

CERTIK AUDIT REPORT FOR SPONB



Request Date: 2019-06-17
Revision Date: 2019-06-18
Platform Name: Ethereum



Contents

Disclaimer	1
About CertiK	2
Exective Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Manual Review Notes	6
Static Analysis Results	7
Formal Verification Results	8
How to read	8
Source Code with CertiK Labels	55

Disclaimer

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

About CertiK

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

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

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

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

Executive Summary

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

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

Vulnerability Classification

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

Critical

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

Medium

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

Low

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

Testing Summary

PASS

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

Jun 18, 2019



Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

None. (Note: The violations in the formal verification result section of this report is for internal evaluation and are not indications of security issues in the user smart contracts. We recommend using `require` in place of `assert` as did in OpenZeppelin's latest `SafeMath` library.)

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- `sponb.sol`
`cef917d19ae144734eed2a5a2efe0a7216b931e28cf8717d5d6412a508fdb7d8`

Summary

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

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

Recommendations

None.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File sponb.sol

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



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

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

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

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


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


Raw code	35	function transferFrom(address from, address 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
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Formal Verification Request 1

Method will not encounter an assertion failure.

 18, Jun 2019

 40.08 ms

Line 30 in File sponb.sol

30 `//@CTK FAIL NO_ASF`

Line 38-49 in File sponb.sol

```
38 function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
39     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
40     // benefit is lost if 'b' is also tested.
41     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
42     if (a == 0) {
43         return 0;
44     }
45
46     c = a * b;
47     assert(c / a == b);
48     return c;
49 }
```

 This code violates the specification.


```
1 Counter Example:
2 Before Execution:
3   Input = {
4     a = 2
5     b = 156
6   }
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24    c = 0
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": "EmptyAddress"
30    }
31  ]
32
```

33 Function invocation is reverted.

Formal Verification Request 2

SafeMath mul

 18, Jun 2019

 294.99 ms

Line 31-37 in File sponb.sol

```
31  /*@CTK "SafeMath mul"
32  @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
33  @post !__reverted -> c == a * b
34  @post !__reverted == !__has_overflow
35  @post !__reverted -> !(__has_assertion_failure)
36  @post !(__has_buf_overflow)
37  */
```

Line 38-49 in File sponb.sol


```
38  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
39  // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
40  // benefit is lost if 'b' is also tested.
41  // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
42  if (a == 0) {
43  return 0;
44  }
45
46  c = a * b;
47  assert(c / a == b);
48  return c;
49  }
```

 The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

 18, Jun 2019

 5.83 ms

Line 54 in File sponb.sol

```
54  /*@CTK FAIL NO_ASF
```

Line 62-67 in File sponb.sol

```
62  function div(uint256 a, uint256 b) internal pure returns (uint256) {
63  // assert(b > 0); // Solidity automatically throws when dividing by 0
64  // uint256 c = a / b;
65  // assert(a == b * c + a % b); // There is no case in which this doesn't hold
66  return a / b;
67  }
```

 This code violates the specification.

```


1 Counter Example:
2 Before Execution:
3   Input = {
4     a = 0
5     b = 0
6   }
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    __return = 0
21    block = {
22      "number": 0,
23      "timestamp": 0
24    }
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": "EmptyAddress"
30    }
31  ]
32
33 Function invocation is reverted.

```

Formal Verification Request 4

SafeMath div

 18, Jun 2019

 0.32 ms

Line 55-61 in File sponb.sol

```

55 /*@CTK "SafeMath div"
56   @post b != 0 -> !__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 62-67 in File sponb.sol

```

62 function div(uint256 a, uint256 b) internal pure returns (uint256) {
63   // assert(b > 0); // Solidity automatically throws when dividing by 0
64   // uint256 c = a / b;
65   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
66   return a / b;

```


67 }

✓ The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

 18, Jun 2019

 11.24 ms

Line 73 in File sponb.sol

73 //@CTK FAIL NO_ASF

Line 81-84 in File sponb.sol


```
81     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
82         assert(b <= a);
83         return a - b;
84     }
```


✗ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3     Input = {
4         a = 0
5         b = 1
6     }
7     Internal = {
8         __has_assertion_failure = false
9         __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14             "gas": 0,
15             "sender": 0,
16             "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22             "number": 0,
23             "timestamp": 0
24         }
25     }
26     Address_Map = [
27         {
28             "key": "ALL_OTHERS",
29             "value": "EmptyAddress"
30         }
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

 18, Jun 2019

 0.8 ms

Line 74-80 in File sponb.sol

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

Line 81-84 in File sponb.sol


```
81  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
82      assert(b <= a);
83      return a - b;
84  }
```

 The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

 18, Jun 2019

 11.92 ms

Line 90 in File sponb.sol

```
90  //@CTK FAIL NO_ASF
```

Line 98-102 in File sponb.sol

```
98  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
99      c = a + b;
100     assert(c >= a);
101     return c;
102 }
```

 This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 191
5          b = 65
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
```

```

13     msg = {
14         "gas": 0,
15         "sender": 0,
16         "value": 0
17     }
18 }
19 Other = {
20     block = {
21         "number": 0,
22         "timestamp": 0
23     }
24     c = 0
25 }
26 Address_Map = [
27     {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30     }
31 ]

```

33 Function invocation is reverted.

Formal Verification Request 8

SafeMath add



18, Jun 2019



2.7 ms

Line 91-97 in File sponb.sol

```

91  /*@CTK "SafeMath add"
92     @post (a + b < a || a + b < b) == __reverted
93     @post !__reverted -> c == a + b
94     @post !__reverted -> !__has_overflow
95     @post !__reverted -> !(__has_assertion_failure)
96     @post !(__has_buf_overflow)
97  */

```

Line 98-102 in File sponb.sol

```

98  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
99      c = a + b;
100      assert(c >= a);
101      return c;
102  }

```



The code meets the specification.

Formal Verification Request 9

If method completes, integer overflow would not happen.



18, Jun 2019



6.58 ms

Line 149 in File sponb.sol

149 `//@CTK NO_OVERFLOW`

Line 157-162 in File sponb.sol

```
157     constructor()
158     public
159     {
160         tokenTransfer = false;
161         owner = msg.sender;
162     }
```

✓ The code meets the specification.

Formal Verification Request 10

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

📅 18, Jun 2019

🕒 0.34 ms

Line 150 in File sponb.sol

150 `//@CTK NO_BUF_OVERFLOW`

Line 157-162 in File sponb.sol

```
157     constructor()
158     public
159     {
160         tokenTransfer = false;
161         owner = msg.sender;
162     }
```

✓ The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.35 ms

Line 151 in File sponb.sol

151 `//@CTK NO_ASF`

Line 157-162 in File sponb.sol


```
157     constructor()
158     public
159     {
160         tokenTransfer = false;
161         owner = msg.sender;
162     }
```

✓ The code meets the specification.

Formal Verification Request 12

Lockable constructor

 18, Jun 2019

 0.83 ms

Line 152-156 in File sponb.sol

```
152  /*@CTK "Lockable constructor"
153      @tag assume_completion
154      @post !__post.tokenTransfer
155      @post __post.owner == msg.sender
156  */
```


Line 157-162 in File sponb.sol


```
157  constructor()
158  public
159  {
160      tokenTransfer = false;
161      owner = msg.sender;
162  }
```

 The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

 18, Jun 2019

 21.93 ms

Line 167 in File sponb.sol

```
167  //@CTK NO_OVERFLOW
```

Line 176-183 in File sponb.sol


```
176  function setLockAddress(address target, bool status)
177  external
178  isOwner
179  {
180      require(owner != target);
181      lockAddress[target] = status;
182      emit Locked(target, status);
183  }
```

 The code meets the specification.

Formal Verification Request 14

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

 18, Jun 2019

 0.49 ms

Line 168 in File sponb.sol

168 `//@CTK NO_BUF_OVERFLOW`

Line 176-183 in File sponb.sol


```
176 function setLockAddress(address target, bool status)
177 external
178 isOwner
179 {
180     require(owner != target);
181     lockAddress[target] = status;
182     emit Locked(target, status);
183 }
```

✓ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

 18, Jun 2019

 0.49 ms

Line 169 in File sponb.sol

169 `//@CTK NO_ASF`

Line 176-183 in File sponb.sol


```
176 function setLockAddress(address target, bool status)
177 external
178 isOwner
179 {
180     require(owner != target);
181     lockAddress[target] = status;
182     emit Locked(target, status);
183 }
```

✓ The code meets the specification.

Formal Verification Request 16

Lockable setLockAddress

 18, Jun 2019

 6.93 ms

Line 170-175 in File sponb.sol

```
170 /*@CTK "Lockable setLockAddress"
171    @tag assume_completion
172    @post owner == msg.sender
173    @post owner != target
174    @post __post.lockAddress[target] == status
175 */
```

Line 176-183 in File sponb.sol


```
176 function setLockAddress(address target, bool status)
177 external
178 isOwner
179 {
180     require(owner != target);
181     lockAddress[target] = status;
182     emit Locked(target, status);
183 }
```

✓ The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

 18, Jun 2019

 13.94 ms

Line 188 in File sponb.sol

```
188 // @CTK_NO_OVERFLOW
```

Line 196-202 in File sponb.sol


```
196 function setUnlockAddress(address target, bool status)
197 external
198 isOwner
199 {
200     unlockAddress[target] = status;
201     emit Unlocked(target, status);
202 }
```

✓ The code meets the specification.

Formal Verification Request 18

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

 18, Jun 2019

 0.41 ms

Line 189 in File sponb.sol

```
189 // @CTK_NO_BUF_OVERFLOW
```


Line 196-202 in File sponb.sol


```
196 function setUnlockAddress(address target, bool status)
197 external
198 isOwner
199 {
200     unlockAddress[target] = status;
201     emit Unlocked(target, status);
202 }
```

✓ The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

 18, Jun 2019

 0.39 ms

Line 190 in File sponb.sol

```
190 // @CTK NO_ASF
```

Line 196-202 in File sponb.sol


```
196 function setUnlockAddress(address target, bool status)
197 external
198 isOwner
199 {
200     unlockAddress[target] = status;
201     emit Unlocked(target, status);
202 }
```

 The code meets the specification.

Formal Verification Request 20

Lockable setUnlockAddress

 18, Jun 2019

 2.72 ms

Line 191-195 in File sponb.sol

```
191 /* @CTK "Lockable setUnlockAddress"
192    @tag assume_completion
193    @post owner == msg.sender
194    @post __post.unlockAddress[target] == status
195 */
```

Line 196-202 in File sponb.sol


```
196 function setUnlockAddress(address target, bool status)
197 external
198 isOwner
199 {
200     unlockAddress[target] = status;
201     emit Unlocked(target, status);
202 }
```

 The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

 18, Jun 2019

 18.17 ms

Line 237 in File sponb.sol

237 `//@CTK NO_OVERFLOW`

Line 246-253 in File sponb.sol


```
246     constructor(uint256 initial_balance)
247     public
248     {
249         require(initial_balance != 0);
250         _supply = initial_balance;
251         _balances[msg.sender] = initial_balance;
252         emit Transfer(address(0), msg.sender, initial_balance);
253     }
```

✓ The code meets the specification.

Formal Verification Request 22

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

 18, Jun 2019

 0.41 ms

Line 238 in File sponb.sol

238 `//@CTK NO_BUF_OVERFLOW`

Line 246-253 in File sponb.sol


```
246     constructor(uint256 initial_balance)
247     public
248     {
249         require(initial_balance != 0);
250         _supply = initial_balance;
251         _balances[msg.sender] = initial_balance;
252         emit Transfer(address(0), msg.sender, initial_balance);
253     }
```

✓ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

 18, Jun 2019

 0.4 ms

Line 239 in File sponb.sol

239 `//@CTK NO_ASF`

Line 246-253 in File sponb.sol


```
246     constructor(uint256 initial_balance)
247     public
248     {
249         require(initial_balance != 0);
250         _supply = initial_balance;
251         _balances[msg.sender] = initial_balance;
```


```
252     emit Transfer(address(0), msg.sender, initial_balance);
253 }
```

✓ The code meets the specification.

Formal Verification Request 24

SPONBToken

 18, Jun 2019

 2.66 ms

Line 240-245 in File sponb.sol

```
240  /*@CTK SPONBToken
241     @tag assume_completion
242     @pre (initial_balance != 0)
243     @post __post._supply == initial_balance
244     @post __post._balances[msg.sender] == initial_balance
245  */
```

Line 246-253 in File sponb.sol


```
246  constructor(uint256 initial_balance)
247  public
248  {
249      require(initial_balance != 0);
250      _supply = initial_balance;
251      _balances[msg.sender] = initial_balance;
252      emit Transfer(address(0), msg.sender, initial_balance);
253  }
```

✓ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

 18, Jun 2019

 5.36 ms

Line 255 in File sponb.sol

```
255  //@CTK NO_OVERFLOW
```


Line 262-267 in File sponb.sol


```
262  function totalSupply()
263  public
264  view
265  returns (uint256) {
266      return _supply;
267  }
```

✓ The code meets the specification.

Formal Verification Request 26

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

 18, Jun 2019

 0.36 ms

Line 256 in File sponb.sol

```
256 // @CTK NO_BUF_OVERFLOW
```


Line 262-267 in File sponb.sol


```
262 function totalSupply()
263 public
264 view
265 returns (uint256) {
266     return _supply;
267 }
```

 The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

 18, Jun 2019

 0.32 ms

Line 257 in File sponb.sol

```
257 // @CTK NO_ASF
```

Line 262-267 in File sponb.sol


```
262 function totalSupply()
263 public
264 view
265 returns (uint256) {
266     return _supply;
267 }
```

 The code meets the specification.

Formal Verification Request 28

totalSupply

 18, Jun 2019

 0.34 ms

Line 258-261 in File sponb.sol

```
258 /* @CTK totalSupply
259     @tag assume_completion
260     @post (__return) == (_supply)
261 */
```



Line 262-267 in File sponb.sol


```
262     function totalSupply()
263     public
264     view
265     returns (uint256) {
266         return _supply;
267     }
```

✓ The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

 18, Jun 2019

 5.59 ms

Line 269 in File sponb.sol

```
269     //@CTK NO_OVERFLOW
```

Line 276-281 in File sponb.sol


```
276     function balanceOf(address who)
277     public
278     view
279     returns (uint256) {
280         return _balances[who];
281     }
```

✓ The code meets the specification.

Formal Verification Request 30

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

 18, Jun 2019

 0.33 ms

Line 270 in File sponb.sol

```
270     //@CTK NO_BUF_OVERFLOW
```

Line 276-281 in File sponb.sol


```
276     function balanceOf(address who)
277     public
278     view
279     returns (uint256) {
280         return _balances[who];
281     }
```

✓ The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

 18, Jun 2019

 0.32 ms

Line 271 in File sponb.sol

```
271  // @CTK NO_ASF
```

Line 276-281 in File sponb.sol


```
276  function balanceOf(address who)
277  public
278  view
279  returns (uint256) {
280      return _balances[who];
281  }
```

 The code meets the specification.

Formal Verification Request 32

balanceOf

 18, Jun 2019

 0.34 ms

Line 272-275 in File sponb.sol

```
272  /* @CTK balanceOf
273     @tag assume_completion
274     @post (__return) == (_balances[who])
275  */
```

Line 276-281 in File sponb.sol


```
276  function balanceOf(address who)
277  public
278  view
279  returns (uint256) {
280      return _balances[who];
281  }
```

 The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

 18, Jun 2019

 133.57 ms

Line 283 in File sponb.sol

```
283  // @CTK NO_OVERFLOW
```

Line 298-310 in File sponb.sol

```
298     function transfer(address to, uint256 value)
299     public
300     isTokenTransfer
301     checkLock
302     returns (bool) {
303         require(to != address(0));
304         require(_balances[msg.sender] >= value);
305
306         _balances[msg.sender] = _balances[msg.sender].sub(value);
307         _balances[to] = _balances[to].add(value);
308         emit Transfer(msg.sender, to, value);
309         return true;
310     }
```

✓ The code meets the specification.

Formal Verification Request 34

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

📅 18, Jun 2019

🕒 31.35 ms

Line 284 in File sponb.sol

```
284     //@CTK NO_BUF_OVERFLOW
```

Line 298-310 in File sponb.sol

```
298     function transfer(address to, uint256 value)
299     public
300     isTokenTransfer
301     checkLock
302     returns (bool) {
303         require(to != address(0));
304         require(_balances[msg.sender] >= value);
305
306         _balances[msg.sender] = _balances[msg.sender].sub(value);
307         _balances[to] = _balances[to].add(value);
308         emit Transfer(msg.sender, to, value);
309         return true;
310     }
```

✓ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 73.62 ms

Line 285 in File sponb.sol

```
285     //@CTK FAIL_NO_ASF
```

Line 298-310 in File sponb.sol

```

298 function transfer(address to, uint256 value)
299 public
300 isTokenTransfer
301 checkLock
302 returns (bool) {
303     require(to != address(0));
304     require(_balances[msg.sender] >= value);
305
306     _balances[msg.sender] = _balances[msg.sender].sub(value);
307     _balances[to] = _balances[to].add(value);
308     emit Transfer(msg.sender, to, value);
309     return true;
310 }

```

✖ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         to = 8
5         value = 169
6     }
7     This = 0
8     Internal = {
9         __has_assertion_failure = false
10        __has_buf_overflow = false
11        __has_overflow = false
12        __has_returned = false
13        __reverted = false
14        msg = {
15            "gas": 0,
16            "sender": 0,
17            "value": 0
18        }
19    }
20    Other = {
21        __return = false
22        block = {
23            "number": 0,
24            "timestamp": 0
25        }
26    }
27    Address_Map = [
28        {
29            "key": 0,
30            "value": {
31                "contract_name": "SPONBToken",
32                "balance": 0,
33                "contract": {
34                    "name": "",
35                    "symbol": "",
36                    "decimals": 0,
37                    "adminMode": false,
38                    "_balances": [
39                        {
40                            "key": 8,
41                            "value": 161
42                        },

```

```


43         {
44             "key": 64,
45             "value": 128
46         },
47         {
48             "key": 1,
49             "value": 0
50         },
51         {
52             "key": "ALL_OTHERS",
53             "value": 169
54         }
55     ],
56     "_approvals": [
57         {
58             "key": "ALL_OTHERS",
59             "value": [
60                 {
61                     "key": "ALL_OTHERS",
62                     "value": 169
63                 }
64             ]
65         }
66     ],
67     "_supply": 0,
68     "tokenTransfer": true,
69     "owner": 0,
70     "unlockAddress": [
71         {
72             "key": "ALL_OTHERS",
73             "value": false
74         }
75     ],
76     "lockAddress": [
77         {
78             "key": 32,
79             "value": true
80         },
81         {
82             "key": "ALL_OTHERS",
83             "value": false
84         }
85     ]
86     },
87     {
88     },
89     {
90         "key": "ALL_OTHERS",
91         "value": "EmptyAddress"
92     }
93 ]


```

95 Function invocation is reverted.

Formal Verification Request 36

transfer

 18, Jun 2019

 356.64 ms

Line 286-297 in File sponb.sol

```

286  /*@CTK transfer
287      @tag assume_completion
288      @post (tokenTransfer || unlockAddress[msg.sender])
289      @post !lockAddress[msg.sender]
290      @pre to != address(0)
291      @pre value <= _balances[msg.sender]
292      @post (msg.sender != to) -> (__post._balances[to] == _balances[to] + value)
293      @post (msg.sender != to) -> (__post._balances[msg.sender] == _balances[msg.
        sender] - value)
294      @post (msg.sender == to) -> (__post._balances[to] == _balances[to])
295      @post (msg.sender == to) -> (__post._balances[msg.sender] == _balances[msg.
        sender])
296      @post __return == true
297  */

```

Line 298-310 in File sponb.sol

```

298  function transfer(address to, uint256 value)
299  public
300  isTokenTransfer
301  checkLock
302  returns (bool) {
303      require(to != address(0));
304      require(_balances[msg.sender] >= value);
305
306      _balances[msg.sender] = _balances[msg.sender].sub(value);
307      _balances[to] = _balances[to].add(value);
308      emit Transfer(msg.sender, to, value);
309      return true;
310  }


```

 The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

 18, Jun 2019

 4.59 ms

Line 312 in File sponb.sol

```

312  /*@CTK NO_OVERFLOW

```

Line 319-324 in File sponb.sol

```

319  function allowance(address owner, address spender)
320  public
321  view
322  returns (uint256) {

```

```
323     return _approvals[owner][spender];
324 }
```

✓ The code meets the specification.

Formal Verification Request 38

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

📅 18, Jun 2019

🕒 0.3 ms

Line 313 in File sponb.sol

```
313 // @CTK NO_BUF_OVERFLOW
```

Line 319-324 in File sponb.sol

```
319 function allowance(address owner, address spender)
320 public
321 view
322 returns (uint256) {
323     return _approvals[owner][spender];
324 }
```

✓ The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.3 ms

Line 314 in File sponb.sol

```
314 // @CTK NO_ASF
```

Line 319-324 in File sponb.sol

```
319 function allowance(address owner, address spender)
320 public
321 view
322 returns (uint256) {
323     return _approvals[owner][spender];
324 }
```

✓ The code meets the specification.

Formal Verification Request 40

allowance

📅 18, Jun 2019

🕒 0.3 ms

Line 315-318 in File sponb.sol

```

315  /*@CTK allowance
316     @tag assume_completion
317     @post (__return) == (_approvals[owner][spender])
318  */

```

Line 319-324 in File sponb.sol

```

319  function allowance(address owner, address spender)
320  public
321  view
322  returns (uint256) {
323      return _approvals[owner][spender];
324  }

```

✓ The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.



18, Jun 2019



157.67 ms

Line 326 in File sponb.sol

```

326  //@CTK NO_OVERFLOW

```

Line 341-354 in File sponb.sol

```

341  function transferFrom(address from, address to, uint256 value)
342  public
343  isTokenTransfer
344  checkLock
345  returns (bool success) {
346      require(!lockAddress[from]);
347      require(_balances[from] >= value);
348      require(_approvals[from][msg.sender] >= value);
349      _balances[from] = _balances[from].sub(value);
350      _balances[to] = _balances[to].add(value);
351      _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352      emit Transfer(from, to, value);
353      return true;
354  }

```

✓ The code meets the specification.

Formal Verification Request 42

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



18, Jun 2019



42.81 ms

Line 327 in File sponb.sol

```

327  //@CTK NO_BUF_OVERFLOW

```

Line 341-354 in File sponb.sol


```

341 function transferFrom(address from, address to, uint256 value)
342 public
343 isTokenTransfer
344 checkLock
345 returns (bool success) {
346     require(!lockAddress[from]);
347     require(_balances[from] >= value);
348     require(_approvals[from][msg.sender] >= value);
349     _balances[from] = _balances[from].sub(value);
350     _balances[to] = _balances[to].add(value);
351     _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352     emit Transfer(from, to, value);
353     return true;
354 }

```

✓ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 208.86 ms

Line 328 in File sponb.sol

```

328 // @CTK FAIL NO_ASF

```

Line 341-354 in File sponb.sol

```

341 function transferFrom(address from, address to, uint256 value)
342 public
343 isTokenTransfer
344 checkLock
345 returns (bool success) {
346     require(!lockAddress[from]);
347     require(_balances[from] >= value);
348     require(_approvals[from][msg.sender] >= value);
349     _balances[from] = _balances[from].sub(value);
350     _balances[to] = _balances[to].add(value);
351     _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352     emit Transfer(from, to, value);
353     return true;
354 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     from = 1
5     to = 0
6     value = 129
7   }
8   This = 0
9   Internal = {
10     __has_assertion_failure = false
11     __has_buf_overflow = false

```

```

12     __has_overflow = false
13     __has_returned = false
14     __reverted = false
15     msg = {
16         "gas": 0,
17         "sender": 0,
18         "value": 0
19     }
20 }
21 Other = {
22     block = {
23         "number": 0,
24         "timestamp": 0
25     }
26     success = false
27 }
28 Address_Map = [
29     {
30         "key": 0,
31         "value": {
32             "contract_name": "SPONBToken",
33             "balance": 0,
34             "contract": {
35                 "name": "",
36                 "symbol": "",
37                 "decimals": 128,
38                 "adminMode": false,
39                 "_balances": [
40                     {
41                         "key": 3,
42                         "value": 32
43                     },
44                     {
45                         "key": 0,
46                         "value": 159
47                     },
48                     {
49                         "key": 36,
50                         "value": 0
51                     },
52                     {
53                         "key": 8,
54                         "value": 0
55                     },
56                     {
57                         "key": 128,
58                         "value": 32
59                     },
60                     {
61                         "key": 4,
62                         "value": 0
63                     },
64                     {
65                         "key": 32,
66                         "value": 32
67                     },
68                     {
69                         "key": 9,

```

```

70         "value": 0
71     },
72     {
73         "key": 2,
74         "value": 0
75     },
76     {
77         "key": 129,
78         "value": 0
79     },
80     {
81         "key": 16,
82         "value": 0
83     },
84     {
85         "key": 1,
86         "value": 192
87     },
88     {
89         "key": "ALL_OTHERS",
90         "value": 170
91     }
92 ],
93 "_approvals": [
94     {
95         "key": 0,
96         "value": [
97             {
98                 "key": 128,
99                 "value": 2
100             },
101             {
102                 "key": 0,
103                 "value": 8
104             },
105             {
106                 "key": "ALL_OTHERS",
107                 "value": 16
108             }
109         ]
110     },
111     {
112         "key": 1,
113         "value": [
114             {
115                 "key": 0,
116                 "value": 170
117             },
118             {
119                 "key": "ALL_OTHERS",
120                 "value": 1
121             }
122         ]
123     },
124     {
125         "key": "ALL_OTHERS",
126         "value": [
127             {

```

```


128         "key": "ALL_OTHERS",
129         "value": 170
130     }
131 ]
132 }
133 ],
134 "_supply": 4,
135 "tokenTransfer": false,
136 "owner": 128,
137 "unlockAddress": [
138     {
139         "key": 32,
140         "value": true
141     },
142     {
143         "key": 0,
144         "value": true
145     },
146     {
147         "key": "ALL_OTHERS",
148         "value": false
149     }
150 ],
151 "lockAddress": [
152     {
153         "key": 128,
154         "value": true
155     },
156     {
157         "key": "ALL_OTHERS",
158         "value": false
159     }
160 ],
161 ]
162 }
163 },
164 {
165     "key": "ALL_OTHERS",
166     "value": "EmptyAddress"
167 }
168 ]
169
170 Function invocation is reverted.

```

Formal Verification Request 44

transferFrom

 18, Jun 2019

 1025.32 ms

Line 329-340 in File sponb.sol

```

329 /*@CTK "transferFrom"
330    @tag assume_completion
331    @pre !lockAddress[from]
332    @pre (value) <= (_balances[from])

```

```

333     @pre (value) <= (_approvals[from][msg.sender])
334     @post (from != to) -> (__post._balances[to] == (_balances[to] + value))
335     @post (from != to) -> (__post._balances[from] == (_balances[from] - value))
336     @post (from == to) -> (__post._balances[to] == _balances[to])
337     @post (from == to) -> (__post._balances[from] == _balances[from])
338     @post (__post._approvals[from][msg.sender]) == (_approvals[from][msg.sender] -
339         value)
340     @post (success) == (true)
341 */

```

Line 341-354 in File sponb.sol

```

341     function transferFrom(address from, address to, uint256 value)
342     public
343     isTokenTransfer
344     checkLock
345     returns (bool success) {
346         require(!lockAddress[from]);
347         require(_balances[from] >= value);
348         require(_approvals[from][msg.sender] >= value);
349         _balances[from] = _balances[from].sub(value);
350         _balances[to] = _balances[to].add(value);
351         _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352         emit Transfer(from, to, value);
353         return true;
354     }


```

✓ The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

 18, Jun 2019

 17.13 ms

Line 365 in File sponb.sol

```

365     //@CTK NO_OVERFLOW

```

Line 373-380 in File sponb.sol

```

373     function approve(address spender, uint256 value)
374     public
375     checkLock
376     returns (bool) {
377         _approvals[msg.sender][spender] = value;
378         emit Approval(msg.sender, spender, value);
379         return true;
380     }


```

✓ The code meets the specification.

Formal Verification Request 46

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

 18, Jun 2019

 0.41 ms

Line 366 in File sponb.sol

```
366 // @CTK NO_BUF_OVERFLOW
```

Line 373-380 in File sponb.sol


```
373 function approve(address spender, uint256 value)
374 public
375 checkLock
376 returns (bool) {
377     _approvals[msg.sender][spender] = value;
378     emit Approval(msg.sender, spender, value);
379     return true;
380 }
```

 The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

 18, Jun 2019

 0.49 ms

Line 367 in File sponb.sol

```
367 // @CTK NO_ASF
```

Line 373-380 in File sponb.sol


```
373 function approve(address spender, uint256 value)
374 public
375 checkLock
376 returns (bool) {
377     _approvals[msg.sender][spender] = value;
378     emit Approval(msg.sender, spender, value);
379     return true;
380 }
```

 The code meets the specification.

Formal Verification Request 48

approve

 18, Jun 2019

 2.31 ms

Line 368-372 in File sponb.sol

```
368 /* @CTK approve
369    @tag assume_completion
370    @post !lockAddress[msg.sender]
371    @post (__post._approvals[msg.sender][spender]) == (value)
372 */
```

Line 373-380 in File sponb.sol

```
373     function approve(address spender, uint256 value)
374     public
375     checkLock
376     returns (bool) {
377         _approvals[msg.sender][spender] = value;
378         emit Approval(msg.sender, spender, value);
379         return true;
380     }
```

✓ The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.



18, Jun 2019



38.87 ms

Line 391 in File sponb.sol

```
391     //@CTK NO_OVERFLOW
```

Line 400-408 in File sponb.sol

```
400     function increaseApproval(address _spender, uint256 _addedValue)
401     public
402     checkLock
403     returns (bool) {
404         _approvals[msg.sender][_spender] = (
405             _approvals[msg.sender][_spender].add(_addedValue));
406         emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
407         return true;
408     }
```

✓ The code meets the specification.

Formal Verification Request 50

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



18, Jun 2019



0.85 ms

Line 392 in File sponb.sol

```
392     //@CTK NO_BUF_OVERFLOW
```

Line 400-408 in File sponb.sol

```
400     function increaseApproval(address _spender, uint256 _addedValue)
401     public
402     checkLock
403     returns (bool) {
404         _approvals[msg.sender][_spender] = (
405             _approvals[msg.sender][_spender].add(_addedValue));
```

```


406     emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
407     return true;
408 }


```

✓ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

 18, Jun 2019

 11.18 ms

Line 393 in File sponb.sol

```

393    //@CTK FAIL NO_ASF

```

Line 400-408 in File sponb.sol

```

400    function increaseApproval(address _spender, uint256 _addedValue)
401    public
402    checkLock
403    returns (bool) {
404        _approvals[msg.sender][_spender] = (
405            _approvals[msg.sender][_spender].add(_addedValue));
406        emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
407        return true;
408    }

```

✗ This code violates the specification.

```

1  Counter Example:
2  Before Execution:
3      Input = {
4          _addedValue = 161
5          _spender = 0
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
12         __has_returned = false
13         __reverted = false
14         msg = {
15             "gas": 0,
16             "sender": 0,
17             "value": 0
18         }
19     }
20     Other = {
21         __return = false
22         block = {
23             "number": 0,
24             "timestamp": 0
25         }
26     }
27     Address_Map = [

```



```

28 {
29   "key": 0,
30   "value": {
31     "contract_name": "SPONBToken",
32     "balance": 0,
33     "contract": {
34       "name": "",
35       "symbol": "",
36       "decimals": 0,
37       "adminMode": false,
38       "_balances": [
39         {
40           "key": 0,
41           "value": 16
42         },
43         {
44           "key": 4,
45           "value": 0
46         },
47         {
48           "key": 32,
49           "value": 32
50         },
51         {
52           "key": "ALL_OTHERS",
53           "value": 161
54         }
55       ],
56       "_approvals": [
57         {
58           "key": 0,
59           "value": [
60             {
61               "key": 0,
62               "value": 95
63             },
64             {
65               "key": 2,
66               "value": 0
67             },
68             {
69               "key": 8,
70               "value": 0
71             },
72             {
73               "key": 32,
74               "value": 0
75             },
76             {
77               "key": "ALL_OTHERS",
78               "value": 161
79             }
80           ]
81         },
82         {
83           "key": "ALL_OTHERS",
84           "value": [
85             {

```

```


86         "key": "ALL_OTHERS",
87         "value": 161
88     }
89 ]
90 }
91 ],
92 "_supply": 0,
93 "tokenTransfer": false,
94 "owner": 0,
95 "unlockAddress": [
96     {
97         "key": 0,
98         "value": true
99     },
100    {
101        "key": "ALL_OTHERS",
102        "value": false
103    }
104 ],
105 "lockAddress": [
106     {
107         "key": "ALL_OTHERS",
108         "value": false
109     }
110 ]
111 }
112 }
113 },
114 {
115     "key": "ALL_OTHERS",
116     "value": "EmptyAddress"
117 }
118 ]
119
120 Function invocation is reverted.

```

Formal Verification Request 52

increaseApproval

 18, Jun 2019

 4.1 ms

Line 394-399 in File sponb.sol

```

394  /*@CTK increaseApproval
395     @tag assume_completion
396     @post !lockAddress[msg.sender]
397     @post (__post._approvals[msg.sender][_spender]) == (_approvals[msg.sender][
398         _spender] + _addedValue)
398     @post (__return) == (true)
399  */

```

Line 400-408 in File sponb.sol

```

400  function increaseApproval(address _spender, uint256 _addedValue)
401  public
402  checkLock

```

```

403     returns (bool) {
404         _approvals[msg.sender][_spender] = (
405             _approvals[msg.sender][_spender].add(_addedValue));
406         emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
407         return true;
408     }


```

✓ The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

 18, Jun 2019

 50.46 ms

Line 419 in File sponb.sol

```

419     //@CTK NO_OVERFLOW

```

Line 429-441 in File sponb.sol

```

429     function decreaseApproval(address _spender, uint256 _subtractedValue)
430     public
431     checkLock
432     returns (bool) {
433         uint256 oldValue = _approvals[msg.sender][_spender];
434         if (_subtractedValue > oldValue) {
435             _approvals[msg.sender][_spender] = 0;
436         } else {
437             _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
438         }
439         emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
440         return true;
441     }


```

✓ The code meets the specification.

Formal Verification Request 54

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

 18, Jun 2019

 0.92 ms

Line 420 in File sponb.sol

```

420     //@CTK NO_BUF_OVERFLOW

```

Line 429-441 in File sponb.sol

```

429     function decreaseApproval(address _spender, uint256 _subtractedValue)
430     public
431     checkLock
432     returns (bool) {
433         uint256 oldValue = _approvals[msg.sender][_spender];
434         if (_subtractedValue > oldValue) {

```

```

435     _approvals[msg.sender][_spender] = 0;
436   } else {
437     _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
438   }
439   emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
440   return true;
441 }

```

✓ The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 1.44 ms

Line 421 in File sponb.sol

```

421  //@CTK NO_ASF

```

Line 429-441 in File sponb.sol

```

429  function decreaseApproval(address _spender, uint256 _subtractedValue)
430  public
431  checkLock
432  returns (bool) {
433      uint256 oldValue = _approvals[msg.sender][_spender];
434      if (_subtractedValue > oldValue) {
435          _approvals[msg.sender][_spender] = 0;
436      } else {
437          _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
438      }
439      emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
440      return true;
441  }

```

✓ The code meets the specification.

Formal Verification Request 56

decreaseApproval

📅 18, Jun 2019

🕒 51.55 ms

Line 422-428 in File sponb.sol

```

422  /*@CTK decreaseApproval
423     @tag assume_completion
424     @post !lockAddress[msg.sender]
425     @pre _spender != msg.sender
426     @post (_subtractedValue > _approvals[msg.sender][_spender]) -> (__post.
         _approvals[msg.sender][_spender] == 0)
427     @post (_subtractedValue <= _approvals[msg.sender][_spender]) -> (__post.
         _approvals[msg.sender][_spender] == _approvals[msg.sender][_spender] -
         _subtractedValue)

```

428 `*/`



Line 429-441 in File sponb.sol

```
429 function decreaseApproval(address _spender, uint256 _subtractedValue)
430 public
431 checkLock
432 returns (bool) {
433     uint256 oldValue = _approvals[msg.sender][_spender];
434     if (_subtractedValue > oldValue) {
435         _approvals[msg.sender][_spender] = 0;
436     } else {
437         _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
438     }
439     emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
440     return true;
441 }
```

 The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

 18, Jun 2019 82.11 ms

Line 447 in File sponb.sol

447 `//@CTK NO_OVERFLOW`


Line 458-468 in File sponb.sol

```
458 function burnTokens(uint256 tokensAmount)
459 public
460 isAdminMode
461 isOwner
462 {
463     require(_balances[msg.sender] >= tokensAmount);
464
465     _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
466     _supply = _supply.sub(tokensAmount);
467     emit TokenBurned(msg.sender, tokensAmount);
468 }
```

 The code meets the specification.

Formal Verification Request 58

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

 18, Jun 2019 14.95 ms

Line 448 in File sponb.sol

448 `//@CTK NO_BUF_OVERFLOW`

Line 458-468 in File sponb.sol

```

458     function burnTokens(uint256 tokensAmount)
459     public
460     isAdminMode
461     isOwner
462     {
463         require(_balances[msg.sender] >= tokensAmount);
464
465         _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
466         _supply = _supply.sub(tokensAmount);
467         emit TokenBurned(msg.sender, tokensAmount);
468     }

```

✓ The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.



18, Jun 2019



42.35 ms

Line 449 in File sponb.sol

```

449     //@CTK FAIL NO_ASF

```

Line 458-468 in File sponb.sol

```

458     function burnTokens(uint256 tokensAmount)
459     public
460     isAdminMode
461     isOwner
462     {
463         require(_balances[msg.sender] >= tokensAmount);
464
465         _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
466         _supply = _supply.sub(tokensAmount);
467         emit TokenBurned(msg.sender, tokensAmount);
468     }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     tokensAmount = 4
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0

```

```

17     }
18 }
19 Other = {
20     block = {
21         "number": 0,
22         "timestamp": 0
23     }
24 }
25 Address_Map = [
26     {
27         "key": 0,
28         "value": {
29             "contract_name": "SPONBToken",
30             "balance": 0,
31             "contract": {
32                 "name": "",
33                 "symbol": "",
34                 "decimals": 0,
35                 "adminMode": true,
36                 "_balances": [
37                     {
38                         "key": 32,
39                         "value": 16
40                     },
41                     {
42                         "key": 64,
43                         "value": 2
44                     },
45                     {
46                         "key": 1,
47                         "value": 0
48                     },
49                     {
50                         "key": 16,
51                         "value": 0
52                     },
53                     {
54                         "key": "ALL_OTHERS",
55                         "value": 128
56                     }
57                 ],
58                 "_approvals": [
59                     {
60                         "key": "ALL_OTHERS",
61                         "value": [
62                             {
63                                 "key": "ALL_OTHERS",
64                                 "value": 128
65                             }
66                         ]
67                     }
68                 ],
69                 "_supply": 0,
70                 "tokenTransfer": false,
71                 "owner": 0,
72                 "unlockAddress": [
73                     {
74                         "key": "ALL_OTHERS",

```


```


75         "value": false
76     }
77 ],
78     "lockAddress": [
79         {
80             "key": 0,
81             "value": true
82         },
83         {
84             "key": "ALL_OTHERS",
85             "value": false
86         }
87     ]
88 }
89 },
90 {
91     "key": "ALL_OTHERS",
92     "value": "EmptyAddress"
93 }
94 ]
95 ]
96
97 Function invocation is reverted.

```

Formal Verification Request 60

burnTokens

 18, Jun 2019

 149.79 ms

Line 450-457 in File sponb.sol

```

450  /*@CTK burnTokens
451     @tag assume_completion
452     @pre adminMode
453     @pre owner == msg.sender
454     @post (tokensAmount <= _balances[msg.sender])
455     @post (__post._supply) == ((_supply) - (tokensAmount))
456     @post (__post._balances[msg.sender]) == ((_balances[msg.sender]) - (tokensAmount
457         ))
458 */

```

Line 458-468 in File sponb.sol

```

458  function burnTokens(uint256 tokensAmount)
459  public
460  isAdminMode
461  isOwner
462  {
463      require(_balances[msg.sender] >= tokensAmount);
464
465      _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
466      _supply = _supply.sub(tokensAmount);
467      emit TokenBurned(msg.sender, tokensAmount);
468  }


```

 The code meets the specification.

Formal Verification Request 61

If method completes, integer overflow would not happen.

 18, Jun 2019

 19.14 ms

Line 479 in File sponb.sol

479 `//@CTK NO_OVERFLOW`


Line 488-495 in File sponb.sol


```
488 function setTokenTransfer(bool _tokenTransfer)
489 external
490 isAdminMode
491 isOwner
492 {
493     tokenTransfer = _tokenTransfer;
494     emit SetTokenTransfer(tokenTransfer);
495 }
```

 The code meets the specification.

Formal Verification Request 62

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

 18, Jun 2019

 0.49 ms

Line 480 in File sponb.sol

480 `//@CTK NO_BUF_OVERFLOW`

Line 488-495 in File sponb.sol


```
488 function setTokenTransfer(bool _tokenTransfer)
489 external
490 isAdminMode
491 isOwner
492 {
493     tokenTransfer = _tokenTransfer;
494     emit SetTokenTransfer(tokenTransfer);
495 }
```

 The code meets the specification.

Formal Verification Request 63

Method will not encounter an assertion failure.

 18, Jun 2019

 0.47 ms

Line 481 in File sponb.sol

481 `//@CTK NO_ASF`

Line 488-495 in File sponb.sol

```
488 function setTokenTransfer(bool _tokenTransfer)
489 external
490 isAdminMode
491 isOwner
492 {
493     tokenTransfer = _tokenTransfer;
494     emit SetTokenTransfer(tokenTransfer);
495 }
```

✓ The code meets the specification.

Formal Verification Request 64

setTokenTransfer

📅 18, Jun 2019

🕒 3.24 ms

Line 482-487 in File sponb.sol

```
482 /*@CTK setTokenTransfer
483 @tag assume_completion
484 @pre adminMode
485 @pre owner == msg.sender
486 @post __post.tokenTransfer == _tokenTransfer
487 */
```

Line 488-495 in File sponb.sol

```
488 function setTokenTransfer(bool _tokenTransfer)
489 external
490 isAdminMode
491 isOwner
492 {
493     tokenTransfer = _tokenTransfer;
494     emit SetTokenTransfer(tokenTransfer);
495 }
```

✓ The code meets the specification.

Formal Verification Request 65

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 14.09 ms

Line 497 in File sponb.sol

497 `//@CTK NO_OVERFLOW`

Line 505-511 in File sponb.sol


```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

✓ The code meets the specification.

Formal Verification Request 66

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

 18, Jun 2019

 0.37 ms

Line 498 in File sponb.sol

```
498     //@CTK NO_BUF_OVERFLOW
```

Line 505-511 in File sponb.sol


```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

✓ The code meets the specification.

Formal Verification Request 67

Method will not encounter an assertion failure.

 18, Jun 2019

 0.35 ms

Line 499 in File sponb.sol

```
499     //@CTK NO_ASF
```


Line 505-511 in File sponb.sol


```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

✓ The code meets the specification.

Formal Verification Request 68

setAdminMode

 18, Jun 2019

 2.43 ms

Line 500-504 in File sponb.sol

```
500  /*@CTK setAdminMode
501      @tag assume_completion
502      @pre owner == msg.sender
503      @post __post.adminMode == _adminMode
504  */
```


Line 505-511 in File sponb.sol


```
505  function setAdminMode(bool _adminMode)
506  public
507  isOwner
508  {
509      adminMode = _adminMode;
510      emit SetAdminMode(adminMode);
511  }
```

 The code meets the specification.

Formal Verification Request 69

If method completes, integer overflow would not happen.

 18, Jun 2019

 85.96 ms

Line 517 in File sponb.sol

```
517  //@CTK NO_OVERFLOW
```


Line 528-541 in File sponb.sol


```
528  function emergencyTransfer(address emergencyAddress)
529  public
530  isAdminMode
531  isOwner
532  returns (bool success) {
533      require(emergencyAddress != owner);
534      _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);
535
536      emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
537      emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);
538
539      _balances[emergencyAddress] = 0;
540      return true;
541  }
```

 The code meets the specification.

Formal Verification Request 70

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

 18, Jun 2019

 5.7 ms

Line 518 in File sponb.sol

518 `//@CTK NO_BUF_OVERFLOW`

Line 528-541 in File sponb.sol


```
528     function emergencyTransfer(address emergencyAddress)
529     public
530     isAdminMode
531     isOwner
532     returns (bool success) {
533         require(emergencyAddress != owner);
534         _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);
535
536         emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
537         emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);
538
539         _balances[emergencyAddress] = 0;
540         return true;
541     }
```

 The code meets the specification.

Formal Verification Request 71

Method will not encounter an assertion failure.

 18, Jun 2019

 40.94 ms

Line 519 in File sponb.sol

519 `//@CTK FAIL_NO_ASF`

Line 528-541 in File sponb.sol

```
528     function emergencyTransfer(address emergencyAddress)
529     public
530     isAdminMode
531     isOwner
532     returns (bool success) {
533         require(emergencyAddress != owner);
534         _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);
535
536         emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
537         emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);
538
539         _balances[emergencyAddress] = 0;
540         return true;
541     }
```

 This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     emergencyAddress = 64
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24    success = false
25  }
26  Address_Map = [
27    {
28      "key": 0,
29      "value": {
30        "contract_name": "SPONBToken",
31        "balance": 0,
32        "contract": {
33          "name": "",
34          "symbol": "",
35          "decimals": 0,
36          "adminMode": true,
37          "_balances": [
38            {
39              "key": 1,
40              "value": 2
41            },
42            {
43              "key": 16,
44              "value": 8
45            },
46            {
47              "key": 0,
48              "value": 192
49            },
50            {
51              "key": 64,
52              "value": 64
53            },
54            {
55              "key": 8,
56              "value": 2
57            },
58            {

```

```

59         "key": "ALL_OTHERS",
60         "value": 0
61     }
62 ],
63     "_approvals": [
64     {
65         "key": "ALL_OTHERS",
66         "value": [
67         {
68             "key": "ALL_OTHERS",
69             "value": 0
70         }
71         ]
72     }
73 ],
74     "_supply": 0,
75     "tokenTransfer": false,
76     "owner": 0,
77     "unlockAddress": [
78     {
79         "key": 0,
80         "value": true
81     },
82     {
83         "key": "ALL_OTHERS",
84         "value": false
85     }
86 ],
87     "lockAddress": [
88     {
89         "key": "ALL_OTHERS",
90         "value": false
91     }
92 ]
93 }
94 }
95 },
96 {
97     "key": "ALL_OTHERS",
98     "value": "EmptyAddress"
99 }
100 ]

```

102 Function invocation is reverted.

Formal Verification Request 72

emergencyTransfer



18, Jun 2019



39.46 ms

Line 520-527 in File sponb.sol

```

520 /*@CTK emergencyTransfer
521    @tag assume_completion
522    @pre adminMode

```

```

523     @pre owner == msg.sender
524     @pre emergencyAddress != owner
525     @post __post._balances[owner] == (_balances[owner] + _balances[emergencyAddress
    ])
526     @post __post._balances[emergencyAddress] == 0
527     */

```

Line 528-541 in File sponb.sol

```

528     function emergencyTransfer(address emergencyAddress)
529     public
530     isAdminMode
531     isOwner
532     returns (bool success) {
533         require(emergencyAddress != owner);
534         _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);
535
536         emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
537         emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);
538
539         _balances[emergencyAddress] = 0;
540         return true;
541     }

```

✓ The code meets the specification.

Source Code with CertiK Labels

File sponb.sol

```

1  pragma solidity ^0.4.24;
2
3  /**
4   *Submitted for verification at Etherscan.io on 2019-05-17
5   */
6
7  /**
8   * @title ERC20 Interface
9   */
10 contract ERC20 {
11     function totalSupply() public view returns (uint256);
12     function balanceOf(address who) public view returns (uint256);
13     function transfer(address to, uint256 value) public returns (bool);
14     event Transfer(address indexed from, address indexed to, uint256 value);
15
16     function allowance(address owner, address spender) public view returns (uint256);
17     function transferFrom(address from, address to, uint256 value) public returns (
18         bool);
19     function approve(address spender, uint256 value) public returns (bool);
20     event Approval(address indexed owner, address indexed spender, uint256 value);
21 }
22 /**
23  * @title SafeMath
24  * @dev Math operations with safety checks that throw on error
25  */
26 library SafeMath {
27     /**
28      * @dev Multiplies two numbers, throws on overflow.
29      */
30     //@CTK FAIL NO_ASF
31     //@CTK "SafeMath mul"
32     @post ((a > 0) && ((a * b) / a) != b) == (__reverted)
33     @post !__reverted -> c == a * b
34     @post !__reverted == !__has_overflow
35     @post !__reverted -> !(__has_assertion_failure)
36     @post !(__has_buf_overflow)
37     */
38     function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
39         // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
40         // benefit is lost if 'b' is also tested.
41         // See: https://github.com/OpenZeppelin/zeppelin-solidity/pull/522
42         if (a == 0) {
43             return 0;
44         }
45
46         c = a * b;
47         assert(c / a == b);
48         return c;
49     }
50
51     /**
52      * @dev Integer division of two numbers, truncating the quotient.
53      */

```

```

54 // @CTK FAIL NO_ASF
55 /* @CTK "SafeMath div"
56     @post b != 0 -> !__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 div(uint256 a, uint256 b) internal pure returns (uint256) {
63     // assert(b > 0); // Solidity automatically throws when dividing by 0
64     // uint256 c = a / b;
65     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
66     return a / b;
67 }
68
69 /**
70  * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
71     than minuend).
72 */
73 // @CTK FAIL NO_ASF
74 /* @CTK "SafeMath sub"
75     @post (a < b) == __reverted
76     @post !__reverted -> __return == a - b
77     @post !__reverted -> !__has_overflow
78     @post !__reverted -> !(__has_assertion_failure)
79     @post !(__has_buf_overflow)
80 */
81 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
82     assert(b <= a);
83     return a - b;
84 }
85
86 /**
87  * @dev Adds two numbers, throws on overflow.
88 */
89 // @CTK FAIL NO_ASF
90 /* @CTK "SafeMath add"
91     @post (a + b < a || a + b < b) == __reverted
92     @post !__reverted -> c == a + b
93     @post !__reverted -> !__has_overflow
94     @post !__reverted -> !(__has_assertion_failure)
95     @post !(__has_buf_overflow)
96 */
97 function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
98     c = a + b;
99     assert(c >= a);
100     return c;
101 }
102 }
103 }
104 /**
105  * @title Lockable Token
106  * @author info@yggdrash.io
107 */
108 contract Lockable {
109     bool public tokenTransfer;
110     address public owner;

```

```

111
112  /**
113   * @dev They can transfer even if tokenTranster flag is false.
114   */
115  mapping(address => bool) public unlockAddress;
116
117  /**
118   * @dev They cannot transfer even if tokenTransfer flag is true.
119   */
120  mapping(address => bool) public lockAddress;
121
122  event Locked(address lockAddress, bool status);
123  event Unlocked(address unlockedAddress, bool status);
124
125  /**
126   * @dev check whether can tranfer tokens or not.
127   */
128  modifier isTokenTransfer {
129      if(!tokenTransfer) {
130          require(unlockAddress[msg.sender]);
131      }
132      _;
133  }
134
135  /**
136   * @dev check whether registered in lockAddress or not
137   */
138  modifier checkLock {
139      require(!lockAddress[msg.sender]);
140      _;
141  }
142
143  modifier isOwner
144  {
145      require(owner == msg.sender);
146      _;
147  }
148
149  //@CTK NO_OVERFLOW
150  //@CTK NO_BUF_OVERFLOW
151  //@CTK NO_ASF
152  /*@CTK "Lockable constructor"
153   @tag assume_completion
154   @post !__post.tokenTransfer
155   @post __post.owner == msg.sender
156  */
157  constructor()
158  public
159  {
160      tokenTransfer = false;
161      owner = msg.sender;
162  }
163
164  /**
165   * @dev add or remove in lockAddress(blacklist)
166   */
167  //@CTK NO_OVERFLOW
168  //@CTK NO_BUF_OVERFLOW

```

```

169 // @CTK NO_ASF
170 /* @CTK "Lockable setLockAddress"
171     @tag assume_completion
172     @post owner == msg.sender
173     @post owner != target
174     @post __post.lockAddress[target] == status
175 */
176 function setLockAddress(address target, bool status)
177 external
178 isOwner
179 {
180     require(owner != target);
181     lockAddress[target] = status;
182     emit Locked(target, status);
183 }
184
185 /**
186  * @dev add or remove in unlockAddress(whitelist)
187  */
188 // @CTK NO_OVERFLOW
189 // @CTK NO_BUF_OVERFLOW
190 // @CTK NO_ASF
191 /* @CTK "Lockable setUnlockAddress"
192     @tag assume_completion
193     @post owner == msg.sender
194     @post __post.unlockAddress[target] == status
195 */
196 function setUnlockAddress(address target, bool status)
197 external
198 isOwner
199 {
200     unlockAddress[target] = status;
201     emit Unlocked(target, status);
202 }
203 }
204 /**
205  * @title YGGDRASH Token Contract.
206  * @author info@yggdrash.io
207  * @notice This contract is the updated version that fixes the unlocking bug.
208  * This source code is audited by external auditors.
209  */
210 contract SPONBToken is ERC20, Lockable {
211
212     string public constant name = "SPONB";
213     string public constant symbol = "SPO";
214     uint8 public constant decimals = 18;
215
216     /**
217      * @dev If this flag is true, admin can use enableTokenTransfer(),
218           emergencyTransfer().
219      */
220     bool public adminMode;
221
222     using SafeMath for uint256;
223
224     mapping(address => uint256) internal _balances;
225     mapping(address => mapping(address => uint256)) internal _approvals;
226     uint256 internal _supply;

```

```

226
227     event TokenBurned(address burnAddress, uint256 amountOfTokens);
228     event SetTokenTransfer(bool transfer);
229     event SetAdminMode(bool adminMode);
230     event EmergencyTransfer(address indexed from, address indexed to, uint256 value);
231
232     modifier isAdminMode {
233         require(adminMode);
234         _;
235     }
236
237     //@CTK NO_OVERFLOW
238     //@CTK NO_BUF_OVERFLOW
239     //@CTK NO_ASF
240     /*CTK SPONBToken
241         @tag assume_completion
242         @pre (initial_balance != 0)
243         @post __post._supply == initial_balance
244         @post __post._balances[msg.sender] == initial_balance
245     */
246     constructor(uint256 initial_balance)
247     public
248     {
249         require(initial_balance != 0);
250         _supply = initial_balance;
251         _balances[msg.sender] = initial_balance;
252         emit Transfer(address(0), msg.sender, initial_balance);
253     }
254
255     //@CTK NO_OVERFLOW
256     //@CTK NO_BUF_OVERFLOW
257     //@CTK NO_ASF
258     /*CTK totalSupply
259         @tag assume_completion
260         @post (__return) == (_supply)
261     */
262     function totalSupply()
263     public
264     view
265     returns (uint256) {
266         return _supply;
267     }
268
269     //@CTK NO_OVERFLOW
270     //@CTK NO_BUF_OVERFLOW
271     //@CTK NO_ASF
272     /*CTK balanceOf
273         @tag assume_completion
274         @post (__return) == (_balances[who])
275     */
276     function balanceOf(address who)
277     public
278     view
279     returns (uint256) {
280         return _balances[who];
281     }
282
283     //@CTK NO_OVERFLOW

```

```

284 // @CTK NO_BUF_OVERFLOW
285 // @CTK FAIL_NO_ASF
286 /* @CTK transfer
287     @tag assume_completion
288     @post (tokenTransfer || unlockAddress[msg.sender])
289     @post !lockAddress[msg.sender]
290     @pre to != address(0)
291     @pre value <= _balances[msg.sender]
292     @post (msg.sender != to) -> (__post._balances[to] == _balances[to] + value)
293     @post (msg.sender != to) -> (__post._balances[msg.sender] == _balances[msg.
        sender] - value)
294     @post (msg.sender == to) -> (__post._balances[to] == _balances[to])
295     @post (msg.sender == to) -> (__post._balances[msg.sender] == _balances[msg.
        sender])
296     @post __return == true
297 */
298 function transfer(address to, uint256 value)
299 public
300 isTokenTransfer
301 checkLock
302 returns (bool) {
303     require(to != address(0));
304     require(_balances[msg.sender] >= value);
305
306     _balances[msg.sender] = _balances[msg.sender].sub(value);
307     _balances[to] = _balances[to].add(value);
308     emit Transfer(msg.sender, to, value);
309     return true;
310 }
311
312 // @CTK NO_OVERFLOW
313 // @CTK NO_BUF_OVERFLOW
314 // @CTK NO_ASF
315 /* @CTK allowance
316     @tag assume_completion
317     @post (__return) == (_approvals[owner][spender])
318 */
319 function allowance(address owner, address spender)
320 public
321 view
322 returns (uint256) {
323     return _approvals[owner][spender];
324 }
325
326 // @CTK NO_OVERFLOW
327 // @CTK NO_BUF_OVERFLOW
328 // @CTK FAIL_NO_ASF
329 /* @CTK "transferFrom"
330     @tag assume_completion
331     @pre !lockAddress[from]
332     @pre (value) <= (_balances[from])
333     @pre (value) <= (_approvals[from][msg.sender])
334     @post (from != to) -> (__post._balances[to] == (_balances[to] + value))
335     @post (from != to) -> (__post._balances[from] == (_balances[from] - value))
336     @post (from == to) -> (__post._balances[to] == _balances[to])
337     @post (from == to) -> (__post._balances[from] == _balances[from])
338     @post (__post._approvals[from][msg.sender]) == (_approvals[from][msg.sender] -
        value)

```

```

339     @post (success) == (true)
340     */
341     function transferFrom(address from, address to, uint256 value)
342     public
343     isTokenTransfer
344     checkLock
345     returns (bool success) {
346         require(!lockAddress[from]);
347         require(_balances[from] >= value);
348         require(_approvals[from][msg.sender] >= value);
349         _balances[from] = _balances[from].sub(value);
350         _balances[to] = _balances[to].add(value);
351         _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352         emit Transfer(from, to, value);
353         return true;
354     }
355
356     /**
357     * @dev Approve the passed address to spend the specified amount of tokens on
358     *     behalf of msg.sender.
359     * Beware that changing an allowance with this method brings the risk that someone
360     *     may use both the old
361     *     and the new allowance by unfortunate transaction ordering. One possible
362     *     solution to mitigate this
363     *     race condition is to first reduce the spender's allowance to 0 and set the
364     *     desired value afterwards:
365     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
366     * @param spender The address which will spend the funds.
367     * @param value The amount of tokens to be spent.
368     */
369     // @CTK NO_OVERFLOW
370     // @CTK NO_BUF_OVERFLOW
371     // @CTK NO_ASF
372     /* @CTK approve
373     @tag assume_completion
374     @post !lockAddress[msg.sender]
375     @post (__post._approvals[msg.sender][spender]) == (value)
376     */
377     function approve(address spender, uint256 value)
378     public
379     checkLock
380     returns (bool) {
381         _approvals[msg.sender][spender] = value;
382         emit Approval(msg.sender, spender, value);
383         return true;
384     }
385
386     /**
387     * @dev Increase the amount of tokens that an owner allowed to a spender.
388     * approve should be called when allowed[_spender] == 0. To increment
389     * allowed value is better to use this function to avoid 2 calls (and wait until
390     * the first transaction is mined)
391     * From MonolithDAO Token.sol
392     * @param _spender The address which will spend the funds.
393     * @param _addedValue The amount of tokens to increase the allowance by.
394     */
395     // @CTK NO_OVERFLOW
396     // @CTK NO_BUF_OVERFLOW

```

```

393 //CTK FAIL NO_ASF
394 /*CTK increaseApproval
395     @tag assume_completion
396     @post !lockAddress[msg.sender]
397     @post (__post._approvals[msg.sender][_spender]) == (_approvals[msg.sender][
        _spender] + _addedValue)
398     @post (__return) == (true)
399 */
400 function increaseApproval(address _spender, uint256 _addedValue)
401 public
402 checkLock
403 returns (bool) {
404     _approvals[msg.sender][_spender] = (
405         _approvals[msg.sender][_spender].add(_addedValue));
406     emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
407     return true;
408 }
409
410 /**
411  * @dev Decrease the amount of tokens that an owner allowed to a spender.
412  * approve should be called when allowed[_spender] == 0. To decrement
413  * allowed value is better to use this function to avoid 2 calls (and wait until
414  * the first transaction is mined)
415  * From MonolithDAO Token.sol
416  * @param _spender The address which will spend the funds.
417  * @param _subtractedValue The amount of tokens to decrease the allowance by.
418  */
419 //CTK NO_OVERFLOW
420 //CTK NO_BUF_OVERFLOW
421 //CTK NO_ASF
422 /*CTK decreaseApproval
423     @tag assume_completion
424     @post !lockAddress[msg.sender]
425     @pre _spender != msg.sender
426     @post (_subtractedValue > _approvals[msg.sender][_spender]) -> (__post.
        _approvals[msg.sender][_spender] == 0)
427     @post (_subtractedValue <= _approvals[msg.sender][_spender]) -> (__post.
        _approvals[msg.sender][_spender] == _approvals[msg.sender][_spender] -
        _subtractedValue)
428 */
429 function decreaseApproval(address _spender, uint256 _subtractedValue)
430 public
431 checkLock
432 returns (bool) {
433     uint256 oldValue = _approvals[msg.sender][_spender];
434     if (_subtractedValue > oldValue) {
435         _approvals[msg.sender][_spender] = 0;
436     } else {
437         _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
438     }
439     emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
440     return true;
441 }
442
443 /**
444  * @dev Burn tokens can only use by owner
445  */
446

```



```

447 // @CTK NO_OVERFLOW
448 // @CTK NO_BUF_OVERFLOW
449 // @CTK FAIL_NO_ASF
450 /* @CTK burnTokens
451     @tag assume_completion
452     @pre adminMode
453     @pre owner == msg.sender
454     @post (tokensAmount <= _balances[msg.sender])
455     @post (__post._supply) == ((_supply) - (tokensAmount))
456     @post (__post._balances[msg.sender]) == ((_balances[msg.sender]) - (tokensAmount
        ))
457 */
458 function burnTokens(uint256 tokensAmount)
459 public
460 isAdminMode
461 isOwner
462 {
463     require(_balances[msg.sender] >= tokensAmount);
464
465     _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
466     _supply = _supply.sub(tokensAmount);
467     emit TokenBurned(msg.sender, tokensAmount);
468 }
469
470 /**
471  * @dev Set the tokenTransfer flag.
472  * If true,
473  * - unregistered lockAddress can transfer()
474  * - registered lockAddress can not transfer()
475  * If false,
476  * - registered unlockAddress & unregistered lockAddress
477  * - can transfer(), unregistered unlockAddress can not transfer()
478  */
479 // @CTK NO_OVERFLOW
480 // @CTK NO_BUF_OVERFLOW
481 // @CTK NO_ASF
482 /* @CTK setTokenTransfer
483     @tag assume_completion
484     @pre adminMode
485     @pre owner == msg.sender
486     @post __post.tokenTransfer == _tokenTransfer
487 */
488 function setTokenTransfer(bool _tokenTransfer)
489 external
490 isAdminMode
491 isOwner
492 {
493     tokenTransfer = _tokenTransfer;
494     emit SetTokenTransfer(tokenTransfer);
495 }
496
497 // @CTK NO_OVERFLOW
498 // @CTK NO_BUF_OVERFLOW
499 // @CTK NO_ASF
500 /* @CTK setAdminMode
501     @tag assume_completion
502     @pre owner == msg.sender
503     @post __post.adminMode == _adminMode

```

```

504  */
505  function setAdminMode(bool _adminMode)
506  public
507  isOwner
508  {
509      adminMode = _adminMode;
510      emit SetAdminMode(adminMode);
511  }
512
513  /**
514   * @dev In emergency situation,
515   * admin can use emergencyTransfer() for protecting user's token.
516   */
517  //@CTK NO_OVERFLOW
518  //@CTK NO_BUF_OVERFLOW
519  //@CTK FAIL NO_ASF
520  /*@CTK emergencyTransfer
521   @tag assume_completion
522   @pre adminMode
523   @pre owner == msg.sender
524   @pre emergencyAddress != owner
525   @post __post._balances[owner] == (_balances[owner] + _balances[emergencyAddress
526       ])
527   @post __post._balances[emergencyAddress] == 0
528  */
529  function emergencyTransfer(address emergencyAddress)
530  public
531  isAdminMode
532  isOwner
533  returns (bool success) {
534      require(emergencyAddress != owner);
535      _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);
536
537      emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
538      emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);
539
540      _balances[emergencyAddress] = 0;
541      return true;
542  }

```