

CERTiK VERIFICATION REPORT FOR BLOCKCLOUD



Request Date: 2019-04-04
Revision Date: 2019-04-16

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 Blockcloud(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.

PASS

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

Apr 16, 2019



Summary

This audit report summarises the smart contract verification service requested by Blockcloud. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security loopholes.

The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the time of the audit.

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	Com-	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Ran-	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	for	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	to	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Variable	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Default	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

`transferFrom` succeeds even when the source address is the same as the destination address. There is a potential side effect that reduce the allowance even though balances remain unchanged.

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.

Source Code with CertiK Labels

File BlockcloudToken.sol

```

1  pragma solidity ^0.4.13;
2
3  library SafeMath {
4
5      /**
6       * @dev Multiplies two numbers, throws on overflow.
7       */
8
9      /*@CTK SafeMath_mul
10       @tag spec
11       @post __reverted == __has_assertion_failure
12       @post __has_assertion_failure == __has_overflow
13       @post __reverted == false -> __return == a * b
14       @post a == 0 -> __return == 0
15       @post msg == msg__post
16       @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
17       @post __addr_map == __addr_map__post
18      */
19     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20         if (a == 0) {
21             return 0;
22         }
23         uint256 c = a * b;
24         assert(c / a == b);
25         return c;
26     }
27
28     /**
29     * @dev Integer division of two numbers, truncating the quotient.
30     */
31
32     /*@CTK SafeMath_div
33     @tag spec
34     @post __reverted == __has_assertion_failure
35     @post b == 0 -> __reverted == true // solidity throws on 0.
36     @post __has_overflow == true -> __has_assertion_failure == true
37     @post __reverted == false -> __return == a / b
38     @post msg == msg__post
39     @post __addr_map == __addr_map__post
40     */
41     function div(uint256 a, uint256 b) internal pure returns (uint256) {
42         // assert(b > 0); // Solidity automatically throws when dividing by 0
43         uint256 c = a / b;
44         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45         return c;
46     }
47
48
49     /**
50     * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
51         than minuend).
52     */
53     /*@CTK SafeMath_sub

```

```

54  @tag spec
55  @post __reverted == __has_assertion_failure
56  @post __has_overflow == true -> __has_assertion_failure == true
57  @post __reverted == false -> __return == a - b
58  @post msg == msg__post
59  @post (a < b) == __has_assertion_failure
60  @post __addr_map == __addr_map__post
61  */
62  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63      assert(b <= a);
64      return a - b;
65  }
66
67  /**
68   * @dev Adds two numbers, throws on overflow.
69   */
70
71  /*@CTK SafeMath_add
72   @tag spec
73   @post __reverted == __has_assertion_failure
74   @post __has_assertion_failure == __has_overflow
75   @post __reverted == false -> __return == a + b
76   @post msg == msg__post
77   @post (a + b < a) == __has_assertion_failure
78   @post __addr_map == __addr_map__post
79   */
80  function add(uint256 a, uint256 b) internal pure returns (uint256) {
81      uint256 c = a + b;
82      assert(c >= a);
83      return c;
84  }
85 }
86
87 contract ERC20Basic {
88     function totalSupply() public view returns (uint256);
89     function balanceOf(address who) public view returns (uint256);
90     function transfer(address to, uint256 value) public returns (bool);
91     event Transfer(address indexed from, address indexed to, uint256 value);
92 }
93
94 contract BasicToken is ERC20Basic {
95     using SafeMath for uint256;
96
97     mapping(address => uint256) balances;
98
99     uint256 totalSupply_;
100
101     /**
102     * @dev total number of tokens in existence
103     */
104
105     /*@CTK TotalSupply_return
106     @tag spec
107     @post __reverted == __has_assertion_failure
108     @post __has_assertion_failure == __has_overflow
109     @post __reverted == false -> __return == totalSupply_
110     @post msg == msg__post
111     */

```

```

112 function totalSupply() public view returns (uint256) {
113     return totalSupply_;
114 }
115
116 /**
117  * @dev transfer token for a specified address
118  * @param _to The address to transfer to.
119  * @param _value The amount to be transferred.
120  */
121
122
123 /*@CTK transfer_failure_addressEqualZero
124    @pre _to == address(0)
125    @pre balances[msg.sender] >= _value
126    @post __reverted == true
127 */
128 /*@CTK transfer_failure_balanceSmallerValue
129    @pre _to != address(0)
130    @pre balances[msg.sender] < _value
131    @post __reverted == true
132 */
133 /*@CTK transfer_conditions
134    @tag assume_completion
135    @pre _to != msg.sender
136    @post __post.balances[_to] == balances[_to] + _value
137    @post __post.balances[msg.sender] == balances[msg.sender] - _value
138 */
139 /*@CTK transfer_same_address
140    @tag assume_completion
141    @tag no_overflow
142    @pre _to == msg.sender
143    @post this == __post
144 */
145 function transfer(address _to, uint256 _value) public returns (bool) {
146     require(_to != address(0));
147     require(_value <= balances[msg.sender]);
148
149     // SafeMath.sub will throw if there is not enough balance.
150     balances[msg.sender] = balances[msg.sender].sub(_value);
151     balances[_to] = balances[_to].add(_value);
152     emit Transfer(msg.sender, _to, _value);
153     return true;
154 }
155
156 /**
157  * @dev Gets the balance of the specified address.
158  * @param _owner The address to query the the balance of.
159  * @return An uint256 representing the amount owned by the passed address.
160  */
161
162 /*@CTK balanceOf
163    @post __reverted == false
164    @post balance == balances[_owner]
165 */
166 function balanceOf(address _owner) public view returns (uint256 balance) {
167     return balances[_owner];
168 }
169

```



```

170 }
171
172 contract ERC20 is ERC20Basic {
173     function allowance(address owner, address spender) public view returns (uint256);
174     function transferFrom(address from, address to, uint256 value) public returns (bool)
175     ;
176     function approve(address spender, uint256 value) public returns (bool);
177     event Approval(address indexed owner, address indexed spender, uint256 value);
178 }
179
180 contract StandardToken is ERC20, BasicToken {
181     mapping (address => mapping (address => uint256)) internal allowed;
182     event Burn(address indexed burner, uint256 value);
183
184
185     /**
186      * @dev Transfer tokens from one address to another
187      * @param _from address The address which you want to send tokens from
188      * @param _to address The address which you want to transfer to
189      * @param _value uint256 the amount of tokens to be transferred
190      */
191
192     /*@CTK transferFrom
193      @tag assume_completion
194      @pre _from != _to
195      @post __return == true
196      @post __post.balances[_to] == balances[_to] + _value
197      @post __post.balances[_from] == balances[_from] - _value
198      @post __has_overflow == false
199     */
200     /*@CTK FAIL "transferFrom_sameOwner"
201      @pre _from == _to
202      @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
203     */
204     function transferFrom(address _from, address _to, uint256 _value) public returns (
205         bool) {
206         require(_to != address(0));
207         require(_value <= balances[_from]);
208         require(_value <= allowed[_from][msg.sender]);
209
210         balances[_from] = balances[_from].sub(_value);
211         balances[_to] = balances[_to].add(_value);
212         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
213         emit Transfer(_from, _to, _value);
214         return true;
215     }
216
217     /**
218      * @dev Approve the passed address to spend the specified amount of tokens on behalf
219      *       of msg.sender.
220      *
221      * Beware that changing an allowance with this method brings the risk that someone
222      * may use both the old
223      * and the new allowance by unfortunate transaction ordering. One possible solution
224      * to mitigate this
225      * race condition is to first reduce the spender's allowance to 0 and set the
226      * desired value afterwards:

```

```

222 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
223 * @param _spender The address which will spend the funds.
224 * @param _value The amount of tokens to be spent.
225 */
226
227 /*@CTK approve_success
228   @post _value == 0 -> __reverted == false
229   @post allowed[msg.sender][_spender] == 0 -> __reverted == false
230 */
231 /*@CTK approve
232   @tag assume_completion
233   @post __post.allowed[msg.sender][_spender] == _value
234 */
235 function approve(address _spender, uint256 _value) public returns (bool) {
236     allowed[msg.sender][_spender] = _value;
237     emit Approval(msg.sender, _spender, _value);
238     return true;
239 }
240
241 /**
242  * @dev Function to check the amount of tokens that an owner allowed to a spender.
243  * @param _owner address The address which owns the funds.
244  * @param _spender address The address which will spend the funds.
245  * @return A uint256 specifying the amount of tokens still available for the spender
246  */
247
248 /*@CTK get_allowance
249   @post __reverted == false
250   @post __return == allowed[_owner][_spender]
251   @post this == __post
252 */
253 function allowance(address _owner, address _spender) public view returns (uint256) {
254     return allowed[_owner][_spender];
255 }
256
257 /**
258  * @dev Increase the amount of tokens that an owner allowed to a spender.
259  *
260  * approve should be called when allowed[_spender] == 0. To increment
261  * allowed value is better to use this function to avoid 2 calls (and wait until
262  * the first transaction is mined)
263  * From MonolithDAO Token.sol
264  * @param _spender The address which will spend the funds.
265  * @param _addedValue The amount of tokens to increase the allowance by.
266  */
267
268 /*@CTK increaseApproval
269   @tag assume_completion
270   @post __post.allowed[msg.sender][_spender] ==
271         allowed[msg.sender][_spender] + _addedValue
272   @post __post.allowed[msg.sender][_spender] ==
273         allowed[msg.sender][_spender] + _addedValue -> __return == true
274 */
275 function increaseApproval(address _spender, uint _addedValue) public returns (bool)
276 {
277     allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
278     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);

```

```

278     return true;
279 }
280
281 /**
282  * @dev Decrease the amount of tokens that an owner allowed to a spender.
283  *
284  * approve should be called when allowed[_spender] == 0. To decrement
285  * allowed value is better to use this function to avoid 2 calls (and wait until
286  * the first transaction is mined)
287  * From MonolithDAO Token.sol
288  * @param _spender The address which will spend the funds.
289  * @param _subtractedValue The amount of tokens to decrease the allowance by.
290  */
291
292 /*@CTK decreaseApproval0
293   @pre __return == true
294   @pre allowed[msg.sender][_spender] <= _subtractedValue
295   @post __post.allowed[msg.sender][_spender] == 0
296 */
297 /*@CTK decreaseApproval
298   @pre __return == true
299   @pre allowed[msg.sender][_spender] > _subtractedValue
300   @post __post.allowed[msg.sender][_spender] ==
301         allowed[msg.sender][_spender] - _subtractedValue
302 */
303 function decreaseApproval(address _spender, uint _subtractedValue) public returns (
304     bool) {
305     uint oldValue = allowed[msg.sender][_spender];
306     if (_subtractedValue > oldValue) {
307         allowed[msg.sender][_spender] = 0;
308     } else {
309         allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
310     }
311     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
312     return true;
313 }
314
315 /**
316  * @dev Burns a specific amount of tokens.
317  * @param _value The amount of token to be burned.
318  */
319
320 /*@CTK burn_success
321   @tag assume_completion
322   @post __post.balances[msg.sender] == balances[msg.sender] - _value
323   @post __post.totalSupply_ == totalSupply_ - _value
324 */
325 /*@CTK burn_failure
326   @pre _value > balances[msg.sender]
327   @post __reverted == true
328 */
329
330 function burn(uint256 _value) public {
331     _burn(msg.sender, _value);
332 }
333
334

```

```

335  /*@CTK _burn_failure
336      @pre _value > balances[_who]
337      @post __reverted == true
338  */
339  /*@CTK _burn_success
340      @tag assume_completion
341      @post __post.balances[_who] == balances[_who] - _value
342      @post __post.totalSupply_ == totalSupply_ - _value
343  */
344  function _burn(address _who, uint256 _value) internal {
345      require(_value <= balances[_who]);
346      // no need to require value <= totalSupply, since that would imply the
347      // sender's balance is greater than the totalSupply, which *should* be an
348      // assertion failure
349      balances[_who] = balances[_who].sub(_value);
350      totalSupply_ = totalSupply_.sub(_value);
351      emit Burn(_who, _value);
352      emit Transfer(_who, address(0), _value);
353  }
354
355 }
356
357 contract BlockcloudToken is StandardToken {
358     string public name = "Blockcloud Token";
359     string public symbol = "BLOC";
360     uint8 public decimals = 18;
361
362
363     // ten billion in initial supply
364     uint256 public constant INITIAL_SUPPLY = 10000000000;
365
366
367     /*@CTK BlockcloudToken_initial
368         @pre __reverted == false
369         @post __post.balances[msg.sender] == __post.totalSupply_
370     */
371     function BlockcloudToken() public {
372         totalSupply_ = INITIAL_SUPPLY * (10 ** uint256(decimals));
373         balances[msg.sender] = totalSupply_;
374     }
375
376
377 }

```

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	56	
	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Static Analysis Request

Formal Verification Request 1

SafeMath_mul

 16, Apr 2019

 390.44 ms

Line 9-18 in File BlockcloudToken.sol

```
9  /*@CTK SafeMath_mul
10  @tag spec
11  @post __reverted == __has_assertion_failure
12  @post __has_assertion_failure == __has_overflow
13  @post __reverted == false -> __return == a * b
14  @post a == 0 -> __return == 0
15  @post msg == msg__post
16  @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
17  @post __addr_map == __addr_map__post
18  */
```

Line 19-26 in File BlockcloudToken.sol


```
19  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20      if (a == 0) {
21          return 0;
22      }
23      uint256 c = a * b;
24      assert(c / a == b);
25      return c;
26  }
```

 The code meets the specification

Formal Verification Request 2

SafeMath_div

 16, Apr 2019

 8.9 ms

Line 32-40 in File BlockcloudToken.sol

```
32  /*@CTK SafeMath_div
33  @tag spec
34  @post __reverted == __has_assertion_failure
35  @post b == 0 -> __reverted == true // solidity throws on 0.
36  @post __has_overflow == true -> __has_assertion_failure == true
37  @post __reverted == false -> __return == a / b
38  @post msg == msg__post
39  @post __addr_map == __addr_map__post
40  */
```

Line 41-46 in File BlockcloudToken.sol

```
41  function div(uint256 a, uint256 b) internal pure returns (uint256) {
42      // assert(b > 0); // Solidity automatically throws when dividing by 0
43      uint256 c = a / b;
```



```
44 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45 return c;
46 }
```

✓ The code meets the specification

Formal Verification Request 3

SafeMath_sub

📅 16, Apr 2019

🕒 15.5 ms

Line 53-61 in File BlockcloudToken.sol

```
53 /*@CTK SafeMath_sub
54 @tag spec
55 @post __reverted == __has_assertion_failure
56 @post __has_overflow == true -> __has_assertion_failure == true
57 @post __reverted == false -> __return == a - b
58 @post msg == msg__post
59 @post (a < b) == __has_assertion_failure
60 @post __addr_map == __addr_map__post
61 */
```

Line 62-65 in File BlockcloudToken.sol

```
62 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63     assert(b <= a);
64     return a - b;
65 }
```

✓ The code meets the specification

Formal Verification Request 4

SafeMath_add

📅 16, Apr 2019

🕒 19.35 ms

Line 71-79 in File BlockcloudToken.sol

```
71 /*@CTK SafeMath_add
72 @tag spec
73 @post __reverted == __has_assertion_failure
74 @post __has_assertion_failure == __has_overflow
75 @post __reverted == false -> __return == a + b
76 @post msg == msg__post
77 @post (a + b < a) == __has_assertion_failure
78 @post __addr_map == __addr_map__post
79 */
```

Line 80-84 in File BlockcloudToken.sol



```
80 function add(uint256 a, uint256 b) internal pure returns (uint256) {
81     uint256 c = a + b;
82     assert(c >= a);
83     return c;
84 }
```

✓ The code meets the specification

Formal Verification Request 5

TotalSupply_return

📅 16, Apr 2019

🕒 6.17 ms

Line 105-111 in File BlockcloudToken.sol

```
105 /*@CTK TotalSupply_return
106     @tag spec
107     @post __reverted == __has_assertion_failure
108     @post __has_assertion_failure == __has_overflow
109     @post __reverted == false -> __return == totalSupply_
110     @post msg == msg__post
111 */
```

Line 112-114 in File BlockcloudToken.sol

```
112 function totalSupply() public view returns (uint256) {
113     return totalSupply_;
114 }
```

✓ The code meets the specification

Formal Verification Request 6

transfer_failure_addressEqualZero

📅 16, Apr 2019

🕒 45.91 ms

Line 123-127 in File BlockcloudToken.sol

```
123 /*@CTK transfer_failure_addressEqualZero
124     @pre _to == address(0)
125     @pre balances[msg.sender] >= _value
126     @post __reverted == true
127 */
```

Line 145-154 in File BlockcloudToken.sol

```
145 function transfer(address _to, uint256 _value) public returns (bool) {
146     require(_to != address(0));
147     require(_value <= balances[msg.sender]);
148
149     // SafeMath.sub will throw if there is not enough balance.
```



```

150     balances[msg.sender] = balances[msg.sender].sub(_value);
151     balances[_to] = balances[_to].add(_value);
152     emit Transfer(msg.sender, _to, _value);
153     return true;
154 }

```

✓ The code meets the specification

Formal Verification Request 7

transfer_failure_balanceSmallerValue

16, Apr 2019

3.52 ms

Line 128-132 in File BlockcloudToken.sol

```

128     /*@CTK transfer_failure_balanceSmallerValue
129         @pre _to != address(0)
130         @pre balances[msg.sender] < _value
131         @post __reverted == true
132     */

```

Line 145-154 in File BlockcloudToken.sol

```

145     function transfer(address _to, uint256 _value) public returns (bool) {
146         require(_to != address(0));
147         require(_value <= balances[msg.sender]);
148
149         // SafeMath.sub will throw if there is not enough balance.
150         balances[msg.sender] = balances[msg.sender].sub(_value);
151         balances[_to] = balances[_to].add(_value);
152         emit Transfer(msg.sender, _to, _value);
153         return true;
154     }

```

✓ The code meets the specification

Formal Verification Request 8

transfer_conditions

16, Apr 2019

106.08 ms

Line 133-138 in File BlockcloudToken.sol

```

133     /*@CTK transfer_conditions
134         @tag assume_completion
135         @pre _to != msg.sender
136         @post __post.balances[_to] == balances[_to] + _value
137         @post __post.balances[msg.sender] == balances[msg.sender] - _value
138     */

```

Line 145-154 in File BlockcloudToken.sol

```

145 function transfer(address _to, uint256 _value) public returns (bool) {
146     require(_to != address(0));
147     require(_value <= balances[msg.sender]);
148
149     // SafeMath.sub will throw if there is not enough balance.
150     balances[msg.sender] = balances[msg.sender].sub(_value);
151     balances[_to] = balances[_to].add(_value);
152     emit Transfer(msg.sender, _to, _value);
153     return true;
154 }


```

✓ The code meets the specification

Formal Verification Request 9

transfer_same_address

 16, Apr 2019

 11.32 ms

Line 139-144 in File BlockcloudToken.sol

```

139 /*@CTK transfer_same_address
140     @tag assume_completion
141     @tag no_overflow
142     @pre _to == msg.sender
143     @post this == __post
144 */

```

Line 145-154 in File BlockcloudToken.sol

```

145 function transfer(address _to, uint256 _value) public returns (bool) {
146     require(_to != address(0));
147     require(_value <= balances[msg.sender]);
148
149     // SafeMath.sub will throw if there is not enough balance.
150     balances[msg.sender] = balances[msg.sender].sub(_value);
151     balances[_to] = balances[_to].add(_value);
152     emit Transfer(msg.sender, _to, _value);
153     return true;
154 }


```

✓ The code meets the specification

Formal Verification Request 10

balanceOf

 16, Apr 2019

 7.26 ms

Line 162-165 in File BlockcloudToken.sol



```
162  /*@CTK balanceOf
163      @post __reverted == false
164      @post balance == balances[_owner]
165  */
```

Line 166-168 in File BlockcloudToken.sol

```
166  function balanceOf(address _owner) public view returns (uint256 balance) {
167      return balances[_owner];
168  }
```

✓ The code meets the specification

Formal Verification Request 11

transferFrom

16, Apr 2019

239.63 ms

Line 192-199 in File BlockcloudToken.sol

```
192  /*@CTK transferFrom
193      @tag assume_completion
194      @pre _from != _to
195      @post __return == true
196      @post __post.balances[_to] == balances[_to] + _value
197      @post __post.balances[_from] == balances[_from] - _value
198      @post __has_overflow == false
199  */
```

Line 204-214 in File BlockcloudToken.sol

```
204  function transferFrom(address _from, address _to, uint256 _value) public returns (
205      bool) {
206      require(_to != address(0));
207      require(_value <= balances[_from]);
208      require(_value <= allowed[_from][msg.sender]);
209
210      balances[_from] = balances[_from].sub(_value);
211      balances[_to] = balances[_to].add(_value);
212      allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
213      emit Transfer(_from, _to, _value);
214      return true;
215  }
```

✓ The code meets the specification

Formal Verification Request 12

transferFrom_sameOwner

16, Apr 2019

68.16 ms

Line 200-203 in File BlockcloudToken.sol

```

200  /*@CTK FAIL "transferFrom_sameOwner"
201     @pre _from == _to
202     @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
203  */

```

Line 204-214 in File BlockcloudToken.sol

```

204  function transferFrom(address _from, address _to, uint256 _value) public returns (
205      bool) {
206      require(_to != address(0));
207      require(_value <= balances[_from]);
208      require(_value <= allowed[_from][msg.sender]);
209
210      balances[_from] = balances[_from].sub(_value);
211      balances[_to] = balances[_to].add(_value);
212      allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
213      emit Transfer(_from, _to, _value);
214      return true;
215  }

```

✖ This code violates the specification

```

1  Counter Example:
2  Before Execution:
3      Input = {
4          _from = 64
5          _to = 64
6          _value = 127
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         __return = false
23         block = {
24             "number": 0,
25             "timestamp": 0
26         }
27     }
28     Address_Map = [
29         {
30             "key": 0,
31             "value": {
32                 "contract_name": "StandardToken",
33                 "balance": 0,
34                 "contract": {
35                     "allowed": [
36                         {
37                             "key": 64,
38                             "value": [

```

```

39         {
40             "key": 4,
41             "value": 0
42         },
43         {
44             "key": 8,
45             "value": 16
46         },
47         {
48             "key": 0,
49             "value": 128
50         },
51         {
52             "key": "ALL_OTHERS",
53             "value": 129
54         }
55     ],
56 },
57 {
58     "key": "ALL_OTHERS",
59     "value": [
60         {
61             "key": "ALL_OTHERS",
62             "value": 129
63         }
64     ]
65 }
66 ],
67 "balances": [
68     {
69         "key": 4,
70         "value": 1
71     },
72     {
73         "key": 8,
74         "value": 0
75     },
76     {
77         "key": 0,
78         "value": 0
79     },
80     {
81         "key": 64,
82         "value": 128
83     },
84     {
85         "key": "ALL_OTHERS",
86         "value": 129
87     }
88 ],
89 "totalSupply_": 0
90 }
91 }
92 },
93 {
94     "key": "ALL_OTHERS",
95     "value": "EmptyAddress"
96 }

```

```

97 ]
98
99 After Execution:
100 Input = {
101     _from = 64
102     _to = 64
103     _value = 127
104 }
105 This = 0
106 Internal = {
107     __has_assertion_failure = false
108     __has_buf_overflow = true
109     __has_overflow = false
110     __has_returned = true
111     __reverted = false
112     msg = {
113         "gas": 0,
114         "sender": 0,
115         "value": 0
116     }
117 }
118 Other = {
119     __return = true
120     block = {
121         "number": 0,
122         "timestamp": 0
123     }
124 }
125 Address_Map = [
126     {
127         "key": 0,
128         "value": {
129             "contract_name": "StandardToken",
130             "balance": 0,
131             "contract": {
132                 "allowed": [
133                     {
134                         "key": 64,
135                         "value": [
136                             {
137                                 "key": 4,
138                                 "value": 0
139                             },
140                             {
141                                 "key": 8,
142                                 "value": 16
143                             },
144                             {
145                                 "key": 0,
146                                 "value": 1
147                             },
148                             {
149                                 "key": "ALL_OTHERS",
150                                 "value": 129
151                             }
152                         ]
153                     },
154                     {

```

```

155         "key": "ALL_OTHERS",
156         "value": [
157             {
158                 "key": "ALL_OTHERS",
159                 "value": 129
160             }
161         ]
162     },
163 ],
164     "balances": [
165         {
166             "key": 4,
167             "value": 1
168         },
169         {
170             "key": 8,
171             "value": 0
172         },
173         {
174             "key": 0,
175             "value": 0
176         },
177         {
178             "key": 64,
179             "value": 128
180         },
181         {
182             "key": "ALL_OTHERS",
183             "value": 129
184         }
185     ],
186     "totalSupply_": 0
187 }
188 }
189 },
190 {
191     "key": "ALL_OTHERS",
192     "value": "EmptyAddress"
193 }
194 ]

```

Formal Verification Request 13

approve_success



16, Apr 2019



14.05 ms

Line 227-230 in File BlockcloudToken.sol

```

227  /*@CTK approve_success
228     @post _value == 0 -> __reverted == false
229     @post allowed[msg.sender][_spender] == 0 -> __reverted == false
230  */

```

Line 235-239 in File BlockcloudToken.sol



```
235 function approve(address _spender, uint256 _value) public returns (bool) {
236     allowed[msg.sender][_spender] = _value;
237     emit Approval(msg.sender, _spender, _value);
238     return true;
239 }
```

✓ The code meets the specification

Formal Verification Request 14

approve

📅 16, Apr 2019

🕒 1.9 ms

Line 231-234 in File BlockcloudToken.sol

```
231 /*@CTK approve
232     @tag assume_completion
233     @post __post.allowed[msg.sender][_spender] == _value
234 */
```

Line 235-239 in File BlockcloudToken.sol

```
235 function approve(address _spender, uint256 _value) public returns (bool) {
236     allowed[msg.sender][_spender] = _value;
237     emit Approval(msg.sender, _spender, _value);
238     return true;
239 }
```

✓ The code meets the specification

Formal Verification Request 15

get_allowance

📅 16, Apr 2019

🕒 7.48 ms

Line 248-252 in File BlockcloudToken.sol

```
248 /*@CTK get_allowance
249     @post __reverted == false
250     @post __return == allowed[_owner][_spender]
251     @post this == __post
252 */
```

Line 253-255 in File BlockcloudToken.sol


```
253 function allowance(address _owner, address _spender) public view returns (uint256) {
254     return allowed[_owner][_spender];
255 }
```

✓ The code meets the specification

Formal Verification Request 16

increaseApproval

 16, Apr 2019

 28.8 ms

Line 268-274 in File BlockcloudToken.sol

```
268  /*@CTK increaseApproval
269    @tag assume_completion
270    @post __post.allowed[msg.sender][_spender] ==
271          allowed[msg.sender][_spender] + _addedValue
272    @post __post.allowed[msg.sender][_spender] ==
273          allowed[msg.sender][_spender] + _addedValue -> __return == true
274  */
```

Line 275-279 in File BlockcloudToken.sol


```
275  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
276  {
277    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
278    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
279    return true;
279  }
```

 The code meets the specification

Formal Verification Request 17

decreaseApproval0

 16, Apr 2019

 78.06 ms

Line 292-296 in File BlockcloudToken.sol

```
292  /*@CTK decreaseApproval0
293    @pre __return == true
294    @pre allowed[msg.sender][_spender] <= _subtractedValue
295    @post __post.allowed[msg.sender][_spender] == 0
296  */
```

Line 303-312 in File BlockcloudToken.sol


```
303  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
304    bool) {
305    uint oldValue = allowed[msg.sender][_spender];
306    if (_subtractedValue > oldValue) {
307      allowed[msg.sender][_spender] = 0;
308    } else {
309      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
310    }
311    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
312    return true;
312  }
```

 The code meets the specification

Formal Verification Request 18

decreaseApproval

 16, Apr 2019

 18.73 ms

Line 297-302 in File BlockcloudToken.sol

```
297  /*@CTK decreaseApproval
298      @pre __return == true
299      @pre allowed[msg.sender][_spender] > _subtractedValue
300      @post __post.allowed[msg.sender][_spender] ==
301            allowed[msg.sender][_spender] - _subtractedValue
302  */
```

Line 303-312 in File BlockcloudToken.sol


```
303  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
304      bool) {
305      uint oldValue = allowed[msg.sender][_spender];
306      if (_subtractedValue > oldValue) {
307          allowed[msg.sender][_spender] = 0;
308      } else {
309          allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
310      }
311      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
312      return true;
313  }
```

 The code meets the specification

Formal Verification Request 19

burn_success

 16, Apr 2019

 119.13 ms

Line 320-324 in File BlockcloudToken.sol

```
320  /*@CTK burn_success
321      @tag assume_completion
322      @post __post.balances[msg.sender] == balances[msg.sender] - _value
323      @post __post.totalSupply_ == totalSupply_ - _value
324  */
```

Line 330-332 in File BlockcloudToken.sol


```
330  function burn(uint256 _value) public {
331      _burn(msg.sender, _value);
332  }
```

 The code meets the specification

Formal Verification Request 20

burn_failure

 16, Apr 2019

 2.62 ms

Line 325-328 in File BlockcloudToken.sol

```
325  /*@CTK burn_failure
326     @pre _value > balances[msg.sender]
327     @post __reverted == true
328  */
```

Line 330-332 in File BlockcloudToken.sol


```
330  function burn(uint256 _value) public {
331      _burn(msg.sender, _value);
332  }
```

 The code meets the specification

Formal Verification Request 21

_burn_failure

 16, Apr 2019

 2.27 ms

Line 335-338 in File BlockcloudToken.sol

```
335  /*@CTK _burn_failure
336     @pre _value > balances[_who]
337     @post __reverted == true
338  */
```

Line 344-353 in File BlockcloudToken.sol


```
344  function _burn(address _who, uint256 _value) internal {
345      require(_value <= balances[_who]);
346      // no need to require value <= totalSupply, since that would imply the
347      // sender's balance is greater than the totalSupply, which *should* be an
348      // assertion failure
349      balances[_who] = balances[_who].sub(_value);
350      totalSupply_ = totalSupply_.sub(_value);
351      emit Burn(_who, _value);
352      emit Transfer(_who, address(0), _value);
353  }
```

 The code meets the specification

Formal Verification Request 22

_burn_success

 16, Apr 2019

 47.45 ms

Line 339-343 in File BlockcloudToken.sol

```
339  /*@CTK _burn_success
340      @tag assume_completion
341      @post __post.balances[_who] == balances[_who] - _value
342      @post __post.totalSupply_ == totalSupply_ - _value
343  */
```

Line 344-353 in File BlockcloudToken.sol


```
344  function _burn(address _who, uint256 _value) internal {
345      require(_value <= balances[_who]);
346      // no need to require value <= totalSupply, since that would imply the
347      // sender's balance is greater than the totalSupply, which *should* be an
348      // assertion failure
349      balances[_who] = balances[_who].sub(_value);
350      totalSupply_ = totalSupply_.sub(_value);
351      emit Burn(_who, _value);
352      emit Transfer(_who, address(0), _value);
353  }
```

 The code meets the specification

Formal Verification Request 23

BlockcloudToken_initial

 16, Apr 2019

 27.64 ms

Line 367-370 in File BlockcloudToken.sol

```
367  /*@CTK BlockcloudToken_initial
368      @pre __reverted == false
369      @post __post.balances[msg.sender] == __post.totalSupply_
370  */
```

Line 371-374 in File BlockcloudToken.sol

```
371  function BlockcloudToken() public {
372      totalSupply_ = INITIAL_SUPPLY * (10 ** uint256(decimals));
373      balances[msg.sender] = totalSupply_;
374  }
```

 The code meets the specification