CertiK Audit Report For Ocean Protocol

Request Date: 2019-06-25
Revision Date: 2019-07-23
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

About CertiK

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

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

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

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

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by Ocean Protocol. This audit was conducted to discover issues and vulnerabilities in the source code of Ocean Protocol'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 issue 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 conditions, 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 vulnerabilies, but no concern found yet.

Testing Summary

PASS

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

Jul 23, 2019

Type of Issues

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

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

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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **PeerManager.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `f4c6a3474f7e66843a8efcf75571a476f0bdf663291dd701f964ad9e716fe5a8`
- **ValidatorSet.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `95f7a9086fd5e457c02eeba4785347db57c626d85e43be76cc7c70ed7720ae83`
- **IPeerManager.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `6b1dc47bde0205bfd5f2e6c5536d96a18e851feb1d1f3a25c95ee0159811dbf9`
- **IValidatorSet.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `25e4a9eced32603cb8713031e8b5daaa3c26503d8c8fd88dab91afa35b6f15c2`
- **IValidatorSetReporting.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `f5de8cadab728ea969af8b429b92d8a6b242ecc52ba3f63db11f9461c2b5fa7f`
- **TestValidatorSet.sol** (commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$)
  `75653b069f94b441fe7ab65c362a4272ae143c9aee1fa33939ca66a2c4ca8ec4`

Summary

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

The ocean team has demonstrated their professional and knowledgeable understanding of the project, by having 1) a production ready repository with high-quality source code; 2) unit tests covering the majority of its business scenarios; 3) accessible, clean, and accurate readme documents for intentions, functionalities, and responsibilities of the smart contracts.

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.

Documentation

CertiK used the following sources of truth about how Ocean Protocol smart contracts should work:

1. Ocean Protocol Website
2. Ocean Protocol Whitepaper
3. Ocean Protocol Governance Contracts Github Code Base
4. Test Scenarios
5. Developer Guide

All listed sources act as a specification. If we discovered inconsistencies within the actual code behavior, we consulted with the Ocean Protocol team for further discussion and confirmation.

Discussion

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

ValidatorSet.sol (commit $3a0ab8bf825322433cc0052e13f6ca946553db23$, previous)

- INFO The visibility of the state variables are currently defined as `public`, which is to be discussed.
  - (Ocean - Confirmed) The contracts are pretty public by design, most of the data must be public available anyway.
- INFO In the constructor function, we are setting the initial validators without setting the state finalized. However, the `validators = pending` gives the impression that only `finalizeChange` should do such task, as ideally any change before `finalized` should not refresh the state variable `validators`. Depending on the business intention, consider either putting `finalizeChange()` in the constructor or explicitly invoke the function after the initialization.
  - (Ocean - Resolved in issue #28) Fixed by removing the setting of `pending to validators`, it is exclusively done by `finalizeChange` now. Integration test has to follow afterwards. See latest commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$.
- DISCUSSION At the stage when a validator is removed but before the invocation of `finalizeChange()`, such validator will be filtered out thanks to the modifier `isValidator()`. However the `getValidators()` will still return validators including the one removed.
  - (Ocean - Confirmed in issue #29) Mitigation by `documentation` and `unit tests`. This is intended behaviour but was hidden well in the code.

PeerManager.sol (commit $3a0ab8bf825322433cc0052e13f6ca946553db23$, previous)

- $\boxed{\text{INFO}}$ The visibility of the state variables are currently defined as `public`, which is to be discussed.
  - (Ocean - Confirmed) The contracts are pretty public by design, most of the data must be public available anyway.
- $\boxed{\text{INFO}}$ The `connectionAllowed` function signature contains `returns (bool allowed)`, but `allowed` is not used in the function body.
  - (Ocean - Resolved in issue #30) Fixed, we are using `allowed` now. See latest commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$.
- $\boxed{\text{INFO}}$ Recommend exiting earlier when two peers are found during the for loop in `connectionAllowed`.
  - (Ocean - Resolved in issue #31) Fixed, we are using `allowed` now. See latest commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$.
- $\boxed{\text{INFO}}$ For `PeerInfo storage peer = peers[i]`, the `memory` storage class can be used since no state change is made. It enhances readability of the contract by letting maintainers worry less about `peer` variable knowing that the variable is not a reference. However, using `storage` is also reasonable as no new memory will be allocated.
  - (Ocean - Resolved in issue #32) Fixed using `memory` instead of `storage`. See latest commit $03c5b1e39ca0b4626f92c3cc9c98fd51c3af1c21$.
- $\boxed{\text{DISCUSSION}}$ Regarding the function `addPeer`, it adds a new peer node and intentionally set connection to `true` for all existing peers. Depending on the business scenario, will there be a case that a peer node may only have partial connections to other nodes? Current smart contract has no connection setters to modify the configuration. The same code from Parity Permissioning explicitly defines the initial config at the constructor level. Given the fact that the current constructor is empty, we assume our client may intend to have a derived smart contract to overwrite and add additional functionalities.
  - (Ocean - Confirmed in issue #33) `PeerManager` is pretty under developed, we decided **we do not deploy this contract** to the network at all because there is no requirement for it.

Best Practice

Solidity Protocol

✓ Use stable solidity version

✓ Handle possible errors properly when making external calls

✓ Provide error message along with require()

✓ Use modifiers properly

✓ Use events to monitor contract activities

✓ Refer and use libraries properly

✓ No compiler warnings

Privilege Control

✓ Restrict access to sensitive functions

Documentation

✓ Provide project readme and execution guidance

✓ Provide inline comment for function intention

✓ Provide instruction to initialize and execute the test files

Testing

✓ Provide test scripts and coverage for potential scenarios

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

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

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File PeerManager.sol

```
pragma solidity 0.5.10;
```

❗ No compiler version found

TIMESTAMP_DEPENDENCY

Line 108 in File PeerManager.sol

```
            block.timestamp
```

❗ ”block.timestamp” can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

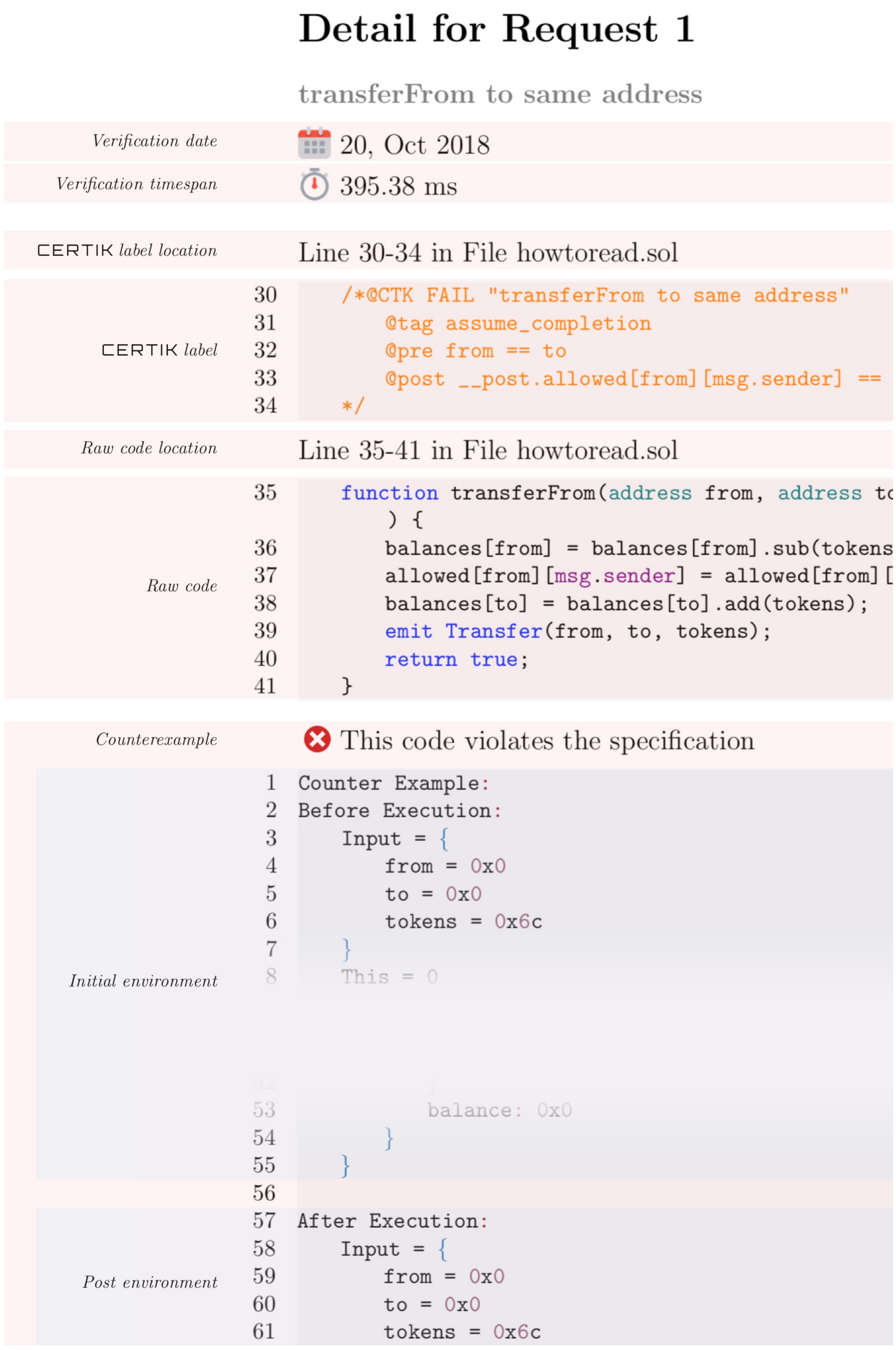
Line 15 in File ValidatorSet.sol

```
pragma solidity 0.5.10;
```

❗ No compiler version found

Formal Verification Results

How to read



Detail for Request 1

transferFrom to same address

Verification date 20, Oct 2018

Verification timespan 395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample This code violates the specification

Initial environment

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

53              balance: 0x0
54          }
55      }
56
```

Post environment

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

Formal Verification Request 1

PeerManager

📅 23, Jul 2019
⏱ 170.37 ms

Line 26-29 in File PeerManager.sol

```
    /*@CTK PeerManager
      @tag assume_completion
      @post __post._owner == _owner
     */
```

Line 30-37 in File PeerManager.sol

```
    function initialize(
        address _owner
    )
        public
        initializer
    {
        Ownable.initialize(_owner);
    }
```

✓ The code meets the specification.

Formal Verification Request 2

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

📅 23, Jul 2019
⏱ 49.31 ms

Line 46 in File PeerManager.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 52-110 in File PeerManager.sol

```
    function addPeer(
        bytes32 _sl,
        bytes32 _sh
    )
        public
        onlyOwner
    {
        /*@IGNORE
        bytes32 peerHash = keccak256(abi.encodePacked(_sh,_sl));

        require(
            !isExist[peerHash],
            'Peer already exists'
        );

        isExist[peerHash] = true;
        @IGNORE*/

        peers[peerCount] = PeerInfo(
            _sl,
            _sh
        );

        bool[] memory newPeer = new bool[](peerCount + 1);

        /*@CTK "addPeer forloop 1"
          @var uint i
          @var PeerManager this
          @var bool[] newPeer
          @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
          @inv this == this__pre
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            newPeer[i] = true;
        }

        allowedConnections.push(newPeer);

        /*@CTK "addPeer forloop 2"
          @var uint i
          @var PeerManager this
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            allowedConnections[i].push(true);
        }

        peerCount++;

        emit PeerAdded(
            _sl,
            _sh,
            /* solium-disable-next-line security/no-block-members */
            block.timestamp
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

23, Jul 2019
0.73 ms

Line 47 in File PeerManager.sol

```
    //@CTK NO_ASF
```

Line 52-110 in File PeerManager.sol

```
    function addPeer(
        bytes32 _sl,
        bytes32 _sh
    )
        public
        onlyOwner
    {
        /*@IGNORE
        bytes32 peerHash = keccak256(abi.encodePacked(_sh,_sl));

        require(
            !isExist[peerHash],
            'Peer already exists'
        );

        isExist[peerHash] = true;
        @IGNORE*/

        peers[peerCount] = PeerInfo(
            _sl,
            _sh
        );

        bool[] memory newPeer = new bool[](peerCount + 1);

        /*@CTK "addPeer forloop 1"
          @var uint i
          @var PeerManager this
          @var bool[] newPeer
          @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
          @inv this == this__pre
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            newPeer[i] = true;
        }

        allowedConnections.push(newPeer);

        /*@CTK "addPeer forloop 2"
          @var uint i
          @var PeerManager this
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            allowedConnections[i].push(true);
        }

        peerCount++;

        emit PeerAdded(
            _sl,
            _sh,
            /* solium-disable-next-line security/no-block-members */
            block.timestamp
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 4

addPeer

📅 23, Jul 2019
⏱ 3.32 ms

Line 48-51 in File PeerManager.sol

```
/*@CTK addPeer
  @tag assume_completion
  @post __post.peerCount == peerCount + 1
 */
```

Line 52-110 in File PeerManager.sol

```
function addPeer(
    bytes32 _sl,
    bytes32 _sh
)
    public
    onlyOwner
{
    /*@IGNORE
    bytes32 peerHash = keccak256(abi.encodePacked(_sh,_sl));

    require(
        !isExist[peerHash],
        'Peer already exists'
    );

    isExist[peerHash] = true;
    @IGNORE*/

    peers[peerCount] = PeerInfo(
        _sl,
        _sh
    );

    bool[] memory newPeer = new bool[](peerCount + 1);

    /*@CTK "addPeer forloop 1"
      @var uint i
      @var PeerManager this
      @var bool[] newPeer
      @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
      @inv this == this__pre
      @post i >= this.peerCount
      @post !__should_return
     */
    for (uint i = 0; i <= peerCount; i++) {
        newPeer[i] = true;
    }

    allowedConnections.push(newPeer);
```

```
 91
 92         /*@CTK "addPeer forloop 2"
 93           @var uint i
 94           @var PeerManager this
 95           @post i >= this.peerCount
 96           @post !__should_return
 97          */
 98         for (uint i = 0; i <= peerCount; i++) {
 99             allowedConnections[i].push(true);
100         }
101
102         peerCount++;
103
104         emit PeerAdded(
105             _sl,
106             _sh,
107             /* solium-disable-next-line security/no-block-members */
108             block.timestamp
109         );
110     }
```

✅ The code meets the specification.

Formal Verification Request 5

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

📅 23, Jul 2019
⏱ 41.71 ms

Line 122 in File PeerManager.sol

```
122     //@CTK NO_BUF_OVERFLOW
```

Line 127-182 in File PeerManager.sol

```
127     function connectionAllowed(
128         bytes32 sl,
129         bytes32 sh,
130         bytes32 pl,
131         bytes32 ph
132     )
133         public view
134         returns (bool allowed)
135     {
136         uint index1 = 0;
137         bool index1_found = false;
138         uint index2 = 0;
139         bool index2_found = false;
140
141         allowed = false;
142         /*@CTK "connectionAllowed ForLoop"
143           @var uint index1
144           @var bool index1_found
145           @var uint index2
146           @var bool index2_found
147           @pre index1_found == false
148           @pre index2_found == false
149          @inv i <= peerCount
150          @inv index1 < peerCount
151          @inv index2 < peerCount
152          @inv this.peers == this__pre.peers
153          @inv this.peerCount == this__pre.peerCount
154          @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
                 peers[j].publicLow) -> (index1_found == true && index1 == j)
155          @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
                 peers[j].publicLow) -> (index2_found == true && index2 == j)
156          @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
                 peers[j].publicHigh /\ sl == peers[j].publicLow)
157          @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
                 peers[j].publicHigh /\ pl == peers[j].publicLow)
158          @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
                 peers[j].publicHigh /\ sl == peers[j].publicLow)
159          @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
                 peers[j].publicHigh /\ pl == peers[j].publicLow)
160          @post !__should_return
161         */
162        for (uint i = 0; i < peerCount; i++) {
163            PeerInfo memory peer = peers[i];
164
165            if (sh == peer.publicHigh && sl == peer.publicLow) {
166                index1 = i;
167                index1_found = true;
168            }
169
170            if (ph == peer.publicHigh && pl == peer.publicLow) {
171                index2 = i;
172                index2_found = true;
173            }
174
175            if (index1_found && index2_found) {
176                allowed = allowedConnections[index1][index2];
177                break;
178            }
179        }
180
181        return allowed;
182    }
```

✅ The code meets the specification.

Formal Verification Request 6

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 0.59 ms

Line 123 in File PeerManager.sol

```
123    //@CTK NO_ASF
```

Line 127-182 in File PeerManager.sol

```
127    function connectionAllowed(
128        bytes32 sl,
        bytes32 sh,
        bytes32 pl,
        bytes32 ph
    )
        public view
        returns (bool allowed)
    {
        uint index1 = 0;
        bool index1_found = false;
        uint index2 = 0;
        bool index2_found = false;

        allowed = false;
        /*@CTK "connectionAllowed ForLoop"
          @var uint index1
          @var bool index1_found
          @var uint index2
          @var bool index2_found
          @pre index1_found == false
          @pre index2_found == false
          @inv i <= peerCount
          @inv index1 < peerCount
          @inv index2 < peerCount
          @inv this.peers == this__pre.peers
          @inv this.peerCount == this__pre.peerCount
          @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
              peers[j].publicLow) -> (index1_found == true && index1 == j)
          @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
              peers[j].publicLow) -> (index2_found == true && index2 == j)
          @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
          @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
              peers[j].publicHigh /\ pl == peers[j].publicLow)
          @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
          @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
              peers[j].publicHigh /\ pl == peers[j].publicLow)
          @post !__should_return
         */
        for (uint i = 0; i < peerCount; i++) {
            PeerInfo memory peer = peers[i];

            if (sh == peer.publicHigh && sl == peer.publicLow) {
                index1 = i;
                index1_found = true;
            }

            if (ph == peer.publicHigh && pl == peer.publicLow) {
                index2 = i;
                index2_found = true;
            }

            if (index1_found && index2_found) {
                allowed = allowedConnections[index1][index2];
                break;
            }
        }

        return allowed;
    }
```

✓ The code meets the specification.

Formal Verification Request 7

connectionAllowed

23, Jul 2019
0.51 ms

Line 124-126 in File PeerManager.sol

```
    /*@CTK connectionAllowed
      @tag assume_completion
     */
```

Line 127-182 in File PeerManager.sol

```
    function connectionAllowed(
        bytes32 sl,
        bytes32 sh,
        bytes32 pl,
        bytes32 ph
    )
        public view
        returns (bool allowed)
    {
        uint index1 = 0;
        bool index1_found = false;
        uint index2 = 0;
        bool index2_found = false;

        allowed = false;
        /*@CTK "connectionAllowed ForLoop"
          @var uint index1
          @var bool index1_found
          @var uint index2
          @var bool index2_found
          @pre index1_found == false
          @pre index2_found == false
          @inv i <= peerCount
          @inv index1 < peerCount
          @inv index2 < peerCount
          @inv this.peers == this__pre.peers
          @inv this.peerCount == this__pre.peerCount
          @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
              peers[j].publicLow) -> (index1_found == true && index1 == j)
          @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
              peers[j].publicLow) -> (index2_found == true && index2 == j)
          @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
          @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
              peers[j].publicHigh /\ pl == peers[j].publicLow)
          @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
```

```
      @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
          peers[j].publicHigh /\ pl == peers[j].publicLow)
      @post !__should_return
     */
    for (uint i = 0; i < peerCount; i++) {
        PeerInfo memory peer = peers[i];

        if (sh == peer.publicHigh && sl == peer.publicLow) {
            index1 = i;
            index1_found = true;
        }

        if (ph == peer.publicHigh && pl == peer.publicLow) {
            index2 = i;
            index2_found = true;
        }

        if (index1_found && index2_found) {
            allowed = allowedConnections[index1][index2];
            break;
        }
    }

    return allowed;
}
```

✓ The code meets the specification.

Formal Verification Request 8

addPeer forloop 1__Generated

📅 23, Jul 2019
⏱ 37.79 ms

(Loop) Line 77-85 in File PeerManager.sol

```
    /*@CTK "addPeer forloop 1"
      @var uint i
      @var PeerManager this
      @var bool[] newPeer
      @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
      @inv this == this__pre
      @post i >= this.peerCount
      @post !__should_return
     */
```

(Loop) Line 77-88 in File PeerManager.sol

```
    /*@CTK "addPeer forloop 1"
      @var uint i
      @var PeerManager this
      @var bool[] newPeer
      @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
      @inv this == this__pre
      @post i >= this.peerCount
      @post !__should_return
     */
```

```
        for (uint i = 0; i <= peerCount; i++) {
            newPeer[i] = true;
        }
```

✓ The code meets the specification.

Formal Verification Request 9

addPeer forloop 2__Generated

23, Jul 2019
20.45 ms

(Loop) Line 92-97 in File PeerManager.sol

```
        /*@CTK "addPeer forloop 2"
          @var uint i
          @var PeerManager this
          @post i >= this.peerCount
          @post !__should_return
         */
```

(Loop) Line 92-100 in File PeerManager.sol

```
        /*@CTK "addPeer forloop 2"
          @var uint i
          @var PeerManager this
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            allowedConnections[i].push(true);
        }
```

✓ The code meets the specification.

Formal Verification Request 10

connectionAllowed ForLoop__Generated

23, Jul 2019
443.91 ms

(Loop) Line 142-161 in File PeerManager.sol

```
        /*@CTK "connectionAllowed ForLoop"
          @var uint index1
          @var bool index1_found
          @var uint index2
          @var bool index2_found
          @pre index1_found == false
          @pre index2_found == false
          @inv i <= peerCount
          @inv index1 < peerCount
          @inv index2 < peerCount
          @inv this.peers == this__pre.peers
```

```
         @inv this.peerCount == this__pre.peerCount
         @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
             peers[j].publicLow) -> (index1_found == true && index1 == j)
         @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
             peers[j].publicLow) -> (index2_found == true && index2 == j)
         @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
             peers[j].publicHigh /\ sl == peers[j].publicLow)
         @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
             peers[j].publicHigh /\ pl == peers[j].publicLow)
         @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
             peers[j].publicHigh /\ sl == peers[j].publicLow)
         @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
             peers[j].publicHigh /\ pl == peers[j].publicLow)
         @post !__should_return
        */
```

(Loop) Line 142-179 in File PeerManager.sol

```
       /*@CTK "connectionAllowed ForLoop"
         @var uint index1
         @var bool index1_found
         @var uint index2
         @var bool index2_found
         @pre index1_found == false
         @pre index2_found == false
         @inv i <= peerCount
         @inv index1 < peerCount
         @inv index2 < peerCount
         @inv this.peers == this__pre.peers
         @inv this.peerCount == this__pre.peerCount
         @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
             peers[j].publicLow) -> (index1_found == true && index1 == j)
         @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
             peers[j].publicLow) -> (index2_found == true && index2 == j)
         @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
             peers[j].publicHigh /\ sl == peers[j].publicLow)
         @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
             peers[j].publicHigh /\ pl == peers[j].publicLow)
         @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
             peers[j].publicHigh /\ sl == peers[j].publicLow)
         @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
             peers[j].publicHigh /\ pl == peers[j].publicLow)
         @post !__should_return
        */
       for (uint i = 0; i < peerCount; i++) {
           PeerInfo memory peer = peers[i];

           if (sh == peer.publicHigh && sl == peer.publicLow) {
               index1 = i;
               index1_found = true;
           }

           if (ph == peer.publicHigh && pl == peer.publicLow) {
               index2 = i;
               index2_found = true;
           }

           if (index1_found && index2_found) {
               allowed = allowedConnections[index1][index2];
```

```
177                 break;
178             }
179         }
```

✓ The code meets the specification.

Formal Verification Request 11

If method completes, integer overflow would not happen.

23, Jul 2019
189.14 ms

Line 130 in File ValidatorSet.sol

```
130     //@CTK NO_OVERFLOW
```

Line 141-173 in File ValidatorSet.sol

```
141     function initialize(
142         address _owner,
143         address[] memory _initial
144     )
145         public
146         initializer
147     {
148         Ownable.initialize(_owner);
149         pending = _initial;
150 
151         recentBlocks = 20;
152 
153         // Don't touch it! It is 2 ^ 160 - 2, the systems signer account.
154         // As stated hereL https://wiki.parity.io/Validator-Set#contracts
155         systemAddress = 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE;
156 
157         /*@CTK "constructor ForLoop"
158           @pre forall j: uint. _initial[j] != 0x0
159           @inv i <= _initial.length
160           @inv _initial == _initial__pre
161           @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
                  true
162           @post i == _initial.length
163           @post !__should_return
164          */
165         for (uint i = 0; i < _initial.length; i++) {
166             require(
167                 _initial[i] != address(0),
168                 'Invalid validator address'
169             );
170             status[_initial[i]].isIn = true;
171             status[_initial[i]].index = i;
172         }
173     }
```

✓ The code meets the specification.

Formal Verification Request 12

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

📅 23, Jul 2019
⏱ 22.14 ms

Line 131 in File ValidatorSet.sol

```
131     //@CTK NO_BUF_OVERFLOW
```

Line 141-173 in File ValidatorSet.sol

```
141     function initialize(
142         address _owner,
143         address[] memory _initial
144     )
145         public
146         initializer
147     {
148         Ownable.initialize(_owner);
149         pending = _initial;
150 
151         recentBlocks = 20;
152 
153         // Don’t touch it! It is 2 ^ 160 - 2, the systems signer account.
154         // As stated hereL https://wiki.parity.io/Validator-Set#contracts
155         systemAddress = 0xffffFFFfFFffffffffffffFfFFFffffFFFfFFFfE;
156 
157         /*@CTK "constructor ForLoop"
158           @pre forall j: uint. _initial[j] != 0x0
159           @inv i <= _initial.length
160           @inv _initial == _initial__pre
161           @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
                  true
162           @post i == _initial.length
163           @post !__should_return
164          */
165         for (uint i = 0; i < _initial.length; i++) {
166             require(
167                 _initial[i] != address(0),
168                 'Invalid validator address'
169             );
170             status[_initial[i]].isIn = true;
171             status[_initial[i]].index = i;
172         }
173     }
```

✅ The code meets the specification.

Formal Verification Request 13

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 20.52 ms

Line 132 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 141-173 in File ValidatorSet.sol

```
    function initialize(
        address _owner,
        address[] memory _initial
    )
        public
        initializer
    {
        Ownable.initialize(_owner);
        pending = _initial;

        recentBlocks = 20;

        // Don't touch it! It is 2 ^ 160 - 2, the systems signer account.
        // As stated hereL https://wiki.parity.io/Validator-Set#contracts
        systemAddress = 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE;

        /*@CTK "constructor ForLoop"
          @pre forall j: uint. _initial[j] != 0x0
          @inv i <= _initial.length
          @inv _initial == _initial__pre
          @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
              true
          @post i == _initial.length
          @post !__should_return
         */
        for (uint i = 0; i < _initial.length; i++) {
            require(
                _initial[i] != address(0),
                'Invalid validator address'
            );
            status[_initial[i]].isIn = true;
            status[_initial[i]].index = i;
        }
    }
```

✓ The code meets the specification.

Formal Verification Request 14

constructor

📅 23, Jul 2019
⏱ 28.4 ms

Line 133-140 in File ValidatorSet.sol

```
    /*@CTK "constructor"
      @tag assume_completion
      @post __post.systemAddress == 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE
      @post __post.recentBlocks == 20
      @post __post._owner == _owner
      @post __post.pending == _initial
      @post forall i: uint. (i >= 0 && i < _initial.length) -> (__post.status[_initial
          [i]].isIn == true)
```

```
     */
```

Line 141-173 in File ValidatorSet.sol

```
    function initialize(
        address _owner,
        address[] memory _initial
    )
        public
        initializer
    {
        Ownable.initialize(_owner);
        pending = _initial;

        recentBlocks = 20;

        // Don't touch it! It is 2 ^ 160 - 2, the systems signer account.
        // As stated hereL https://wiki.parity.io/Validator-Set#contracts
        systemAddress = 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE;

        /*@CTK "constructor ForLoop"
          @pre forall j: uint. _initial[j] != 0x0
          @inv i <= _initial.length
          @inv _initial == _initial__pre
          @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
              true
          @post i == _initial.length
          @post !__should_return
         */
        for (uint i = 0; i < _initial.length; i++) {
            require(
                _initial[i] != address(0),
                'Invalid validator address'
            );
            status[_initial[i]].isIn = true;
            status[_initial[i]].index = i;
        }
    }
```

✓ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

23, Jul 2019
117.29 ms

Line 185 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 202-220 in File ValidatorSet.sol

```
    function addValidator(
        address _validator
    )
        external
        onlyOwner
        isNotValidator(_validator)
    {
        require(
            _validator != address(0),
            'Invalid validator address'
        );

        status[_validator].isIn = true;
        status[_validator].index = pending.length;

        pending.push(_validator);

        triggerChange();
    }
```

✓ The code meets the specification.

Formal Verification Request 16

addValidator

📅 23, Jul 2019
⏱ 9.08 ms

Line 186-201 in File ValidatorSet.sol

```
    /*@CTK "addValidator"
      @tag assume_completion
      @post _owner == msg.sender
      @pre forall addr: address. pending[status[addr].index] == addr
      @pre forall i: uint. status[pending[i]].isIn == true && status[pending[i]].index
           == i
      @post forall addr: address. __post.pending[__post.status[addr].index] == addr
      @post forall i: uint. __post.status[__post.pending[i]].isIn == true && __post.
          status[__post.pending[i]].index == i
      @post status[_validator].isIn == false
      @post _validator != 0
      @post __post.status[_validator].isIn == true
      @post __post.status[_validator].index == pending.length
      @post __post.pending.length == pending.length + 1
      @post __post.pending[__post.status[_validator].index] == _validator
      @post finalized == true
      @post __post.finalized == false
     */
```

Line 202-220 in File ValidatorSet.sol

```
    function addValidator(
        address _validator
    )
        external
        onlyOwner
        isNotValidator(_validator)
    {
        require(
            _validator != address(0),
            'Invalid validator address'
```

```
212         );
213 
214         status[_validator].isIn = true;
215         status[_validator].index = pending.length;
216 
217         pending.push(_validator);
218 
219         triggerChange();
220     }
```

✅ The code meets the specification.

Formal Verification Request 17

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 122.66 ms

Line 230 in File ValidatorSet.sol

```
230     //@CTK NO_ASF
```

Line 251-277 in File ValidatorSet.sol

```
251     function removeValidator(
252         address _validator
253     )
254         external
255         onlyOwner
256         isValidator(_validator)
257     {
258         require(
259             pending.length > 1,
260             'Requires at least one live validator in the system'
261         );
262 
263         // Remove validator from pending by moving the
264         // last element to its slot
265         uint index = status[_validator].index;
266 
267         pending[index] = pending[pending.length - 1];
268         status[pending[index]].index = index;
269 
270         delete pending[pending.length - 1];
271         pending.length--;
272 
273         // Reset address status including `isIn` and it's `index`
274         delete status[_validator];
275 
276         triggerChange();
277     }
```

✅ The code meets the specification.

Formal Verification Request 18

removeValidator

23, Jul 2019
595.17 ms

Line 231-250 in File ValidatorSet.sol

```
    /*@CTK "removeValidator"
      @tag assume_completion
      @pre forall addr: address. pending[status[addr].index] == addr
      @pre forall i: uint. status[pending[i]].isIn == true && status[pending[i]].index
          == i
      @post _owner == msg.sender
      @post pending.length > 1
      @post validators[status[_validator].index] == _validator
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post finalized == true
      @post __post.status[_validator].isIn == false
      @post __post.status[_validator].index == 0
      @post __post.pending.length == pending.length - 1
      @post __post.pending[pending.length - 1] == 0x0
      @post status[_validator].index != pending.length - 1
           -> __post.pending[status[_validator].index] == pending[pending.length - 1]
           && __post.status[pending[pending.length - 1]].index == status[_validator].
               index
           && __post.status[pending[pending.length - 1]].isIn == true
      @post __post.finalized == false
     */
```

Line 251-277 in File ValidatorSet.sol

```
    function removeValidator(
        address _validator
    )
        external
        onlyOwner
        isValidator(_validator)
    {
        require(
            pending.length > 1,
            'Requires at least one live validator in the system'
        );

        // Remove validator from pending by moving the
        // last element to its slot
        uint index = status[_validator].index;

        pending[index] = pending[pending.length - 1];
        status[pending[index]].index = index;

        delete pending[pending.length - 1];
        pending.length--;

        // Reset address status including ‘isIn‘ and it’s ‘index‘
        delete status[_validator];

```

```
276             triggerChange();
277         }
```

The code meets the specification.

Formal Verification Request 19

If method completes, integer overflow would not happen.

23, Jul 2019
20.99 ms

Line 286 in File ValidatorSet.sol

```
286         //@CTK NO_OVERFLOW
```

Line 294-301 in File ValidatorSet.sol

```
294         function setRecentBlocks(
295             uint _recentBlocks
296         )
297             external
298             onlyOwner
299         {
300             recentBlocks = _recentBlocks;
301         }
```

The code meets the specification.

Formal Verification Request 20

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

23, Jul 2019
0.49 ms

Line 287 in File ValidatorSet.sol

```
287         //@CTK NO_BUF_OVERFLOW
```

Line 294-301 in File ValidatorSet.sol

```
294         function setRecentBlocks(
295             uint _recentBlocks
296         )
297             external
298             onlyOwner
299         {
300             recentBlocks = _recentBlocks;
301         }
```

The code meets the specification.

Formal Verification Request 21

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 0.48 ms

Line 288 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 294-301 in File ValidatorSet.sol

```
    function setRecentBlocks(
        uint _recentBlocks
    )
        external
        onlyOwner
    {
        recentBlocks = _recentBlocks;
    }
```

✓ The code meets the specification.

Formal Verification Request 22

setRecentBlocks

📅 23, Jul 2019
⏱ 1.01 ms

Line 289-293 in File ValidatorSet.sol

```
    /*@CTK "setRecentBlocks"
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.recentBlocks == _recentBlocks
     */
```

Line 294-301 in File ValidatorSet.sol

```
    function setRecentBlocks(
        uint _recentBlocks
    )
        external
        onlyOwner
    {
        recentBlocks = _recentBlocks;
    }
```

✓ The code meets the specification.

Formal Verification Request 23

If method completes, integer overflow would not happen.

📅 23, Jul 2019
⏱ 5.45 ms

Line 308 in File ValidatorSet.sol

```
     //@CTK NO_OVERFLOW
```

Line 314-319 in File ValidatorSet.sol

```
    function getValidators()
        external view
        returns (address[] memory _validators)
    {
        return validators;
    }
```

✓ The code meets the specification.

Formal Verification Request 24

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

23, Jul 2019
0.33 ms

Line 309 in File ValidatorSet.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 314-319 in File ValidatorSet.sol

```
    function getValidators()
        external view
        returns (address[] memory _validators)
    {
        return validators;
    }
```

✓ The code meets the specification.

Formal Verification Request 25

Method will not encounter an assertion failure.

23, Jul 2019
0.35 ms

Line 310 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 314-319 in File ValidatorSet.sol

```
    function getValidators()
        external view
        returns (address[] memory _validators)
    {
        return validators;
    }
```

✓ The code meets the specification.

Formal Verification Request 26

getValidators

23, Jul 2019
0.36 ms

Line 311-313 in File ValidatorSet.sol

```
    /*@CTK "getValidators"
      @post _validators == validators
     */
```

Line 314-319 in File ValidatorSet.sol

```
    function getValidators()
        external view
        returns (address[] memory _validators)
    {
        return validators;
    }
```

✓ The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

23, Jul 2019
5.54 ms

Line 326 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 330-335 in File ValidatorSet.sol

```
    function getPending()
        external view
        returns (address[] memory)
    {
        return pending;
    }
```

✓ The code meets the specification.

Formal Verification Request 28

getPending

23, Jul 2019
0.46 ms

Line 327-329 in File ValidatorSet.sol

```
    /*@CTK "getPending"
      @post __return == pending
     */
```

Line 330-335 in File ValidatorSet.sol

```
    function getPending()
        external view
        returns (address[] memory)
    {
        return pending;
    }
```

✓ The code meets the specification.

Formal Verification Request 29

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 61.35 ms

Line 344 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 356-370 in File ValidatorSet.sol

```
    function reportBenign(
        address _validator,
        uint _blockNumber
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            false
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 30

reportBenign

📅 23, Jul 2019
⏱ 25.26 ms

Line 345-355 in File ValidatorSet.sol

```
    /*@CTK "reportBenign"
      @tag assume_completion
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post validators[status[_validator].index] == _validator
      @post status[msg.sender].isIn == true
      @post status[msg.sender].index < validators.length
      @post validators[status[msg.sender].index] == msg.sender
      @post _blockNumber < block.number
      @post _blockNumber + recentBlocks >= block.number
     */
```

Line 356-370 in File ValidatorSet.sol

```
    function reportBenign(
        address _validator,
        uint _blockNumber
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            false
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 59.58 ms

Line 383 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 395-410 in File ValidatorSet.sol

```
    function reportMalicious(
        address _validator,
        uint _blockNumber,
        bytes calldata _proof
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            true
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 32

reportMalicious

📅 23, Jul 2019
⏱ 25.19 ms

Line 384-394 in File ValidatorSet.sol

```
    /*@CTK "reportMalicious"
      @tag assume_completion
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post validators[status[_validator].index] == _validator
      @post status[msg.sender].isIn == true
      @post status[msg.sender].index < validators.length
      @post validators[status[msg.sender].index] == msg.sender
      @post _blockNumber < block.number
      @post _blockNumber + recentBlocks >= block.number
     */
```

Line 395-410 in File ValidatorSet.sol

```
    function reportMalicious(
        address _validator,
        uint _blockNumber,
        bytes calldata _proof
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            true
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 33

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 30.75 ms

Line 417 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 425-433 in File ValidatorSet.sol

```
    function finalizeChange()
        external
        whenNotFinalized
        onlySystem
    {
        validators = pending;
        finalized = true;
        emit ChangeFinalized(validators);
    }
```

✅ The code meets the specification.

Formal Verification Request 34

finalizeChange

📅 23, Jul 2019
⏱ 3.31 ms

Line 418-424 in File ValidatorSet.sol

```
    /*@CTK "finalizeChange"
      @tag assume_completion
      @post systemAddress == msg.sender
      @post finalized == false
      @post __post.finalized == true
      @post __post.validators == pending
     */
```

Line 425-433 in File ValidatorSet.sol

```
    function finalizeChange()
        external
        whenNotFinalized
        onlySystem
    {
        validators = pending;
        finalized = true;
        emit ChangeFinalized(validators);
    }
```

✅ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

📅 23, Jul 2019
⏱ 0.45 ms

Line 441 in File ValidatorSet.sol

```
    //@CTK NO_ASF
```

Line 447-456 in File ValidatorSet.sol

```
    function triggerChange()
        private
        whenFinalized
    {
        finalized = false;
        emit InitiateChange(
            blockhash(block.number - 1),
            pending
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 36

triggerChange

23, Jul 2019
1.78 ms

Line 442-446 in File ValidatorSet.sol

```
    /*@CTK triggerChange
      @tag assume_completion
      @post finalized == true
      @post __post.finalized == false
     */
```

Line 447-456 in File ValidatorSet.sol

```
    function triggerChange()
        private
        whenFinalized
    {
        finalized = false;
        emit InitiateChange(
            blockhash(block.number - 1),
            pending
        );
    }
```

✓ The code meets the specification.

Formal Verification Request 37

constructor ForLoop__Generated

23, Jul 2019
133.23 ms

(Loop) Line 157-164 in File ValidatorSet.sol

```
        /*@CTK "constructor ForLoop"
          @pre forall j: uint. _initial[j] != 0x0
          @inv i <= _initial.length
          @inv _initial == _initial__pre
          @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
              true
          @post i == _initial.length
          @post !__should_return
         */
```

(Loop) Line 157-172 in File ValidatorSet.sol

```
        /*@CTK "constructor ForLoop"
          @pre forall j: uint. _initial[j] != 0x0
          @inv i <= _initial.length
          @inv _initial == _initial__pre
          @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
              true
          @post i == _initial.length
          @post !__should_return
         */
        for (uint i = 0; i < _initial.length; i++) {
            require(
                _initial[i] != address(0),
                'Invalid validator address'
            );
            status[_initial[i]].isIn = true;
            status[_initial[i]].index = i;
        }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File PeerManager.sol

```
pragma solidity 0.5.10;

import 'openzeppelin-eth/contracts/ownership/Ownable.sol';
import './interfaces/IPeerManager.sol';

/**
 * @title PeerManager
 * @dev PeerManager contract manages the peer life cycle including
 * adding peers and allowed connections between peers
 * TODO: The current implementation does not include removing peers
 * TODO: It also assumes that all peers have the same allowed connections
 */

contract PeerManager is Ownable, IPeerManager {

    struct PeerInfo {
        bytes32 publicLow;
        bytes32 publicHigh;
    }

    mapping(uint => PeerInfo) public peers;
    bool[][] public allowedConnections;
    uint public peerCount;
    mapping(bytes32 => bool) isExist;

    /*@CTK PeerManager
      @tag assume_completion
      @post __post._owner == _owner
     */
    function initialize(
        address _owner
    )
        public
        initializer
    {
        Ownable.initialize(_owner);
    }

    /**
     * @dev addPeer adds peers to peer registry where each peer is
     * represented by node address a 64 bytes long. For more technical
     * information please refer to https://wiki.parity.io/Permissioning
     * @param _sl refers to lower 32 bytes of the node address
     * @param _sh refers to higher 32 bytes of the node address
     */
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK addPeer
      @tag assume_completion
      @post __post.peerCount == peerCount + 1
     */
    function addPeer(
        bytes32 _sl,
        bytes32 _sh
    )
        public
        onlyOwner
    {
        bytes32 peerHash = keccak256(abi.encodePacked(_sh,_sl));

        require(
            !isExist[peerHash],
            'Peer already exists'
        );

        isExist[peerHash] = true;

        peers[peerCount] = PeerInfo(
            _sl,
            _sh
        );

        bool[] memory newPeer = new bool[](peerCount + 1);

        /*@CTK "addPeer forloop 1"
          @var uint i
          @var PeerManager this
          @var bool[] newPeer
          @inv forall j: uint. (j >= 0 /\ j < i) -> newPeer[j] == true
          @inv this == this__pre
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            newPeer[i] = true;
        }

        allowedConnections.push(newPeer);

        /*@CTK "addPeer forloop 2"
          @var uint i
          @var PeerManager this
          @post i >= this.peerCount
          @post !__should_return
         */
        for (uint i = 0; i <= peerCount; i++) {
            allowedConnections[i].push(true);
        }

        peerCount++;

        emit PeerAdded(
            _sl,
            _sh,
            /* solium-disable-next-line security/no-block-members */
            block.timestamp
        );
    }

    /**
     * @dev connectionAllowed check if the connection between two peers
     * is allowed or not. For more info, please refer to this documentation
     * https://wiki.parity.io/Permissioning
     * @param sl refers to lower 32 bytes of the node address
     * @param sh refers to higher 32 bytes of the node address
     * @param pl peer public low address (lower 32 bytes)
     * @param ph peer public high address (higher 32 bytes)
     * @return true if connection is allowed
     */
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK connectionAllowed
      @tag assume_completion
     */
    function connectionAllowed(
        bytes32 sl,
        bytes32 sh,
        bytes32 pl,
        bytes32 ph
    )
        public view
        returns (bool allowed)
    {
        uint index1 = 0;
        bool index1_found = false;
        uint index2 = 0;
        bool index2_found = false;

        allowed = false;
        /*@CTK "connectionAllowed ForLoop"
          @var uint index1
          @var bool index1_found
          @var uint index2
          @var bool index2_found
          @pre index1_found == false
          @pre index2_found == false
          @inv i <= peerCount
          @inv index1 < peerCount
          @inv index2 < peerCount
          @inv this.peers == this__pre.peers
          @inv this.peerCount == this__pre.peerCount
          @inv exists j: uint. (j >= 0 /\ j < i /\ sh == peers[j].publicHigh /\ sl ==
              peers[j].publicLow) -> (index1_found == true && index1 == j)
          @inv exists j: uint. (j >= 0 /\ j < i /\ ph == peers[j].publicHigh /\ pl ==
              peers[j].publicLow) -> (index2_found == true && index2 == j)
          @inv index1_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
          @inv index2_found == false -> forall j: uint. (j >= 0 /\ j < i) -> ~(ph ==
              peers[j].publicHigh /\ pl == peers[j].publicLow)
          @post !index1_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(sh ==
              peers[j].publicHigh /\ sl == peers[j].publicLow)
          @post !index2_found -> forall j: uint. (j >= 0 /\ j < peerCount) -> ~(ph ==
              peers[j].publicHigh /\ pl == peers[j].publicLow)
          @post !__should_return
         */
        for (uint i = 0; i < peerCount; i++) {
            PeerInfo memory peer = peers[i];

            if (sh == peer.publicHigh && sl == peer.publicLow) {
                index1 = i;
                index1_found = true;
            }

            if (ph == peer.publicHigh && pl == peer.publicLow) {
                index2 = i;
                index2_found = true;
            }

            if (index1_found && index2_found) {
                allowed = allowedConnections[index1][index2];
                break;
            }
        }

        return allowed;
    }
}
```

File ValidatorSet.sol

```
// Copyright 2018, Parity Technologies Ltd.
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//    http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

pragma solidity 0.5.10;

import 'openzeppelin-eth/contracts/ownership/Ownable.sol';
import './interfaces/IValidatorSet.sol';
import './interfaces/IValidatorSetReporting.sol';

/**
 * @title ValidatorSet
 * @dev an owned validator set contract where the owner can add or remove validators.
 * This is an abstract contract that provides the base logic for adding/removing
 * validators and provides base implementations for the `ValidatorSet`
 * interface. The base implementations of the misbehavior reporting functions
 * perform validation on the reported and reporter validators according to the
 * currently active validator set. The base implementation of `finalizeChange`
 * validates that there are existing unfinalized changes.
 *
 * A validator that is pending to be added is not considered a validator, only when
 * that change is finalized will this method return true. A validator that
 * is pending to be removed is immediately not considered a validator
 * (before the change is finalized).
 *
 * For the purposes of this contract one of the consequences is that you
 * can't report on a validator that is currently active but pending to be
 * removed. This is a compromise for simplicity since the reporting
 * functions only emit events which can be tracked off-chain.
 */
contract ValidatorSet is Ownable, IValidatorSet, IValidatorSetReporting {

    // Was the last validator change finalized. Implies validators == pending
    bool public finalized;

    // Don't touch it! It is 2 ^ 160 - 2, the systems signer account.
    // As stated hereL https://wiki.parity.io/Validator-Set#contracts
    address public systemAddress;

    // TYPES
    struct AddressStatus {
        bool isIn;
        uint index;
    }

    // STATE
    uint public recentBlocks;

    // Current list of addresses entitled to participate in the consensus.
    address[] private validators;
    address[] private pending;
    mapping(address => AddressStatus) status;

    // Asserts whether a given address is currently a validator.
    modifier isValidator(
        address _someone
    ) {
        bool isIn = status[_someone].isIn;
        uint index = status[_someone].index;

        require(
            isIn &&
            index < validators.length &&
            validators[index] == _someone,
            'given address is not an validator'
        );
        _;
    }

    // Asserts whether a given address is currently not a validator
    modifier isNotValidator(
        address _someone
    ) {
        require(
            !status[_someone].isIn,
            'given address is an validator'
        );
        _;
    }

    // Asserts whether a given block number is recent block
    modifier isRecent(
        uint _blockNumber
    ) {
        require(
            block.number <= _blockNumber + recentBlocks &&
            _blockNumber < block.number,
            ’it is not recent’
        );
        _;
    }

    // Assert whether finalized signal is true
    modifier whenFinalized() {
        require(
            finalized,
            ’it is not finalized’
        );
        _;
    }

    // Assert whether finalized signal is false
    modifier whenNotFinalized() {
        require(
            !finalized,
            ’it is finalized’
        );
        _;
    }

    // Assert the method is called by system account
    modifier onlySystem() {
        require(
            msg.sender == systemAddress,
            ’not system account’
        );
        _;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "constructor"
      @tag assume_completion
      @post __post.systemAddress == 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE
      @post __post.recentBlocks == 20
      @post __post._owner == _owner
      @post __post.pending == _initial
      @post forall i: uint. (i >= 0 && i < _initial.length) -> (__post.status[_initial
          [i]].isIn == true)
     */
    function initialize(
        address _owner,
        address[] memory _initial
    )
        public
        initializer
    {
        Ownable.initialize(_owner);
        pending = _initial;

        recentBlocks = 20;

        // Don’t touch it! It is 2 ˆ 160 - 2, the systems signer account.
        // As stated hereL https://wiki.parity.io/Validator-Set#contracts
        systemAddress = 0xffffFFFfFFffffffffffffffFfFFFfffFFFfFFfE;

        /*@CTK "constructor ForLoop"
          @pre forall j: uint. _initial[j] != 0x0
          @inv i <= _initial.length
          @inv _initial == _initial__pre
          @inv forall j: uint. (j >= 0 /\ j < i) -> this.status[_initial[j]].isIn ==
              true
          @post i == _initial.length
          @post !__should_return
         */
        for (uint i = 0; i < _initial.length; i++) {
            require(
                _initial[i] != address(0),
                'Invalid validator address'
            );
            status[_initial[i]].isIn = true;
            status[_initial[i]].index = i;
        }
    }

    /**
     * @dev addValidator adds validator to the validator set
     * checks either a validator already exists or not, also
     * only owner can call this function. Once the validator is
     * added to the pending validator set, it triggers change,
     * sets finalized to false to notify the system in order to
     * finalize the change in the validator set (system reaches
     * finality)
     * @param _validator validator address
     */
    //@CTK NO_ASF
    /*@CTK "addValidator"
      @tag assume_completion
      @post _owner == msg.sender
      @pre forall addr: address. pending[status[addr].index] == addr
      @pre forall i: uint. status[pending[i]].isIn == true && status[pending[i]].index
          == i
      @post forall addr: address. __post.pending[__post.status[addr].index] == addr
      @post forall i: uint. __post.status[__post.pending[i]].isIn == true && __post.
          status[__post.pending[i]].index == i
      @post status[_validator].isIn == false
      @post _validator != 0
      @post __post.status[_validator].isIn == true
      @post __post.status[_validator].index == pending.length
      @post __post.pending.length == pending.length + 1
      @post __post.pending[__post.status[_validator].index] == _validator
      @post finalized == true
      @post __post.finalized == false
     */
    function addValidator(
        address _validator
    )
        external
        onlyOwner
        isNotValidator(_validator)
    {
        require(
            _validator != address(0),
            'Invalid validator address'
        );

        status[_validator].isIn = true;
        status[_validator].index = pending.length;

        pending.push(_validator);

        triggerChange();
    }

    /**
     * @dev removeValidator adds validator from the validator set
     * checks either a validator already exists or not, also
     * only owner can call this function. Any change happens in the
     * pending validator set will trigger change signal to reach
     * finality
     * @param _validator validator address
     */
    //@CTK NO_ASF
    /*@CTK "removeValidator"
      @tag assume_completion
      @pre forall addr: address. pending[status[addr].index] == addr
      @pre forall i: uint. status[pending[i]].isIn == true && status[pending[i]].index
          == i
      @post _owner == msg.sender
      @post pending.length > 1
      @post validators[status[_validator].index] == _validator
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post finalized == true
      @post __post.status[_validator].isIn == false
      @post __post.status[_validator].index == 0
      @post __post.pending.length == pending.length - 1
      @post __post.pending[pending.length - 1] == 0x0
      @post status[_validator].index != pending.length - 1
            -> __post.pending[status[_validator].index] == pending[pending.length - 1]
            && __post.status[pending[pending.length - 1]].index == status[_validator].
                index
            && __post.status[pending[pending.length - 1]].isIn == true
      @post __post.finalized == false
     */
    function removeValidator(
        address _validator
    )
        external
        onlyOwner
        isValidator(_validator)
    {
        require(
            pending.length > 1,
            'Requires at least one live validator in the system'
        );

        // Remove validator from pending by moving the
        // last element to its slot
        uint index = status[_validator].index;

        pending[index] = pending[pending.length - 1];
        status[pending[index]].index = index;

        delete pending[pending.length - 1];
        pending.length--;

        // Reset address status including ‘isIn‘ and it’s ‘index‘
        delete status[_validator];

        triggerChange();
    }

    /**
     * @dev setRecentBlocks called only by the contract
     * owner in which sets the recentBlocks number. It acts
     * as a time window between two sequential malicious/benign
     * validator reports.
     * @param _recentBlocks the new value for the recent blocks
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "setRecentBlocks"
      @tag assume_completion
      @post _owner == msg.sender
      @post __post.recentBlocks == _recentBlocks
     */
    function setRecentBlocks(
        uint _recentBlocks
    )
        external
        onlyOwner
    {
        recentBlocks = _recentBlocks;
    }

    /**
     * @dev getValidators called to determine the current
     * set of validators.
     * @return the current validators set
     */
     //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "getValidators"
      @post _validators == validators
     */
    function getValidators()
        external view
        returns (address[] memory _validators)
    {
        return validators;
    }

    /**
     * @dev getPending called to determine the pending
     * set of validators.
     * @return the current pending validators set
     */
    //@CTK NO_ASF
    /*@CTK "getPending"
      @post __return == pending
     */
    function getPending()
        external view
        returns (address[] memory)
    {
        return pending;
    }

    /**
     * @dev reportBenign reports that a validator has
     * misbehaved in a benign way.
     * @param _validator validator address
     * @param _blockNumber is used to check whether the
     * report is recent
     */
    //@CTK NO_ASF
    /*@CTK "reportBenign"
      @tag assume_completion
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post validators[status[_validator].index] == _validator
      @post status[msg.sender].isIn == true
      @post status[msg.sender].index < validators.length
      @post validators[status[msg.sender].index] == msg.sender
      @post _blockNumber < block.number
      @post _blockNumber + recentBlocks >= block.number
     */
    function reportBenign(
        address _validator,
        uint _blockNumber
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            false
        );
    }

    /**
     * @dev reportMalicious reports that a validator has
     * misbehaved maliciously.
     * @param _validator validator address
     * @param _blockNumber is used to check whether the
     * report is recent
     * @param _proof (Not used) only emits events which
     * can be tracked off-chain. But we should implements the
     * same interface, for more information please refer to
     * https://wiki.parity.io/Validator-Set.html#reporting-contract
     */
    //@CTK NO_ASF
    /*@CTK "reportMalicious"
      @tag assume_completion
      @post status[_validator].isIn == true
      @post status[_validator].index < validators.length
      @post validators[status[_validator].index] == _validator
      @post status[msg.sender].isIn == true
      @post status[msg.sender].index < validators.length
      @post validators[status[msg.sender].index] == msg.sender
      @post _blockNumber < block.number
      @post _blockNumber + recentBlocks >= block.number
     */
    function reportMalicious(
        address _validator,
        uint _blockNumber,
        bytes calldata _proof
    )
        external
        isValidator(msg.sender)
        isValidator(_validator)
        isRecent(_blockNumber)
    {
        emit Report(
            msg.sender,
            _validator,
            true
        );
    }

    /**
     * @dev finalizeChange called when an initiated change
     * reaches finality and is activated. Only system account
     * call call this method
     */
    //@CTK NO_ASF
    /*@CTK "finalizeChange"
      @tag assume_completion
      @post systemAddress == msg.sender
      @post finalized == false
      @post __post.finalized == true
      @post __post.validators == pending
     */
    function finalizeChange()
        external
        whenNotFinalized
        onlySystem
    {
        validators = pending;
        finalized = true;
        emit ChangeFinalized(validators);
    }

    /**
     * @dev triggerChange trigger change by emitting an
     * event to report the change including the current
     * the hash of the block number and the pending
     * validator set
     */
    //@CTK NO_ASF
    /*@CTK triggerChange
      @tag assume_completion
      @post finalized == true
      @post __post.finalized == false
     */
    function triggerChange()
        private
        whenFinalized
    {
        finalized = false;
        emit InitiateChange(
            blockhash(block.number - 1),
            pending
        );
    }
}
```