender != _sale)
      @post balances[msg.sender] >= _value1 + _value2 + _fee
      @post __post.balances[msg.sender] == balances[msg.sender] - (_value1 + _value2 +
          _fee)
     */
    function referralDrop(address _to1, uint256 _value1, address _to2, uint256 _value2
        , address _sale, uint256 _fee) external onlyWhenNotStopped returns (bool)
    {
        require(_to1 != address(0), ERROR_ADDRESS_NOT_VALID);
        require(_to2 != address(0), ERROR_ADDRESS_NOT_VALID);
        require(_sale != address(0), ERROR_ADDRESS_NOT_VALID);
        require(balances[msg.sender] >= _value1.add(_value2).add(_fee),
            ERROR_VALUE_NOT_VALID);
        require(!isLocked(msg.sender, _value1.add(_value2).add(_fee)), ERROR_LOCKED);
        require(msg.sender != _to1 && msg.sender != _to2 && msg.sender != _sale,
            ERROR_DUPLICATE_ADDRESS);

        balances[msg.sender] = balances[msg.sender].sub(_value1.add(_value2).add(_fee))
            ;

        if(_value1 > 0)
        {
            balances[_to1] = balances[_to1].add(_value1);
        }

        if(_value2 > 0)
        {
            balances[_to2] = balances[_to2].add(_value2);
        }

        if(_fee > 0)
        {
            balances[_sale] = balances[_sale].add(_fee);
        }

        emit ReferralDrop(msg.sender, _to1, _value1, _to2, _value2);
        return true;
    }
}
```

File SafeMath.sol

```
pragma solidity ^0.5.8;

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

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

        return c;
    }

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

        return c;
    }

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

        return c;
    }

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

        return c;
    }

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

File BaseToken.sol

```
pragma solidity ^0.5.8;

import "./SafeMath.sol";
import "./Ownable.sol";

contract BaseToken is Ownable
{
    using SafeMath for uint256;

    // MARK: error message.
    string constant internal ERROR_APPROVED_BALANCE_NOT_ENOUGH = 'Reason: Approved
        balance is not enough.';
    string constant internal ERROR_BALANCE_NOT_ENOUGH      = 'Reason: Balance is not
        enough.';
    string constant internal ERROR_LOCKED                  = 'Reason: Locked.';
    string constant internal ERROR_ADDRESS_NOT_VALID       = 'Reason: Address is not
        valid.';
    string constant internal ERROR_ADDRESS_IS_SAME         = 'Reason: Address is same.';
    string constant internal ERROR_VALUE_NOT_VALID         = 'Reason: Value must be
        greater than 0.';
    string constant internal ERROR_NO_LOCKUP               = 'Reason: There is no lockup
        .';
    string constant internal ERROR_DATE_TIME_NOT_VALID     = 'Reason: Datetime must
        grater or equals than zero.';
    string constant internal ERROR_OUT_OF_INDEX            = 'Reason: Out of index.';
    string constant internal ERROR_TIME_IS_PAST            = 'Reason: Time is past.';

    // MARK: basic token information.
    uint256 constant internal E18   = 1000000000000000000;
    uint256 constant public decimals = 18;
    uint256 public totalSupply;

    struct Lock {
        uint256 amount;
        uint256 expiresAt;
    }

    mapping (address => uint256) public balances;
    mapping (address => mapping ( address => uint256 )) public approvals;
    mapping (address => Lock[]) public lockup;


    // MARK: events
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    event Locked(address _who,uint256 _index);
    event UnlockedAll(address _who);
    event UnlockedIndex(address _who, uint256 _index);
    event Burn(address indexed from, uint256 indexed value);

    /*@CTK BaseToken
      @post __post.balances[msg.sender] == totalSupply
     */
    constructor() public
    {
        balances[msg.sender] = totalSupply;
    }

    modifier transferParamsValidation(address _from, address _to, uint256 _value)
    {
        require(_from != address(0), ERROR_ADDRESS_NOT_VALID);
        require(_to != address(0), ERROR_ADDRESS_NOT_VALID);
        require(_value > 0, ERROR_VALUE_NOT_VALID);
        require(balances[_from] >= _value, ERROR_BALANCE_NOT_ENOUGH);
        require(!isLocked(_from, _value), ERROR_LOCKED);
        _;
    }

    // MARK: functions for view data
    /*@CTK balanceOf
      @post __return == balances[_who]
     */
    function balanceOf(address _who) view public returns (uint256)
    {
        return balances[_who];
    }

    function lockedBalanceOf(address _who) view public returns (uint256)
    {
        require(_who != address(0), ERROR_ADDRESS_NOT_VALID);

        uint256 lockedBalance = 0;
        if(lockup[_who].length > 0)
        {
            Lock[] storage locks = lockup[_who];

            uint256 length = locks.length;
            /*@CTK "lockedBalanceOf for"
              @inv lockedBalance >= 0
              @post !__should_return
             */
            for (uint i = 0; i < length; i++)
            {
                if (now < locks[i].expiresAt)
                {
                    lockedBalance = lockedBalance.add(locks[i].amount);
                }
            }
        }

        return lockedBalance;
    }

    /*@CTK allowance
      @post __return == approvals[_owner][_spender]
     */
    function allowance(address _owner, address _spender) view external returns (
        uint256)
    {
        return approvals[_owner][_spender];
    }

    // true: _who can transfer token
    // false: _who can't transfer token
    function isLocked(address _who, uint256 _value) view public returns(bool)
    {
        uint256 lockedBalance = lockedBalanceOf(_who);
        uint256 balance = balanceOf(_who);

        if(lockedBalance <= 0)
        {
            return false;
        }
        else
        {
            return !(balance > lockedBalance && balance.sub(lockedBalance) >= _value);
        }
    }

    // MARK: functions for token transfer
    /*@CTK transfer
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !(stopped)
      @post _to != address(0)
      @post _value > 0
      @post balances[msg.sender] >= _value
      @post (msg.sender != _to) -> __post.balances[msg.sender] == balances[msg.sender]
          - _value
      @post (msg.sender != _to) -> __post.balances[_to] == balances[_to] + _value
     */
    function transfer(address _to, uint256 _value) external onlyWhenNotStopped
        transferParamsValidation(msg.sender, _to, _value) returns (bool)
    {
        _transfer(msg.sender, _to, _value);

        return true;
    }

    /*@CTK transferFrom
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !stopped
      @post _from != address(0)
      @post _to != address(0)
      @post _value > 0
      @post balances[_from] >= _value
      @post approvals[_from][msg.sender] >= _value
      @post __post.approvals[_from][msg.sender] == approvals[_from][msg.sender] -
```

```
        _value
      @post (_from != _to) -> __post.balances[_from] == balances[_from] - _value
      @post (_from != _to) -> __post.balances[_to] == balances[_to] + _value
     */
    function transferFrom(address _from, address _to, uint256 _value) external
        onlyWhenNotStopped transferParamsValidation(_from, _to, _value) returns (bool)
    {
        require(approvals[_from][msg.sender] >= _value,
            ERROR_APPROVED_BALANCE_NOT_ENOUGH);

        approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);

        _transfer(_from, _to, _value);

        return true;
    }

    /*@CTK transferWithLock
      @tag assume_completion
      @post _owner == msg.sender
      @post _to != address(0)
      @post _value > 0
      @post balances[msg.sender] >= _value
      @post __post.lockup[_to][lockup[_to].length].amount == _value
      @post __post.lockup[_to][lockup[_to].length].expiresAt == _time
      @post (msg.sender != _to) -> __post.balances[msg.sender] == balances[msg.sender]
          - _value
      @post (msg.sender != _to) -> __post.balances[_to] == balances[_to] + _value
     */
    function transferWithLock(address _to, uint256 _value, uint256 _time) onlyOwner
        transferParamsValidation(msg.sender, _to, _value) external returns (bool)
    {
        require(_time > now, ERROR_TIME_IS_PAST);

        _lock(_to, _value, _time);
        _transfer(msg.sender, _to, _value);

        return true;
    }

    // MARK: utils for transfer authentication
    /*@CTK approve
      @tag assume_completion
      @post (msg.sender == _owner) || (_allowed[msg.sender]) || !stopped
      @post _spender != address(0)
      @post balances[msg.sender] >= _value
      @post msg.sender != _spender
      @post __post.approvals[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint256 _value) external onlyWhenNotStopped
        returns (bool)
    {
        require(_spender != address(0), ERROR_VALUE_NOT_VALID);
        require(balances[msg.sender] >= _value, ERROR_BALANCE_NOT_ENOUGH);
        require(msg.sender != _spender, ERROR_ADDRESS_IS_SAME);

        approvals[msg.sender][_spender] = _value;

        emit Approval(msg.sender, _spender, _value);

        return true;
    }

    // MARK: utils for amount of token
    // Lock up token until specific date time.
    /*@CTK unlock
      @tag assume_completion
      @post _owner == msg.sender
      @post lockup[_who].length > _index
      @post __post.lockup[_who][_index].amount == lockup[_who][lockup[_who].length -
          1].amount
      @post __post.lockup[_who][_index].expiresAt == lockup[_who][lockup[_who].length
          - 1].expiresAt
      @post __post.lockup[_who].length == lockup[_who].length - 1
     */
    function unlock(address _who, uint256 _index) onlyOwner external returns (bool)
    {
        uint256 length = lockup[_who].length;
        require(length > _index, ERROR_OUT_OF_INDEX);

        lockup[_who][_index] = lockup[_who][length - 1];
        lockup[_who].length--;

        emit UnlockedIndex(_who, _index);

        return true;
    }

    /*@CTK unlockAll
      @tag assume_completion
      @post _owner == msg.sender
      @post lockup[_who].length > 0
      @post __post.lockup[_who].length == 0
     */
    function unlockAll(address _who) onlyOwner external returns (bool)
    {
        require(lockup[_who].length > 0, ERROR_NO_LOCKUP);

        delete lockup[_who];
        emit UnlockedAll(_who);

        return true;
    }

    /*@CTK burn
      @tag assume_completion
      @post balances[msg.sender] >= _value
      @post _value > 0
      @post __post.balances[msg.sender] == balances[msg.sender] - _value
      @post __post.totalSupply == totalSupply - _value
     */
    function burn(uint256 _value) external
    {
        require(balances[msg.sender] >= _value, ERROR_BALANCE_NOT_ENOUGH);
        require(_value > 0, ERROR_VALUE_NOT_VALID);

        balances[msg.sender] = balances[msg.sender].sub(_value);

        totalSupply = totalSupply.sub(_value);

        emit Burn(msg.sender, _value);
    }

    // MARK: internal functions
    /*@CTK _transfer
      @tag assume_completion
      @post (_from == _to) -> __post.balances[_from] == balances[_from]
      @post (_from != _to) -> __post.balances[_to] == balances[_to] + _value
      @post (_from != _to) -> __post.balances[_from] == balances[_from] - _value
     */
    function _transfer(address _from, address _to, uint256 _value) internal
    {
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);

        emit Transfer(_from, _to, _value);
    }

    /*@CTK _lock
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.lockup[_who][lockup[_who].length].amount == _value
      @post __post.lockup[_who][lockup[_who].length].expiresAt == _dateTime
      @post __post.lockup[_who].length == lockup[_who].length + 1
     */
    function _lock(address _who, uint256 _value, uint256 _dateTime) onlyOwner internal
    {
        lockup[_who].push(Lock(_value, _dateTime));

        emit Locked(_who, lockup[_who].length - 1);
    }

    // destruct for only after token upgrade
    function close() onlyOwner public
    {
        selfdestruct(msg.sender);
    }
}
```