

# CERTIK AUDIT REPORT FOR FIRMACHAIN



Request Date: 2019-06-05  
Revision Date: 2019-06-18  
Platform Name: Ethereum



# Contents

<b>Disclaimer</b>	<b>1</b>
<b>About CertiK</b>	<b>2</b>
<b>Exective Summary</b>	<b>3</b>
<b>Vulnerability Classification</b>	<b>3</b>
<b>Testing Summary</b>	<b>4</b>
Audit Score . . . . .	4
Type of Issues . . . . .	4
Vulnerability Details . . . . .	5
<b>Manual Review Notes</b>	<b>6</b>
<b>Static Analysis Results</b>	<b>7</b>
<b>Formal Verification Results</b>	<b>8</b>
How to read . . . . .	8
<b>Source Code with CertiK Labels</b>	<b>61</b>

## 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 FirmaChain(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 FirmaChain. This audit was conducted to discover issues and vulnerabilities in the source code of FirmaChain'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.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

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

## Vulnerability Details

### Critical

No issue found.

### Medium

No issue found.

### Low

MultiOwnable:

- Missing address check `require(msg.sender != newOwner, ...)` in function `function transferOwnership(address newOwner)`, which may lead to possible ownership loss.

SafeMath:

- Recommend switching `assert` statements to `require` as in the latest OpenZeppelin SafeMath library.

(Note: The violations in the formal verification result section of the report is for internal evaluation and do not reflect security issue in the user contracts.)

# Manual Review Notes

## Review Details

### Source Code SHA-256 Checksum

- **FCTV.sol**

a4ba4caf3d0df81e1f9fbee6455fd31a9f324d46c9f89d4100ebc456cd92682f

### Summary

CertiK was chosen by FirmaChain to audit the design and implementation of its FCTV smart contracts. 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 FCTV.sol

```
1 pragma solidity ^0.4.23;
```



! Version to compile has the following bug: 0.4.23: DynamicConstructorArgumentsClipped-ABIv2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIv2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClipped-ABIv2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClipped-ABIv2

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

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

Raw code	<pre> 35      function transferFrom(address from, address to 36         ) { 37         balances[from] = balances[from].sub(tokens 38         allowed[from] [msg.sender] = allowed[from] [ 39         balances[to] = balances[to].add(tokens); 40         emit Transfer(from, to, tokens); 41         return true; 42     } </pre>
----------	---


Counterexample	 This code violates the specification
----------------	--

Initial environment	<pre> 1 Counter Example: 2 Before Execution: 3   Input = { 4     from = 0x0 5     to = 0x0 6     tokens = 0x6c 7   } 8   This = 0 </pre>
Post environment	<pre> 52   } 53   balance: 0x0 54 } 55 } 56 57 After Execution: 58   Input = { 59     from = 0x0 60     to = 0x0 61     tokens = 0x6c </pre>

## Formal Verification Request 1

Method will not encounter an assertion failure.

 18, Jun 2019

 46.0 ms

Line 12 in File FCTV.sol

```
12 // @CTK FAIL NO_ASF
```

Line 20-31 in File FCTV.sol

```
20 function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
21     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
22     // benefit is lost if 'b' is also tested.
23     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
24     if (a == 0) {
25         return 0;
26     }
27
28     c = a * b;
29     assert(c / a == b);
30     return c;
31 }
```

 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

 289.27 ms

Line 13-19 in File FCTV.sol

```
13  /*@CTK "SafeMath mul"
14     @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
15     @post !__reverted -> c == a * b
16     @post !__reverted == !__has_overflow
17     @post !__reverted -> !(__has_assertion_failure)
18     @post !(__has_buf_overflow)
19  */
```

Line 20-31 in File FCTV.sol


```
20  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
21     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
22     // benefit is lost if 'b' is also tested.
23     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
24     if (a == 0) {
25         return 0;
26     }
27
28     c = a * b;
29     assert(c / a == b);
30     return c;
31 }
```

 The code meets the specification.

## Formal Verification Request 3

Method will not encounter an assertion failure.

 18, Jun 2019

 6.22 ms

Line 36 in File FCTV.sol

```
36  //@CTK FAIL NO_ASF
```

Line 44-49 in File FCTV.sol

```
44  function div(uint256 a, uint256 b) internal pure returns (uint256) {
45     // assert(b > 0); // Solidity automatically throws when dividing by 0
46     // uint256 c = a / b;
47     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
48     return a / b;
49 }
```

 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

 0.33 ms

Line 37-43 in File FCTV.sol

```

37 /*@CTK "SafeMath div"
38   @post b != 0 -> !__reverted
39   @post !__reverted -> __return == a / b
40   @post !__reverted -> !__has_overflow
41   @post !__reverted -> !(__has_assertion_failure)
42   @post !(__has_buf_overflow)
43 */

```

Line 44-49 in File FCTV.sol

```

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

```

49 }  
}

✔ The code meets the specification.

## Formal Verification Request 5

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 10.54 ms

Line 54 in File FCTV.sol

54 //OCTK FAIL NO\_ASF

Line 62-65 in File FCTV.sol


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


✘ 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 55-61 in File FCTV.sol

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

Line 62-65 in File FCTV.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 7

Method will not encounter an assertion failure.

 18, Jun 2019

 12.47 ms

Line 70 in File FCTV.sol

```
70  //@CTK FAIL_NO_ASF
```

Line 78-82 in File FCTV.sol

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

 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

 2.52 ms

Line 71-77 in File FCTV.sol

```

71  /*@CTK "SafeMath add"
72     @post (a + b < a || a + b < b) == __reverted
73     @post !__reverted -> c == a + b
74     @post !__reverted -> !__has_overflow
75     @post !__reverted -> !(__has_assertion_failure)
76     @post !(__has_buf_overflow)
77  */

```

Line 78-82 in File FCTV.sol

```

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


```

 The code meets the specification.

## Formal Verification Request 9

If method completes, integer overflow would not happen.

 18, Jun 2019

 4.66 ms

Line 111 in File FCTV.sol



```
111 // @CTK_NO_OVERFLOW
```

Line 118-120 in File FCTV.sol

```
118 function totalSupply() public view returns (uint256) {  
119     return totalSupply_;  
120 }
```

✔ The code meets the specification.

## Formal Verification Request 10

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

📅 18, Jun 2019

🕒 0.37 ms

Line 112 in File FCTV.sol

```
112 // @CTK_NO_BUF_OVERFLOW
```

Line 118-120 in File FCTV.sol

```
118 function totalSupply() public view returns (uint256) {  
119     return totalSupply_;  
120 }
```

✔ The code meets the specification.

## Formal Verification Request 11

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.34 ms

Line 113 in File FCTV.sol

```
113 // @CTK_NO_ASF
```

Line 118-120 in File FCTV.sol

```
118 function totalSupply() public view returns (uint256) {  
119     return totalSupply_;  
120 }
```

✔ The code meets the specification.

## Formal Verification Request 12

totalSupply

📅 18, Jun 2019

🕒 0.32 ms

Line 114-117 in File FCTV.sol

```

114  /*@CTK totalSupply
115     @tag assume_completion
116     @post (__return) == (totalSupply_)
117  */

```

Line 118-120 in File FCTV.sol

```

118  function totalSupply() public view returns (uint256) {
119     return totalSupply_;
120  }

```

✔ The code meets the specification.

## Formal Verification Request 13

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 77.21 ms

Line 127 in File FCTV.sol

```

127  //@CTK NO_OVERFLOW

```

Line 140-148 in File FCTV.sol

```

140  function transfer(address _to, uint256 _value) public returns (bool) {
141     require(_to != address(0));
142     require(_value <= balances[msg.sender]);
143
144     balances[msg.sender] = balances[msg.sender].sub(_value);
145     balances[_to] = balances[_to].add(_value);
146     emit Transfer(msg.sender, _to, _value);
147     return true;
148  }

```

✔ The code meets the specification.

## Formal Verification Request 14

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

📅 18, Jun 2019

🕒 18.18 ms

Line 128 in File FCTV.sol

```

128  //@CTK NO_BUF_OVERFLOW

```

Line 140-148 in File FCTV.sol

```

140  function transfer(address _to, uint256 _value) public returns (bool) {
141     require(_to != address(0));
142     require(_value <= balances[msg.sender]);
143
144     balances[msg.sender] = balances[msg.sender].sub(_value);
145     balances[_to] = balances[_to].add(_value);
146     emit Transfer(msg.sender, _to, _value);

```


```
147     return true;
148 }
```

✔ The code meets the specification.

## Formal Verification Request 15

Method will not encounter an assertion failure.

 18, Jun 2019

 43.87 ms

Line 129 in File FCTV.sol

```
129 //@CTK FAIL NO_ASF
```

Line 140-148 in File FCTV.sol

```
140 function transfer(address _to, uint256 _value) public returns (bool) {
141     require(_to != address(0));
142     require(_value <= balances[msg.sender]);
143
144     balances[msg.sender] = balances[msg.sender].sub(_value);
145     balances[_to] = balances[_to].add(_value);
146     emit Transfer(msg.sender, _to, _value);
147     return true;
148 }
```

✘ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     _to = 128
5     _value = 1
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": "BasicToken",
32         "balance": 0,
33         "contract": {
34             "balances": [
35                 {
36                     "key": 128,
37                     "value": 255
38                 },
39                 {
40                     "key": 0,
41                     "value": 128
42                 },
43                 {
44                     "key": "ALL_OTHERS",
45                     "value": 1
46                 }
47             ],
48             "totalSupply_": 0
49         }
50     },
51 },
52 {
53     "key": "ALL_OTHERS",
54     "value": "EmptyAddress"
55 }
56 ]
57
58 Function invocation is reverted.

```

## Formal Verification Request 16

transfer

 18, Jun 2019

 128.83 ms

Line 130-139 in File FCTV.sol

```

130  /*@CTK transfer
131     @tag assume_completion
132     @pre _to != address(0)
133     @pre _value <= balances[msg.sender]
134     @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
135     @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
136         - _value)
137     @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
138     @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
139     @post __return == true
140 */

```

Line 140-148 in File FCTV.sol

```

140  function transfer(address _to, uint256 _value) public returns (bool) {
141      require(_to != address(0));
142      require(_value <= balances[msg.sender]);
143  }

```

```

144     balances[msg.sender] = balances[msg.sender].sub(_value);
145     balances[_to] = balances[_to].add(_value);
146     emit Transfer(msg.sender, _to, _value);
147     return true;
148 }

```

✔ The code meets the specification.

## Formal Verification Request 17

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 4.44 ms

Line 155 in File FCTV.sol

```

155 // @CTK_NO_OVERFLOW

```

Line 162-164 in File FCTV.sol

```

162 function balanceOf(address _owner) public view returns (uint256) {
163     return balances[_owner];
164 }

```

✔ The code meets the specification.

## Formal Verification Request 18

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

📅 18, Jun 2019

🕒 0.3 ms

Line 156 in File FCTV.sol

```

156 // @CTK_NO_BUF_OVERFLOW

```

Line 162-164 in File FCTV.sol

```

162 function balanceOf(address _owner) public view returns (uint256) {
163     return balances[_owner];
164 }

```

✔ The code meets the specification.

## Formal Verification Request 19

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.3 ms

Line 157 in File FCTV.sol

```

157 // @CTK_NO_ASF

```

Line 162-164 in File FCTV.sol

```
162  function balanceOf(address _owner) public view returns (uint256) {
163      return balances[_owner];
164  }
```

✔ The code meets the specification.

## Formal Verification Request 20

balanceOf

📅 18, Jun 2019

🕒 0.32 ms

Line 158-161 in File FCTV.sol

```
158  /*@CTK balanceOf
159      @tag assume_completion
160      @post (__return) == (balances[_owner])
161  */
```

Line 162-164 in File FCTV.sol

```
162  function balanceOf(address _owner) public view returns (uint256) {
163      return balances[_owner];
164  }
```

✔ The code meets the specification.

## Formal Verification Request 21

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 93.81 ms

Line 205 in File FCTV.sol

```
205  //@CTK NO_OVERFLOW
```

Line 220-237 in File FCTV.sol

```
220  function transferFrom(
221      address _from,
222      address _to,
223      uint256 _value
224  )
225      public
226      returns (bool)
227  {
228      require(_to != address(0));
229      require(_value <= balances[_from]);
230      require(_value <= allowed[_from][msg.sender]);
231
232      balances[_from] = balances[_from].sub(_value);
233      balances[_to] = balances[_to].add(_value);
```

```
234     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
235     emit Transfer(_from, _to, _value);
236     return true;
237 }
```

✔ The code meets the specification.

## Formal Verification Request 22

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

📅 18, Jun 2019

🕒 14.08 ms

Line 206 in File FCTV.sol

```
206 // @CTK_NO_BUF_OVERFLOW
```

Line 220-237 in File FCTV.sol

```
220 function transferFrom(
221     address _from,
222     address _to,
223     uint256 _value
224 )
225     public
226     returns (bool)
227 {
228     require(_to != address(0));
229     require(_value <= balances[_from]);
230     require(_value <= allowed[_from][msg.sender]);
231
232     balances[_from] = balances[_from].sub(_value);
233     balances[_to] = balances[_to].add(_value);
234     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
235     emit Transfer(_from, _to, _value);
236     return true;
237 }
```

✔ The code meets the specification.

## Formal Verification Request 23

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 90.98 ms

Line 207 in File FCTV.sol

```
207 // @CTK_FAIL_NO_ASF
```

Line 220-237 in File FCTV.sol

```
220 function transferFrom(
221     address _from,
222     address _to,
```

```

223     uint256 _value
224 )
225 public
226 returns (bool)
227 {
228     require(_to != address(0));
229     require(_value <= balances[_from]);
230     require(_value <= allowed[_from][msg.sender]);
231
232     balances[_from] = balances[_from].sub(_value);
233     balances[_to] = balances[_to].add(_value);
234     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
235     emit Transfer(_from, _to, _value);
236     return true;
237 }

```

✘ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _from = 0
5     _to = 64
6     _value = 1
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": 0,
38              "value": [
39                {
40                  "key": 0,
41                  "value": 128

```



```

42         },
43         {
44             "key": "ALL_OTHERS",
45             "value": 1
46         }
47     ]
48 },
49 {
50     "key": 1,
51     "value": [
52         {
53             "key": 0,
54             "value": 16
55         },
56         {
57             "key": "ALL_OTHERS",
58             "value": 128
59         }
60     ]
61 },
62 {
63     "key": "ALL_OTHERS",
64     "value": [
65         {
66             "key": "ALL_OTHERS",
67             "value": 1
68         }
69     ]
70 }
71 ],
72 "balances": [
73     {
74         "key": 32,
75         "value": 0
76     },
77     {
78         "key": 8,
79         "value": 16
80     },
81     {
82         "key": 2,
83         "value": 2
84     },
85     {
86         "key": 1,
87         "value": 4
88     },
89     {
90         "key": 64,
91         "value": 255
92     },
93     {
94         "key": 128,
95         "value": 0
96     },
97     {
98         "key": "ALL_OTHERS",
99         "value": 1

```

```

100     }
101     ],
102     "totalSupply_": 0
103     }
104     }
105     },
106     {
107     "key": "ALL_OTHERS",
108     "value": "EmptyAddress"
109     }
110 ]


```

112 Function invocation is reverted.

## Formal Verification Request 24

transferFrom

 18, Jun 2019

 366.31 ms

Line 208-219 in File FCTV.sol

```

208 /*@CTK "transferFrom"
209 @tag assume_completion
210 @pre (_to) != (address(0))
211 @pre (_value) <= (balances[_from])
212 @pre (_value) <= (allowed[_from][msg.sender])
213 @post (_from != _to) -> (__post.balances[_to] == (balances[_to] + _value))
214 @post (_from != _to) -> (__post.balances[_from] == (balances[_from] - _value))
215 @post (_from == _to) -> (__post.balances[_to] == balances[_to])
216 @post (_from == _to) -> (__post.balances[_from] == balances[_from])
217 @post (__post.allowed[_from][msg.sender]) == (allowed[_from][msg.sender] - _value)
218 @post (__return) == (true)
219 */

```

Line 220-237 in File FCTV.sol

```

220 function transferFrom(
221     address _from,
222     address _to,
223     uint256 _value
224 )
225 public
226 returns (bool)
227 {
228     require(_to != address(0));
229     require(_value <= balances[_from]);
230     require(_value <= allowed[_from][msg.sender]);
231
232     balances[_from] = balances[_from].sub(_value);
233     balances[_to] = balances[_to].add(_value);
234     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
235     emit Transfer(_from, _to, _value);
236     return true;
237 }


```

 The code meets the specification.

## Formal Verification Request 25

If method completes, integer overflow would not happen.

 18, Jun 2019

 8.84 ms

Line 249 in File FCTV.sol

```
249 // @CTK_NO_OVERFLOW
```


Line 256-260 in File FCTV.sol


```
256 function approve(address _spender, uint256 _value) public returns (bool) {
257     allowed[msg.sender][_spender] = _value;
258     emit Approval(msg.sender, _spender, _value);
259     return true;
260 }
```

 The code meets the specification.

## Formal Verification Request 26

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

 18, Jun 2019

 0.37 ms

Line 250 in File FCTV.sol

```
250 // @CTK_NO_BUF_OVERFLOW
```

Line 256-260 in File FCTV.sol


```
256 function approve(address _spender, uint256 _value) public returns (bool) {
257     allowed[msg.sender][_spender] = _value;
258     emit Approval(msg.sender, _spender, _value);
259     return true;
260 }
```

 The code meets the specification.

## Formal Verification Request 27

Method will not encounter an assertion failure.

 18, Jun 2019

 0.33 ms

Line 251 in File FCTV.sol

```
251 // @CTK_NO_ASF
```

Line 256-260 in File FCTV.sol

```
256 function approve(address _spender, uint256 _value) public returns (bool) {
257     allowed[msg.sender][_spender] = _value;
258     emit Approval(msg.sender, _spender, _value);
259     return true;
260 }
```

✔ The code meets the specification.

## Formal Verification Request 28

approve

📅 18, Jun 2019

🕒 1.24 ms

Line 252-255 in File FCTV.sol

```
252  /*@CTK approve
253     @tag assume_completion
254     @post ( __post.allowed[msg.sender][_spender]) == (_value)
255  */
```

Line 256-260 in File FCTV.sol

```
256  function approve(address _spender, uint256 _value) public returns (bool) {
257      allowed[msg.sender][_spender] = _value;
258      emit Approval(msg.sender, _spender, _value);
259      return true;
260  }
```

✔ The code meets the specification.

## Formal Verification Request 29

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 6.33 ms

Line 268 in File FCTV.sol

```
268  //@CTK NO_OVERFLOW
```


Line 275-284 in File FCTV.sol


```
275  function allowance(
276      address _owner,
277      address _spender
278  )
279      public
280      view
281      returns (uint256)
282  {
283      return allowed[_owner][_spender];
284  }
```

✔ The code meets the specification.

## Formal Verification Request 30

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

 18, Jun 2019

 0.34 ms

Line 269 in File FCTV.sol

```
269 // @CTK_NO_BUF_OVERFLOW
```


Line 275-284 in File FCTV.sol


```
275 function allowance(  
276     address _owner,  
277     address _spender  
278 )  
279     public  
280     view  
281     returns (uint256)  
282 {  
283     return allowed[_owner][_spender];  
284 }
```

 The code meets the specification.

## Formal Verification Request 31

Method will not encounter an assertion failure.

 18, Jun 2019

 0.34 ms

Line 270 in File FCTV.sol

```
270 // @CTK_NO_ASF
```

Line 275-284 in File FCTV.sol


```
275 function allowance(  
276     address _owner,  
277     address _spender  
278 )  
279     public  
280     view  
281     returns (uint256)  
282 {  
283     return allowed[_owner][_spender];  
284 }
```

 The code meets the specification.

## Formal Verification Request 32

allowance

 18, Jun 2019

 0.35 ms

Line 271-274 in File FCTV.sol

```
271  /*@CTK allowance
272     @tag assume_completion
273     @post ().__return == (allowed[_owner][_spender])
274  */
```

Line 275-284 in File FCTV.sol

```
275  function allowance(
276     address _owner,
277     address _spender
278  )
279  public
280  view
281  returns (uint256)
282  {
283     return allowed[_owner][_spender];
284  }
```

✔ The code meets the specification.

## Formal Verification Request 33

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 34.18 ms

Line 296 in File FCTV.sol

```
296  //@CTK NO_OVERFLOW
```

Line 305-316 in File FCTV.sol

```
305  function increaseApproval(
306     address _spender,
307     uint _addedValue
308  )
309  public
310  returns (bool)
311  {
312     allowed[msg.sender][_spender] = (
313         allowed[msg.sender][_spender].add(_addedValue));
314     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
315     return true;
316  }
```

✔ The code meets the specification.

## Formal Verification Request 34

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

📅 18, Jun 2019

🕒 0.63 ms

Line 297 in File FCTV.sol

```
297 // @CTK_NO_BUF_OVERFLOW
```

Line 305-316 in File FCTV.sol

```
305 function increaseApproval(
306     address _spender,
307     uint _addedValue
308 )
309     public
310     returns (bool)
311 {
312     allowed[msg.sender][_spender] = (
313         allowed[msg.sender][_spender].add(_addedValue));
314     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
315     return true;
316 }
```

✔ The code meets the specification.

## Formal Verification Request 35

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 6.94 ms

Line 298 in File FCTV.sol

```
298 // @CTK_FAIL_NO_ASF
```

Line 305-316 in File FCTV.sol

```
305 function increaseApproval(
306     address _spender,
307     uint _addedValue
308 )
309     public
310     returns (bool)
311 {
312     allowed[msg.sender][_spender] = (
313         allowed[msg.sender][_spender].add(_addedValue));
314     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
315     return true;
316 }
```

✘ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     _addedValue = 1
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": "StandardToken",
32             "balance": 0,
33             "contract": {
34                 "allowed": [
35                     {
36                         "key": 0,
37                         "value": [
38                             {
39                                 "key": 1,
40                                 "value": 0
41                             },
42                             {
43                                 "key": 0,
44                                 "value": 255
45                             },
46                             {
47                                 "key": "ALL_OTHERS",
48                                 "value": 1
49                             }
50                         ]
51                     },
52                     {
53                         "key": "ALL_OTHERS",
54                         "value": [
55                             {
56                                 "key": "ALL_OTHERS",
57                                 "value": 1
58                             }
59                         ]
60                     }
61                 ],
62                 "balances": [
63                     {
64                         "key": 1,
65                         "value": 4
66                     },
67                     {
68                         "key": "ALL_OTHERS",

```



```

69         "value": 1
70     }
71 ],
72     "totalSupply_": 0
73 }
74 }
75 },
76 {
77     "key": "ALL_OTHERS",
78     "value": "EmptyAddress"
79 }
80 ]


```

82 Function invocation is reverted.

## Formal Verification Request 36

### increaseApproval

 18, Jun 2019

 1.85 ms

Line 299-304 in File FCTV.sol

```

299  /*@CTK increaseApproval
300     @tag assume_completion
301     @pre _spender != 0x0
302     @post (__post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
           _addedValue)
303     @post (__return) == (true)
304  */

```

Line 305-316 in File FCTV.sol

```

305  function increaseApproval(
306     address _spender,
307     uint _addedValue
308  )
309     public
310     returns (bool)
311  {
312     allowed[msg.sender][_spender] = (
313         allowed[msg.sender][_spender].add(_addedValue));
314     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
315     return true;
316  }


```

 The code meets the specification.

## Formal Verification Request 37

If method completes, integer overflow would not happen.

 18, Jun 2019

 40.83 ms

Line 328 in File FCTV.sol

```
328 // @CTK_NO_OVERFLOW
```

Line 337-352 in File FCTV.sol

```
337 function decreaseApproval(  
338     address _spender,  
339     uint _subtractedValue  
340 )  
341     public  
342     returns (bool)  
343 {  
344     uint oldValue = allowed[msg.sender][_spender];  
345     if (_subtractedValue > oldValue) {  
346         allowed[msg.sender][_spender] = 0;  
347     } else {  
348         allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);  
349     }  
350     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);  
351     return true;  
352 }
```

✔ The code meets the specification.

## Formal Verification Request 38

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

📅 18, Jun 2019

🕒 0.73 ms

Line 329 in File FCTV.sol

```
329 // @CTK_NO_BUF_OVERFLOW
```

Line 337-352 in File FCTV.sol


```
337 function decreaseApproval(  
338     address _spender,  
339     uint _subtractedValue  
340 )  
341     public  
342     returns (bool)  
343 {  
344     uint oldValue = allowed[msg.sender][_spender];  
345     if (_subtractedValue > oldValue) {  
346         allowed[msg.sender][_spender] = 0;  
347     } else {  
348         allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);  
349     }  
350     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);  
351     return true;  
352 }
```

✔ The code meets the specification.

## Formal Verification Request 39

Method will not encounter an assertion failure.

 18, Jun 2019

 1.21 ms

Line 330 in File FCTV.sol

```
330 // @CTK NO_ASF
```

Line 337-352 in File FCTV.sol


```
337 function decreaseApproval(
338     address _spender,
339     uint _subtractedValue
340 )
341     public
342     returns (bool)
343 {
344     uint oldValue = allowed[msg.sender][_spender];
345     if (_subtractedValue > oldValue) {
346         allowed[msg.sender][_spender] = 0;
347     } else {
348         allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
349     }
350     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
351     return true;
352 }
```

 The code meets the specification.

## Formal Verification Request 40

decreaseApproval

 18, Jun 2019

 3.9 ms

Line 331-336 in File FCTV.sol

```
331 /* @CTK "decreaseApproval"
332     @tag assume_completion
333     @pre _spender != 0x0
334     @post (_subtractedValue > allowed[msg.sender][_spender]) -> (
335         __post.allowed[msg.sender][_spender] == 0)
335     @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (
336         __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
336         _subtractedValue)
336 */
```

Line 337-352 in File FCTV.sol

```
337 function decreaseApproval(
338     address _spender,
339     uint _subtractedValue
340 )
341     public
342     returns (bool)
343 {
```

```

344     uint oldValue = allowed[msg.sender][_spender];
345     if (_subtractedValue > oldValue) {
346         allowed[msg.sender][_spender] = 0;
347     } else {
348         allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
349     }
350     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
351     return true;
352 }

```

✔ The code meets the specification.

## Formal Verification Request 41

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 6.59 ms

Line 369 in File FCTV.sol

```

369     // @CTK_NO_OVERFLOW

```

Line 377-380 in File FCTV.sol

```

377     constructor() public {
378         owners[msg.sender] = true;
379         unremovableOwner = msg.sender;
380     }

```

✔ The code meets the specification.

## Formal Verification Request 42

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

📅 18, Jun 2019

🕒 0.33 ms

Line 370 in File FCTV.sol

```

370     // @CTK_NO_BUF_OVERFLOW

```

Line 377-380 in File FCTV.sol

```


377     constructor() public {
378         owners[msg.sender] = true;
379         unremovableOwner = msg.sender;
380     }


```

✔ The code meets the specification.

## Formal Verification Request 43

Method will not encounter an assertion failure.

 18, Jun 2019

 0.31 ms

Line 371 in File FCTV.sol

```
371 // @CTK NO_ASF
```


Line 377-380 in File FCTV.sol


```
377 constructor() public {
378     owners[msg.sender] = true;
379     unremovableOwner = msg.sender;
380 }
```

 The code meets the specification.

## Formal Verification Request 44

MultiOwnable constructor

 18, Jun 2019

 1.11 ms

Line 372-376 in File FCTV.sol

```
372 /* @CTK "MultiOwnable constructor"
373     @tag assume_completion
374     @post __post.owners[msg.sender]
375     @post __post.unremovableOwner == msg.sender
376 */
```

Line 377-380 in File FCTV.sol


```
377 constructor() public {
378     owners[msg.sender] = true;
379     unremovableOwner = msg.sender;
380 }
```

 The code meets the specification.

## Formal Verification Request 45

If method completes, integer overflow would not happen.

 18, Jun 2019

 18.65 ms

Line 382 in File FCTV.sol

```
382 // @CTK NO_OVERFLOW
```

Line 391-395 in File FCTV.sol


```
391     function addOwner(address guest) onlyOwner public {
392         require(guest != address(0));
393         owners[guest] = true;
394         emit OwnershipExtended(msg.sender, guest);
395     }
```

✔ The code meets the specification.

## Formal Verification Request 46

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

 18, Jun 2019

 0.44 ms

Line 383 in File FCTV.sol

```
383     //@CTK_NO_BUF_OVERFLOW
```

Line 391-395 in File FCTV.sol


```
391     function addOwner(address guest) onlyOwner public {
392         require(guest != address(0));
393         owners[guest] = true;
394         emit OwnershipExtended(msg.sender, guest);
395     }
```

✔ The code meets the specification.

## Formal Verification Request 47

Method will not encounter an assertion failure.

 18, Jun 2019

 0.43 ms

Line 384 in File FCTV.sol

```
384     //@CTK_NO_ASF
```

Line 391-395 in File FCTV.sol


```
391     function addOwner(address guest) onlyOwner public {
392         require(guest != address(0));
393         owners[guest] = true;
394         emit OwnershipExtended(msg.sender, guest);
395     }
```

✔ The code meets the specification.

## Formal Verification Request 48

addOwner

 18, Jun 2019

 1.69 ms

Line 385-390 in File FCTV.sol

```
385  /*@CTK addOwner
386     @tag assume_completion
387     @pre owners[msg.sender]
388     @pre guest != address(0)
389     @post __post.owners[guest]
390  */
```

Line 391-395 in File FCTV.sol

```
391  function addOwner(address guest) onlyOwner public {
392      require(guest != address(0));
393      owners[guest] = true;
394      emit OwnershipExtended(msg.sender, guest);
395  }
```

✔ The code meets the specification.

## Formal Verification Request 49

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 25.66 ms

Line 397 in File FCTV.sol

```
397  //@CTK NO_OVERFLOW
```

Line 407-412 in File FCTV.sol

```
407  function removeOwner(address removedOwner) onlyOwner public {
408      require(removedOwner != address(0));
409      require(unremovableOwner != removedOwner);
410      delete owners[removedOwner];
411      emit OwnershipRemoved(removedOwner);
412  }
```

✔ The code meets the specification.

## Formal Verification Request 50

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

📅 18, Jun 2019

🕒 1.6 ms

Line 398 in File FCTV.sol

```
398  //@CTK NO_BUF_OVERFLOW
```

Line 407-412 in File FCTV.sol

```
407  function removeOwner(address removedOwner) onlyOwner public {
408      require(removedOwner != address(0));
409      require(unremovableOwner != removedOwner);
```


```
410     delete owners[removedOwner];
411     emit OwnershipRemoved(removedOwner);
412 }
```

✔ The code meets the specification.

## Formal Verification Request 51

Method will not encounter an assertion failure.

 18, Jun 2019

 1.55 ms

Line 399 in File FCTV.sol

```
399     //@CTK NO_ASF
```

Line 407-412 in File FCTV.sol


```
407     function removeOwner(address removedOwner) onlyOwner public {
408         require(removedOwner != address(0));
409         require(unremovableOwner != removedOwner);
410         delete owners[removedOwner];
411         emit OwnershipRemoved(removedOwner);
412     }
```

✔ The code meets the specification.

## Formal Verification Request 52

removeOwner

 18, Jun 2019

 1.75 ms

Line 400-406 in File FCTV.sol

```
400     /*@CTK removeOwner
401         @tag assume_completion
402         @pre owners[msg.sender]
403         @pre removedOwner != address(0)
404         @pre unremovableOwner != removedOwner
405         @post !(__post.owners[removedOwner])
406     */
```

Line 407-412 in File FCTV.sol

```
407     function removeOwner(address removedOwner) onlyOwner public {
408         require(removedOwner != address(0));
409         require(unremovableOwner != removedOwner);
410         delete owners[removedOwner];
411         emit OwnershipRemoved(removedOwner);
412     }
```


✔ The code meets the specification.



## Formal Verification Request 53

If method completes, integer overflow would not happen.

 18, Jun 2019

 29.28 ms

Line 414 in File FCTV.sol

414 `//@CTK_NO_OVERFLOW`

Line 426-432 in File FCTV.sol


```
426     function transferOwnership(address newOwner) onlyOwner public {
427         require(newOwner != address(0));
428         require(unremovableOwner != msg.sender);
429         owners[newOwner] = true;
430         delete owners[msg.sender];
431         emit OwnershipTransferred(msg.sender, newOwner);
432     }
```

 The code meets the specification.

## Formal Verification Request 54

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

 18, Jun 2019

 2.14 ms

Line 415 in File FCTV.sol

415 `//@CTK_NO_BUF_OVERFLOW`

Line 426-432 in File FCTV.sol


```
426     function transferOwnership(address newOwner) onlyOwner public {
427         require(newOwner != address(0));
428         require(unremovableOwner != msg.sender);
429         owners[newOwner] = true;
430         delete owners[msg.sender];
431         emit OwnershipTransferred(msg.sender, newOwner);
432     }
```

 The code meets the specification.

## Formal Verification Request 55

Method will not encounter an assertion failure.

 18, Jun 2019

 2.02 ms

Line 416 in File FCTV.sol

416 `//@CTK_NO_ASF`

Line 426-432 in File FCTV.sol

```

426     function transferOwnership(address newOwner) onlyOwner public {
427         require(newOwner != address(0));
428         require(unremovableOwner != msg.sender);
429         owners[newOwner] = true;
430         delete owners[msg.sender];
431         emit OwnershipTransferred(msg.sender, newOwner);
432     }

```

✔ The code meets the specification.

## Formal Verification Request 56

transferOwnership

📅 18, Jun 2019

🕒 2.48 ms

Line 417-425 in File FCTV.sol

```

417     /*@CTK transferOwnership
418         @tag assume_completion
419         @pre owners[msg.sender]
420         @pre newOwner != address(0)
421         @pre msg.sender != newOwner
422         @pre unremovableOwner != msg.sender
423         @post (__post.owners[newOwner])
424         @post !(__post.owners[msg.sender])
425     */

```

Line 426-432 in File FCTV.sol

```

426     function transferOwnership(address newOwner) onlyOwner public {
427         require(newOwner != address(0));
428         require(unremovableOwner != msg.sender);
429         owners[newOwner] = true;
430         delete owners[msg.sender];
431         emit OwnershipTransferred(msg.sender, newOwner);
432     }

```

✔ The code meets the specification.

## Formal Verification Request 57

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 5.36 ms

Line 434 in File FCTV.sol

```

434     //@CTK NO_OVERFLOW

```

Line 441-443 in File FCTV.sol

```

441     function isOwner(address addr) public view returns(bool){
442         return owners[addr];
443     }

```

✔ The code meets the specification.

## Formal Verification Request 58

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

📅 18, Jun 2019

🕒 0.35 ms

Line 435 in File FCTV.sol

```
435 // @CTK_NO_BUF_OVERFLOW
```

Line 441-443 in File FCTV.sol

```
441 function isOwner(address addr) public view returns(bool){
442     return owners[addr];
443 }
```

✔ The code meets the specification.

## Formal Verification Request 59

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 0.31 ms

Line 436 in File FCTV.sol

```
436 // @CTK_NO_ASF
```

Line 441-443 in File FCTV.sol

```
441 function isOwner(address addr) public view returns(bool){
442     return owners[addr];
443 }
```

✔ The code meets the specification.

## Formal Verification Request 60

isOwner

📅 18, Jun 2019

🕒 0.34 ms

Line 437-440 in File FCTV.sol

```
437 /* @CTK isOwner
438     @tag assume_completion
439     @post (__return) == (owners[addr])
440 */
```

Line 441-443 in File FCTV.sol

```
441     function isOwner(address addr) public view returns(bool){
442         return owners[addr];
443     }
```

✔ The code meets the specification.

## Formal Verification Request 61

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 60.75 ms

Line 458 in File FCTV.sol

```
458     //@CTK_NO_OVERFLOW
```

Line 464-468 in File FCTV.sol

```
464     constructor() public {
465         totalSupply_ = TOTAL_CAP.mul(10 ** decimals);
466         balances[msg.sender] = totalSupply_;
467         emit Transfer(address(0), msg.sender, balances[msg.sender]);
468     }
```

✔ The code meets the specification.

## Formal Verification Request 62

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

📅 18, Jun 2019

🕒 0.8 ms

Line 459 in File FCTV.sol

```
459     //@CTK_NO_BUF_OVERFLOW
```

Line 464-468 in File FCTV.sol

```
464     constructor() public {
465         totalSupply_ = TOTAL_CAP.mul(10 ** decimals);
466         balances[msg.sender] = totalSupply_;
467         emit Transfer(address(0), msg.sender, balances[msg.sender]);
468     }
```

✔ The code meets the specification.

## Formal Verification Request 63

FCTV constructor

📅 18, Jun 2019

🕒 5.1 ms

Line 460-463 in File FCTV.sol

```

460  /*@CTK "FCTV constructor"
461     @tag assume_completion
462     @post __post.balances[msg.sender] == __post.totalSupply_
463  */

```

Line 464-468 in File FCTV.sol

```

464  constructor() public {
465      totalSupply_ = TOTAL_CAP.mul(10 ** decimals);
466      balances[msg.sender] = totalSupply_;
467      emit Transfer(address(0), msg.sender, balances[msg.sender]);
468  }

```

✔ The code meets the specification.

## Formal Verification Request 64

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 14.1 ms

Line 470 in File FCTV.sol

```

470  //@CTK NO_OVERFLOW

```

Line 477-479 in File FCTV.sol

```

477  function unlock() external onlyOwner {
478      isTransferable = true;
479  }

```

✔ The code meets the specification.

## Formal Verification Request 65

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

📅 18, Jun 2019

🕒 0.42 ms

Line 471 in File FCTV.sol

```

471  //@CTK NO_BUF_OVERFLOW

```

Line 477-479 in File FCTV.sol

```

477  function unlock() external onlyOwner {
478      isTransferable = true;
479  }


```

✔ The code meets the specification.

## Formal Verification Request 66

Method will not encounter an assertion failure.

 18, Jun 2019

 0.4 ms

Line 472 in File FCTV.sol

```
472 // @CTK_NO_ASF
```

Line 477-479 in File FCTV.sol


```
477 function unlock() external onlyOwner {  
478     isTransferable = true;  
479 }
```

 The code meets the specification.

## Formal Verification Request 67

unlock

 18, Jun 2019

 1.96 ms

Line 473-476 in File FCTV.sol

```
473 /* @CTK_unlock  
474     @pre owners[msg.sender]  
475     @post (__post.isTransferable) == (true)  
476 */
```

Line 477-479 in File FCTV.sol


```
477 function unlock() external onlyOwner {  
478     isTransferable = true;  
479 }
```

 The code meets the specification.

## Formal Verification Request 68

If method completes, integer overflow would not happen.

 18, Jun 2019

 14.34 ms

Line 481 in File FCTV.sol

```
481 // @CTK_NO_OVERFLOW
```


Line 488-490 in File FCTV.sol


```
488 function lock() external onlyOwner {  
489     isTransferable = false;  
490 }
```

 The code meets the specification.

## Formal Verification Request 69

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

 18, Jun 2019

 0.44 ms

Line 482 in File FCTV.sol

```
482 // @CTK_NO_BUF_OVERFLOW
```


Line 488-490 in File FCTV.sol


```
488 function lock() external onlyOwner {
489     isTransferable = false;
490 }
```

 The code meets the specification.

## Formal Verification Request 70

Method will not encounter an assertion failure.

 18, Jun 2019

 0.4 ms

Line 483 in File FCTV.sol

```
483 // @CTK_NO_ASF
```


Line 488-490 in File FCTV.sol


```
488 function lock() external onlyOwner {
489     isTransferable = false;
490 }
```

 The code meets the specification.

## Formal Verification Request 71

lock

 18, Jun 2019

 2.06 ms

Line 484-487 in File FCTV.sol

```
484 /* @CTK lock
485    @pre owners[msg.sender]
486    @post (__post.isTransferable) == (false)
487    */
```


Line 488-490 in File FCTV.sol


```
488 function lock() external onlyOwner {
489     isTransferable = false;
490 }
```

 The code meets the specification.

## Formal Verification Request 72

If method completes, integer overflow would not happen.

 18, Jun 2019

 226.88 ms

Line 492 in File FCTV.sol

492 `//@CTK_NO_OVERFLOW`


Line 505-508 in File FCTV.sol


```
505     function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool) {
506         require(isTransferable || owners[msg.sender]);
507         return super.transferFrom(_from, _to, _value);
508     }
```

 The code meets the specification.

## Formal Verification Request 73

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

 18, Jun 2019

 36.8 ms

Line 493 in File FCTV.sol

493 `//@CTK_NO_BUF_OVERFLOW`

Line 505-508 in File FCTV.sol


```
505     function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool) {
506         require(isTransferable || owners[msg.sender]);
507         return super.transferFrom(_from, _to, _value);
508     }
```

 The code meets the specification.

## Formal Verification Request 74

Method will not encounter an assertion failure.

 18, Jun 2019

 1122.66 ms

Line 494 in File FCTV.sol

494 `//@CTK_FAIL_NO_ASF`

Line 505-508 in File FCTV.sol

```
505     function transferFrom(address _from, address _to, uint256 _value) public returns (
        bool) {
506         require(isTransferable || owners[msg.sender]);
507         return super.transferFrom(_from, _to, _value);
508     }
```



✘ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _from = 0
5     _to = 1
6     _value = 1
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": "FCTV",
33        "balance": 0,
34        "contract": {
35          "TOTAL_CAP": 0,
36          "name": "",
37          "symbol": "",
38          "decimals": 0,
39          "isTransferable": false,
40          "owners": [
41            {
42              "key": 0,
43              "value": true
44            },
45            {
46              "key": "ALL_OTHERS",
47              "value": false
48            }
49          ],
50          "unremovableOwner": 0,
51          "allowed": [
52            {
53              "key": 0,
54              "value": [
55                {
56                  "key": 8,
57                  "value": 128

```

```

58         },
59         {
60             "key": 32,
61             "value": 0
62         },
63         {
64             "key": 0,
65             "value": 32
66         },
67         {
68             "key": "ALL_OTHERS",
69             "value": 5
70         }
71     ]
72 },
73 {
74     "key": "ALL_OTHERS",
75     "value": [
76         {
77             "key": "ALL_OTHERS",
78             "value": 5
79         }
80     ]
81 }
82 ],
83 "balances": [
84     {
85         "key": 1,
86         "value": 255
87     },
88     {
89         "key": 16,
90         "value": 0
91     },
92     {
93         "key": 2,
94         "value": 32
95     },
96     {
97         "key": 8,
98         "value": 0
99     },
100    {
101        "key": 32,
102        "value": 128
103    },
104    {
105        "key": 64,
106        "value": 64
107    },
108    {
109        "key": 9,
110        "value": 0
111    },
112    {
113        "key": "ALL_OTHERS",
114        "value": 5
115    }

```


```


116     ],
117     "totalSupply_": 0
118   }
119 }
120 },
121 {
122   "key": "ALL_OTHERS",
123   "value": "EmptyAddress"
124 }
125 ]
126
127 Function invocation is reverted.

```

## Formal Verification Request 75

### FCTV transferFrom

 18, Jun 2019

 471.05 ms

Line 495-504 in File FCTV.sol

```

495  /*@CTK "FCTV transferFrom"
496     @tag assume_completion
497     @post (isTransferable || owners[msg.sender])
498     @post ((_from) != (_to)) -> ((__post.balances[_to]) == ((balances[_to]) + (
499         _value)))
500     @post ((_from) != (_to)) -> ((__post.balances[_from]) == ((balances[_from]) - (
501         _value)))
502     @post ((_from) == (_to)) -> ((__post.balances[_to]) == (balances[_to]))
503     @post ((_from) == (_to)) -> ((__post.balances[_from]) == (balances[_from]))
504     @post (__post.allowed[_from][msg.sender]) == ((allowed[_from][msg.sender]) - (
505         _value))
506     @post (__return) == (true)
507 */

```

Line 505-508 in File FCTV.sol

```

505  function transferFrom(address _from, address _to, uint256 _value) public returns (
506     bool) {
507     require(isTransferable || owners[msg.sender]);
508     return super.transferFrom(_from, _to, _value);
509 }


```

 The code meets the specification.

## Formal Verification Request 76

If method completes, integer overflow would not happen.

 18, Jun 2019

 185.09 ms

Line 510 in File FCTV.sol

```

510  //@CTK NO_OVERFLOW

```

Line 522-525 in File FCTV.sol

```
522     function transfer(address _to, uint256 _value) public returns (bool) {
523         require(isTransferable || owners[msg.sender]);
524         return super.transfer(_to, _value);
525     }
```

✔ The code meets the specification.

## Formal Verification Request 77

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

📅 18, Jun 2019

🕒 52.78 ms

Line 511 in File FCTV.sol

```
511     //@CTK_NO_BUF_OVERFLOW
```

Line 522-525 in File FCTV.sol

```
522     function transfer(address _to, uint256 _value) public returns (bool) {
523         require(isTransferable || owners[msg.sender]);
524         return super.transfer(_to, _value);
525     }
```

✔ The code meets the specification.

## Formal Verification Request 78

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 105.47 ms

Line 512 in File FCTV.sol

```
512     //@CTK_FAIL_NO_ASF
```

Line 522-525 in File FCTV.sol

```
522     function transfer(address _to, uint256 _value) public returns (bool) {
523         require(isTransferable || owners[msg.sender]);
524         return super.transfer(_to, _value);
525     }
```

✘ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3     Input = {
4         _to = 32
5         _value = 24
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": "FCTV",
32             "balance": 0,
33             "contract": {
34                 "TOTAL_CAP": 0,
35                 "name": "",
36                 "symbol": "",
37                 "decimals": 0,
38                 "isTransferable": false,
39                 "owners": [
40                     {
41                         "key": 0,
42                         "value": true
43                     },
44                     {
45                         "key": "ALL_OTHERS",
46                         "value": false
47                     }
48                 ],
49                 "unremovableOwner": 0,
50                 "allowed": [
51                     {
52                         "key": "ALL_OTHERS",
53                         "value": [
54                             {
55                                 "key": "ALL_OTHERS",
56                                 "value": 232
57                             }
58                         ]
59                     }
60                 ],
61                 "balances": [
62                     {
63                         "key": 2,
64                         "value": 0
65                     },
66                     {
67                         "key": 1,

```

```

68         "value": 32
69     },
70     {
71         "key": 8,
72         "value": 0
73     },
74     {
75         "key": 0,
76         "value": 24
77     },
78     {
79         "key": 32,
80         "value": 240
81     },
82     {
83         "key": 16,
84         "value": 0
85     },
86     {
87         "key": 4,
88         "value": 0
89     },
90     {
91         "key": "ALL_OTHERS",
92         "value": 232
93     }
94 ],
95     "totalSupply_": 0
96 }
97 }
98 },
99 {
100     "key": "ALL_OTHERS",
101     "value": "EmptyAddress"
102 }
103 ]


```

105 Function invocation is reverted.

## Formal Verification Request 79

### FCTV transfer

 18, Jun 2019

 154.17 ms

Line 513-521 in File FCTV.sol

```

513     /*@CTK "FCTV transfer"
514     @tag assume_completion
515     @post (isTransferable || owners[msg.sender])
516     @post ((_to) != (msg.sender)) -> (__post.balances[msg.sender]) == (balances[msg.
517         sender] - _value)
518     @post ((_to) != (msg.sender)) -> (__post.balances[_to]) == (balances[_to] +
519         _value)
520     @post ((_to) == (msg.sender)) -> (__post.balances[msg.sender]) == balances[msg.
521         sender]

```

```
519     @post ((_to) == (msg.sender)) -> (__post.balances[_to]) == balances[_to]
520     @post (__return) == (true)
521     */
```

Line 522-525 in File FCTV.sol

```
522     function transfer(address _to, uint256 _value) public returns (bool) {
523         require(isTransferable || owners[msg.sender]);
524         return super.transfer(_to, _value);
525     }
```

✔ The code meets the specification.

## Formal Verification Request 80

If method completes, integer overflow would not happen.

📅 18, Jun 2019

🕒 76.87 ms

Line 528 in File FCTV.sol

```
528     //@CTK NO_OVERFLOW
```

Line 538-548 in File FCTV.sol

```
538     function mint(address _to, uint256 _amount) onlyOwner public returns (bool) {
539         require(_to != address(0));
540
541         totalSupply_ = totalSupply_.add(_amount);
542         balances[_to] = balances[_to].add(_amount);
543
544         emit Mint(_to, _amount);
545         emit Transfer(address(0), _to, _amount);
546
547         return true;
548     }
```

✔ The code meets the specification.

## Formal Verification Request 81

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

📅 18, Jun 2019

🕒 13.11 ms

Line 529 in File FCTV.sol

```
529     //@CTK NO_BUF_OVERFLOW
```

Line 538-548 in File FCTV.sol

```
538     function mint(address _to, uint256 _amount) onlyOwner public returns (bool) {
539         require(_to != address(0));
540
541         totalSupply_ = totalSupply_.add(_amount);
542         balances[_to] = balances[_to].add(_amount);
```

```

543
544     emit Mint(_to, _amount);
545     emit Transfer(address(0), _to, _amount);
546
547     return true;
548 }

```

✔ The code meets the specification.

## Formal Verification Request 82

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 69.72 ms

Line 530 in File FCTV.sol

```
530 //CTK FAIL NO_ASF
```

Line 538-548 in File FCTV.sol

```

538     function mint(address _to, uint256 _amount) onlyOwner public returns (bool) {
539         require(_to != address(0));
540
541         totalSupply_ = totalSupply_.add(_amount);
542         balances[_to] = balances[_to].add(_amount);
543
544         emit Mint(_to, _amount);
545         emit Transfer(address(0), _to, _amount);
546
547         return true;
548     }

```

✘ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3     Input = {
4         _amount = 148
5         _to = 2
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": "FCTV",
32       "balance": 0,
33       "contract": {
34         "TOTAL_CAP": 0,
35         "name": "",
36         "symbol": "",
37         "decimals": 0,
38         "isTransferable": false,
39         "owners": [
40           {
41             "key": 0,
42             "value": true
43           },
44           {
45             "key": "ALL_OTHERS",
46             "value": false
47           }
48         ],
49         "unremovableOwner": 0,
50         "allowed": [
51           {
52             "key": "ALL_OTHERS",
53             "value": [
54               {
55                 "key": "ALL_OTHERS",
56                 "value": 2
57               }
58             ]
59           }
60         ],
61         "balances": [
62           {
63             "key": 16,
64             "value": 0
65           },
66           {
67             "key": 0,
68             "value": 4
69           },
70           {
71             "key": 2,
72             "value": 140
73           },
74           {
75             "key": "ALL_OTHERS",
76             "value": 2
77           }
78         ],
79         "totalSupply_": 36
80       }

```


```


81     }
82     },
83     {
84         "key": "ALL_OTHERS",
85         "value": "EmptyAddress"
86     }
87 ]
88
89 Function invocation is reverted.

```

## Formal Verification Request 83

### FCTV mint

 18, Jun 2019

 143.92 ms

Line 531-537 in File FCTV.sol

```

531  /*@CTK "FCTV mint"
532     @tag assume_completion
533     @pre owners[msg.sender]
534     @post (_to != address(0))
535     @post (__post.totalSupply_) == ((totalSupply_) + (_amount))
536     @post (__post.balances[_to]) == ((balances[_to]) + (_amount))
537  */

```

Line 538-548 in File FCTV.sol

```

538  function mint(address _to, uint256 _amount) onlyOwner public returns (bool) {
539      require(_to != address(0));
540
541      totalSupply_ = totalSupply_.add(_amount);
542      balances[_to] = balances[_to].add(_amount);
543
544      emit Mint(_to, _amount);
545      emit Transfer(address(0), _to, _amount);
546
547      return true;
548  }


```

 The code meets the specification.

## Formal Verification Request 84

If method completes, integer overflow would not happen.

 18, Jun 2019

 67.69 ms

Line 551 in File FCTV.sol

```

551  //@CTK NO_OVERFLOW

```

Line 561-569 in File FCTV.sol

```

561     function burn(uint256 _amount) onlyOwner public {
562         require(_amount <= balances[msg.sender]);
563
564         totalSupply_ = totalSupply_.sub(_amount);
565         balances[msg.sender] = balances[msg.sender].sub(_amount);
566
567         emit Burn(msg.sender, _amount);
568         emit Transfer(msg.sender, address(0), _amount);
569     }

```

✔ The code meets the specification.

## Formal Verification Request 85

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

📅 18, Jun 2019

🕒 11.17 ms

Line 552 in File FCTV.sol

```

552     // @CTK_NO_BUF_OVERFLOW

```

Line 561-569 in File FCTV.sol

```

561     function burn(uint256 _amount) onlyOwner public {
562         require(_amount <= balances[msg.sender]);
563
564         totalSupply_ = totalSupply_.sub(_amount);
565         balances[msg.sender] = balances[msg.sender].sub(_amount);
566
567         emit Burn(msg.sender, _amount);
568         emit Transfer(msg.sender, address(0), _amount);
569     }

```

✔ The code meets the specification.

## Formal Verification Request 86

Method will not encounter an assertion failure.

📅 18, Jun 2019

🕒 73.28 ms

Line 553 in File FCTV.sol

```

553     // @CTK_FAIL_NO_ASF

```

Line 561-569 in File FCTV.sol

```

561     function burn(uint256 _amount) onlyOwner public {
562         require(_amount <= balances[msg.sender]);
563
564         totalSupply_ = totalSupply_.sub(_amount);
565         balances[msg.sender] = balances[msg.sender].sub(_amount);
566
567         emit Burn(msg.sender, _amount);

```

```
568     emit Transfer(msg.sender, address(0), _amount);
569 }
```

✘ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     _amount = 128
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": "FCTV",
30        "balance": 0,
31        "contract": {
32          "TOTAL_CAP": 0,
33          "name": "",
34          "symbol": "",
35          "decimals": 0,
36          "isTransferable": false,
37          "owners": [
38            {
39              "key": 0,
40              "value": true
41            },
42            {
43              "key": "ALL_OTHERS",
44              "value": false
45            }
46          ],
47          "unremovableOwner": 0,
48          "allowed": [
49            {
50              "key": 0,
51              "value": [
52                {
53                  "key": 0,
54                  "value": 128
```

```

55         },
56     },
57     "key": "ALL_OTHERS",
58     "value": 72
59 }
60 ]
61 },
62 {
63     "key": "ALL_OTHERS",
64     "value": [
65     {
66         "key": "ALL_OTHERS",
67         "value": 128
68     }
69 ]
70 }
71 ],
72 "balances": [
73 {
74     "key": 2,
75     "value": 0
76 },
77 {
78     "key": 128,
79     "value": 1
80 },
81 {
82     "key": 64,
83     "value": 16
84 },
85 {
86     "key": 1,
87     "value": 32
88 },
89 {
90     "key": 4,
91     "value": 1
92 },
93 {
94     "key": 16,
95     "value": 0
96 },
97 {
98     "key": "ALL_OTHERS",
99     "value": 128
100 }
101 ],
102 "totalSupply_": 64
103 }
104 }
105 },
106 {
107     "key": "ALL_OTHERS",
108     "value": "EmptyAddress"
109 }
110 ]


```

112 Function invocation is reverted.

## Formal Verification Request 87

### FCTV burn

 18, Jun 2019

 187.47 ms

Line 554-560 in File FCTV.sol

```
554  /*@CTK "FCTV burn"
555     @tag assume_completion
556     @pre owners[msg.sender]
557     @post (_amount <= balances[msg.sender])
558     @post (__post.totalSupply_) == ((totalSupply_) - (_amount))
559     @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_amount))
560  */
```

Line 561-569 in File FCTV.sol

```
561  function burn(uint256 _amount) onlyOwner public {
562      require(_amount <= balances[msg.sender]);
563
564      totalSupply_ = totalSupply_.sub(_amount);
565      balances[msg.sender] = balances[msg.sender].sub(_amount);
566
567      emit Burn(msg.sender, _amount);
568      emit Transfer(msg.sender, address(0), _amount);
569  }
```

 The code meets the specification.

## Source Code with CertiK Labels

File FCTV.sol

```

1  pragma solidity ^0.4.23;
2
3  /**
4   * @title SafeMath
5   * @dev Math operations with safety checks that throw on error
6   */
7  library SafeMath {
8
9   /**
10  * @dev Multiplies two numbers, throws on overflow.
11  */
12  //@CTK FAIL NO_ASF
13  /*@CTK "SafeMath mul"
14   @post ((a > 0) && ((a * b) / a) != b) == (__reverted)
15   @post !__reverted -> c == a * b
16   @post !__reverted == !__has_overflow
17   @post !__reverted -> !(__has_assertion_failure)
18   @post !(__has_buf_overflow)
19  */
20  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
21   // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
22   // benefit is lost if 'b' is also tested.
23   // See: https://github.com/OpenZeppelin/zeppelin-solidity/pull/522
24   if (a == 0) {
25     return 0;
26   }
27
28   c = a * b;
29   assert(c / a == b);
30   return c;
31  }
32
33  /**
34  * @dev Integer division of two numbers, truncating the quotient.
35  */
36  //@CTK FAIL NO_ASF
37  /*@CTK "SafeMath div"
38   @post b != 0 -> !__reverted
39   @post !__reverted -> __return == a / b
40   @post !__reverted -> !__has_overflow
41   @post !__reverted -> !(__has_assertion_failure)
42   @post !(__has_buf_overflow)
43  */
44  function div(uint256 a, uint256 b) internal pure returns (uint256) {
45   // assert(b > 0); // Solidity automatically throws when dividing by 0
46   // uint256 c = a / b;
47   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
48   return a / b;
49  }
50
51  /**
52  * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
53   minuend).
54  */

```

```

54 // @CTK FAIL NO_ASF
55 /* @CTK "SafeMath sub"
56     @post (a < b) == __reverted
57     @post !__reverted -> __return == a - b
58     @post !__reverted -> !__has_overflow
59     @post !__reverted -> !(__has_assertion_failure)
60     @post !(__has_buf_overflow)
61 */
62 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63     assert(b <= a);
64     return a - b;
65 }
66
67 /**
68 * @dev Adds two numbers, throws on overflow.
69 */
70 // @CTK FAIL NO_ASF
71 /* @CTK "SafeMath add"
72     @post (a + b < a || a + b < b) == __reverted
73     @post !__reverted -> c == a + b
74     @post !__reverted -> !__has_overflow
75     @post !__reverted -> !(__has_assertion_failure)
76     @post !(__has_buf_overflow)
77 */
78 function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
79     c = a + b;
80     assert(c >= a);
81     return c;
82 }
83 }
84
85 /**
86 * @title ERC20Basic
87 * @dev Simpler version of ERC20 interface
88 * @dev see https://github.com/ethereum/EIPs/issues/179
89 */
90 contract ERC20Basic {
91     function totalSupply() public view returns (uint256);
92     function balanceOf(address who) public view returns (uint256);
93     function transfer(address to, uint256 value) public returns (bool);
94     event Transfer(address indexed from, address indexed to, uint256 value);
95 }
96
97 /**
98 * @title Basic token
99 * @dev Basic version of StandardToken, with no allowances.
100 */
101 contract BasicToken is ERC20Basic {
102     using SafeMath for uint256;
103
104     mapping(address => uint256) balances;
105
106     uint256 totalSupply_;
107
108     /**
109     * @dev total number of tokens in existence
110     */
111     // @CTK NO_OVERFLOW

```



```

112 //@CTK NO_BUF_OVERFLOW
113 //@CTK NO_ASF
114 /*@CTK totalSupply
115     @tag assume_completion
116     @post (__return) == (totalSupply_)
117 */
118 function totalSupply() public view returns (uint256) {
119     return totalSupply_;
120 }
121
122 /**
123  * @dev transfer token for a specified address
124  * @param _to The address to transfer to.
125  * @param _value The amount to be transferred.
126  */
127 //@CTK NO_OVERFLOW
128 //@CTK NO_BUF_OVERFLOW
129 //@CTK FAIL NO_ASF
130 /*@CTK transfer
131     @tag assume_completion
132     @pre _to != address(0)
133     @pre _value <= balances[msg.sender]
134     @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
135     @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
136         - _value)
137     @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
138     @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
139     @post __return == true
140 */
141 function transfer(address _to, uint256 _value) public returns (bool) {
142     require(_to != address(0));
143     require(_value <= balances[msg.sender]);
144
145     balances[msg.sender] = balances[msg.sender].sub(_value);
146     balances[_to] = balances[_to].add(_value);
147     emit Transfer(msg.sender, _to, _value);
148     return true;
149 }
150
151 /**
152  * @dev Gets the balance of the specified address.
153  * @param _owner The address to query the the balance of.
154  * @return An uint256 representing the amount owned by the passed address.
155  */
156 //@CTK NO_OVERFLOW
157 //@CTK NO_BUF_OVERFLOW
158 //@CTK NO_ASF
159 /*@CTK balanceOf
160     @tag assume_completion
161     @post (__return) == (balances[_owner])
162 */
163 function balanceOf(address _owner) public view returns (uint256) {
164     return balances[_owner];
165 }
166 }
167
168 /**

```

```

169 * @title ERC20 interface
170 * @dev see https://github.com/ethereum/EIPs/issues/20
171 */
172 contract ERC20 is ERC20Basic {
173     function allowance(address owner, address spender)
174         public view returns (uint256);
175
176     function transferFrom(address from, address to, uint256 value)
177         public returns (bool);
178
179     function approve(address spender, uint256 value) public returns (bool);
180     event Approval(
181         address indexed owner,
182         address indexed spender,
183         uint256 value
184     );
185 }
186
187 /**
188 * @title Standard ERC20 token
189 *
190 * @dev Implementation of the basic standard token.
191 * @dev https://github.com/ethereum/EIPs/issues/20
192 * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master
193     /smart_contract/FirstBloodToken.sol
194 */
195 contract StandardToken is ERC20, BasicToken {
196     mapping (address => mapping (address => uint256)) internal allowed;
197
198
199     /**
200     * @dev Transfer tokens from one address to another
201     * @param _from address The address which you want to send tokens from
202     * @param _to address The address which you want to transfer to
203     * @param _value uint256 the amount of tokens to be transferred
204     */
205     //@CTK NO_OVERFLOW
206     //@CTK NO_BUF_OVERFLOW
207     //@CTK FAIL_NO_ASF
208     /*CTK "transferFrom"
209     @tag assume_completion
210     @pre (_to) != (address(0))
211     @pre (_value) <= (balances[_from])
212     @pre (_value) <= (allowed[_from][msg.sender])
213     @post (_from != _to) -> (__post.balances[_to] == (balances[_to] + _value))
214     @post (_from != _to) -> (__post.balances[_from] == (balances[_from] - _value))
215     @post (_from == _to) -> (__post.balances[_to] == balances[_to])
216     @post (_from == _to) -> (__post.balances[_from] == balances[_from])
217     @post (__post.allowed[_from][msg.sender]) == (allowed[_from][msg.sender] - _value)
218     @post (__return) == (true)
219     */
220     function transferFrom(
221         address _from,
222         address _to,
223         uint256 _value
224     )
225     public

```

```

226     returns (bool)
227     {
228         require(_to != address(0));
229         require(_value <= balances[_from]);
230         require(_value <= allowed[_from][msg.sender]);
231
232         balances[_from] = balances[_from].sub(_value);
233         balances[_to] = balances[_to].add(_value);
234         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
235         emit Transfer(_from, _to, _value);
236         return true;
237     }
238
239     /**
240      * @dev Approve the passed address to spend the specified amount of tokens on behalf
241      *       of msg.sender.
242      *
243      * Beware that changing an allowance with this method brings the risk that someone
244      * may use both the old
245      * and the new allowance by unfortunate transaction ordering. One possible solution
246      * to mitigate this
247      * race condition is to first reduce the spender's allowance to 0 and set the
248      * desired value afterwards:
249      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
250      * @param _spender The address which will spend the funds.
251      * @param _value The amount of tokens to be spent.
252      */
253     // @CTK NO_OVERFLOW
254     // @CTK NO_BUF_OVERFLOW
255     // @CTK NO_ASF
256     /*CTK approve
257      @tag assume_completion
258      @post ( __post.allowed[msg.sender][_spender]) == (_value)
259     */
260     function approve(address _spender, uint256 _value) public returns (bool) {
261         allowed[msg.sender][_spender] = _value;
262         emit Approval(msg.sender, _spender, _value);
263         return true;
264     }
265
266     /**
267      * @dev Function to check the amount of tokens that an owner allowed to a spender.
268      * @param _owner address The address which owns the funds.
269      * @param _spender address The address which will spend the funds.
270      * @return A uint256 specifying the amount of tokens still available for the spender
271      *
272      */
273     // @CTK NO_OVERFLOW
274     // @CTK NO_BUF_OVERFLOW
275     // @CTK NO_ASF
276     /*CTK allowance
277      @tag assume_completion
278      @post ( __return ) == (allowed[_owner][_spender])
279     */
280     function allowance(
281         address _owner,
282         address _spender
283     )

```

```

279     public
280     view
281     returns (uint256)
282     {
283         return allowed[_owner][_spender];
284     }
285
286     /**
287     * @dev Increase the amount of tokens that an owner allowed to a spender.
288     *
289     * approve should be called when allowed[_spender] == 0. To increment
290     * allowed value is better to use this function to avoid 2 calls (and wait until
291     * the first transaction is mined)
292     * From MonolithDAO Token.sol
293     * @param _spender The address which will spend the funds.
294     * @param _addedValue The amount of tokens to increase the allowance by.
295     */
296     //@CTK NO_OVERFLOW
297     //@CTK NO_BUF_OVERFLOW
298     //@CTK FAIL_NO_ASF
299     /*CTK increaseApproval
300     @tag assume_completion
301     @pre _spender != 0x0
302     @post ( __post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
303         _addedValue)
304     @post ( __return) == (true)
305     */
306     function increaseApproval(
307         address _spender,
308         uint _addedValue
309     )
310     public
311     returns (bool)
312     {
313         allowed[msg.sender][_spender] = (
314             allowed[msg.sender][_spender].add(_addedValue));
315         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
316         return true;
317     }
318
319     /**
320     * @dev Decrease the amount of tokens that an owner allowed to a spender.
321     *
322     * approve should be called when allowed[_spender] == 0. To decrement
323     * allowed value is better to use this function to avoid 2 calls (and wait until
324     * the first transaction is mined)
325     * From MonolithDAO Token.sol
326     * @param _spender The address which will spend the funds.
327     * @param _subtractedValue The amount of tokens to decrease the allowance by.
328     */
329     //@CTK NO_OVERFLOW
330     //@CTK NO_BUF_OVERFLOW
331     //@CTK NO_ASF
332     /*CTK "decreaseApproval"
333     @tag assume_completion
334     @pre _spender != 0x0
335     @post (_subtractedValue > allowed[msg.sender][_spender]) -> ( __post.allowed[msg.
336         sender][_spender] == 0)

```

```

335     @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (__post.allowed[msg.
        sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue)
336     */
337     function decreaseApproval(
338         address _spender,
339         uint _subtractedValue
340     )
341     public
342     returns (bool)
343     {
344         uint oldValue = allowed[msg.sender][_spender];
345         if (_subtractedValue > oldValue) {
346             allowed[msg.sender][_spender] = 0;
347         } else {
348             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
349         }
350         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
351         return true;
352     }
353 }
354 }
355
356 contract MultiOwnable {
357     mapping (address => bool) owners;
358     address unremovableOwner;
359
360     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
        );
361     event OwnershipExtended(address indexed host, address indexed guest);
362     event OwnershipRemoved(address indexed removedOwner);
363
364     modifier onlyOwner() {
365         require(owners[msg.sender]);
366         _;
367     }
368
369     //@CTK NO_OVERFLOW
370     //@CTK NO_BUF_OVERFLOW
371     //@CTK NO_ASF
372     /*@CTK "MultiOwnable constructor"
373         @tag assume_completion
374         @post __post.owners[msg.sender]
375         @post __post.unremovableOwner == msg.sender
376     */
377     constructor() public {
378         owners[msg.sender] = true;
379         unremovableOwner = msg.sender;
380     }
381
382     //@CTK NO_OVERFLOW
383     //@CTK NO_BUF_OVERFLOW
384     //@CTK NO_ASF
385     /*@CTK addOwner
386         @tag assume_completion
387         @pre owners[msg.sender]
388         @pre guest != address(0)
389         @post __post.owners[guest]
390     */

```

```

391 function addOwner(address guest) onlyOwner public {
392     require(guest != address(0));
393     owners[guest] = true;
394     emit OwnershipExtended(msg.sender, guest);
395 }
396
397 //@CTK NO_OVERFLOW
398 //@CTK NO_BUF_OVERFLOW
399 //@CTK NO_ASF
400 /*@CTK removeOwner
401     @tag assume_completion
402     @pre owners[msg.sender]
403     @pre removedOwner != address(0)
404     @pre unremovableOwner != removedOwner
405     @post !(__post.owners[removedOwner])
406 */
407 function removeOwner(address removedOwner) onlyOwner public {
408     require(removedOwner != address(0));
409     require(unremovableOwner != removedOwner);
410     delete owners[removedOwner];
411     emit OwnershipRemoved(removedOwner);
412 }
413
414 //@CTK NO_OVERFLOW
415 //@CTK NO_BUF_OVERFLOW
416 //@CTK NO_ASF
417 /*@CTK transferOwnership
418     @tag assume_completion
419     @pre owners[msg.sender]
420     @pre newOwner != address(0)
421     @pre msg.sender != newOwner
422     @pre unremovableOwner != msg.sender
423     @post (__post.owners[newOwner])
424     @post !(__post.owners[msg.sender])
425 */
426 function transferOwnership(address newOwner) onlyOwner public {
427     require(newOwner != address(0));
428     require(unremovableOwner != msg.sender);
429     owners[newOwner] = true;
430     delete owners[msg.sender];
431     emit OwnershipTransferred(msg.sender, newOwner);
432 }
433
434 //@CTK NO_OVERFLOW
435 //@CTK NO_BUF_OVERFLOW
436 //@CTK NO_ASF
437 /*@CTK isOwner
438     @tag assume_completion
439     @post (__return) == (owners[addr])
440 */
441 function isOwner(address addr) public view returns(bool){
442     return owners[addr];
443 }
444 }
445
446 contract FCTV is StandardToken, MultiOwnable {
447
448     using SafeMath for uint256;

```

```

449
450     uint256 public constant TOTAL_CAP = 600000000;
451
452     string public constant name = "[FCT] FirmaChain Token";
453     string public constant symbol = "FCT";
454     uint256 public constant decimals = 18;
455
456     bool isTransferable = false;
457
458     //@CTK NO_OVERFLOW
459     //@CTK NO_BUF_OVERFLOW
460     /*@CTK "FCTV constructor"
461         @tag assume_completion
462         @post __post.balances[msg.sender] == __post.totalSupply_
463     */
464     constructor() public {
465         totalSupply_ = TOTAL_CAP.mul(10 ** decimals);
466         balances[msg.sender] = totalSupply_;
467         emit Transfer(address(0), msg.sender, balances[msg.sender]);
468     }
469
470     //@CTK NO_OVERFLOW
471     //@CTK NO_BUF_OVERFLOW
472     //@CTK NO_ASF
473     /*@CTK unlock
474         @pre owners[msg.sender]
475         @post (__post.isTransferable) == (true)
476     */
477     function unlock() external onlyOwner {
478         isTransferable = true;
479     }
480
481     //@CTK NO_OVERFLOW
482     //@CTK NO_BUF_OVERFLOW
483     //@CTK NO_ASF
484     /*@CTK lock
485         @pre owners[msg.sender]
486         @post (__post.isTransferable) == (false)
487     */
488     function lock() external onlyOwner {
489         isTransferable = false;
490     }
491
492     //@CTK NO_OVERFLOW
493     //@CTK NO_BUF_OVERFLOW
494     //@CTK FAIL_NO_ASF
495     /*@CTK "FCTV transferFrom"
496         @tag assume_completion
497         @post (isTransferable || owners[msg.sender])
498         @post ((_from) != (_to)) -> ((__post.balances[_to]) == ((balances[_to]) + (
499             _value)))
500             @post ((_from) != (_to)) -> ((__post.balances[_from]) == ((balances[_from]) - (
501                 _value)))
502             @post ((_from) == (_to)) -> ((__post.balances[_to]) == (balances[_to]))
503             @post ((_from) == (_to)) -> ((__post.balances[_from]) == (balances[_from]))
504             @post (__post.allowed[_from][msg.sender]) == ((allowed[_from][msg.sender]) - (
505                 _value))
506         @post (__return) == (true)

```

```

504  */
505  function transferFrom(address _from, address _to, uint256 _value) public returns (
506      bool) {
507      require(isTransferable || owners[msg.sender]);
508      return super.transferFrom(_from, _to, _value);
509  }
510  // @CTK NO_OVERFLOW
511  // @CTK NO_BUF_OVERFLOW
512  // @CTK FAIL_NO_ASF
513  /* @CTK "FCTV transfer"
514     @tag assume_completion
515     @post (isTransferable || owners[msg.sender])
516     @post ((_to) != (msg.sender)) -> (__post.balances[msg.sender]) == (balances[msg.
517         sender] - _value)
518     @post ((_to) != (msg.sender)) -> (__post.balances[_to]) == (balances[_to] +
519         _value)
520     @post ((_to) == (msg.sender)) -> (__post.balances[msg.sender]) == balances[msg.
521         sender]
522     @post ((_to) == (msg.sender)) -> (__post.balances[_to]) == balances[_to]
523     @post (__return) == (true)
524  */
525  function transfer(address _to, uint256 _value) public returns (bool) {
526      require(isTransferable || owners[msg.sender]);
527      return super.transfer(_to, _value);
528  }
529  // NOTE: _amount of 1 FCT is 10 ** decimals
530  // @CTK NO_OVERFLOW
531  // @CTK NO_BUF_OVERFLOW
532  // @CTK FAIL_NO_ASF
533  /* @CTK "FCTV mint"
534     @tag assume_completion
535     @pre owners[msg.sender]
536     @post (_to != address(0))
537     @post (__post.totalSupply_) == ((totalSupply_) + (_amount))
538     @post (__post.balances[_to]) == ((balances[_to]) + (_amount))
539  */
540  function mint(address _to, uint256 _amount) onlyOwner public returns (bool) {
541      require(_to != address(0));
542
543      totalSupply_ = totalSupply_.add(_amount);
544      balances[_to] = balances[_to].add(_amount);
545
546      emit Mint(_to, _amount);
547      emit Transfer(address(0), _to, _amount);
548
549      return true;
550  }
551  // NOTE: _amount of 1 FCT is 10 ** decimals
552  // @CTK NO_OVERFLOW
553  // @CTK NO_BUF_OVERFLOW
554  // @CTK FAIL_NO_ASF
555  /* @CTK "FCTV burn"
556     @tag assume_completion
557     @pre owners[msg.sender]
558     @post (_amount <= balances[msg.sender])

```



```
558     @post (__post.totalSupply_) == ((totalSupply_) - (_amount))
559     @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_amount))
560     */
561     function burn(uint256 _amount) onlyOwner public {
562         require(_amount <= balances[msg.sender]);
563
564         totalSupply_ = totalSupply_.sub(_amount);
565         balances[msg.sender] = balances[msg.sender].sub(_amount);
566
567         emit Burn(msg.sender, _amount);
568         emit Transfer(msg.sender, address(0), _amount);
569     }
570
571     event Mint(address indexed _to, uint256 _amount);
572     event Burn(address indexed _from, uint256 _amount);
573 }
```