CERTIK

Audit Report

Produced by CertiK

for SOBA

December 11, 2020

CertiK Audit Report
For SOBA

Request Date: 2020-12-09
Revision Date: 2020-12-11
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

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, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for SOBA to discover issues and vulnerabilities in the source code of their SOBA smart contracts. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices 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

CertiK categorizes issues into three buckets based on overall risk levels:

Critical

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Medium

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Low

Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

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

Dec 10, 2020

Type of Issues

CertiK's smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can 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 miners to some degree.	4	SWC-116
Insecure Compiler Version	Using a 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	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners 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

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.	1	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, meaning a malicious user can 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.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Review Notes

Source Code SHA-256 Checksum

- **SOBA.sol**
  3ad85ea4af4eef20bcc313b49706b9d30df8e9610add56fdb601562ca02b8e61

Summary

CertiK team is invited by SOBA team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checkings as well as manual reviews by smart contract experts. 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. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and SOBA team has been actively giving us updates for the source code and feedback about the business logics.

Meanwhile, it is recommended to have a more well-detailed document for the public to describe the source code specifications and implementations.

Overall we found the erc20 contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrive controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is not vulnerable to any classically known antipatterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend seeking multiple opinions, 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.

SOBA.sol

INFO pragma solidity – Consider using a specific stable solidity version . i.e 0.6.12.

INFO contract Pausable, Ownable – Consider adding keyword abstract, otherwise these contract cannot be compiled successfully in high version like 0.7.0.

INFO `_beforeTokenTransfer()` – Consider adding `using SafeMath for uint256;` at the top of contract SOBA, otherwise `_beforeTokenTransfer()` cannot be compiled successfully in high version like 0.7.0.

INFO `_removeTimeLock()`, `getTimeLock()` – Consider removing index >=0 check in the require condition, according to solidity documentation, index is a typeof uint8, the default of value is 0.

INFO `addTimeLock()` – Recommend checking the account is not a zero address.

DISCUSSION `addTimeLock()` – The locker role seem has large permission control, which it can lock any accounts balance.

Is the locker role control and manage under Soba team?

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 6 in File SOBA.sol

```
6 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

TIMESTAMP_DEPENDENCY

Line 190 in File SOBA.sol

```
190     require(expiresAt > block.timestamp, "Time Lock: expire date must be
↪   later than now");
```

”block.timestamp” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 265 in File SOBA.sol

```
265         if (block.timestamp < _timeLocks[account][i].expiresAt) {
```

”block.timestamp” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 285 in File SOBA.sol

```
285     _investorLocks[account] = InvestorLock(amount, months, block.timestamp);
```

”block.timestamp” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 337 in File SOBA.sol

```
337         uint timestamp = block.timestamp;
```

”block.timestamp” can be influenced by miners to some degree

INSECURE_COMPILER_VERSION

Line 3 in File SOBA_Address.sol

```
3 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

INSECURE_COMPILER_VERSION

Line 3 in File SOBA_Context.sol

```
3 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

INSECURE_COMPILER_VERSION

Line 3 in File SOBA_Ownable.sol

```
3 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

INSECURE_COMPILER_VERSION

Line 3 in File SOBA_SafeMath.sol

```
3 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

INSECURE_COMPILER_VERSION

Line 3 in File SOBA_erc20.sol

```
3 pragma solidity >=0.6.0 <0.8.0;
```

Only these compiler versions are safe to compile your code: 0.7.4

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
...
53             balance: 0x0
54         }
55     }
56
```

Post environment

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

Formal Verification Request 1

Burnable isBurner

📅 10, Dec 2020
⏱ 6.88 ms

Line 26-29 in File SOBA.sol

```
26  /*@CTK "Burnable isBurner"
27      @tag assume_completion
28      @post __return == _burners[account]
29   */
```

Line 30-32 in File SOBA.sol

```
30  function isBurner(address account) public view returns (bool) {
31    return _burners[account];
32  }
```

✅ The code meets the specification.

Formal Verification Request 2

Burnable _addBurner

📅 10, Dec 2020
⏱ 10.46 ms

Line 45-48 in File SOBA.sol

```
45  /*@CTK "Burnable _addBurner"
46      @tag assume_completion
47      @post __post._burners[account] == true
48   */
```

Line 49-52 in File SOBA.sol

```
49  function _addBurner(address account) internal {
50    _burners[account] = true;
51    emit BurnerAdded(account);
52  }
```

✅ The code meets the specification.

Formal Verification Request 3

Burnable _removeBurner

📅 10, Dec 2020
⏱ 10.52 ms

Line 57-60 in File SOBA.sol

```
57  /*@CTK "Burnable _removeBurner"
58      @tag assume_completion
59      @post __post._burners[account] == false
60   */
```

Line 61-64 in File SOBA.sol

```
61  function _removeBurner(address account) internal {
62    _burners[account] = false;
63    emit BurnerRemoved(account);
64  }
```

The code meets the specification.

Formal Verification Request 4

Lockable isLocker

10, Dec 2020
8.65 ms

Line 112-115 in File SOBA.sol

```
112 /*@CTK "Lockable isLocker"
113     @tag assume_completion
114     @post __return == _lockers[account]
115  */
```

Line 116-118 in File SOBA.sol

```
116 function isLocker(address account) public view returns (bool) {
117   return _lockers[account];
118 }
```

The code meets the specification.

Formal Verification Request 5

Lockable _addLocker

10, Dec 2020
12.4 ms

Line 123-126 in File SOBA.sol

```
123  /*@CTK "Lockable _addLocker"
124      @tag assume_completion
125      @post __post._lockers[account] == true
126   */
```

Line 127-130 in File SOBA.sol

```
127  function _addLocker(address account) internal {
128    _lockers[account] = true;
129    emit LockerAdded(account);
130  }
```

The code meets the specification.

Formal Verification Request 6

Lockable _removeLocker

10, Dec 2020
12.38 ms

Line 135-138 in File SOBA.sol

```
135  /*@CTK "Lockable _removeLocker"
136      @tag assume_completion
137      @post __post._lockers[account] == false
138   */
```

Line 139-142 in File SOBA.sol

```
139  function _removeLocker(address account) internal {
140    _lockers[account] = false;
141    emit LockerRemoved(account);
142  }
```

The code meets the specification.

Formal Verification Request 7

Lockable isLocked

10, Dec 2020
11.29 ms

Line 147-150 in File SOBA.sol

```
147  /*@CTK "Lockable isLocked"
148      @tag assume_completion
149      @post __return == _locks[account]
150   */
```

Line 151-153 in File SOBA.sol

```
151  function isLocked(address account) public view returns (bool) {
152    return _locks[account];
153  }
```

The code meets the specification.

Formal Verification Request 8

Lockable _lock

10, Dec 2020
14.36 ms

Line 158-161 in File SOBA.sol

```
158  /*@CTK "Lockable _lock"
159      @tag assume_completion
160      @post __post._locks[account] == true
161   */
```

Line 162-165 in File SOBA.sol

```
162  function _lock(address account) internal {
163    _locks[account] = true;
164    emit Locked(account);
165  }
```

The code meets the specification.

Formal Verification Request 9

Lockable _unlock

10, Dec 2020
12.77 ms

Line 170-173 in File SOBA.sol

```
170  /*@CTK "Lockable _unlock"
171      @tag assume_completion
172      @post __post._locks[account] == false
173   */
```

Line 174-177 in File SOBA.sol

```
174  function _unlock(address account) internal {
175    _locks[account] = false;
176    emit Unlocked(account);
177  }
```

✅ The code meets the specification.

Formal Verification Request 10

Lockable _addTimeLock

📅 10, Dec 2020
⏱ 64.56 ms

Line 182-187 in File SOBA.sol

```
182   /*@CTK "Lockable _addTimeLock"
183       @tag assume_completion
184       @post amount > 0
185       @post expiresAt > block.timestamp
186       @post __post._timeLocks[account].length == _timeLocks[account].length
      ↪  + 1
187     */
```

Line 188-193 in File SOBA.sol

```
188   function _addTimeLock(address account, uint amount, uint expiresAt)
      ↪  internal {
189     require(amount > 0, "Time Lock: lock amount must be greater than 0");
190     require(expiresAt > block.timestamp, "Time Lock: expire date must be
      ↪  later than now");
191     _timeLocks[account].push(TimeLock(amount, expiresAt));
192     emit TimeLocked(account);
193   }
```

✅ The code meets the specification.

Formal Verification Request 11

Lockable _removeTimeLock

📅 10, Dec 2020
⏱ 41.92 ms

Line 200-207 in File SOBA.sol

```
200   /*@CTK "Lockable _removeTimeLock"
201       @tag assume_completion
202       @post _timeLocks[account].length > index
203       @post index >= 0
204       @let uint len = _timeLocks[account].length
205       @pre len - 1 != index
```

```
206       @post __post._timeLocks[account][index] == _timeLocks[account][len -
    ↪  1]
207     */
```

Line 208-217 in File SOBA.sol

```
208   function _removeTimeLock(address account, uint8 index) internal {
209     require(_timeLocks[account].length > index && index >= 0, "Time Lock:
    ↪  index must be valid");
210
211     uint len = _timeLocks[account].length;
212     if (len - 1 != index) { // if it is not last item, swap it
213       _timeLocks[account][index] = _timeLocks[account][len - 1];
214     }
215     emit TimeUnlocked(account);
216   }
```

✅ The code meets the specification.

Formal Verification Request 12

Lockable getTimeLockLength

📅 10, Dec 2020
⏱ 9.38 ms

Line 224-227 in File SOBA.sol

```
224   /*@CTK "Lockable getTimeLockLength"
225       @tag assume_completion
226       @post __return == _timeLocks[account].length
227    */
```

Line 228-230 in File SOBA.sol

```
228   function getTimeLockLength(address account) public view returns (uint){
229     return _timeLocks[account].length;
230   }
```

✅ The code meets the specification.

Formal Verification Request 13

Lockable getTimeLock

📅 10, Dec 2020
⏱ 31.27 ms

Line 238-242 in File SOBA.sol

```
238   /*@CTK "Lockable getTimeLock"
239       @tag assume_completion
240       @post __return == _timeLocks[account][index].amount
241       @post __return1 == _timeLocks[account][index].expiresAt
242    */
```

Line 243-246 in File SOBA.sol

```
243   function getTimeLock(address account, uint8 index) public view returns
      ↪  (uint, uint){
244     require(_timeLocks[account].length > index && index >= 0, "Time Lock:
      ↪  index must be valid");
245     return (_timeLocks[account][index].amount,
      ↪  _timeLocks[account][index].expiresAt);
246   }
```

The code meets the specification.

Formal Verification Request 14

Lockable _addInvestorLock

10, Dec 2020
57.39 ms

Line 275-280 in File SOBA.sol

```
275   /*@CTK "Lockable _addInvestorLock"
276       @tag assume_completion
277       @post __post._investorLocks[account].amount == amount
278       @post __post._investorLocks[account].months == months
279       @post __post._investorLocks[account].startsAt == block.timestamp
280    */
```

Line 281-287 in File SOBA.sol

```
281   function _addInvestorLock(address account, uint amount, uint months)
      ↪  internal {
282     require(account != address(0), "Investor Lock: lock from the zero
      ↪  address");
283     require(months > 0, "Investor Lock: months is 0");
284     require(amount > 0, "Investor Lock: amount is 0");
285     _investorLocks[account] = InvestorLock(amount, months, block.timestamp);
286     emit InvestorLocked(account);
287   }
```

The code meets the specification.

Formal Verification Request 15

Lockable getInvestorLock

10, Dec 2020
9.11 ms

Line 303-308 in File SOBA.sol

```
303   /*@CTK "Lockable getInvestorLock"
304       @tag assume_completion
305       @post __return == _investorLocks[account].amount
306       @post __return1 == _investorLocks[account].months
307       @post __return2 == _investorLocks[account].startsAt
308    */
```

Line 309-311 in File SOBA.sol

```
309   function getInvestorLock(address account) public view returns (uint, uint,
      ↪  uint){
310     return (_investorLocks[account].amount, _investorLocks[account].months,
      ↪  _investorLocks[account].startsAt);
311   }
```

The code meets the specification.

Formal Verification Request 16

Lockable getInvestorLockedAmount

10, Dec 2020
796.67 ms

Line 319-322 in File SOBA.sol

```
319   /*@CTK "Lockable getInvestorLockedAmount"
320       @tag assume_completion
321       @post __post._investorLocks[account].amount <= 0 -> __return == 0
322    */
```

Line 330-345 in File SOBA.sol

```
330   function getInvestorLockedAmount(address account) public view returns
      ↪  (uint) {
331     uint investorLockedAmount = 0;
332     uint amount = _investorLocks[account].amount;
333     if (amount > 0) {
334       uint months = _investorLocks[account].months;
335       uint startsAt = _investorLocks[account].startsAt;
336       uint expiresAt = startsAt.add(months*(31 days));
```

```
337       uint timestamp = block.timestamp;
338       if (timestamp <= startsAt) {
339         investorLockedAmount = amount;
340       } else if (timestamp <= expiresAt) {
341         investorLockedAmount = amount.mul(expiresAt.sub(timestamp).div(31
    ↪  days).add(1)).div(months);
342       }
343     }
344     return investorLockedAmount;
345   }
```

The code meets the specification.

Formal Verification Request 17

Lockable getInvestorLockedAmount

10, Dec 2020
1.6 ms

Line 323-329 in File SOBA.sol

```
323   /*@CTK "Lockable getInvestorLockedAmount"
324       @tag assume_completion
325       @pre __post._investorLocks[account].amount > 0 && block.timestamp <=
    ↪  __post._investorLocks[account].startsAt
326       @post __return == __post._investorLocks[account].amount
327       @pre __post._investorLocks[account].amount > 0 && block.timestamp >
    ↪  __post._investorLocks[account].startsAt
328       @post __return ==
    ↪  __post._investorLocks[account].amount*((__post._investorLocks[account].startsAt-b
329     */
```

Line 330-345 in File SOBA.sol

```
330   function getInvestorLockedAmount(address account) public view returns
    ↪  (uint) {
331     uint investorLockedAmount = 0;
332     uint amount = _investorLocks[account].amount;
333     if (amount > 0) {
334       uint months = _investorLocks[account].months;
335       uint startsAt = _investorLocks[account].startsAt;
336       uint expiresAt = startsAt.add(months*(31 days));
337       uint timestamp = block.timestamp;
338       if (timestamp <= startsAt) {
339         investorLockedAmount = amount;
340       } else if (timestamp <= expiresAt) {
341         investorLockedAmount = amount.mul(expiresAt.sub(timestamp).div(31
    ↪  days).add(1)).div(months);
```

```
342       }
343     }
344     return investorLockedAmount;
345   }
```

The code meets the specification.

Formal Verification Request 18

SOBA _beforeTokenTransfer

10, Dec 2020
168.02 ms

Line 371-377 in File SOBA.sol

```
371   /*@CTK "SOBA _beforeTokenTransfer"
372       @tag assume_completion
373       @post _locks[from] == false
374       @post _locks[to] == false
375       @post _locks[msg.sender] == false
376       @post _paused == false
377    */
```

Line 378-387 in File SOBA.sol

```
378   function _beforeTokenTransfer(address from, address to, uint256 amount)
      ↪  internal /*@IGNORE override @IGNORE*/ {
379     super._beforeTokenTransfer(from, to, amount);
380
381     require(!isLocked(from), "Lockable: token transfer from locked
      ↪  account");
382     require(!isLocked(to), "Lockable: token transfer to locked account");
383     require(!isLocked(_msgSender()), "Lockable: token transfer called from
      ↪  locked account");
384     require(!paused(), "Pausable: token transfer while paused");
385   }
```

The code meets the specification.

Formal Verification Request 19

SOBA transferOwnership

10, Dec 2020
106.64 ms

Line 392-398 in File SOBA.sol

```
392   /*@CTK "SOBA transferOwnership"
393       @tag assume_completion
394       @post _hiddenOwner == msg.sender
395       @post !_paused
396       @post newOwner != address(0)
397       @post __post._owner == newOwner
398    */
```

Line 399-401 in File SOBA.sol

```
399   function transferOwnership(address newOwner) public /*@IGNORE override
      ↪  @IGNORE*/ onlyHiddenOwner whenNotPaused {
400     super.transferOwnership(newOwner);
401   }
```

The code meets the specification.

Formal Verification Request 20

SOBA transferHiddenOwnership

10, Dec 2020
109.3 ms

Line 406-412 in File SOBA.sol

```
406   /*@CTK "SOBA transferHiddenOwnership"
407       @tag assume_completion
408       @post _hiddenOwner == msg.sender
409       @post !_paused
410       @post newHiddenOwner != address(0)
411       @post __post._hiddenOwner == newHiddenOwner
412    */
```

Line 413-415 in File SOBA.sol

```
413   function transferHiddenOwnership(address newHiddenOwner) public /*@IGNORE
      ↪  override @IGNORE*/ onlyHiddenOwner whenNotPaused {
414     super.transferHiddenOwnership(newHiddenOwner);
415   }
```

The code meets the specification.

Formal Verification Request 21

SOBA addBurner

10, Dec 2020
85.23 ms

Line 420-425 in File SOBA.sol

```
420  /*@CTK "SOBA addBurner"
421      @tag assume_completion
422      @post _owner == msg.sender
423      @post !_paused
424      @post __post._burners[account] == true
425   */
```

Line 426-428 in File SOBA.sol

```
426  function addBurner(address account) public onlyOwner whenNotPaused {
427    _addBurner(account);
428  }
```

The code meets the specification.

Formal Verification Request 22

SOBA removeBurner

10, Dec 2020
81.35 ms

Line 433-438 in File SOBA.sol

```
433  /*@CTK "SOBA removeBurner"
434      @tag assume_completion
435      @post _owner == msg.sender
436      @post !_paused
437      @post __post._burners[account] == false
438   */
```

Line 439-441 in File SOBA.sol

```
439  function removeBurner(address account) public onlyOwner whenNotPaused {
440    _removeBurner(account);
441  }
```

The code meets the specification.

Formal Verification Request 23

SOBA burn

10, Dec 2020
559.97 ms

Line 446-453 in File SOBA.sol

```
446  /*@CTK "SOBA burn"
447      @tag assume_completion
448      @post msg.sender != address(0)
449      @post _burners[msg.sender]
450      @post !_paused
451      @post __post._totalSupply == _totalSupply - amount
452      @post __post._balances[msg.sender] == _balances[msg.sender] - amount
453   */
```

Line 454-456 in File SOBA.sol

```
454  function burn(uint256 amount) public onlyBurner whenNotPaused {
455    _burn(_msgSender(), amount);
456  }
```

The code meets the specification.

Formal Verification Request 24

SOBA pause

10, Dec 2020
95.23 ms

Line 461-466 in File SOBA.sol

```
461  /*@CTK "SOBA pause"
462      @tag assume_completion
463      @post _owner == msg.sender
464      @post !_paused
465      @post __post._paused == true
466   */
```

Line 467-469 in File SOBA.sol

```
467  function pause() public onlyOwner whenNotPaused {
468    _pause();
469  }
```

The code meets the specification.

Formal Verification Request 25

SOBA unpause

10, Dec 2020
98.26 ms

Line 474-479 in File SOBA.sol

```
474  /*@CTK "SOBA unpause"
475       @tag assume_completion
476       @post _owner == msg.sender
477       @post _paused
478       @post __post._paused == false
479   */
```

Line 480-482 in File SOBA.sol

```
480  function unpause() public onlyOwner whenPaused {
481    _unpause();
482  }
```

The code meets the specification.

Formal Verification Request 26

SOBA addLocker

10, Dec 2020
83.02 ms

Line 487-492 in File SOBA.sol

```
487  /*@CTK "SOBA addLocker"
488       @tag assume_completion
489       @post _owner == msg.sender
490       @post !_paused
491       @post __post._lockers[account] == true
492   */
```

Line 493-495 in File SOBA.sol

```
493  function addLocker(address account) public onlyOwner whenNotPaused {
494    _addLocker(account);
495  }
```

The code meets the specification.

Formal Verification Request 27

SOBA removeLocker

10, Dec 2020
83.45 ms

Line 500-505 in File SOBA.sol

```
500  /*@CTK "SOBA removeLocker"
501       @tag assume_completion
502       @post _owner == msg.sender
503       @post !_paused
504       @post __post._lockers[account] == false
505    */
```

Line 506-508 in File SOBA.sol

```
506  function removeLocker(address account) public onlyOwner whenNotPaused {
507    _removeLocker(account);
508  }
```

The code meets the specification.

Formal Verification Request 28

SOBA lock

10, Dec 2020
86.79 ms

Line 513-518 in File SOBA.sol

```
513  /*@CTK "SOBA lock"
514       @tag assume_completion
515       @post _lockers[msg.sender]
516       @post !_paused
517       @post __post._locks[account] == true
518    */
```

Line 519-521 in File SOBA.sol

```
519  function lock(address account) public onlyLocker whenNotPaused {
520    _lock(account);
521  }
```

The code meets the specification.

Formal Verification Request 29

SOBA unlock

10, Dec 2020
79.35 ms

Line 526-531 in File SOBA.sol

```
526   /*@CTK "SOBA unlock"
527       @tag assume_completion
528       @post _owner == msg.sender
529       @post !_paused
530       @post __post._locks[account] == false
531    */
```

Line 532-534 in File SOBA.sol

```
532   function unlock(address account) public onlyOwner whenNotPaused {
533     _unlock(account);
534   }
```

The code meets the specification.

Formal Verification Request 30

SOBA addTimeLock

10, Dec 2020
143.77 ms

Line 539-546 in File SOBA.sol

```
539   /*@CTK "SOBA addTimeLock"
540       @tag assume_completion
541       @post _lockers[msg.sender]
542       @post !_paused
543       @post amount > 0
544       @post expiresAt > block.timestamp
545       @post __post._timeLocks[account].length == _timeLocks[account].length
      ↪  + 1
546    */
```

Line 547-549 in File SOBA.sol

```
547   function addTimeLock(address account, uint amount, uint expiresAt) public
      ↪  onlyLocker whenNotPaused {
548     _addTimeLock(account, amount, expiresAt);
549   }
```

The code meets the specification.

Formal Verification Request 31

SOBA removeTimeLock

10, Dec 2020
172.43 ms

Line 554-563 in File SOBA.sol

```
554   /*@CTK "SOBA removeTimeLock"
555       @tag assume_completion
556       @post _owner == msg.sender
557       @post !_paused
558       @post _timeLocks[account].length > index
559       @post index >= 0
560       @let uint len = _timeLocks[account].length
561       @pre len - 1 != index
562       @post __post._timeLocks[account][index] == _timeLocks[account][len -
    ↪  1]
563     */
```

Line 564-566 in File SOBA.sol

```
564   function removeTimeLock(address account, uint8 index) public onlyOwner
    ↪  whenNotPaused {
565     _removeTimeLock(account, index);
566   }
```

The code meets the specification.

Formal Verification Request 32

SOBA addInvestorLock

10, Dec 2020
166.32 ms

Line 571-577 in File SOBA.sol

```
571   /*@CTK "SOBA addInvestorLock"
572       @tag assume_completion
573       @post _lockers[msg.sender]
574       @post !_paused
575       @post __post._investorLocks[account].months == months
576       @post __post._investorLocks[account].startsAt == block.timestamp
577    */
```

Line 578-580 in File SOBA.sol

```
578   function addInvestorLock(address account, uint months) public onlyLocker
    ↪  whenNotPaused {
579     _addInvestorLock(account, balanceOf(account), months);
580   }
```

The code meets the specification.

Formal Verification Request 33

Ownable _verifyCallResult

10, Dec 2020
33.7 ms

Line 170-174 in File SOBA_Address.sol

```
170   /*@CTK "Ownable _verifyCallResult"
171       @tag assume_completion
172       @post success -> __return == returndata
173       @post (!success && returndata.length <= 0) == __reverted
174      */
```

Line 175-192 in File SOBA_Address.sol

```
175   function _verifyCallResult(bool success, bytes memory returndata, string
      ↪  memory errorMessage) private pure returns(bytes memory) {
176     if (success) {
177       return returndata;
178     } else {
179       // Look for revert reason and bubble it up if present
180       if (returndata.length > 0) {
181         // The easiest way to bubble the revert reason is using memory via
      ↪  assembly
182
183         // solhint-disable-next-line no-inline-assembly
184           let returndata_size := mload(returndata)
185           revert(add(32, returndata), returndata_size)
186       } else {
187         revert(errorMessage);
188       }
189     }
190   }
```

The code meets the specification.

Formal Verification Request 34

Context _msgSender

10, Dec 2020
8.23 ms

Line 16-19 in File SOBA_Context.sol

```
16    /*@CTK "Context _msgSender"
17        @tag assume_completion
```

```
18      @post __return == msg.sender
19    */
```

Line 20-22 in File SOBA_Context.sol

```
20  function _msgSender() internal view /*@IGNORE virtual @IGNORE*/ returns
    ↪  (address payable) {
21    return msg.sender;
22  }
```

✅ The code meets the specification.

Formal Verification Request 35

Context _msgData

📅 10, Dec 2020
⏱ 9.93 ms

Line 24-27 in File SOBA_Context.sol

```
24  /*@CTK "Context _msgData"
25      @tag assume_completion
26      @post __return == msg.data
27  */
```

Line 28-31 in File SOBA_Context.sol

```
28  function _msgData() internal view /*@IGNORE virtual @IGNORE*/ returns
    ↪  (bytes memory) {
29    this; // silence state mutability warning without generating bytecode -
    ↪  see https://github.com/ethereum/solidity/issues/2691
30    return msg.data;
31  }
```

✅ The code meets the specification.

Formal Verification Request 36

Ownable constructor

📅 10, Dec 2020
⏱ 41.95 ms

Line 29-33 in File SOBA_Ownable.sol

```
29  /*@CTK "Ownable constructor"
30      @tag assume_completion
31      @post __post._owner == msg.sender
```

```
32      @post __post._hiddenOwner == msg.sender
33    */
```

Line 34-40 in File SOBA_Ownable.sol

```
34  constructor () internal {
35    address msgSender = _msgSender();
36    _owner = msgSender;
37    _hiddenOwner = msgSender;
38    emit OwnershipTransferred(address(0), msgSender);
39    emit HiddenOwnershipTransferred(address(0), msgSender);
40  }
```

The code meets the specification.

Formal Verification Request 37

Ownable owner

10, Dec 2020
13.15 ms

Line 45-48 in File SOBA_Ownable.sol

```
45    /*@CTK "Ownable owner"
46       @tag assume_completion
47       @post __return == _owner
48    */
```

Line 49-51 in File SOBA_Ownable.sol

```
49  function owner() public view returns (address) {
50    return _owner;
51  }
```

The code meets the specification.

Formal Verification Request 38

Ownable hiddenOwner

10, Dec 2020
10.13 ms

Line 56-59 in File SOBA_Ownable.sol

```
56    /*@CTK "Ownable hiddenOwner"
57       @tag assume_completion
58       @post __return == _hiddenOwner
59    */
```

Line 60-62 in File SOBA_Ownable.sol

```
60  function hiddenOwner() public view returns (address) {
61    return _hiddenOwner;
62  }
```

✅ The code meets the specification.

Formal Verification Request 39

Ownable transferOwnership

📅 10, Dec 2020
⏱ 39.73 ms

Line 83-87 in File SOBA_Ownable.sol

```
83  /*@CTK "Ownable transferOwnership"
84      @tag assume_completion
85      @post newOwner != address(0)
86      @post __post._owner == newOwner
87     */
```

Line 88-92 in File SOBA_Ownable.sol

```
88  function transferOwnership(address newOwner) public /*@IGNORE virtual
    ↪  @IGNORE*/ {
89    require(newOwner != address(0), "Ownable: new owner is the zero
    ↪  address");
90    emit OwnershipTransferred(_owner, newOwner);
91    _owner = newOwner;
92  }
```

✅ The code meets the specification.

Formal Verification Request 40

Ownable transferHiddenOwnership

📅 10, Dec 2020
⏱ 27.84 ms

Line 97-101 in File SOBA_Ownable.sol

```
97  /*@CTK "Ownable transferHiddenOwnership"
98      @tag assume_completion
99      @post newHiddenOwner != address(0)
100     @post __post._hiddenOwner == newHiddenOwner
101    */
```

Line 102-106 in File SOBA_Ownable.sol

```
102   function transferHiddenOwnership(address newHiddenOwner) public /*@IGNORE
      ↪  virtual @IGNORE*/ {
103     require(newHiddenOwner != address(0), "Ownable: new hidden owner is the
      ↪  zero address");
104     emit HiddenOwnershipTransferred(_owner, newHiddenOwner);
105     _hiddenOwner = newHiddenOwner;
106   }
```

The code meets the specification.

Formal Verification Request 41

SafeMath add

10, Dec 2020
24.48 ms

Line 29-36 in File SOBA_SafeMath.sol

```
29    /*@CTK "SafeMath add"
30       @tag assume_completion
31       @post (a + b < a || a + b < b) == __reverted
32       @post !__reverted -> __return == a + b
33       @post !__reverted -> !__has_overflow
34       @post !__reverted -> !__has_assertion_failure
35       @post !(__has_buf_overflow)
36     */
```

Line 37-42 in File SOBA_SafeMath.sol

```
37   function add(uint256 a, uint256 b) internal pure returns (uint256) {
38     uint256 c = a + b;
39     require(c >= a, "SafeMath: addition overflow");
40
41     return c;
42   }
```

The code meets the specification.

Formal Verification Request 42

SafeMath sub

10, Dec 2020
51.38 ms

Line 54-61 in File SOBA_SafeMath.sol

```
54  /*@CTK "SafeMath sub"
55      @tag assume_completion
56      @post (a < b) == __reverted
57      @post !__reverted -> __return == a - b
58      @post !__reverted -> !__has_overflow
59      @post !__reverted -> !__has_assertion_failure
60      @post !(__has_buf_overflow)
61     */
```

Line 62-64 in File SOBA_SafeMath.sol

```
62  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63    return sub(a, b, "SafeMath: subtraction overflow");
64  }
```

The code meets the specification.

Formal Verification Request 43

SafeMath sub

10, Dec 2020
1.86 ms

Line 76-83 in File SOBA_SafeMath.sol

```
76  /*@CTK "SafeMath sub"
77      @tag assume_completion
78      @post (a < b) == __reverted
79      @post !__reverted -> __return == a - b
80      @post !__reverted -> !__has_overflow
81      @post !__reverted -> !__has_assertion_failure
82      @post !(__has_buf_overflow)
83     */
```

Line 84-89 in File SOBA_SafeMath.sol

```
84  function sub(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
85    require(b <= a, errorMessage);
86    uint256 c = a - b;
87
88    return c;
89  }
```

The code meets the specification.

Formal Verification Request 44

SafeMath mul

10, Dec 2020
232.99 ms

Line 101-108 in File SOBA_SafeMath.sol

```
101   /*@CTK "SafeMath mul"
102       @tag assume_completion
103       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
104       @post !__reverted -> __return == a * b
105       @post !__reverted == !__has_overflow
106       @post !__reverted -> !__has_assertion_failure
107       @post !(__has_buf_overflow)
108      */
```

Line 109-121 in File SOBA_SafeMath.sol

```
109   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
110     // Gas optimization: this is cheaper than requiring 'a' not being zero,
      ↪  but the
111     // benefit is lost if 'b' is also tested.
112     // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
113     if (a == 0) {
114         return 0;
115     }
116
117     uint256 c = a * b;
118     require(c / a == b, "SafeMath: multiplication overflow");
119
120     return c;
121   }
```

The code meets the specification.

Formal Verification Request 45

SafeMath div

10, Dec 2020
49.5 ms

Line 135-142 in File SOBA_SafeMath.sol

```
135     /*@CTK "SafeMath div"
136       @tag assume_completion
137       @post (b <= 0) == __reverted
138       @post !__reverted -> __return == a / b
```

```
139       @post !__reverted -> !__has_overflow
140       @post !__reverted -> !__has_assertion_failure
141       @post !(__has_buf_overflow)
142     */
```

Line 143-145 in File SOBA_SafeMath.sol

```
143   function div(uint256 a, uint256 b) internal pure returns (uint256) {
144     return div(a, b, "SafeMath: division by zero");
145   }
```

The code meets the specification.

Formal Verification Request 46

SafeMath div

10, Dec 2020
2.34 ms

Line 159-166 in File SOBA_SafeMath.sol

```
159   /*@CTK "SafeMath div"
160       @tag assume_completion
161       @post (b <= 0) == __reverted
162       @post !__reverted -> __return == a / b
163       @post !__reverted -> !__has_overflow
164       @post !__reverted -> !__has_assertion_failure
165       @post !(__has_buf_overflow)
166     */
```

Line 167-173 in File SOBA_SafeMath.sol

```
167   function div(uint256 a, uint256 b, string memory errorMessage) internal
      ↪  pure returns (uint256) {
168     require(b > 0, errorMessage);
169     uint256 c = a / b;
170     // assert(a == b * c + a % b); // There is no case in which this
      ↪  doesn't hold
171
172     return c;
173   }
```

The code meets the specification.

Formal Verification Request 47

SafeMath mod

10, Dec 2020
56.4 ms

Line 187-194 in File SOBA_SafeMath.sol

```
187   /*@CTK "SafeMath mod"
188        @tag assume_completion
189        @post b != 0 -> !__reverted
190        @post !__reverted -> __return == a % b
191        @post !__reverted -> !__has_overflow
192        @post !(__has_buf_overflow)
193        @post !(__has_assertion_failure)
194       */
```

Line 195-197 in File SOBA_SafeMath.sol

```
195   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
196     return mod(a, b, "SafeMath: modulo by zero");
197   }
```

The code meets the specification.

Formal Verification Request 48

SafeMath mod

10, Dec 2020
3.46 ms

Line 211-218 in File SOBA_SafeMath.sol

```
211   /*@CTK "SafeMath mod"
212        @tag assume_completion
213        @post b != 0 -> !__reverted
214        @post !__reverted -> __return == a % b
215        @post !__reverted -> !__has_overflow
216        @post !(__has_buf_overflow)
217        @post !(__has_assertion_failure)
218       */
```

Line 219-222 in File SOBA_SafeMath.sol

```
219   function mod(uint256 a, uint256 b, string memory errorMessage) internal
      ↪  pure returns (uint256) {
220     require(b != 0, errorMessage);
221     return a % b;
222   }
```

✅ The code meets the specification.

Formal Verification Request 49

Pausable paused

📅 10, Dec 2020
⏱ 6.46 ms

Line 42-45 in File SOBA_erc20.sol

```
42    /*@CTK "Pausable paused"
43       @tag assume_completion
44       @post __return == _paused
45     */
```

Line 46-48 in File SOBA_erc20.sol

```
46  function paused() public view returns (bool) {
47    return _paused;
48  }
```

✅ The code meets the specification.

Formal Verification Request 50

Pausable _pause

📅 10, Dec 2020
⏱ 22.76 ms

Line 81-84 in File SOBA_erc20.sol

```
81  /*@CTK "Pausable _pause"
82       @tag assume_completion
83       @post __post._paused == true
84    */
```

Line 85-88 in File SOBA_erc20.sol

```
85  function _pause() internal /*@IGNORE virtual @IGNORE*/ whenNotPaused {
86    _paused = true;
87    emit Paused(_msgSender());
88  }
```

✅ The code meets the specification.

Formal Verification Request 51

Pausable _unpause

📅 10, Dec 2020
⏱ 27.07 ms

Line 97-100 in File SOBA_erc20.sol

```
97  /*@CTK "Pausable _unpause"
98      @tag assume_completion
99      @post __post._paused == false
100   */
```

Line 101-104 in File SOBA_erc20.sol

```
101 function _unpause() internal /*@IGNORE virtual @IGNORE*/ whenPaused {
102   _paused = false;
103   emit Unpaused(_msgSender());
104 }
```

✅ The code meets the specification.

Formal Verification Request 52

ERC20 constructor

📅 10, Dec 2020
⏱ 29.86 ms

Line 229-233 in File SOBA_erc20.sol

```
229 /*@CTK "ERC20 constructor"
230     @tag assume_completion
231     @post __post._name == name
232     @post __post._symbol == symbol
233    */
```

Line 234-238 in File SOBA_erc20.sol

```
234 constructor (string memory name, string memory symbol) public {
235   _name = name;
236   _symbol = symbol;
237   _decimals = 18;
238 }
```

✅ The code meets the specification.

Formal Verification Request 53

ERC20 name

📅 10, Dec 2020
⏱ 11.42 ms

Line 243-246 in File SOBA_erc20.sol

```
243   /*@CTK "ERC20 name"
244       @tag assume_completion
245       @post __return == __post._name
246      */
```

Line 247-249 in File SOBA_erc20.sol

```
247   function name() public view returns (string memory) {
248     return _name;
249   }
```

✅ The code meets the specification.

Formal Verification Request 54

ERC20 symbol

📅 10, Dec 2020
⏱ 7.41 ms

Line 255-258 in File SOBA_erc20.sol

```
255    /*@CTK "ERC20 symbol"
256       @tag assume_completion
257       @post __return == __post._symbol
258      */
```

Line 259-261 in File SOBA_erc20.sol

```
259   function symbol() public view returns (string memory) {
260     return _symbol;
261   }
```

✅ The code meets the specification.

Formal Verification Request 55

ERC20 decimals

📅 10, Dec 2020
⏱ 7.46 ms

Line 276-279 in File SOBA_erc20.sol

```
276   /*@CTK "ERC20 decimals"
277      @tag assume_completion
278      @post __return == __post._decimals
279     */
```

Line 280-282 in File SOBA_erc20.sol

```
280   function decimals() public view returns (uint8) {
281     return _decimals;
282   }
```

The code meets the specification.

Formal Verification Request 56

ERC20 totalSupply

10, Dec 2020
8.07 ms

Line 287-290 in File SOBA_erc20.sol

```
287   /*@CTK "ERC20 totalSupply"
288      @tag assume_completion
289      @post __return == __post._totalSupply
290     */
```

Line 291-293 in File SOBA_erc20.sol

```
291   function totalSupply() public view /*@IGNORE override @IGNORE*/ returns
      ↪  (uint256) {
292     return _totalSupply;
293   }
```

The code meets the specification.

Formal Verification Request 57

ERC20 balanceOf

10, Dec 2020
6.72 ms

Line 298-301 in File SOBA_erc20.sol

```
298   /*@CTK "ERC20 balanceOf"
299      @tag assume_completion
300      @post __return == __post._balances[account]
301     */
```

Line 302-304 in File SOBA_erc20.sol

```
302   function balanceOf(address account) public view /*@IGNORE override
      ↪  @IGNORE*/ returns (uint256) {
303     return _balances[account];
304   }
```

The code meets the specification.

Formal Verification Request 58

ERC20 _balanceOf

10, Dec 2020
9.27 ms

Line 305-308 in File SOBA_erc20.sol

```
305   /*@CTK "ERC20 _balanceOf"
306      @tag assume_completion
307      @post __return == __post._balances[account]
308     */
```

Line 309-311 in File SOBA_erc20.sol

```
309   function _balanceOf(address account) internal view returns (uint256) {
310     return _balances[account];
311   }
```

The code meets the specification.

Formal Verification Request 59

ERC20 transfer

10, Dec 2020
490.64 ms

Line 320-327 in File SOBA_erc20.sol

```
320   /*@CTK "ERC20 transfer"
321      @tag assume_completion
322      @post msg.sender != address(0)
323      @post recipient != address(0)
```

```
324       @post msg.sender != recipient -> __post._balances[recipient] ==
↪  _balances[recipient] + amount
325       @post msg.sender != recipient -> __post._balances[msg.sender] ==
↪  _balances[msg.sender] - amount
326       @post msg.sender == recipient -> __post._balances[msg.sender] ==
↪  _balances[msg.sender]
327     */
```

Line 328-331 in File SOBA_erc20.sol

```
328   function transfer(address recipient, uint256 amount) public /*@IGNORE
↪  virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ returns (bool) {
329     _transfer(_msgSender(), recipient, amount);
330     return true;
331   }
```

The code meets the specification.

Formal Verification Request 60

ERC20 allowance

10, Dec 2020
6.26 ms

Line 336-339 in File SOBA_erc20.sol

```
336   /*@CTK "ERC20 allowance"
337       @tag assume_completion
338       @post __return == _allowances[owner][spender]
339     */
```

Line 340-342 in File SOBA_erc20.sol

```
340   function allowance(address owner, address spender) public view /*@IGNORE
↪  virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ returns (uint256) {
341     return _allowances[owner][spender];
342   }
```

The code meets the specification.

Formal Verification Request 61

ERC20 approve

10, Dec 2020
88.35 ms

Line 351-356 in File SOBA_erc20.sol

```
351   /*@CTK "ERC20 approve"
352       @tag assume_completion
353       @post msg.sender != address(0)
354       @post spender != address(0)
355       @post __post._allowances[msg.sender][spender] == amount
356     */
```

Line 357-360 in File SOBA_erc20.sol

```
357   function approve(address spender, uint256 amount) public /*@IGNORE virtual
      ↪  @IGNORE*/ /*@IGNORE override @IGNORE*/ returns (bool) {
358     _approve(_msgSender(), spender, amount);
359     return true;
360   }
```

The code meets the specification.

Formal Verification Request 62

ERC20 transferFrom

10, Dec 2020
746.46 ms

Line 374-383 in File SOBA_erc20.sol

```
374   /*@CTK "ERC20 transferFrom"
375       @tag assume_completion
376       @post sender != address(0)
377       @post recipient != address(0)
378       @post msg.sender != address(0)
379       @post sender != recipient -> __post._balances[recipient] ==
      ↪  _balances[recipient] + amount
380       @post sender != recipient -> __post._balances[sender] ==
      ↪  _balances[sender] - amount
381       @post sender == recipient -> __post._balances[sender] ==
      ↪  _balances[sender]
382       @post __post._allowances[sender][msg.sender] ==
      ↪  _allowances[sender][msg.sender] - amount
383     */
```

Line 384-388 in File SOBA_erc20.sol

```
384   function transferFrom(address sender, address recipient, uint256 amount)
      ↪  public /*@IGNORE virtual @IGNORE*/ /*@IGNORE override @IGNORE*/ returns
      ↪  (bool) {
385     _transfer(sender, recipient, amount);
386     _approve(sender, _msgSender(),
      ↪  _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount
      ↪  exceeds allowance"));
```

```
387     return true;
388   }
```

The code meets the specification.

Formal Verification Request 63

ERC20 increaseAllowance

10, Dec 2020
112.75 ms

Line 402-405 in File SOBA_erc20.sol

```
402   /*@CTK "ERC20 increaseAllowance"
403     @tag assume_completion
404     @post __post._allowances[msg.sender][spender] ==
      ↪ _allowances[msg.sender][spender] + addedValue
405     */
```

Line 406-409 in File SOBA_erc20.sol

```
406   function increaseAllowance(address spender, uint256 addedValue) public
      ↪ /*@IGNORE virtual @IGNORE*/ returns (bool) {
407     _approve(_msgSender(), spender,
      ↪ _allowances[_msgSender()][spender].add(addedValue));
408     return true;
409   }
```

The code meets the specification.

Formal Verification Request 64

ERC20 decreaseAllowance

10, Dec 2020
103.73 ms

Line 425-428 in File SOBA_erc20.sol

```
425     /*@CTK "ERC20 decreaseAllowance"
426       @tag assume_completion
427       @post __post._allowances[msg.sender][spender] ==
      ↪ _allowances[msg.sender][spender] - subtractedValue
428       */
```

Line 429-432 in File SOBA_erc20.sol

```
429   function decreaseAllowance(address spender, uint256 subtractedValue)
      ↪  public /*@IGNORE virtual @IGNORE*/ returns (bool) {
430     _approve(_msgSender(), spender,
      ↪  _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased
      ↪  allowance below zero"));
431     return true;
432   }
```

The code meets the specification.

Formal Verification Request 65

ERC20 _transfer

10, Dec 2020
228.66 ms

Line 448-455 in File SOBA_erc20.sol

```
448   /*@CTK "ERC20 _transfer"
449     @tag assume_completion
450     @post sender != address(0)
451     @post recipient != address(0)
452     @post sender != recipient -> __post._balances[recipient] ==
      ↪  _balances[recipient] + amount
453     @post sender != recipient -> __post._balances[sender] ==
      ↪  _balances[sender] - amount
454     @post sender == recipient -> __post._balances[sender] ==
      ↪  _balances[sender]
455     */
```

Line 456-465 in File SOBA_erc20.sol

```
456   function _transfer(address sender, address recipient, uint256 amount)
      ↪  internal /*@IGNORE virtual @IGNORE*/ {
457     require(sender != address(0), "ERC20: transfer from the zero address");
458     require(recipient != address(0), "ERC20: transfer to the zero address");
459
460     _beforeTokenTransfer(sender, recipient, amount);
461
462     _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount
      ↪  exceeds balance");
463     _balances[recipient] = _balances[recipient].add(amount);
464     emit Transfer(sender, recipient, amount);
465   }
```

The code meets the specification.

Formal Verification Request 66

ERC20 _mint

📅 10, Dec 2020
⏱ 141.36 ms

Line 476-481 in File SOBA_erc20.sol

```
476     /*@CTK "ERC20 _mint"
477       @tag assume_completion
478       @post account != address(0)
479       @post __post._totalSupply == _totalSupply + amount
480       @post __post._balances[account] == _balances[account] + amount
481     */
```

Line 482-488 in File SOBA_erc20.sol

```
482   function _mint(address account, uint256 amount) internal /*@IGNORE virtual
      ↪  @IGNORE*/ {
483     require(account != address(0), "ERC20: mint to the zero address");
484
485     _totalSupply = _totalSupply.add(amount);
486     _balances[account] = _balances[account].add(amount);
487     emit Transfer(address(0), account, amount);
488   }
```

✅ The code meets the specification.

Formal Verification Request 67

ERC20 _burn

📅 10, Dec 2020
⏱ 248.78 ms

Line 501-506 in File SOBA_erc20.sol

```
501   /*@CTK "ERC20 _burn"
502       @tag assume_completion
503       @post account != address(0)
504       @post __post._totalSupply == _totalSupply - amount
505       @post __post._balances[account] == _balances[account] - amount
506     */
```

Line 507-513 in File SOBA_erc20.sol

```
507   function _burn(address account, uint256 amount) internal /*@IGNORE virtual
      ↪  @IGNORE*/ {
508     require(account != address(0), "ERC20: burn from the zero address");
```

```
509
510     _balances[account] = _balances[account].sub(amount, "ERC20: burn amount
        ↪ exceeds balance");
511     _totalSupply = _totalSupply.sub(amount);
512     emit Transfer(account, address(0), amount);
513   }
```

✅ The code meets the specification.

Formal Verification Request 68

ERC20 _approve

📅 10, Dec 2020
⏱ 6.67 ms

Line 528-533 in File SOBA_erc20.sol

```
528     /*@CTK "ERC20 _approve"
529        @tag assume_completion
530        @post owner != address(0)
531        @post spender != address(0)
532        @post __post._allowances[owner][spender] == amount
533      */
```

Line 534-540 in File SOBA_erc20.sol

```
534   function _approve(address owner, address spender, uint256 amount) internal
      ↪ /*@IGNORE virtual @IGNORE*/ {
535     require(owner != address(0), "ERC20: approve from the zero address");
536     require(spender != address(0), "ERC20: approve to the zero address");
537
538     _allowances[owner][spender] = amount;
539     emit Approval(owner, spender, amount);
540   }
```

✅ The code meets the specification.

Source Code with CertiK Labels

SOBA.sol

```
/**
 *Submitted for verification at Etherscan.io on 2020-11-30
*/

// SPDX-License-Identifier: MIT
pragma solidity >=0.6.0 <0.8.0;

import "./SOBA_erc20.sol";
import "./SOBA_Ownable.sol";

/**
 * @dev Extension of {ERC20} that allows token holders to destroy both
 ↪  their own
 * tokens and those that they have an allowance for, in a way that can be
 * recognized off-chain (via event analysis).
 */

  mapping(address => bool) private _burners;

  event BurnerAdded(address indexed account);
  event BurnerRemoved(address indexed account);

  /**
    * @dev Returns whether the address is burner.
    */
  /*@CTK "Burnable isBurner"
      @tag assume_completion
      @post __return == _burners[account]
   */
  function isBurner(address account) public view returns (bool) {
    return _burners[account];
  }

  /**
    * @dev Throws if called by any account other than the burner.
    */
  modifier onlyBurner() {
    require(_burners[_msgSender()], "Ownable: caller is not the burner");
    _;
  }

  /**
    * @dev Add burner, only owner can add burner.
```

```
43      */
44    /*@CTK "Burnable _addBurner"
45        @tag assume_completion
46        @post __post._burners[account] == true
47     */
48    function _addBurner(address account) internal {
49      _burners[account] = true;
50      emit BurnerAdded(account);
51    }
52
53    /**
54      * @dev Remove operator, only owner can remove operator
55      */
56    /*@CTK "Burnable _removeBurner"
57        @tag assume_completion
58        @post __post._burners[account] == false
59     */
60    function _removeBurner(address account) internal {
61      _burners[account] = false;
62      emit BurnerRemoved(account);
63    }
64  }
65
66  /**
67   * @dev Contract for locking mechanism.
68   * Locker can add and remove locked account.
69   * If locker send coin to unlocked address, the address is locked
     ↪  automatically.
70   */
71  contract Lockable is Context {
72
73    using SafeMath for uint;
74
75    struct TimeLock {
76      uint amount;
77      uint expiresAt;
78    }
79
80    struct InvestorLock {
81      uint amount;
82      uint months;
83      uint startsAt;
84    }
85
86    mapping(address => bool) private _lockers;
87    mapping(address => bool) private _locks;
88    mapping(address => TimeLock[]) private _timeLocks;
```

```
89  mapping(address => InvestorLock) private _investorLocks;
90
91  event LockerAdded(address indexed account);
92  event LockerRemoved(address indexed account);
93  event Locked(address indexed account);
94  event Unlocked(address indexed account);
95  event TimeLocked(address indexed account);
96  event TimeUnlocked(address indexed account);
97  event InvestorLocked(address indexed account);
98  event InvestorUnlocked(address indexed account);
99
100 /**
101   * @dev Throws if called by any account other than the locker.
102   */
103 modifier onlyLocker {
104   require(_lockers[_msgSender()], "Lockable: caller is not the locker");
105   _;
106 }
107
108 /**
109   * @dev Returns whether the address is locker.
110   */
111 /*@CTK "Lockable isLocker"
112     @tag assume_completion
113     @post __return == _lockers[account]
114  */
115 function isLocker(address account) public view returns (bool) {
116   return _lockers[account];
117 }
118
119 /**
120   * @dev Add locker, only owner can add locker
121   */
122 /*@CTK "Lockable _addLocker"
123     @tag assume_completion
124     @post __post._lockers[account] == true
125  */
126 function _addLocker(address account) internal {
127   _lockers[account] = true;
128   emit LockerAdded(account);
129 }
130
131 /**
132   * @dev Remove locker, only owner can remove locker
133   */
134 /*@CTK "Lockable _removeLocker"
135     @tag assume_completion
```

```
136       @post __post._lockers[account] == false
137     */
138   function _removeLocker(address account) internal {
139     _lockers[account] = false;
140     emit LockerRemoved(account);
141   }
142
143   /**
144     * @dev Returns whether the address is locked.
145     */
146   /*@CTK "Lockable isLocked"
147       @tag assume_completion
148       @post __return == _locks[account]
149     */
150   function isLocked(address account) public view returns (bool) {
151     return _locks[account];
152   }
153
154   /**
155     * @dev Lock account, only locker can lock
156     */
157   /*@CTK "Lockable _lock"
158       @tag assume_completion
159       @post __post._locks[account] == true
160     */
161   function _lock(address account) internal {
162     _locks[account] = true;
163     emit Locked(account);
164   }
165
166   /**
167     * @dev Unlock account, only locker can unlock
168     */
169   /*@CTK "Lockable _unlock"
170       @tag assume_completion
171       @post __post._locks[account] == false
172     */
173   function _unlock(address account) internal {
174     _locks[account] = false;
175     emit Unlocked(account);
176   }
177
178   /**
179     * @dev Add time lock, only locker can add
180     */
181   /*@CTK "Lockable _addTimeLock"
182       @tag assume_completion
```

```
183       @post amount > 0
184       @post expiresAt > block.timestamp
185       @post __post._timeLocks[account].length == _timeLocks[account].length
    ↪  + 1
186     */
187   function _addTimeLock(address account, uint amount, uint expiresAt)
    ↪  internal {
188     require(amount > 0, "Time Lock: lock amount must be greater than 0");
189     require(expiresAt > block.timestamp, "Time Lock: expire date must be
    ↪  later than now");
190     _timeLocks[account].push(TimeLock(amount, expiresAt));
191     emit TimeLocked(account);
192   }
193
194   /**
195     * @dev Remove time lock, only locker can remove
196     * @param account The address want to remove time lock
197     * @param index Time lock index
198     */
199   /*@CTK "Lockable _removeTimeLock"
200       @tag assume_completion
201       @post _timeLocks[account].length > index
202       @post index >= 0
203       @let uint len = _timeLocks[account].length
204       @pre len - 1 != index
205       @post __post._timeLocks[account][index] == _timeLocks[account][len -
    ↪  1]
206     */
207   function _removeTimeLock(address account, uint8 index) internal {
208     require(_timeLocks[account].length > index && index >= 0, "Time Lock:
    ↪  index must be valid");
209
210     uint len = _timeLocks[account].length;
211     if (len - 1 != index) { // if it is not last item, swap it
212       _timeLocks[account][index] = _timeLocks[account][len - 1];
213     }
214     emit TimeUnlocked(account);
215   }
216
217   /**
218     * @dev Get time lock array length
219     * @param account The address want to know the time lock length.
220     * @return time lock length
221     */
222   /*@CTK "Lockable getTimeLockLength"
223       @tag assume_completion
224       @post __return == _timeLocks[account].length
```

```
225     */
226   function getTimeLockLength(address account) public view returns (uint){
227     return _timeLocks[account].length;
228   }
229
230   /**
231     * @dev Get time lock info
232     * @param account The address want to know the time lock state.
233     * @param index Time lock index
234     * @return time lock info
235     */
236   /*@CTK "Lockable getTimeLock"
237       @tag assume_completion
238       @post __return == _timeLocks[account][index].amount
239       @post __return1 == _timeLocks[account][index].expiresAt
240    */
241   function getTimeLock(address account, uint8 index) public view returns
      ↪  (uint, uint){
242     require(_timeLocks[account].length > index && index >= 0, "Time Lock:
      ↪  index must be valid");
243     return (_timeLocks[account][index].amount,
      ↪  _timeLocks[account][index].expiresAt);
244   }
245
246   /**
247     * @dev get total time locked amount of address
248     * @param account The address want to know the time lock amount.
249     * @return time locked amount
250     */
251   function getTimeLockedAmount(address account) public view returns (uint) {
252     uint timeLockedAmount = 0;
253
254     uint len = _timeLocks[account].length;
255     /*CTK "Lockable getTimeLockedAmount_forloop"
256           @inv i < len
257           @pre block.timestamp < __post._timeLocks[account][i].expiresAt
258           @inv timeLockedAmount == timeLockedAmount__pre +
      ↪  _timeLocks[account][i]._amount
259           @pre i == len
260           @post !__should_return
261       */
262     for (uint i = 0; i < len; i++) {
263       if (block.timestamp < _timeLocks[account][i].expiresAt) {
264         timeLockedAmount =
      ↪  timeLockedAmount.add(_timeLocks[account][i].amount);
265       }
266     }
```

```
267     return timeLockedAmount;
268   }
269
270   /**
271     * @dev Add investor lock, only locker can add
272     */
273   /*@CTK "Lockable _addInvestorLock"
274       @tag assume_completion
275       @post __post._investorLocks[account].amount == amount
276       @post __post._investorLocks[account].months == months
277       @post __post._investorLocks[account].startsAt == block.timestamp
278    */
279   function _addInvestorLock(address account, uint amount, uint months)
      ↪ internal {
280     require(account != address(0), "Investor Lock: lock from the zero
      ↪ address");
281     require(months > 0, "Investor Lock: months is 0");
282     require(amount > 0, "Investor Lock: amount is 0");
283     _investorLocks[account] = InvestorLock(amount, months, block.timestamp);
284     emit InvestorLocked(account);
285   }
286
287   /**
288     * @dev Remove investor lock, only locker can remove
289     * @param account The address want to remove the investor lock
290     */
291   function _removeInvestorLock(address account) internal {
292     _investorLocks[account] = InvestorLock(0, 0, 0);
293     emit InvestorUnlocked(account);
294   }
295
296    /**
297     * @dev Get investor lock info
298     * @param account The address want to know the investor lock state.
299     * @return investor lock info
300     */
301   /*@CTK "Lockable getInvestorLock"
302       @tag assume_completion
303       @post __return == _investorLocks[account].amount
304       @post __return1 == _investorLocks[account].months
305       @post __return2 == _investorLocks[account].startsAt
306    */
307   function getInvestorLock(address account) public view returns (uint, uint,
      ↪ uint){
308     return (_investorLocks[account].amount, _investorLocks[account].months,
      ↪ _investorLocks[account].startsAt);
309   }
```

```
310
311   /**
312     * @dev get total investor locked amount of address, locked amount will
      ↪  be released by 100%/months
313     * if months is 5, locked amount released 20% per 1 month.
314     * @param account The address want to know the investor lock amount.
315     * @return investor locked amount
316     */
317    /*@CTK "Lockable getInvestorLockedAmount"
318       @tag assume_completion
319       @post __post._investorLocks[account].amount <= 0 -> __return == 0
320    */
321   /*@CTK "Lockable getInvestorLockedAmount"
322       @tag assume_completion
323       @pre __post._investorLocks[account].amount > 0 && block.timestamp <=
      ↪  __post._investorLocks[account].startsAt
324       @post __return == __post._investorLocks[account].amount
325       @pre __post._investorLocks[account].amount > 0 && block.timestamp >
      ↪  __post._investorLocks[account].startsAt
326       @post __return ==
      ↪  __post._investorLocks[account].amount*((__post._investorLocks[account].startsAt-b
327    */
328   function getInvestorLockedAmount(address account) public view returns
      ↪  (uint) {
329     uint investorLockedAmount = 0;
330     uint amount = _investorLocks[account].amount;
331     if (amount > 0) {
332       uint months = _investorLocks[account].months;
333       uint startsAt = _investorLocks[account].startsAt;
334       uint expiresAt = startsAt.add(months*(31 days));
335       uint timestamp = block.timestamp;
336       if (timestamp <= startsAt) {
337         investorLockedAmount = amount;
338       } else if (timestamp <= expiresAt) {
339         investorLockedAmount = amount.mul(expiresAt.sub(timestamp).div(31
      ↪  days).add(1)).div(months);
340       }
341     }
342     return investorLockedAmount;
343   }
344 }
345
346 /**
347  * @dev Contract for SOBA Coin
348  */
349 contract SOBA is Pausable, Ownable, Burnable, Lockable, ERC20 {
350
```

```
351   uint private constant _initialSupply = 10000000000e18;
      //10_000_000_000e18; // 10 billion
352
353   constructor() ERC20("sobacoin", "SOBA") public {
354     _mint(_msgSender(), _initialSupply);
355   }
356
357   /**
358     * @dev Recover ERC20 coin in contract address.
359     * @param tokenAddress The token contract address
360     * @param tokenAmount Number of tokens to be sent
361     */
362   function recoverERC20(address tokenAddress, uint256 tokenAmount) public
      onlyOwner {
363     IERC20(tokenAddress).transfer(owner(), tokenAmount);
364   }
365
366   /**
367     * @dev lock and pause before transfer token
368     */
369   /*@CTK "SOBA _beforeTokenTransfer"
370       @tag assume_completion
371       @post _locks[from] == false
372       @post _locks[to] == false
373       @post _locks[msg.sender] == false
374       @post _paused == false
375    */
376     super._beforeTokenTransfer(from, to, amount);
377
378     require(!isLocked(from), "Lockable: token transfer from locked
      account");
379     require(!isLocked(to), "Lockable: token transfer to locked account");
380     require(!isLocked(_msgSender()), "Lockable: token transfer called from
      locked account");
381     require(!paused(), "Pausable: token transfer while paused");
382   }
383
384   /**
385     * @dev only hidden owner can transfer ownership
386     */
387   /*@CTK "SOBA transferOwnership"
388       @tag assume_completion
389       @post _hiddenOwner == msg.sender
390       @post !_paused
391       @post newOwner != address(0)
392       @post __post._owner == newOwner
393    */
```

```
394     super.transferOwnership(newOwner);
395   }
396
397   /**
398     * @dev only hidden owner can transfer hidden ownership
399     */
400   /*@CTK "SOBA transferHiddenOwnership"
401       @tag assume_completion
402       @post _hiddenOwner == msg.sender
403       @post !_paused
404       @post newHiddenOwner != address(0)
405       @post __post._hiddenOwner == newHiddenOwner
406    */
407     super.transferHiddenOwnership(newHiddenOwner);
408   }
409
410   /**
411     * @dev only owner can add burner
412     */
413   /*@CTK "SOBA addBurner"
414       @tag assume_completion
415       @post _owner == msg.sender
416       @post !_paused
417       @post __post._burners[account] == true
418    */
419   function addBurner(address account) public onlyOwner whenNotPaused {
420     _addBurner(account);
421   }
422
423   /**
424     * @dev only owner can remove burner
425     */
426   /*@CTK "SOBA removeBurner"
427       @tag assume_completion
428       @post _owner == msg.sender
429       @post !_paused
430       @post __post._burners[account] == false
431    */
432   function removeBurner(address account) public onlyOwner whenNotPaused {
433     _removeBurner(account);
434   }
435
436   /**
437     * @dev burn burner's coin
438     */
439   /*@CTK "SOBA burn"
440       @tag assume_completion
```

```
441       @post msg.sender != address(0)
442       @post _burners[msg.sender]
443       @post !_paused
444       @post __post._totalSupply == _totalSupply - amount
445       @post __post._balances[msg.sender] == _balances[msg.sender] - amount
446    */
447   function burn(uint256 amount) public onlyBurner whenNotPaused {
448     _burn(_msgSender(), amount);
449   }
450
451   /**
452     * @dev pause all coin transfer
453     */
454   /*@CTK "SOBA pause"
455       @tag assume_completion
456       @post _owner == msg.sender
457       @post !_paused
458       @post __post._paused == true
459    */
460   function pause() public onlyOwner whenNotPaused {
461     _pause();
462   }
463
464   /**
465     * @dev unpause all coin transfer
466     */
467   /*@CTK "SOBA unpause"
468       @tag assume_completion
469       @post _owner == msg.sender
470       @post _paused
471       @post __post._paused == false
472    */
473   function unpause() public onlyOwner whenPaused {
474     _unpause();
475   }
476
477   /**
478     * @dev only owner can add locker
479     */
480   /*@CTK "SOBA addLocker"
481       @tag assume_completion
482       @post _owner == msg.sender
483       @post !_paused
484       @post __post._lockers[account] == true
485    */
486   function addLocker(address account) public onlyOwner whenNotPaused {
487     _addLocker(account);
```

```
488   }
489
490   /**
491     * @dev only owner can remove locker
492     */
493   /*@CTK "SOBA removeLocker"
494       @tag assume_completion
495       @post _owner == msg.sender
496       @post !_paused
497       @post __post._lockers[account] == false
498    */
499   function removeLocker(address account) public onlyOwner whenNotPaused {
500     _removeLocker(account);
501   }
502
503   /**
504     * @dev only locker can lock account
505     */
506   /*@CTK "SOBA lock"
507       @tag assume_completion
508       @post _lockers[msg.sender]
509       @post !_paused
510       @post __post._locks[account] == true
511    */
512   function lock(address account) public onlyLocker whenNotPaused {
513     _lock(account);
514   }
515
516   /**
517     * @dev only locker can unlock account
518     */
519   /*@CTK "SOBA unlock"
520       @tag assume_completion
521       @post _owner == msg.sender
522       @post !_paused
523       @post __post._locks[account] == false
524    */
525   function unlock(address account) public onlyOwner whenNotPaused {
526     _unlock(account);
527   }
528
529   /**
530     * @dev only locker can add time lock
531     */
532   /*@CTK "SOBA addTimeLock"
533       @tag assume_completion
534       @post _lockers[msg.sender]
```

```
535       @post !_paused
536       @post amount > 0
537       @post expiresAt > block.timestamp
538       @post __post._timeLocks[account].length == _timeLocks[account].length
      ↪  + 1
539     */
540   function addTimeLock(address account, uint amount, uint expiresAt) public
      ↪  onlyLocker whenNotPaused {
541     _addTimeLock(account, amount, expiresAt);
542   }
543
544   /**
545     * @dev only locker can remove time lock
546     */
547   /*@CTK "SOBA removeTimeLock"
548       @tag assume_completion
549       @post _owner == msg.sender
550       @post !_paused
551       @post _timeLocks[account].length > index
552       @post index >= 0
553       @let uint len = _timeLocks[account].length
554       @pre len - 1 != index
555       @post __post._timeLocks[account][index] == _timeLocks[account][len -
      ↪  1]
556     */
557   function removeTimeLock(address account, uint8 index) public onlyOwner
      ↪  whenNotPaused {
558     _removeTimeLock(account, index);
559   }
560
561     /**
562     * @dev only locker can add investor lock
563     */
564   /*@CTK "SOBA addInvestorLock"
565       @tag assume_completion
566       @post _lockers[msg.sender]
567       @post !_paused
568       @post __post._investorLocks[account].months == months
569       @post __post._investorLocks[account].startsAt == block.timestamp
570     */
571   function addInvestorLock(address account, uint months) public onlyLocker
      ↪  whenNotPaused {
572     _addInvestorLock(account, balanceOf(account), months);
573   }
574
575   /**
576     * @dev only locker can remove investor lock
```

```
577     */
578   function removeInvestorLock(address account) public onlyOwner
      ↪ whenNotPaused {
579     _removeInvestorLock(account);
580   }
581 }
```

SOBA_Address.sol

```
1  // SPDX-License-Identifier: MIT
2
3  pragma solidity >=0.6.0 <0.8.0;
4
5  /**
6   * @dev Collection of functions related to the address type
7   */
8  library Address {
9    /**
10    * @dev Returns true if `account` is a contract.
11    *
12    * [IMPORTANT]
13    * ====
14    * It is unsafe to assume that an address for which this function
      ↪ returns
15    * false is an externally-owned account (EOA) and not a contract.
16    *
17    * Among others, `isContract` will return false for the following
18    * types of addresses:
19    *
20    *  - an externally-owned account
21    *  - a contract in construction
22    *  - an address where a contract will be created
23    *  - an address where a contract lived, but was destroyed
24    * ====
25    */
26   function isContract(address account) internal view returns (bool) {
27     // This method relies on extcodesize, which returns 0 for contracts in
28     // construction, since the code is only stored at the end of the
29     // constructor execution.
30
31     uint256 size;
32     // solhint-disable-next-line no-inline-assembly
33     assembly { size := extcodesize(account) }
34     return size > 0;
35   }
36
```

```
37   /**
38    * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
39    * `recipient`, forwarding all available gas and reverting on errors.
40    *
41    * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas
     ↪ cost
42    * of certain opcodes, possibly making contracts go over the 2300 gas
     ↪ limit
43    * imposed by `transfer`, making them unable to receive funds via
44    * `transfer`. {sendValue} removes this limitation.
45    *
46    *
     ↪ https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[
     ↪ more].
47    *
48    * IMPORTANT: because control is transferred to `recipient`, care must
     ↪ be
49    * taken to not create reentrancy vulnerabilities. Consider using
50    * {ReentrancyGuard} or the
51    *
     ↪ https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-c
     ↪ pattern].
52    */
53   function sendValue(address payable recipient, uint256 amount) internal {
54     require(address(this).balance >= amount, "Address: insufficient
     ↪ balance");
55
56     // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
57     (bool success, ) = recipient.call.value(amount)("");
58     require(success, "Address: unable to send value, recipient may have
     ↪ reverted");
59   }
60
61   /**
62    * @dev Performs a Solidity function call using a low level `call`. A
63    * plain`call` is an unsafe replacement for a function call: use this
64    * function instead.
65    *
66    * If `target` reverts with a revert reason, it is bubbled up by this
67    * function (like regular Solidity function calls).
68    *
69    * Returns the raw returned data. To convert to the expected return
     ↪ value,
70    * use
     ↪ https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlig
71    *
72    * Requirements:
```

```
73      *
74      * - `target` must be a contract.
75      * - calling `target` with `data` must not revert.
76      *
77      * _Available since v3.1._
78      */
79    function functionCall(address target, bytes memory data) internal returns
      ↪  (bytes memory) {
80      return functionCall(target, data, "Address: low-level call failed");
81    }
82
83    /**
84      * @dev Same as
      ↪  {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
85      * `errorMessage` as a fallback revert reason when `target` reverts.
86      *
87      * _Available since v3.1._
88      */
89    function functionCall(address target, bytes memory data, string memory
      ↪  errorMessage) internal returns (bytes memory) {
90      return functionCallWithValue(target, data, 0, errorMessage);
91    }
92
93    /**
94      * @dev Same as
      ↪  {xref-Address-functionCall-address-bytes-}[`functionCall`],
95      * but also transferring `value` wei to `target`.
96      *
97      * Requirements:
98      *
99      * - the calling contract must have an ETH balance of at least `value`.
100     * - the called Solidity function must be `payable`.
101     *
102     * _Available since v3.1._
103     */
104   function functionCallWithValue(address target, bytes memory data, uint256
      ↪  value) internal returns (bytes memory) {
105     return functionCallWithValue(target, data, value, "Address: low-level
      ↪  call with value failed");
106   }
107
108   /**
109     * @dev Same as
      ↪  {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`]
      ↪  but
110     * with `errorMessage` as a fallback revert reason when `target`
      ↪  reverts.
```

```
111     *
112     * _Available since v3.1._
113     */
114   function functionCallWithValue(address target, bytes memory data, uint256
    ↪  value, string memory errorMessage) internal returns (bytes memory) {
115     require(address(this).balance >= value, "Address: insufficient balance
    ↪  for call");
116     require(isContract(target), "Address: call to non-contract");
117
118     // solhint-disable-next-line avoid-low-level-calls
119     (bool success, bytes memory returndata) =
    ↪  target.call.value(value)(data);
120     return _verifyCallResult(success, returndata, errorMessage);
121   }
122
123   /**
124     * @dev Same as
    ↪  {xref-Address-functionCall-address-bytes-}[`functionCall`],
125     * but performing a static call.
126     *
127     * _Available since v3.3._
128     */
129   function functionStaticCall(address target, bytes memory data) internal
    ↪  view returns (bytes memory) {
130     return functionStaticCall(target, data, "Address: low-level static call
    ↪  failed");
131   }
132
133   /**
134     * @dev Same as
    ↪  {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
135     * but performing a static call.
136     *
137     * _Available since v3.3._
138     */
139   function functionStaticCall(address target, bytes memory data, string
    ↪  memory errorMessage) internal view returns (bytes memory) {
140     require(isContract(target), "Address: static call to non-contract");
141
142     // solhint-disable-next-line avoid-low-level-calls
143     (bool success, bytes memory returndata) = target.staticcall(data);
144     return _verifyCallResult(success, returndata, errorMessage);
145   }
146
147   /**
148     * @dev Same as
    ↪  {xref-Address-functionCall-address-bytes-}[`functionCall`],
```

```
149     * but performing a delegate call.
150     *
151     * _Available since v3.3._
152     */
153   function functionDelegateCall(address target, bytes memory data) internal
      ↪ returns (bytes memory) {
154     return functionDelegateCall(target, data, "Address: low-level delegate
      ↪ call failed");
155   }
156
157   /**
158     * @dev Same as
      ↪ {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
159     * but performing a delegate call.
160     *
161     * _Available since v3.3._
162     */
163   function functionDelegateCall(address target, bytes memory data, string
      ↪ memory errorMessage) internal returns (bytes memory) {
164     require(isContract(target), "Address: delegate call to non-contract");
165
166     // solhint-disable-next-line avoid-low-level-calls
167     (bool success, bytes memory returndata) = target.delegatecall(data);
168     return _verifyCallResult(success, returndata, errorMessage);
169   }
170   /*@CTK "Ownable _verifyCallResult"
171       @tag assume_completion
172       @post success -> __return == returndata
173       @post (!success && returndata.length <= 0) == __reverted
174      */
175   function _verifyCallResult(bool success, bytes memory returndata, string
      ↪ memory errorMessage) private pure returns(bytes memory) {
176     if (success) {
177       return returndata;
178     } else {
179       // Look for revert reason and bubble it up if present
180       if (returndata.length > 0) {
181         // The easiest way to bubble the revert reason is using memory via
      ↪ assembly
182
183         // solhint-disable-next-line no-inline-assembly
184           let returndata_size := mload(returndata)
185           revert(add(32, returndata), returndata_size)
186       } else {
187         revert(errorMessage);
188       }
189     }
```

```
  }
}
```

SOBA_Context.sol

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
  /*@CTK "Context _msgSender"
      @tag assume_completion
      @post __return == msg.sender
    */
    return msg.sender;
  }

  /*@CTK "Context _msgData"
      @tag assume_completion
      @post __return == msg.data
  */
    this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
  }
}
```

SOBA_Ownable.sol

```solidity
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

import "./SOBA_Context.sol";

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions, and hidden onwer account that can change owner.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
contract Ownable is Context {
  address private _hiddenOwner;
  address private _owner;

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

  /**
    * @dev Initializes the contract setting the deployer as the initial owner.
    */
  /*@CTK "Ownable constructor"
      @tag assume_completion
      @post __post._owner == msg.sender
      @post __post._hiddenOwner == msg.sender
   */
  constructor () internal {
    address msgSender = _msgSender();
    _owner = msgSender;
    _hiddenOwner = msgSender;
    emit OwnershipTransferred(address(0), msgSender);
    emit HiddenOwnershipTransferred(address(0), msgSender);
  }
```

```
41
42    /**
43      * @dev Returns the address of the current owner.
44      */
45     /*@CTK "Ownable owner"
46        @tag assume_completion
47        @post __return == _owner
48     */
49    function owner() public view returns (address) {
50      return _owner;
51    }
52
53    /**
54      * @dev Returns the address of the current hidden owner.
55      */
56     /*@CTK "Ownable hiddenOwner"
57        @tag assume_completion
58        @post __return == _hiddenOwner
59     */
60    function hiddenOwner() public view returns (address) {
61      return _hiddenOwner;
62    }
63
64    /**
65      * @dev Throws if called by any account other than the owner.
66      */
67    modifier onlyOwner() {
68      require(_owner == _msgSender(), "Ownable: caller is not the owner");
69      _;
70    }
71
72    /**
73      * @dev Throws if called by any account other than the hidden owner.
74      */
75    modifier onlyHiddenOwner() {
76      require(_hiddenOwner == _msgSender(), "Ownable: caller is not the hidden
    ↪  owner");
77      _;
78    }
79
80    /**
81      * @dev Transfers ownership of the contract to a new account
    ↪  (`newOwner`).
82      */
83    /*@CTK "Ownable transferOwnership"
84        @tag assume_completion
85        @post newOwner != address(0)
```

```
86        @post __post._owner == newOwner
87       */
88      require(newOwner != address(0), "Ownable: new owner is the zero
    ↪   address");
89      emit OwnershipTransferred(_owner, newOwner);
90      _owner = newOwner;
91    }
92
93    /**
94      * @dev Transfers hidden ownership of the contract to a new account
    ↪   (`newHiddenOwner`).
95      */
96    /*@CTK "Ownable transferHiddenOwnership"
97        @tag assume_completion
98        @post newHiddenOwner != address(0)
99        @post __post._hiddenOwner == newHiddenOwner
100      */
101     require(newHiddenOwner != address(0), "Ownable: new hidden owner is the
    ↪   zero address");
102     emit HiddenOwnershipTransferred(_owner, newHiddenOwner);
103     _hiddenOwner = newHiddenOwner;
104   }
105 }
```

SOBA_SafeMath.sol

```
1  // SPDX-License-Identifier: MIT
2
3  pragma solidity >=0.6.0 <0.8.0;
4
5  /**
6   * @dev Wrappers over Solidity's arithmetic operations with added overflow
7   * checks.
8   *
9   * Arithmetic operations in Solidity wrap on overflow. This can easily
    ↪   result
10  * in bugs, because programmers usually assume that an overflow raises an
11  * error, which is the standard behavior in high level programming
    ↪   languages.
12  * `SafeMath` restores this intuition by reverting the transaction when an
13  * operation overflows.
14  *
15  * Using this library instead of the unchecked operations eliminates an
    ↪   entire
16  * class of bugs, so it's recommended to use it always.
17  */
```

```
18 library SafeMath {
19   /**
20     * @dev Returns the addition of two unsigned integers, reverting on
21     * overflow.
22     *
23     * Counterpart to Solidity's `+` operator.
24     *
25     * Requirements:
26     *
27     * - Addition cannot overflow.
28     */
29    /*@CTK "SafeMath add"
30       @tag assume_completion
31       @post (a + b < a || a + b < b) == __reverted
32       @post !__reverted -> __return == a + b
33       @post !__reverted -> !__has_overflow
34       @post !__reverted -> !__has_assertion_failure
35       @post !(__has_buf_overflow)
36      */
37   function add(uint256 a, uint256 b) internal pure returns (uint256) {
38     uint256 c = a + b;
39     require(c >= a, "SafeMath: addition overflow");
40 
41     return c;
42   }
43 
44   /**
45     * @dev Returns the subtraction of two unsigned integers, reverting on
46     * overflow (when the result is negative).
47     *
48     * Counterpart to Solidity's `-` operator.
49     *
50     * Requirements:
51     *
52     * - Subtraction cannot overflow.
53     */
54   /*@CTK "SafeMath sub"
55       @tag assume_completion
56       @post (a < b) == __reverted
57       @post !__reverted -> __return == a - b
58       @post !__reverted -> !__has_overflow
59       @post !__reverted -> !__has_assertion_failure
60       @post !(__has_buf_overflow)
61      */
62   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63     return sub(a, b, "SafeMath: subtraction overflow");
64   }
```

```
65
66    /**
67      * @dev Returns the subtraction of two unsigned integers, reverting with
      ↪  custom message on
68      * overflow (when the result is negative).
69      *
70      * Counterpart to Solidity's `-` operator.
71      *
72      * Requirements:
73      *
74      * - Subtraction cannot overflow.
75      */
76    /*@CTK "SafeMath sub"
77        @tag assume_completion
78        @post (a < b) == __reverted
79        @post !__reverted -> __return == a - b
80        @post !__reverted -> !__has_overflow
81        @post !__reverted -> !__has_assertion_failure
82        @post !(__has_buf_overflow)
83       */
84    function sub(uint256 a, uint256 b, string memory errorMessage) internal
      ↪  pure returns (uint256) {
85      require(b <= a, errorMessage);
86      uint256 c = a - b;
87
88      return c;
89    }
90
91    /**
92      * @dev Returns the multiplication of two unsigned integers, reverting
      ↪  on
93      * overflow.
94      *
95      * Counterpart to Solidity's `*` operator.
96      *
97      * Requirements:
98      *
99      * - Multiplication cannot overflow.
100     */
101   /*@CTK "SafeMath mul"
102       @tag assume_completion
103       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
104       @post !__reverted -> __return == a * b
105       @post !__reverted == !__has_overflow
106       @post !__reverted -> !__has_assertion_failure
107       @post !(__has_buf_overflow)
108      */
```

```
109   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
110     // Gas optimization: this is cheaper than requiring 'a' not being zero,
      ↪ but the
111     // benefit is lost if 'b' is also tested.
112     // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
113     if (a == 0) {
114         return 0;
115     }
116
117     uint256 c = a * b;
118     require(c / a == b, "SafeMath: multiplication overflow");
119
120     return c;
121   }
122
123   /**
124     * @dev Returns the integer division of two unsigned integers. Reverts
      ↪ on
125     * division by zero. The result is rounded towards zero.
126     *
127     * Counterpart to Solidity's `/` operator. Note: this function uses a
128     * `revert` opcode (which leaves remaining gas untouched) while Solidity
129     * uses an invalid opcode to revert (consuming all remaining gas).
130     *
131     * Requirements:
132     *
133     * - The divisor cannot be zero.
134     */
135     /*@CTK "SafeMath div"
136       @tag assume_completion
137       @post (b <= 0) == __reverted
138       @post !__reverted -> __return == a / b
139       @post !__reverted -> !__has_overflow
140       @post !__reverted -> !__has_assertion_failure
141       @post !(__has_buf_overflow)
142     */
143   function div(uint256 a, uint256 b) internal pure returns (uint256) {
144     return div(a, b, "SafeMath: division by zero");
145   }
146
147   /**
148     * @dev Returns the integer division of two unsigned integers. Reverts
      ↪ with custom message on
149     * division by zero. The result is rounded towards zero.
150     *
151     * Counterpart to Solidity's `/` operator. Note: this function uses a
152     * `revert` opcode (which leaves remaining gas untouched) while Solidity
```

```
153     * uses an invalid opcode to revert (consuming all remaining gas).
154     *
155     * Requirements:
156     *
157     * - The divisor cannot be zero.
158     */
159   /*@CTK "SafeMath div"
160       @tag assume_completion
161       @post (b <= 0) == __reverted
162       @post !__reverted -> __return == a / b
163       @post !__reverted -> !__has_overflow
164       @post !__reverted -> !__has_assertion_failure
165       @post !(__has_buf_overflow)
166     */
167   function div(uint256 a, uint256 b, string memory errorMessage) internal
    ↪  pure returns (uint256) {
168     require(b > 0, errorMessage);
169     uint256 c = a / b;
170     // assert(a == b * c + a % b); // There is no case in which this
    ↪  doesn't hold
171
172     return c;
173   }
174
175   /**
176     * @dev Returns the remainder of dividing two unsigned integers.
    ↪  (unsigned integer modulo),
177     * Reverts when dividing by zero.
178     *
179     * Counterpart to Solidity's `%` operator. This function uses a `revert`
180     * opcode (which leaves remaining gas untouched) while Solidity uses an
181     * invalid opcode to revert (consuming all remaining gas).
182     *
183     * Requirements:
184     *
185     * - The divisor cannot be zero.
186     */
187   /*@CTK "SafeMath mod"
188       @tag assume_completion
189       @post b != 0 -> !__reverted
190       @post !__reverted -> __return == a % b
191       @post !__reverted -> !__has_overflow
192       @post !(__has_buf_overflow)
193       @post !(__has_assertion_failure)
194      */
195   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
196     return mod(a, b, "SafeMath: modulo by zero");
```

```
  }

  /**
    * @dev Returns the remainder of dividing two unsigned integers.
↪  (unsigned integer modulo),
    * Reverts with custom message when dividing by zero.
    *
    * Counterpart to Solidity's `%` operator. This function uses a `revert`
    * opcode (which leaves remaining gas untouched) while Solidity uses an
    * invalid opcode to revert (consuming all remaining gas).
    *
    * Requirements:
    *
    * - The divisor cannot be zero.
    */
  /*@CTK "SafeMath mod"
      @tag assume_completion
      @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, string memory errorMessage) internal
↪  pure returns (uint256) {
    require(b != 0, errorMessage);
    return a % b;
  }
}
```

SOBA_erc20.sol

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

import "./SOBA_Address.sol";
import "./SOBA_Context.sol";
import "./SOBA_SafeMath.sol";


/**
 * @dev Contract module which allows children to implement an emergency
↪  stop
 * mechanism that can be triggered by an authorized account.
 *
```

```
14  * This module is used through inheritance. It will make available the
15  * modifiers `whenNotPaused` and `whenPaused`, which can be applied to
16  * the functions of your contract. Note that they will not be pausable by
17  * simply including this module, only once the modifiers are put in place.
18  */
19 contract Pausable is Context {
20   /**
21     * @dev Emitted when the pause is triggered by `account`.
22     */
23   event Paused(address account);
24
25   /**
26     * @dev Emitted when the pause is lifted by `account`.
27     */
28   event Unpaused(address account);
29
30   bool private _paused;
31
32   /**
33     * @dev Initializes the contract in unpaused state.
34     */
35   constructor () internal {
36     _paused = false;
37   }
38
39   /**
40     * @dev Returns true if the contract is paused, and false otherwise.
41     */
42    /*@CTK "Pausable paused"
43       @tag assume_completion
44       @post __return == _paused
45     */
46   function paused() public view returns (bool) {
47     return _paused;
48   }
49
50   /**
51     * @dev Modifier to make a function callable only when the contract is
    ↪  not paused.
52     *
53     * Requirements:
54     *
55     * - The contract must not be paused.
56     */
57   modifier whenNotPaused() {
58     require(!_paused, "Pausable: paused");
59     _;
```

```
60    }
61
62    /**
63      * @dev Modifier to make a function callable only when the contract is
      ↪  paused.
64      *
65      * Requirements:
66      *
67      * - The contract must be paused.
68      */
69    modifier whenPaused() {
70      require(_paused, "Pausable: not paused");
71      _;
72    }
73
74    /**
75      * @dev Triggers stopped state.
76      *
77      * Requirements:
78      *
79      * - The contract must not be paused.
80      */
81    /*@CTK "Pausable _pause"
82        @tag assume_completion
83        @post __post._paused == true
84      */
85      _paused = true;
86      emit Paused(_msgSender());
87    }
88
89    /**
90      * @dev Returns to normal state.
91      *
92      * Requirements:
93      *
94      * - The contract must be paused.
95      */
96    /*@CTK "Pausable _unpause"
97        @tag assume_completion
98        @post __post._paused == false
99      */
100     _paused = false;
101     emit Unpaused(_msgSender());
102   }
103 }
104
105 /**
```

```
106  * @dev Interface of the ERC20 standard as defined in the EIP.
107  */
108 interface IERC20 {
109   /**
110     * @dev Returns the amount of tokens in existence.
111     */
112   function totalSupply() external view returns (uint256);
113
114   /**
115     * @dev Returns the amount of tokens owned by `account`.
116     */
117   function balanceOf(address account) external view returns (uint256);
118
119   /**
120     * @dev Moves `amount` tokens from the caller's account to `recipient`.
121     *
122     * Returns a boolean value indicating whether the operation succeeded.
123     *
124     * Emits a {Transfer} event.
125     */
126   function transfer(address recipient, uint256 amount) external returns
      ↪  (bool);
127
128   /**
129     * @dev Returns the remaining number of tokens that `spender` will be
130     * allowed to spend on behalf of `owner` through {transferFrom}. This is
131     * zero by default.
132     *
133     * This value changes when {approve} or {transferFrom} are called.
134     */
135   function allowance(address owner, address spender) external view returns
      ↪  (uint256);
136
137   /**
138     * @dev Sets `amount` as the allowance of `spender` over the caller's
      ↪  tokens.
139     *
140     * Returns a boolean value indicating whether the operation succeeded.
141     *
142     * IMPORTANT: Beware that changing an allowance with this method brings
      ↪  the risk
143     * that someone may use both the old and the new allowance by
      ↪  unfortunate
144     * transaction ordering. One possible solution to mitigate this race
145     * condition is to first reduce the spender's allowance to 0 and set the
146     * desired value afterwards:
147     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
```

```
148     *
149     * Emits an {Approval} event.
150     */
151   function approve(address spender, uint256 amount) external returns (bool);
152
153   /**
154     * @dev Moves `amount` tokens from `sender` to `recipient` using the
155     * allowance mechanism. `amount` is then deducted from the caller's
156     * allowance.
157     *
158     * Returns a boolean value indicating whether the operation succeeded.
159     *
160     * Emits a {Transfer} event.
161     */
162   function transferFrom(address sender, address recipient, uint256 amount)
    ↪  external returns (bool);
163
164   /**
165     * @dev Emitted when `value` tokens are moved from one account (`from`)
    ↪  to
166     * another (`to`).
167     *
168     * Note that `value` may be zero.
169     */
170   event Transfer(address indexed from, address indexed to, uint256 value);
171
172   /**
173     * @dev Emitted when the allowance of a `spender` for an `owner` is set
    ↪  by
174     * a call to {approve}. `value` is the new allowance.
175     */
176   event Approval(address indexed owner, address indexed spender, uint256
    ↪  value);
177 }
178
179
180 /**
181  * @dev Implementation of the {IERC20} interface.
182  *
183  * This implementation is agnostic to the way tokens are created. This
    ↪  means
184  * that a supply mechanism has to be added in a derived contract using
    ↪  {_mint}.
185  * For a generic mechanism see {ERC20PresetMinterPauser}.
186  *
187  * TIP: For a detailed writeup see our guide
188  *
    ↪  https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms/226[H
```

```
189  * to implement supply mechanisms].
190  *
191  * We have followed general OpenZeppelin guidelines: functions revert
     ↪  instead
192  * of returning `false` on failure. This behavior is nonetheless
     ↪  conventional
193  * and does not conflict with the expectations of ERC20 applications.
194  *
195  * Additionally, an {Approval} event is emitted on calls to {transferFrom}.
196  * This allows applications to reconstruct the allowance for all accounts
     ↪  just
197  * by listening to said events. Other implementations of the EIP may not
     ↪  emit
198  * these events, as it isn't required by the specification.
199  *
200  * Finally, the non-standard {decreaseAllowance} and {increaseAllowance}
201  * functions have been added to mitigate the well-known issues around
     ↪  setting
202  * allowances. See {IERC20-approve}.
203  */
204 contract ERC20 is Context, IERC20 {
205   using SafeMath for uint256;
206   using Address for address;
207
208   mapping (address => uint256) private _balances;
209
210   mapping (address => mapping (address => uint256)) private _allowances;
211
212   uint256 private _totalSupply;
213
214   string private _name;
215   string private _symbol;
216   uint8 private _decimals;
217
218   /**
219     * @dev Sets the values for {name} and {symbol}, initializes {decimals}
     ↪  with
220     * a default value of 18.
221     *
222     * To select a different value for {decimals}, use {_setupDecimals}.
223     *
224     * All three of these values are immutable: they can only be set once
     ↪  during
225     * construction.
226     */
227   /*@CTK "ERC20 constructor"
228       @tag assume_completion
```

```
229       @post __post._name == name
230       @post __post._symbol == symbol
231      */
232   constructor (string memory name, string memory symbol) public {
233     _name = name;
234     _symbol = symbol;
235     _decimals = 18;
236   }
237
238   /**
239     * @dev Returns the name of the token.
240     */
241   /*@CTK "ERC20 name"
242       @tag assume_completion
243       @post __return == __post._name
244      */
245   function name() public view returns (string memory) {
246     return _name;
247   }
248
249   /**
250     * @dev Returns the symbol of the token, usually a shorter version of
      ↪  the
251     * name.
252     */
253    /*@CTK "ERC20 symbol"
254       @tag assume_completion
255       @post __return == __post._symbol
256      */
257   function symbol() public view returns (string memory) {
258     return _symbol;
259   }
260
261   /**
262     * @dev Returns the number of decimals used to get its user
      ↪  representation.
263     * For example, if `decimals` equals `2`, a balance of `505` tokens
      ↪  should
264     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
265     *
266     * Tokens usually opt for a value of 18, imitating the relationship
      ↪  between
267     * Ether and Wei. This is the value {ERC20} uses, unless
      ↪  {_setupDecimals} is
268     * called.
269     *
270     * NOTE: This information is only used for _display_ purposes: it in
```

```
271     * no way affects any of the arithmetic of the contract, including
272     * {IERC20-balanceOf} and {IERC20-transfer}.
273     */
274    /*@CTK "ERC20 decimals"
275       @tag assume_completion
276       @post __return == __post._decimals
277      */
278    function decimals() public view returns (uint8) {
279      return _decimals;
280    }
281
282    /**
283     * @dev See {IERC20-totalSupply}.
284     */
285    /*@CTK "ERC20 totalSupply"
286       @tag assume_completion
287       @post __return == __post._totalSupply
288      */
289      return _totalSupply;
290    }
291
292    /**
293     * @dev See {IERC20-balanceOf}.
294     */
295    /*@CTK "ERC20 balanceOf"
296       @tag assume_completion
297       @post __return == __post._balances[account]
298      */
299      return _balances[account];
300    }
301    /*@CTK "ERC20 _balanceOf"
302       @tag assume_completion
303       @post __return == __post._balances[account]
304      */
305    function _balanceOf(address account) internal view returns (uint256) {
306      return _balances[account];
307    }
308    /**
309     * @dev See {IERC20-transfer}.
310     *
311     * Requirements:
312     *
313     * - `recipient` cannot be the zero address.
314     * - the caller must have a balance of at least `amount`.
315     */
316    /*@CTK "ERC20 transfer"
317       @tag assume_completion
```

```
318       @post msg.sender != address(0)
319       @post recipient != address(0)
320       @post msg.sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
321       @post msg.sender != recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender] - amount
322       @post msg.sender == recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender]
323     */
324     _transfer(_msgSender(), recipient, amount);
325     return true;
326   }
327
328   /**
329     * @dev See {IERC20-allowance}.
330     */
331   /*@CTK "ERC20 allowance"
332       @tag assume_completion
333       @post __return == _allowances[owner][spender]
334     */
335     return _allowances[owner][spender];
336   }
337
338   /**
339     * @dev See {IERC20-approve}.
340     *
341     * Requirements:
342     *
343     * - `spender` cannot be the zero address.
344     */
345    /*@CTK "ERC20 approve"
346        @tag assume_completion
347        @post msg.sender != address(0)
348        @post spender != address(0)
349        @post __post._allowances[msg.sender][spender] == amount
350      */
351     _approve(_msgSender(), spender, amount);
352     return true;
353   }
354
355   /**
356     * @dev See {IERC20-transferFrom}.
357     *
358     * Emits an {Approval} event indicating the updated allowance. This is
    ↪  not
359     * required by the EIP. See the note at the beginning of {ERC20};
360     *
```

```
361     * Requirements:
362     * - `sender` and `recipient` cannot be the zero address.
363     * - `sender` must have a balance of at least `amount`.
364     * - the caller must have allowance for ``sender``'s tokens of at least
365     * `amount`.
366     */
367   /*@CTK "ERC20 transferFrom"
368       @tag assume_completion
369       @post sender != address(0)
370       @post recipient != address(0)
371       @post msg.sender != address(0)
372       @post sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
373       @post sender != recipient -> __post._balances[sender] ==
    ↪  _balances[sender] - amount
374       @post sender == recipient -> __post._balances[sender] ==
    ↪  _balances[sender]
375       @post __post._allowances[sender][msg.sender] ==
    ↪  _allowances[sender][msg.sender] - amount
376     */
377     _transfer(sender, recipient, amount);
378     _approve(sender, _msgSender(),
    ↪  _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount
    ↪  exceeds allowance"));
379     return true;
380   }
381
382   /**
383     * @dev Atomically increases the allowance granted to `spender` by the
    ↪  caller.
384     *
385     * This is an alternative to {approve} that can be used as a mitigation
    ↪  for
386     * problems described in {IERC20-approve}.
387     *
388     * Emits an {Approval} event indicating the updated allowance.
389     *
390     * Requirements:
391     *
392     * - `spender` cannot be the zero address.
393     */
394   /*@CTK "ERC20 increaseAllowance"
395       @tag assume_completion
396       @post __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] + addedValue
397     */
398     _approve(_msgSender(), spender,
    ↪  _allowances[_msgSender()][spender].add(addedValue));
```

```
399     return true;
400   }
401
402   /**
403     * @dev Atomically decreases the allowance granted to `spender` by the
    ↪  caller.
404     *
405     * This is an alternative to {approve} that can be used as a mitigation
    ↪  for
406     * problems described in {IERC20-approve}.
407     *
408     * Emits an {Approval} event indicating the updated allowance.
409     *
410     * Requirements:
411     *
412     * - `spender` cannot be the zero address.
413     * - `spender` must have allowance for the caller of at least
414     * `subtractedValue`.
415     */
416     /*@CTK "ERC20 decreaseAllowance"
417       @tag assume_completion
418       @post __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] - subtractedValue
419     */
420     _approve(_msgSender(), spender,
    ↪  _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased
    ↪  allowance below zero"));
421     return true;
422   }
423
424   /**
425     * @dev Moves tokens `amount` from `sender` to `recipient`.
426     *
427     * This is internal function is equivalent to {transfer}, and can be
    ↪  used to
428     * e.g. implement automatic token fees, slashing mechanisms, etc.
429     *
430     * Emits a {Transfer} event.
431     *
432     * Requirements:
433     *
434     * - `sender` cannot be the zero address.
435     * - `recipient` cannot be the zero address.
436     * - `sender` must have a balance of at least `amount`.
437     */
438    /*@CTK "ERC20 _transfer"
439       @tag assume_completion
```

```
440       @post sender != address(0)
441       @post recipient != address(0)
442       @post sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
443       @post sender != recipient -> __post._balances[sender] ==
    ↪  _balances[sender] - amount
444       @post sender == recipient -> __post._balances[sender] ==
    ↪  _balances[sender]
445     */
446     require(sender != address(0), "ERC20: transfer from the zero address");
447     require(recipient != address(0), "ERC20: transfer to the zero address");
448
449     _beforeTokenTransfer(sender, recipient, amount);
450
451     _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount
    ↪  exceeds balance");
452     _balances[recipient] = _balances[recipient].add(amount);
453     emit Transfer(sender, recipient, amount);
454   }
455
456   /** @dev Creates `amount` tokens and assigns them to `account`,
    ↪  increasing
457     * the total supply.
458     *
459     * Emits a {Transfer} event with `from` set to the zero address.
460     *
461     * Requirements
462     *
463     * - `to` cannot be the zero address.
464     */
465     /*@CTK "ERC20 _mint"
466       @tag assume_completion
467       @post account != address(0)
468       @post __post._totalSupply == _totalSupply + amount
469       @post __post._balances[account] == _balances[account] + amount
470     */
471     require(account != address(0), "ERC20: mint to the zero address");
472
473     _totalSupply = _totalSupply.add(amount);
474     _balances[account] = _balances[account].add(amount);
475     emit Transfer(address(0), account, amount);
476   }
477
478   /**
479     * @dev Destroys `amount` tokens from `account`, reducing the
480     * total supply.
481     *
```

```
482     * Emits a {Transfer} event with `to` set to the zero address.
483     *
484     * Requirements
485     *
486     * - `account` cannot be the zero address.
487     * - `account` must have at least `amount` tokens.
488     */
489    /*@CTK "ERC20 _burn"
490       @tag assume_completion
491       @post account != address(0)
492       @post __post._totalSupply == _totalSupply - amount
493       @post __post._balances[account] == _balances[account] - amount
494     */
495     require(account != address(0), "ERC20: burn from the zero address");
496
497     _balances[account] = _balances[account].sub(amount, "ERC20: burn amount
    ↪  exceeds balance");
498     _totalSupply = _totalSupply.sub(amount);
499     emit Transfer(account, address(0), amount);
500   }
501
502   /**
503     * @dev Sets `amount` as the allowance of `spender` over the `owner`s
    ↪  tokens.
504     *
505     * This is internal function is equivalent to `approve`, and can be used
    ↪  to
506     * e.g. set automatic allowances for certain subsystems, etc.
507     *
508     * Emits an {Approval} event.
509     *
510     * Requirements:
511     *
512     * - `owner` cannot be the zero address.
513     * - `spender` cannot be the zero address.
514     */
515     /*@CTK "ERC20 _approve"
516        @tag assume_completion
517        @post owner != address(0)
518        @post spender != address(0)
519        @post __post._allowances[owner][spender] == amount
520      */
521     require(owner != address(0), "ERC20: approve from the zero address");
522     require(spender != address(0), "ERC20: approve to the zero address");
523
524     _allowances[owner][spender] = amount;
525     emit Approval(owner, spender, amount);
```

```
526   }
527
528   /**
529     * @dev Hook that is called before any transfer of tokens. This includes
530     * minting and burning.
531     *
532     * Calling conditions:
533     *
534     * - when `from` and `to` are both non-zero, `amount` of ``from``'s
      ↪ tokens
535     * will be to transferred to `to`.
536     * - when `from` is zero, `amount` tokens will be minted for `to`.
537     * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
538     * - `from` and `to` are never both zero.
539     *
540     * To learn more about hooks, head to
      ↪ xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
541     */
542 }
```