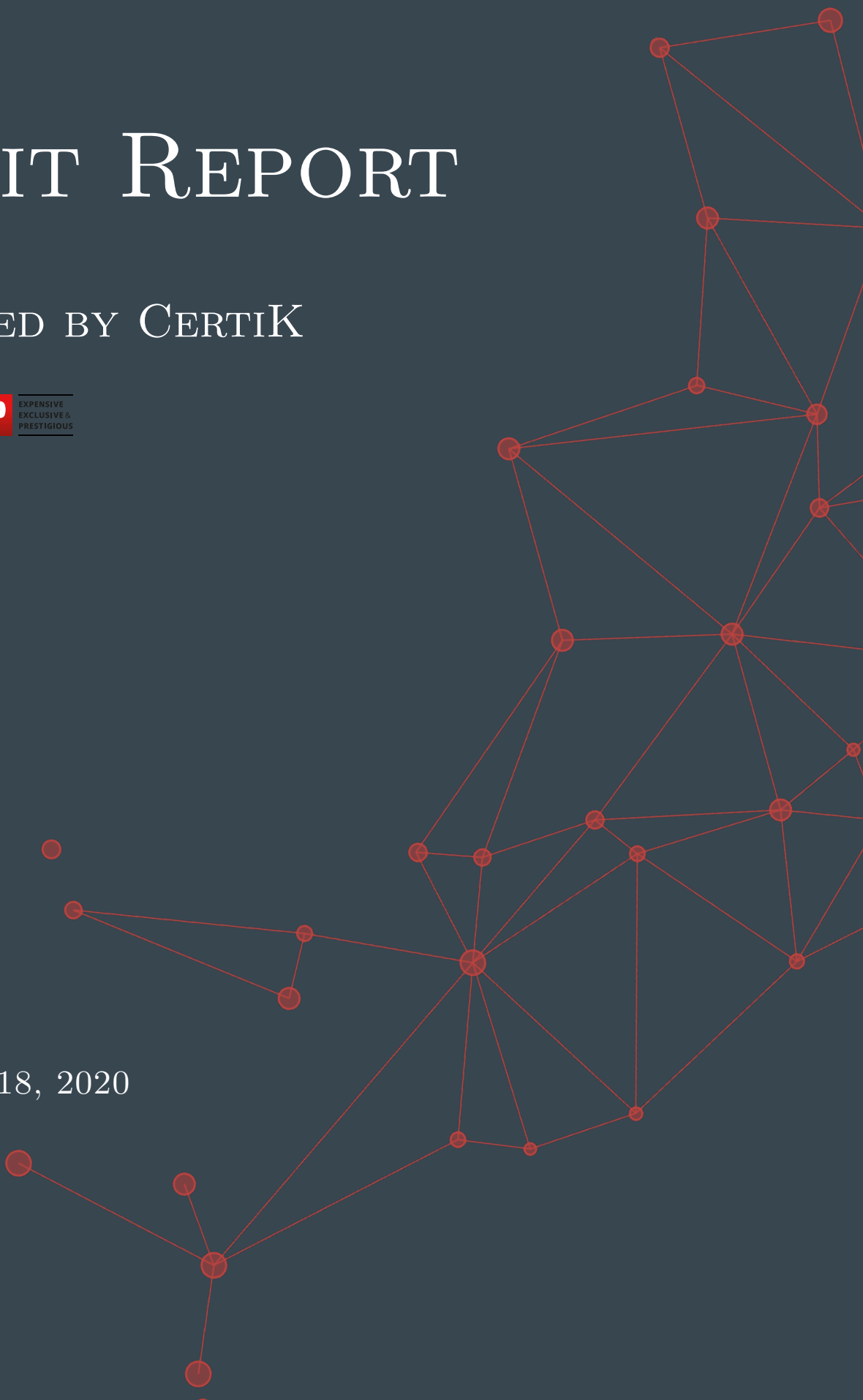
CERTIK

Audit Report

Produced by CertiK

for E2P EXPENSIVE EXCLUSIVE & PRESTIGIOUS



December 18, 2020

CertiK Audit Report
For E2P

Request Date: 2020-12-16
Revision Date: 2020-12-18
Platform Name: Ethereum

Contents

Disclaimer

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

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

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

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

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

What is a CertiK report?

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

About CertiK

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

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

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

Executive Summary

This report has been prepared for E2P to discover issues and vulnerabilities in the source code of their E2P smart contracts. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Vulnerability Classification

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

Critical

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

Medium

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

Low

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

Testing Summary

PASS

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

Dec 18, 2020

Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by miners to some degree.	7	SWC-116
Insecure Compiler Version	Using a fixed outdated compiler version or floating pragma can be problematic if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners to some degree.	0	SWC-120
"tx.origin" for Authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so the target and arguments provided must be sanitized.	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, meaning a malicious user can make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized Variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Review Notes

Source Code SHA-256 Checksum

- **E2P.sol**[1]
  `fecad53d8982656d0d4e2d12532ad4c69cda9323b5e9f0dbab440e51a7a23ee6`

Summary

CertiK team is invited by E2P team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checkings as well as manual reviews by smart contract experts. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and E2P team has been actively giving us updates for the source code and feedback about the business logics.

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

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

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes. Recommendations are labeled CRITICAL, MAJOR, MINOR, INFO, and DISCUSSION in decreasing significance level.

1. INFO – Consider using `add()` function instead of `+`.
   i.e. `now + _afterTime` -> `now.add(_afterTime)`. Please check Line 645, 647, 683, 686, 696, 712.

2. DISSCUSSION – Owner have permissions to freeze any account, lock any account's balance, mint tokens to any account, burn any account's tokens. Is this owner role under the E2P team's control and management?
   - (E2P - confirmed) owner is E2P's CEO, they know all of owner's permissions and the risk if transfer owner role to a bad guy.

[1] `<https://etherscan.io/address/0xA0b84460a1e78339692C7463009c35f0B9A6AE4C>`

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 5 in File E2P.sol

```
5 pragma solidity 0.5.8;
```

Version to compile has the following bug:
0.5.8: EmptyByteArrayCopy, DynamicArrayCleanup, ImplicitConstructorCallvalueCheck, TupleAssignmentMultiStackSlotComponents, MemoryArrayCreationOverflow, privateCanBeOverridden, YulOptimizerRedundantAssignmentBreakContinue0.5, ABIEncoderV2CalldataStructsWithStatica
SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClippedABIV2

TIMESTAMP_DEPENDENCY

Line 811 in File E2P.sol

```
811             if (lockInfo[_holder][i].releaseTime <= now) {
```

”now” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 870 in File E2P.sol

```
870             LockInfo(now + _afterTime, _amount)
```

”now” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 872 in File E2P.sol

```
872         emit Lock(_holder, _amount, now + _afterTime);
```

”now” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 936 in File E2P.sol

```
936             LockInfo(now + _afterTime, _value)
```

”now” can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 939 in File E2P.sol

```
939         emit Lock(_to, _value, now + _afterTime);
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 947 in File E2P.sol

```
947         return now;
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 953 in File E2P.sol

```
953         return now + _value;
```

! "now" can be influenced by miners to some degree

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

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

Post environment

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

Formal Verification Request 1

SafeMath add

18, Dec 2020
14.69 ms

Line 109-116 in File E2P.sol

```
109     /*@CTK "SafeMath add"
110       @tag assume_completion
111       @post (a + b < a || a + b < b) == __reverted
112       @post !__reverted -> __return == a + b
113       @post !__reverted -> !__has_overflow
114       @post !__reverted -> !__has_assertion_failure
115       @post !(__has_buf_overflow)
116      */
```

Line 117-122 in File E2P.sol

```
117     function add(uint256 a, uint256 b) internal pure returns (uint256) {
118         uint256 c = a + b;
119         require(c >= a, "SafeMath: addition overflow");
120 
121         return c;
122     }
```

The code meets the specification.

Formal Verification Request 2

SafeMath sub

18, Dec 2020
15.99 ms

Line 133-140 in File E2P.sol

```
133     /*@CTK "SafeMath sub"
134       @tag assume_completion
135       @post (a < b) == __reverted
136       @post !__reverted -> __return == a - b
137       @post !__reverted -> !__has_overflow
138       @post !__reverted -> !__has_assertion_failure
139       @post !(__has_buf_overflow)
140      */
```

Line 141-146 in File E2P.sol

```
141     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
142         require(b <= a, "SafeMath: subtraction overflow");
143         uint256 c = a - b;
144 
145         return c;
146     }
```

✅ The code meets the specification.

Formal Verification Request 3

SafeMath mul

📅 18, Dec 2020
⏱ 159.61 ms

Line 157-164 in File E2P.sol

```
157     /*@CTK "SafeMath mul"
158       @tag assume_completion
159       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
160       @post !__reverted -> __return == a * b
161       @post !__reverted == !__has_overflow
162       @post !__reverted -> !__has_assertion_failure
163       @post !(__has_buf_overflow)
164      */
```

Line 165-177 in File E2P.sol

```
165     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
166         // Gas optimization: this is cheaper than requiring 'a' not being
    ↪ zero, but the
167         // benefit is lost if 'b' is also tested.
168         // See:
    ↪ https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
169         if (a == 0) {
170             return 0;
171         }
172 
173         uint256 c = a * b;
174         require(c / a == b, "SafeMath: multiplication overflow");
175 
176         return c;
177     }
```

✅ The code meets the specification.

Formal Verification Request 4

SafeMath div

18, Dec 2020
14.2 ms

Line 190-197 in File E2P.sol

```
190     /*@CTK "SafeMath div"
191       @tag assume_completion
192       @post (b <= 0) == __reverted
193       @post !__reverted -> __return == a / b
194       @post !__reverted -> !__has_overflow
195       @post !__reverted -> !__has_assertion_failure
196       @post !(__has_buf_overflow)
197     */
```

Line 198-205 in File E2P.sol

```
198     function div(uint256 a, uint256 b) internal pure returns (uint256) {
199         // Solidity only automatically asserts when dividing by 0
200         require(b > 0, "SafeMath: division by zero");
201         uint256 c = a / b;
202         // assert(a == b * c + a % b); // There is no case in which this
    ↪  doesn't hold
203 
204         return c;
205     }
```

The code meets the specification.

Formal Verification Request 5

SafeMath mod

18, Dec 2020
13.84 ms

Line 218-225 in File E2P.sol

```
218     /*@CTK "SafeMath mod"
219       @tag assume_completion
220       @post b != 0 -> !__reverted
221       @post !__reverted -> __return == a % b
222       @post !__reverted -> !__has_overflow
223       @post !(__has_buf_overflow)
224       @post !(__has_assertion_failure)
225      */
```

Line 226-229 in File E2P.sol

```
226     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
227         require(b != 0, "SafeMath: modulo by zero");
228         return a % b;
229     }
```

✅ The code meets the specification.

Formal Verification Request 6

ERC20 totalSupply

📅 18, Dec 2020
⏱ 4.99 ms

Line 269-272 in File E2P.sol

```
269     /*@CTK "ERC20 totalSupply"
270       @tag assume_completion
271       @post __return == __post._totalSupply
272      */
```

Line 273-275 in File E2P.sol

```
273     function totalSupply() public view returns (uint256) {
274         return _totalSupply;
275     }
```

✅ The code meets the specification.

Formal Verification Request 7

ERC20 balanceOf

📅 18, Dec 2020
⏱ 5.61 ms

Line 280-283 in File E2P.sol

```
280     /*@CTK "ERC20 balanceOf"
281       @tag assume_completion
282       @post __return == __post._balances[account]
283      */
```

Line 284-286 in File E2P.sol

```
284     function balanceOf(address account) public view returns (uint256) {
285         return _balances[account];
286     }
```

The code meets the specification.

Formal Verification Request 8

ERC20 transfer

18, Dec 2020
226.14 ms

Line 296-303 in File E2P.sol

```
296     /*@CTK "ERC20 transfer"
297       @tag assume_completion
298       @post msg.sender != address(0)
299       @post recipient != address(0)
300       @post msg.sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
301       @post msg.sender != recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender] - amount
302       @post msg.sender == recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender]
303     */
```

Line 304-307 in File E2P.sol

```
304     function transfer(address recipient, uint256 amount) public returns
    ↪  (bool) {
305         _transfer(msg.sender, recipient, amount);
306         return true;
307     }
```

The code meets the specification.

Formal Verification Request 9

ERC20 allowance

18, Dec 2020
5.09 ms

Line 312-315 in File E2P.sol

```
312     /*@CTK "ERC20 allowance"
313       @tag assume_completion
314       @post __return == _allowances[owner][spender]
315     */
```

Line 316-318 in File E2P.sol

```
316     function allowance(address owner, address spender) public view returns
    ↪   (uint256) {
317         return _allowances[owner][spender];
318     }
```

The code meets the specification.

Formal Verification Request 10

ERC20 approve

18, Dec 2020
53.18 ms

Line 327-332 in File E2P.sol

```
327     /*@CTK "ERC20 approve"
328        @tag assume_completion
329        @post msg.sender != address(0)
330        @post spender != address(0)
331        @post __post._allowances[msg.sender][spender] == value
332      */
```

Line 333-336 in File E2P.sol

```
333     function approve(address spender, uint256 value) public returns (bool) {
334         _approve(msg.sender, spender, value);
335         return true;
336     }
```

The code meets the specification.

Formal Verification Request 11

ERC20 transferFrom

18, Dec 2020
420.41 ms

Line 350-359 in File E2P.sol

```
350     /*@CTK "ERC20 transferFrom"
351       @tag assume_completion
352       @post sender != address(0)
353       @post recipient != address(0)
354       @post msg.sender != address(0)
355       @post sender != recipient -> __post._balances[recipient] ==
    ↪   _balances[recipient] + amount
356       @post sender != recipient -> __post._balances[sender] ==
    ↪   _balances[sender] - amount
```

```
357       @post sender == recipient -> __post._balances[sender] ==
    ↪ _balances[sender]
358       @post __post._allowances[sender][msg.sender] ==
    ↪ _allowances[sender][msg.sender] - amount
359       */
```

Line 360-364 in File E2P.sol

```
360       function transferFrom(address sender, address recipient, uint256 amount)
    ↪ public returns (bool) {
361           _transfer(sender, recipient, amount);
362           _approve(sender, msg.sender,
    ↪ _allowances[sender][msg.sender].sub(amount));
363           return true;
364       }
```

✅ The code meets the specification.

Formal Verification Request 12

ERC20 increaseAllowance

📅 18, Dec 2020
⏱ 47.9 ms

Line 378-381 in File E2P.sol

```
378       /*@CTK "ERC20 increaseAllowance"
379         @tag assume_completion
380         @post __post._allowances[msg.sender][spender] ==
    ↪ _allowances[msg.sender][spender] + addedValue
381       */
```

Line 382-385 in File E2P.sol

```
382       function increaseAllowance(address spender, uint256 addedValue) public
    ↪ returns (bool) {
383           _approve(msg.sender, spender,
    ↪ _allowances[msg.sender][spender].add(addedValue));
384           return true;
385       }
```

✅ The code meets the specification.

Formal Verification Request 13

ERC20 decreaseAllowance

📅 18, Dec 2020
⏱ 51.32 ms

Line 401-404 in File E2P.sol

```
401     /*@CTK "ERC20 decreaseAllowance"
402       @tag assume_completion
403       @post __post._allowances[msg.sender][spender] ==
    ↪ _allowances[msg.sender][spender] - subtractedValue
404     */
```

Line 405-408 in File E2P.sol

```
405     function decreaseAllowance(address spender, uint256 subtractedValue)
    ↪ public returns (bool) {
406         _approve(msg.sender, spender,
    ↪ _allowances[msg.sender][spender].sub(subtractedValue));
407         return true;
408     }
```

The code meets the specification.

Formal Verification Request 14

ERC20 _transfer

18, Dec 2020
98.75 ms

Line 424-431 in File E2P.sol

```
424     /*@CTK "ERC20 _transfer"
425       @tag assume_completion
426       @post sender != address(0)
427       @post recipient != address(0)
428       @post sender != recipient -> __post._balances[recipient] ==
    ↪ _balances[recipient] + amount
429       @post sender != recipient -> __post._balances[sender] ==
    ↪ _balances[sender] - amount
430       @post sender == recipient -> __post._balances[sender] ==
    ↪ _balances[sender]
431     */
```

Line 432-439 in File E2P.sol

```
432     function _transfer(address sender, address recipient, uint256 amount)
    ↪ internal {
433         require(sender != address(0), "ERC20: transfer from the zero
    ↪ address");
434         require(recipient != address(0), "ERC20: transfer to the zero
    ↪ address");
435
```

```
436         _balances[sender] = _balances[sender].sub(amount);
437         _balances[recipient] = _balances[recipient].add(amount);
438         emit Transfer(sender, recipient, amount);
439     }
```

✅ The code meets the specification.

Formal Verification Request 15

ERC20 _mint

📅 18, Dec 2020
⏱ 87.74 ms

Line 450-455 in File E2P.sol

```
450     /*@CTK "ERC20 _mint"
451       @tag assume_completion
452       @post account != address(0)
453       @post __post._totalSupply == _totalSupply + amount
454       @post __post._balances[account] == _balances[account] + amount
455     */
```

Line 456-462 in File E2P.sol

```
456     function _mint(address account, uint256 amount) internal {
457         require(account != address(0), "ERC20: mint to the zero address");
458 
459         _totalSupply = _totalSupply.add(amount);
460         _balances[account] = _balances[account].add(amount);
461         emit Transfer(address(0), account, amount);
462     }
```

✅ The code meets the specification.

Formal Verification Request 16

ERC20 _burn

📅 18, Dec 2020
⏱ 72.94 ms

Line 475-480 in File E2P.sol

```
475     /*@CTK "ERC20 _burn"
476       @tag assume_completion
477       @post account != address(0)
478       @post __post._totalSupply == _totalSupply - value
479       @post __post._balances[account] == _balances[account] - value
480     */
```

Line 481-487 in File E2P.sol

```
481     function _burn(address account, uint256 value) internal {
482         require(account != address(0), "ERC20: burn from the zero address");
483 
484         _totalSupply = _totalSupply.sub(value);
485         _balances[account] = _balances[account].sub(value);
486         emit Transfer(account, address(0), value);
487     }
```

✅ The code meets the specification.

Formal Verification Request 17

ERC20 _approve

📅 18, Dec 2020
⏱ 2.89 ms

Line 502-507 in File E2P.sol

```
502     /*@CTK "ERC20 _approve"
503        @tag assume_completion
504        @post owner != address(0)
505        @post spender != address(0)
506        @post __post._allowances[owner][spender] == value
507      */
```

Line 508-514 in File E2P.sol

```
508     function _approve(address owner, address spender, uint256 value)
    ↪  internal {
509         require(owner != address(0), "ERC20: approve from the zero
    ↪  address");
510         require(spender != address(0), "ERC20: approve to the zero
    ↪  address");
511 
512         _allowances[owner][spender] = value;
513         emit Approval(owner, spender, value);
514     }
```

✅ The code meets the specification.

Formal Verification Request 18

ERC20 burnFrom

📅 18, Dec 2020
⏱ 210.62 ms

Line 522-529 in File E2P.sol

```
522     /*@CTK "ERC20 burnFrom"
523       @tag assume_completion
524       @post account != 0x0
525       @post amount <= _balances[account] && amount <=
    ↪  _allowances[account][msg.sender]
526       @post __post._balances[account] == _balances[account] - amount
527       @post __post._totalSupply == _totalSupply - amount
528       @post __post._allowances[account][msg.sender] ==
    ↪  _allowances[account][msg.sender] - amount
529      */
```

Line 530-533 in File E2P.sol

```
530     function _burnFrom(address account, uint256 amount) internal {
531         _burn(account, amount);
532         _approve(account, msg.sender,
    ↪  _allowances[account][msg.sender].sub(amount));
533     }
```

The code meets the specification.

Formal Verification Request 19

E2P constructor

18, Dec 2020
119.53 ms

Line 543-549 in File E2P.sol

```
543     /*@CTK "E2P constructor"
544       @tag assume_completion
545       @pre msg.sender != address(0)
546       @post __post._totalSupply == _totalSupply + initialSupply
547       @post __post._balances[msg.sender] == _balances[msg.sender] +
    ↪  initialSupply
548       @post __post.owner == msg.sender
549     */
```

Line 550-553 in File E2P.sol

```
550     constructor() public {
551         super._mint(msg.sender, initialSupply);
552         owner = msg.sender;
553     }
```

The code meets the specification.

Formal Verification Request 20

E2P renounceOwnership

18, Dec 2020
15.93 ms

Line 575-579 in File E2P.sol

```
575     /*@CTK "E2P renounceOwnership"
576      @tag assume_completion
577      @post owner == msg.sender
578      @post __post.owner == address(0)
579     */
```

Line 580-583 in File E2P.sol

```
580     function renounceOwnership() public onlyOwner {
581         emit OwnershipRenounced(owner);
582         owner = address(0);
583     }
```

The code meets the specification.

Formal Verification Request 21

E2P transferOwnership

18, Dec 2020
50.3 ms

Line 589-594 in File E2P.sol

```
589     /*@CTK "E2P transferOwnership"
590      @tag assume_completion
591      @post owner == msg.sender
592      @post _newOwner != address(0)
593      @post __post.owner == _newOwner
594     */
```

Line 595-597 in File E2P.sol

```
595     function transferOwnership(address _newOwner) public onlyOwner {
596         _transferOwnership(_newOwner);
597     }
```

The code meets the specification.

Formal Verification Request 22

E2P _transferOwnership

18, Dec 2020
2.13 ms

Line 603-607 in File E2P.sol

```
603     /*@CTK "E2P _transferOwnership"
604       @tag assume_completion
605       @post _newOwner != address(0)
606       @post __post.owner == _newOwner
607      */
```

Line 608-612 in File E2P.sol

```
608     function _transferOwnership(address _newOwner) internal {
609         require(_newOwner != address(0), "Already owner");
610         emit OwnershipTransferred(owner, _newOwner);
611         owner = _newOwner;
612     }
```

The code meets the specification.

Formal Verification Request 23

E2P pause

18, Dec 2020
27.77 ms

Line 639-644 in File E2P.sol

```
639     /*@CTK "E2P pause"
640       @tag assume_completion
641       @post owner == msg.sender
642       @post !paused
643       @post __post.paused == true
644     */
```

Line 645-648 in File E2P.sol

```
645     function pause() public onlyOwner whenNotPaused {
646         paused = true;
647         emit Pause();
648     }
```

The code meets the specification.

Formal Verification Request 24

E2P unpause

18, Dec 2020
26.64 ms

Line 653-658 in File E2P.sol

```
653     /*@CTK "E2P unpause"
654       @tag assume_completion
655       @post owner == msg.sender
656       @post paused
657       @post __post.paused == false
658     */
```

Line 659-662 in File E2P.sol

```
659     function unpause() public onlyOwner whenPaused {
660         paused = false;
661         emit Unpause();
662     }
```

The code meets the specification.

Formal Verification Request 25

E2P freeze

18, Dec 2020
17.95 ms

Line 674-678 in File E2P.sol

```
674     /*@CTK "E2P freeze"
675       @tag assume_completion
676       @pre msg.sender == owner
677       @post __post.freezes[_target] == true
678     */
```

Line 679-682 in File E2P.sol

```
679     function freeze(address _target) public onlyOwner {
680         freezes[_target] = true;
681         emit Frozen(_target);
682     }
```

The code meets the specification.

Formal Verification Request 26

E2P unfreeze

18, Dec 2020
54.82 ms

Line 683-687 in File E2P.sol

```
683     /*@CTK "E2P unfreeze"
684       @tag assume_completion
685       @pre msg.sender == owner
686       @post __post.freezes[_target] == false
687     */
```

Line 688-691 in File E2P.sol

```
688     function unfreeze(address _target) public onlyOwner {
689         freezes[_target] = false;
690         emit Unfrozen(_target);
691     }
```

The code meets the specification.

Formal Verification Request 27

E2P isFrozen

18, Dec 2020
6.5 ms

Line 692-695 in File E2P.sol

```
692     /*@CTK "E2P isFrozen"
693       @tag assume_completion
694       @post __return == freezes[_target]
695     */
```

Line 696-698 in File E2P.sol

```
696     function isFrozen(address _target) public view returns (bool) {
697         return freezes[_target];
698     }
```

The code meets the specification.

Formal Verification Request 28

E2P transfer

18, Dec 2020
235.3 ms

Line 699-703 in File E2P.sol

```
699     /*@CTK "E2P transfer"
700       @tag assume_completion
701       @post !paused
702       @post !freezes[msg.sender]
703     */
```

Line 704-715 in File E2P.sol

```
704     function transfer(
705         address _to,
706         uint256 _value
707     )
708       public
709       whenNotFrozen
710       whenNotPaused
711       returns (bool)
712     {
713         return super.transfer(_to, _value);
714     }
```

The code meets the specification.

Formal Verification Request 29

E2P transferFrom

18, Dec 2020
271.28 ms

Line 716-720 in File E2P.sol

```
716     /*@CTK "E2P transferFrom"
717       @tag assume_completion
718       @post !paused
719       @post !freezes[_from]
720     */
```

Line 721-733 in File E2P.sol

```
721     function transferFrom(
722         address _from,
723         address _to,
724         uint256 _value
725     )
726       public
727       whenNotPaused
```

```
728       returns (bool)
729     {
730         require(!freezes[_from], "From account is locked.");
731         return super.transferFrom(_from, _to, _value);
732     }
```

The code meets the specification.

Formal Verification Request 30

E2P mint

18, Dec 2020
144.74 ms

Line 737-744 in File E2P.sol

```
737     /*@CTK "E2P mint"
738       @tag assume_completion
739       @pre _to != address(0)
740       @pre msg.sender == owner
741       @post __post._totalSupply == _totalSupply + _amount
742       @post __post._balances[_to] == _balances[_to] + _amount
743       @post __return == true
744     */
```

Line 745-756 in File E2P.sol

```
745     function mint(
746         address _to,
747         uint256 _amount
748     )
749       public
750       onlyOwner
751       returns (bool)
752     {
753         super._mint(_to, _amount);
754         emit Mint(_to, _amount);
755         return true;
756     }
```

The code meets the specification.

Formal Verification Request 31

E2P burn

18, Dec 2020
242.12 ms

Line 760-767 in File E2P.sol

```
760     /*@CTK "E2P burn"
761       @tag assume_completion
762       @pre _who != address(0)
763       @pre msg.sender == owner
764       @pre _value <= _balances[_who]
765       @post __post._totalSupply == _totalSupply - _value
766       @post __post._balances[_who] == _balances[_who] - _value
767     */
```

Line 768-773 in File E2P.sol

```
768     function burn(address _who, uint256 _value) public onlyOwner {
769         require(_value <= super.balanceOf(_who), "Balance is too small.");
770 
771         _burn(_who, _value);
772         emit Burn(_who, _value);
773     }
```

The code meets the specification.

Formal Verification Request 32

Method will not encounter an assertion failure.

18, Dec 2020
8.54 ms

Line 784 in File E2P.sol

```
784     //@CTK NO_ASF
```

Line 785-800 in File E2P.sol

```
785     function balanceOf(address _holder) public view returns (uint256
    ↪ balance) {
786         uint256 lockedBalance = 0;
787         /*@CTK "E2P balanceOf_loop"
788           @inv _holder == _holder__pre
789           @inv i <= this.lockInfo[_holder].length
790           @inv lockedBalance >= lockedBalance
791           @post i == this.lockInfo[_holder].length
792           @post !__should_return
793         */
794         for(uint256 i = 0; i < lockInfo[_holder].length ; i++ ) {
795             lockedBalance = lockedBalance.add(lockInfo[_holder][i].balance);
796         }
```

```
797         return super.balanceOf(_holder).add(lockedBalance);
798     }
```

The code meets the specification.

Formal Verification Request 33

E2P lockCount

18, Dec 2020
5.14 ms

Line 825-828 in File E2P.sol

```
825     /*@CTK "E2P lockCount"
826       @tag assume_completion
827       @post __return == lockInfo[_holder].length
828     */
```

Line 829-831 in File E2P.sol

```
829     function lockCount(address _holder) public view returns (uint256) {
830         return lockInfo[_holder].length;
831     }
```

The code meets the specification.

Formal Verification Request 34

E2P lockState

18, Dec 2020
7.68 ms

Line 832-836 in File E2P.sol

```
832     /*@CTK "E2P lockState"
833       @tag assume_completion
834       @post __return == lockInfo[_holder][_idx].releaseTime
835       @post __return1 == lockInfo[_holder][_idx].balance
836     */
```

Line 837-839 in File E2P.sol

```
837     function lockState(address _holder, uint256 _idx) public view returns
    ↪  (uint256, uint256) {
838         return (lockInfo[_holder][_idx].releaseTime,
    ↪  lockInfo[_holder][_idx].balance);
839     }
```

The code meets the specification.

Formal Verification Request 35

E2P lock

18, Dec 2020
96.54 ms

Line 840-848 in File E2P.sol

```
840     /*@CTK "E2P lock"
841       @tag assume_completion
842       @pre msg.sender == owner
843       @post (_balances[_holder] < _amount) -> __reverted == true
844       @post __post._balances[_holder] == _balances[_holder] - _amount
845       @post __post.lockInfo[_holder].length == lockInfo[_holder].length + 1
846       @post __post.lockInfo[_holder][lockInfo[_holder].length].releaseTime
    ↪   == _releaseTime
847       @post __post.lockInfo[_holder][lockInfo[_holder].length].balance ==
    ↪   _amount
848     */
```

Line 849-856 in File E2P.sol

```
849     function lock(address _holder, uint256 _amount, uint256 _releaseTime)
    ↪   public onlyOwner {
850         require(super.balanceOf(_holder) >= _amount, "Balance is too
    ↪   small.");
851         _balances[_holder] = _balances[_holder].sub(_amount);
852         lockInfo[_holder].push(
853             LockInfo(_releaseTime, _amount)
854         );
855         emit Lock(_holder, _amount, _releaseTime);
856     }
```

The code meets the specification.

Formal Verification Request 36

E2P lockAfter

18, Dec 2020
104.27 ms

Line 857-865 in File E2P.sol

```
857     /*@CTK "E2P lockAfter"
858       @tag assume_completion
859       @pre msg.sender == owner
860       @post (_balances[_holder] < _amount) -> __reverted == true
861       @post __post._balances[_holder] == _balances[_holder] - _amount
```

```
862       @post __post.lockInfo[_holder].length == lockInfo[_holder].length + 1
863       @post __post.lockInfo[_holder][lockInfo[_holder].length].releaseTime
    ↪  == now + _afterTime
864       @post __post.lockInfo[_holder][lockInfo[_holder].length].balance ==
    ↪  _amount
865     */
```

Line 866-873 in File E2P.sol

```
866     function lockAfter(address _holder, uint256 _amount, uint256 _afterTime)
    ↪  public onlyOwner {
867         require(super.balanceOf(_holder) >= _amount, "Balance is too
    ↪  small.");
868         _balances[_holder] = _balances[_holder].sub(_amount);
869         lockInfo[_holder].push(
870             LockInfo(now + _afterTime, _amount)
871         );
872         emit Lock(_holder, _amount, now + _afterTime);
873     }
```

The code meets the specification.

Formal Verification Request 37

E2P unlock

18, Dec 2020
240.59 ms

Line 874-882 in File E2P.sol

```
874     /*@CTK "E2P unlock"
875       @tag assume_completion
876       @post msg.sender == owner
877       @post (i >= lockInfo[_holder].length) -> __reverted == true
878       @post __post._balances[_holder] == _balances[_holder] +
    ↪  lockInfo[_holder][i].balance
879       @post (i == lockInfo[_holder].length - 1) ->
    ↪  __post.lockInfo[_holder][i].balance == 0
880       @post (i != lockInfo[_holder].length - 1) ->
    ↪  __post.lockInfo[_holder][i] ==
    ↪  lockInfo[_holder][lockInfo[_holder].length - 1]
881       @post __post.lockInfo[_holder].length == lockInfo[_holder].length - 1
882     */
```

Line 883-894 in File E2P.sol

```
883     function unlock(address _holder, uint256 i) public onlyOwner {
884         require(i < lockInfo[_holder].length, "No lock information.");
885 
886         _balances[_holder] =
    ↪ _balances[_holder].add(lockInfo[_holder][i].balance);
887         emit Unlock(_holder, lockInfo[_holder][i].balance);
888         lockInfo[_holder][i].balance = 0;
889 
890         if (i != lockInfo[_holder].length - 1) {
891             lockInfo[_holder][i] = lockInfo[_holder][lockInfo[_holder].length
    ↪ - 1];
892         }
893         lockInfo[_holder].length--;
894     }
```

The code meets the specification.

Formal Verification Request 38

E2P transferWithLock

18, Dec 2020
120.86 ms

Line 895-905 in File E2P.sol

```
895     /*@CTK "E2P transferWithLock"
896       @tag assume_completion
897       @post msg.sender == owner
898       @post (_to == address(0)) -> __reverted == true
899       @post (_value > _balances[owner]) -> __reverted == true
900       @post __post._balances[owner] == _balances[owner] - _value
901       @post __post.lockInfo[_to].length == lockInfo[_to].length + 1
902       @post __post.lockInfo[_to][lockInfo[_to].length].releaseTime ==
    ↪ _releaseTime
903       @post __post.lockInfo[_to][lockInfo[_to].length].balance == _value
904       @post __return == true
905     */
```

Line 906-918 in File E2P.sol

```
906     function transferWithLock(address _to, uint256 _value, uint256
    ↪ _releaseTime) public onlyOwner returns (bool) {
907         require(_to != address(0), "wrong address");
908         require(_value <= super.balanceOf(owner), "Not enough balance");
909 
910         _balances[owner] = _balances[owner].sub(_value);
911         lockInfo[_to].push(
```

```
912             LockInfo(_releaseTime, _value)
913         );
914         emit Transfer(owner, _to, _value);
915         emit Lock(_to, _value, _releaseTime);
916 
917         return true;
918     }
```

The code meets the specification.

Formal Verification Request 39

E2P transferWithLockAfter

18, Dec 2020
127.71 ms

Line 919-929 in File E2P.sol

```
919     /*@CTK "E2P transferWithLockAfter"
920       @tag assume_completion
921       @post msg.sender == owner
922       @post (_to == address(0)) -> __reverted == true
923       @post (_value > _balances[owner]) -> __reverted == true
924       @post __post._balances[owner] == _balances[owner] - _value
925       @post __post.lockInfo[_to].length == lockInfo[_to].length + 1
926       @post __post.lockInfo[_to][lockInfo[_to].length].releaseTime == now +
    ↪  _afterTime
927       @post __post.lockInfo[_to][lockInfo[_to].length].balance == _value
928       @post __return == true
929     */
```

Line 930-942 in File E2P.sol

```
930     function transferWithLockAfter(address _to, uint256 _value, uint256
    ↪  _afterTime) public onlyOwner returns (bool) {
931         require(_to != address(0), "wrong address");
932         require(_value <= super.balanceOf(owner), "Not enough balance");
933 
934         _balances[owner] = _balances[owner].sub(_value);
935         lockInfo[_to].push(
936             LockInfo(now + _afterTime, _value)
937         );
938         emit Transfer(owner, _to, _value);
939         emit Lock(_to, _value, now + _afterTime);
940 
941         return true;
942     }
```

The code meets the specification.

Formal Verification Request 40

E2P currentTime

18, Dec 2020
4.33 ms

Line 943-945 in File E2P.sol

```
943     /*@CTK "E2P currentTime"
944       @post __return == now
945     */
```

Line 946-948 in File E2P.sol

```
946     function currentTime() public view returns (uint256) {
947         return now;
948     }
```

The code meets the specification.

Formal Verification Request 41

E2P afterTime

18, Dec 2020
7.04 ms

Line 949-951 in File E2P.sol

```
949     /*@CTK "E2P afterTime"
950       @post __return == now + _value
951     */
```

Line 952-954 in File E2P.sol

```
952     function afterTime(uint256 _value) public view returns (uint256) {
953         return now + _value;
954     }
```

The code meets the specification.

Formal Verification Request 42

E2P balanceOf_loop___Generated

18, Dec 2020
51.94 ms

(Loop) Line 787-793 in File E2P.sol

```
787         /*@CTK "E2P balanceOf_loop"
788           @inv _holder == _holder__pre
789           @inv i <= this.lockInfo[_holder].length
790           @inv lockedBalance >= lockedBalance
791           @post i == this.lockInfo[_holder].length
792           @post !__should_return
793         */
```

(Loop) Line 787-796 in File E2P.sol

```
787         /*@CTK "E2P balanceOf_loop"
788           @inv _holder == _holder__pre
789           @inv i <= this.lockInfo[_holder].length
790           @inv lockedBalance >= lockedBalance
791           @post i == this.lockInfo[_holder].length
792           @post !__should_return
793         */
794         for(uint256 i = 0; i < lockInfo[_holder].length ; i++ ) {
795             lockedBalance = lockedBalance.add(lockInfo[_holder][i].balance);
796         }
```

The code meets the specification.

Source Code with CertiK Labels

E2P.sol

```
/**
 *Submitted for verification at Etherscan.io on 2020-06-23
*/

pragma solidity 0.5.8;

// File:
 ↪  node_modules\openzeppelin-solidity\contracts\token\ERC20\IERC20.sol

/**
 * @dev Interface of the ERC20 standard as defined in the EIP. Does not
 ↪  include
 * the optional functions; to access them see `ERC20Detailed`.
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a `Transfer` event.
     */
    function transfer(address recipient, uint256 amount) external returns
 ↪  (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through `transferFrom`. This
 ↪  is
     * zero by default.
     *
     * This value changes when `approve` or `transferFrom` are called.
     */
```

```
40      function allowance(address owner, address spender) external view returns
    ↪   (uint256);
41
42      /**
43       * @dev Sets `amount` as the allowance of `spender` over the caller's
    ↪   tokens.
44       *
45       * Returns a boolean value indicating whether the operation succeeded.
46       *
47       * > Beware that changing an allowance with this method brings the risk
48       * that someone may use both the old and the new allowance by
    ↪   unfortunate
49       * transaction ordering. One possible solution to mitigate this race
50       * condition is to first reduce the spender's allowance to 0 and set
    ↪   the
51       * desired value afterwards:
52       * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
53       *
54       * Emits an `Approval` event.
55       */
56      function approve(address spender, uint256 amount) external returns
    ↪   (bool);
57
58      /**
59       * @dev Moves `amount` tokens from `sender` to `recipient` using the
60       * allowance mechanism. `amount` is then deducted from the caller's
61       * allowance.
62       *
63       * Returns a boolean value indicating whether the operation succeeded.
64       *
65       * Emits a `Transfer` event.
66       */
67      function transferFrom(address sender, address recipient, uint256 amount)
    ↪   external returns (bool);
68
69      /**
70       * @dev Emitted when `value` tokens are moved from one account (`from`)
    ↪   to
71       * another (`to`).
72       *
73       * Note that `value` may be zero.
74       */
75      event Transfer(address indexed from, address indexed to, uint256 value);
76
77      /**
78       * @dev Emitted when the allowance of a `spender` for an `owner` is set
    ↪   by
```

```
79          * a call to `approve`. `value` is the new allowance.
80          */
81         event Approval(address indexed owner, address indexed spender, uint256
    ↪  value);
82     }
83
84     // File: node_modules\openzeppelin-solidity\contracts\math\SafeMath.sol
85
86     /**
87      * @dev Wrappers over Solidity's arithmetic operations with added overflow
88      * checks.
89      *
90      * Arithmetic operations in Solidity wrap on overflow. This can easily
    ↪  result
91      * in bugs, because programmers usually assume that an overflow raises an
92      * error, which is the standard behavior in high level programming
    ↪  languages.
93      * `SafeMath` restores this intuition by reverting the transaction when an
94      * operation overflows.
95      *
96      * Using this library instead of the unchecked operations eliminates an
    ↪  entire
97      * class of bugs, so it's recommended to use it always.
98      */
99     library SafeMath {
100        /**
101         * @dev Returns the addition of two unsigned integers, reverting on
102         * overflow.
103         *
104         * Counterpart to Solidity's `+` operator.
105         *
106         * Requirements:
107         * - Addition cannot overflow.
108         */
109        /*@CTK "SafeMath add"
110          @tag assume_completion
111          @post (a + b < a || a + b < b) == __reverted
112          @post !__reverted -> __return == a + b
113          @post !__reverted -> !__has_overflow
114          @post !__reverted -> !__has_assertion_failure
115          @post !(__has_buf_overflow)
116         */
117        function add(uint256 a, uint256 b) internal pure returns (uint256) {
118            uint256 c = a + b;
119            require(c >= a, "SafeMath: addition overflow");
120
121            return c;
```

```
122     }
123
124     /**
125      * @dev Returns the subtraction of two unsigned integers, reverting on
126      * overflow (when the result is negative).
127      *
128      * Counterpart to Solidity's `-` operator.
129      *
130      * Requirements:
131      * - Subtraction cannot overflow.
132      */
133     /*@CTK "SafeMath sub"
134       @tag assume_completion
135       @post (a < b) == __reverted
136       @post !__reverted -> __return == a - b
137       @post !__reverted -> !__has_overflow
138       @post !__reverted -> !__has_assertion_failure
139       @post !(__has_buf_overflow)
140      */
141     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
142         require(b <= a, "SafeMath: subtraction overflow");
143         uint256 c = a - b;
144
145         return c;
146     }
147
148     /**
149      * @dev Returns the multiplication of two unsigned integers, reverting
    ↪ on
150      * overflow.
151      *
152      * Counterpart to Solidity's `*` operator.
153      *
154      * Requirements:
155      * - Multiplication cannot overflow.
156      */
157     /*@CTK "SafeMath mul"
158       @tag assume_completion
159       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
160       @post !__reverted -> __return == a * b
161       @post !__reverted == !__has_overflow
162       @post !__reverted -> !__has_assertion_failure
163       @post !(__has_buf_overflow)
164      */
165     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
166         // Gas optimization: this is cheaper than requiring 'a' not being
    ↪ zero, but the
```

```
167         // benefit is lost if 'b' is also tested.
168         // See:
    ↪ https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
169         if (a == 0) {
170             return 0;
171         }
172
173         uint256 c = a * b;
174         require(c / a == b, "SafeMath: multiplication overflow");
175
176         return c;
177     }
178
179     /**
180      * @dev Returns the integer division of two unsigned integers. Reverts
    ↪ on
181      * division by zero. The result is rounded towards zero.
182      *
183      * Counterpart to Solidity's `/` operator. Note: this function uses a
184      * `revert` opcode (which leaves remaining gas untouched) while
    ↪ Solidity
185      * uses an invalid opcode to revert (consuming all remaining gas).
186      *
187      * Requirements:
188      * - The divisor cannot be zero.
189      */
190     /*@CTK "SafeMath div"
191       @tag assume_completion
192       @post (b <= 0) == __reverted
193       @post !__reverted -> __return == a / b
194       @post !__reverted -> !__has_overflow
195       @post !__reverted -> !__has_assertion_failure
196       @post !(__has_buf_overflow)
197     */
198     function div(uint256 a, uint256 b) internal pure returns (uint256) {
199         // Solidity only automatically asserts when dividing by 0
200         require(b > 0, "SafeMath: division by zero");
201         uint256 c = a / b;
202         // assert(a == b * c + a % b); // There is no case in which this
    ↪ doesn't hold
203
204         return c;
205     }
206
207     /**
208      * @dev Returns the remainder of dividing two unsigned integers.
    ↪ (unsigned integer modulo),
```

```
209         * Reverts when dividing by zero.
210         *
211         * Counterpart to Solidity's `%` operator. This function uses a
    ↪  `revert`
212         * opcode (which leaves remaining gas untouched) while Solidity uses an
213         * invalid opcode to revert (consuming all remaining gas).
214         *
215         * Requirements:
216         * - The divisor cannot be zero.
217         */
218        /*@CTK "SafeMath mod"
219          @tag assume_completion
220          @post b != 0 -> !__reverted
221          @post !__reverted -> __return == a % b
222          @post !__reverted -> !__has_overflow
223          @post !(__has_buf_overflow)
224          @post !(__has_assertion_failure)
225         */
226        function mod(uint256 a, uint256 b) internal pure returns (uint256) {
227            require(b != 0, "SafeMath: modulo by zero");
228            return a % b;
229        }
230    }
231
232    // File: node_modules\openzeppelin-solidity\contracts\token\ERC20\ERC20.sol
233
234    /**
235     * @dev Implementation of the `IERC20` interface.
236     *
237     * This implementation is agnostic to the way tokens are created. This
    ↪  means
238     * that a supply mechanism has to be added in a derived contract using
    ↪  `_mint`.
239     * For a generic mechanism see `ERC20Mintable`.
240     *
241     * *For a detailed writeup see our guide [How to implement supply
242     *
    ↪  mechanisms](https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mech
243     *
244     * We have followed general OpenZeppelin guidelines: functions revert
    ↪  instead
245     * of returning `false` on failure. This behavior is nonetheless
    ↪  conventional
246     * and does not conflict with the expectations of ERC20 applications.
247     *
248     * Additionally, an `Approval` event is emitted on calls to `transferFrom`.
249     * This allows applications to reconstruct the allowance for all accounts
    ↪  just
```

```
 * by listening to said events. Other implementations of the EIP may not emit
 * these events, as it isn't required by the specification.
 *
 * Finally, the non-standard `decreaseAllowance` and `increaseAllowance`
 * functions have been added to mitigate the well-known issues around setting
 * allowances. See `IERC20.approve`.
 */
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) internal _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;

    /**
     * @dev See `IERC20.totalSupply`.
     */
    /*@CTK "ERC20 totalSupply"
      @tag assume_completion
      @post __return == __post._totalSupply
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev See `IERC20.balanceOf`.
     */
    /*@CTK "ERC20 balanceOf"
      @tag assume_completion
      @post __return == __post._balances[account]
     */
    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }

    /**
     * @dev See `IERC20.transfer`.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
```

```
295      */
296     /*@CTK "ERC20 transfer"
297       @tag assume_completion
298       @post msg.sender != address(0)
299       @post recipient != address(0)
300       @post msg.sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
301       @post msg.sender != recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender] - amount
302       @post msg.sender == recipient -> __post._balances[msg.sender] ==
    ↪  _balances[msg.sender]
303     */
304     function transfer(address recipient, uint256 amount) public returns
    ↪  (bool) {
305         _transfer(msg.sender, recipient, amount);
306         return true;
307     }
308
309     /**
310      * @dev See `IERC20.allowance`.
311      */
312     /*@CTK "ERC20 allowance"
313       @tag assume_completion
314       @post __return == _allowances[owner][spender]
315     */
316     function allowance(address owner, address spender) public view returns
    ↪  (uint256) {
317         return _allowances[owner][spender];
318     }
319
320     /**
321      * @dev See `IERC20.approve`.
322      *
323      * Requirements:
324      *
325      * - `spender` cannot be the zero address.
326      */
327     /*@CTK "ERC20 approve"
328        @tag assume_completion
329        @post msg.sender != address(0)
330        @post spender != address(0)
331        @post __post._allowances[msg.sender][spender] == value
332      */
333     function approve(address spender, uint256 value) public returns (bool) {
334         _approve(msg.sender, spender, value);
335         return true;
336     }
```

```
337
338     /**
339      * @dev See `IERC20.transferFrom`.
340      *
341      * Emits an `Approval` event indicating the updated allowance. This is
    ↪ not
342      * required by the EIP. See the note at the beginning of `ERC20`;
343      *
344      * Requirements:
345      * - `sender` and `recipient` cannot be the zero address.
346      * - `sender` must have a balance of at least `value`.
347      * - the caller must have allowance for `sender`'s tokens of at least
348      * `amount`.
349      */
350     /*@CTK "ERC20 transferFrom"
351       @tag assume_completion
352       @post sender != address(0)
353       @post recipient != address(0)
354       @post msg.sender != address(0)
355       @post sender != recipient -> __post._balances[recipient] ==
    ↪ _balances[recipient] + amount
356       @post sender != recipient -> __post._balances[sender] ==
    ↪ _balances[sender] - amount
357       @post sender == recipient -> __post._balances[sender] ==
    ↪ _balances[sender]
358       @post __post._allowances[sender][msg.sender] ==
    ↪ _allowances[sender][msg.sender] - amount
359     */
360     function transferFrom(address sender, address recipient, uint256 amount)
    ↪ public returns (bool) {
361         _transfer(sender, recipient, amount);
362         _approve(sender, msg.sender,
    ↪ _allowances[sender][msg.sender].sub(amount));
363         return true;
364     }
365
366     /**
367      * @dev Atomically increases the allowance granted to `spender` by the
    ↪ caller.
368      *
369      * This is an alternative to `approve` that can be used as a mitigation
    ↪ for
370      * problems described in `IERC20.approve`.
371      *
372      * Emits an `Approval` event indicating the updated allowance.
373      *
374      * Requirements:
```

```
375      *
376      * - `spender` cannot be the zero address.
377      */
378     /*@CTK "ERC20 increaseAllowance"
379       @tag assume_completion
380       @post __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] + addedValue
381     */
382     function increaseAllowance(address spender, uint256 addedValue) public
    ↪  returns (bool) {
383         _approve(msg.sender, spender,
    ↪  _allowances[msg.sender][spender].add(addedValue));
384         return true;
385     }
386
387     /**
388      * @dev Atomically decreases the allowance granted to `spender` by the
    ↪  caller.
389      *
390      * This is an alternative to `approve` that can be used as a mitigation
    ↪  for
391      * problems described in `IERC20.approve`.
392      *
393      * Emits an `Approval` event indicating the updated allowance.
394      *
395      * Requirements:
396      *
397      * - `spender` cannot be the zero address.
398      * - `spender` must have allowance for the caller of at least
399      * `subtractedValue`.
400      */
401     /*@CTK "ERC20 decreaseAllowance"
402       @tag assume_completion
403       @post __post._allowances[msg.sender][spender] ==
    ↪  _allowances[msg.sender][spender] - subtractedValue
404     */
405     function decreaseAllowance(address spender, uint256 subtractedValue)
    ↪  public returns (bool) {
406         _approve(msg.sender, spender,
    ↪  _allowances[msg.sender][spender].sub(subtractedValue));
407         return true;
408     }
409
410     /**
411      * @dev Moves tokens `amount` from `sender` to `recipient`.
412      *
413      * This is internal function is equivalent to `transfer`, and can be
    ↪  used to
```

```
414      * e.g. implement automatic token fees, slashing mechanisms, etc.
415      *
416      * Emits a `Transfer` event.
417      *
418      * Requirements:
419      *
420      * - `sender` cannot be the zero address.
421      * - `recipient` cannot be the zero address.
422      * - `sender` must have a balance of at least `amount`.
423      */
424     /*@CTK "ERC20 _transfer"
425       @tag assume_completion
426       @post sender != address(0)
427       @post recipient != address(0)
428       @post sender != recipient -> __post._balances[recipient] ==
    ↪  _balances[recipient] + amount
429       @post sender != recipient -> __post._balances[sender] ==
    ↪  _balances[sender] - amount
430       @post sender == recipient -> __post._balances[sender] ==
    ↪  _balances[sender]
431     */
432     function _transfer(address sender, address recipient, uint256 amount)
    ↪  internal {
433         require(sender != address(0), "ERC20: transfer from the zero
    ↪  address");
434         require(recipient != address(0), "ERC20: transfer to the zero
    ↪  address");
435
436         _balances[sender] = _balances[sender].sub(amount);
437         _balances[recipient] = _balances[recipient].add(amount);
438         emit Transfer(sender, recipient, amount);
439     }
440
441     /** @dev Creates `amount` tokens and assigns them to `account`,
    ↪  increasing
442      * the total supply.
443      *
444      * Emits a `Transfer` event with `from` set to the zero address.
445      *
446      * Requirements
447      *
448      * - `to` cannot be the zero address.
449      */
450     /*@CTK "ERC20 _mint"
451       @tag assume_completion
452       @post account != address(0)
453       @post __post._totalSupply == _totalSupply + amount
```

```
454       @post __post._balances[account] == _balances[account] + amount
455     */
456     function _mint(address account, uint256 amount) internal {
457         require(account != address(0), "ERC20: mint to the zero address");
458 
459         _totalSupply = _totalSupply.add(amount);
460         _balances[account] = _balances[account].add(amount);
461         emit Transfer(address(0), account, amount);
462     }
463 
464      /**
465       * @dev Destoys `amount` tokens from `account`, reducing the
466       * total supply.
467       *
468       * Emits a `Transfer` event with `to` set to the zero address.
469       *
470       * Requirements
471       *
472       * - `account` cannot be the zero address.
473       * - `account` must have at least `amount` tokens.
474       */
475     /*@CTK "ERC20 _burn"
476       @tag assume_completion
477       @post account != address(0)
478       @post __post._totalSupply == _totalSupply - value
479       @post __post._balances[account] == _balances[account] - value
480     */
481     function _burn(address account, uint256 value) internal {
482         require(account != address(0), "ERC20: burn from the zero address");
483 
484         _totalSupply = _totalSupply.sub(value);
485         _balances[account] = _balances[account].sub(value);
486         emit Transfer(account, address(0), value);
487     }
488 
489     /**
490      * @dev Sets `amount` as the allowance of `spender` over the `owner`s
    ↪ tokens.
491      *
492      * This is internal function is equivalent to `approve`, and can be
    ↪ used to
493      * e.g. set automatic allowances for certain subsystems, etc.
494      *
495      * Emits an `Approval` event.
496      *
497      * Requirements:
498      *
```

```
499      * - `owner` cannot be the zero address.
500      * - `spender` cannot be the zero address.
501      */
502     /*@CTK "ERC20 _approve"
503       @tag assume_completion
504       @post owner != address(0)
505       @post spender != address(0)
506       @post __post._allowances[owner][spender] == value
507      */
508     function _approve(address owner, address spender, uint256 value)
    ↪  internal {
509         require(owner != address(0), "ERC20: approve from the zero
    ↪  address");
510         require(spender != address(0), "ERC20: approve to the zero
    ↪  address");
511 
512         _allowances[owner][spender] = value;
513         emit Approval(owner, spender, value);
514     }
515 
516     /**
517      * @dev Destoys `amount` tokens from `account`.`amount` is then
    ↪  deducted
518      * from the caller's allowance.
519      *
520      * See `_burn` and `_approve`.
521      */
522     /*@CTK "ERC20 burnFrom"
523       @tag assume_completion
524       @post account != 0x0
525       @post amount <= _balances[account] && amount <=
    ↪  _allowances[account][msg.sender]
526       @post __post._balances[account] == _balances[account] - amount
527       @post __post._totalSupply == _totalSupply - amount
528       @post __post._allowances[account][msg.sender] ==
    ↪  _allowances[account][msg.sender] - amount
529      */
530     function _burnFrom(address account, uint256 amount) internal {
531         _burn(account, amount);
532         _approve(account, msg.sender,
    ↪  _allowances[account][msg.sender].sub(amount));
533     }
534 }
535 
536 // File: J\contracts\E2P.sol
537 
538 contract E2P is ERC20 {
```

```
539     string public constant name = "E2P";
540     string public constant symbol = "E2P";
541     uint8 public constant decimals = 18;
542     uint256 public constant initialSupply = 5000000000 * (10 **
    ↪  uint256(decimals));
543     /*@CTK "E2P constructor"
544       @tag assume_completion
545       @pre msg.sender != address(0)
546       @post __post._totalSupply == _totalSupply + initialSupply
547       @post __post._balances[msg.sender] == _balances[msg.sender] +
    ↪  initialSupply
548       @post __post.owner == msg.sender
549     */
550     constructor() public {
551         super._mint(msg.sender, initialSupply);
552         owner = msg.sender;
553     }
554 
555     //ownership
556     address public owner;
557 
558     event OwnershipRenounced(address indexed previousOwner);
559     event OwnershipTransferred(
560     address indexed previousOwner,
561     address indexed newOwner
562     );
563 
564     modifier onlyOwner() {
565         require(msg.sender == owner, "Not owner");
566         _;
567     }
568 
569   /**
570    * @dev Allows the current owner to relinquish control of the contract.
571    * @notice Renouncing to ownership will leave the contract without an
    ↪   owner.
572    * It will not be possible to call the functions with the `onlyOwner`
573    * modifier anymore.
574    */
575    /*@CTK "E2P renounceOwnership"
576     @tag assume_completion
577     @post owner == msg.sender
578     @post __post.owner == address(0)
579    */
580     function renounceOwnership() public onlyOwner {
581         emit OwnershipRenounced(owner);
582         owner = address(0);
```

```
583     }
584
585   /**
586     * @dev Allows the current owner to transfer control of the contract to a
    ↪  newOwner.
587     * @param _newOwner The address to transfer ownership to.
588     */
589     /*@CTK "E2P transferOwnership"
590       @tag assume_completion
591       @post owner == msg.sender
592       @post _newOwner != address(0)
593       @post __post.owner == _newOwner
594     */
595     function transferOwnership(address _newOwner) public onlyOwner {
596         _transferOwnership(_newOwner);
597     }
598
599   /**
600     * @dev Transfers control of the contract to a newOwner.
601     * @param _newOwner The address to transfer ownership to.
602     */
603     /*@CTK "E2P _transferOwnership"
604        @tag assume_completion
605        @post _newOwner != address(0)
606        @post __post.owner == _newOwner
607      */
608     function _transferOwnership(address _newOwner) internal {
609         require(_newOwner != address(0), "Already owner");
610         emit OwnershipTransferred(owner, _newOwner);
611         owner = _newOwner;
612     }
613
614     //pausable
615     event Pause();
616     event Unpause();
617
618     bool public paused = false;
619
620     /**
621     * @dev Modifier to make a function callable only when the contract is
    ↪  not paused.
622     */
623     modifier whenNotPaused() {
624         require(!paused, "Paused by owner");
625         _;
626     }
627
```

```
628     /**
629     * @dev Modifier to make a function callable only when the contract is
    ↪  paused.
630     */
631     modifier whenPaused() {
632         require(paused, "Not paused now");
633         _;
634     }
635 
636     /**
637     * @dev called by the owner to pause, triggers stopped state
638     */
639     /*@CTK "E2P pause"
640       @tag assume_completion
641       @post owner == msg.sender
642       @post !paused
643       @post __post.paused == true
644     */
645     function pause() public onlyOwner whenNotPaused {
646         paused = true;
647         emit Pause();
648     }
649 
650     /**
651     * @dev called by the owner to unpause, returns to normal state
652     */
653     /*@CTK "E2P unpause"
654       @tag assume_completion
655       @post owner == msg.sender
656       @post paused
657       @post __post.paused == false
658     */
659     function unpause() public onlyOwner whenPaused {
660         paused = false;
661         emit Unpause();
662     }
663 
664     //freezable
665     event Frozen(address target);
666     event Unfrozen(address target);
667 
668     mapping(address => bool) internal freezes;
669 
670     modifier whenNotFrozen() {
671         require(!freezes[msg.sender], "Sender account is locked.");
672         _;
673     }
```

```
674     /*@CTK "E2P freeze"
675       @tag assume_completion
676       @pre msg.sender == owner
677       @post __post.freezes[_target] == true
678     */
679     function freeze(address _target) public onlyOwner {
680         freezes[_target] = true;
681         emit Frozen(_target);
682     }
683     /*@CTK "E2P unfreeze"
684       @tag assume_completion
685       @pre msg.sender == owner
686       @post __post.freezes[_target] == false
687     */
688     function unfreeze(address _target) public onlyOwner {
689         freezes[_target] = false;
690         emit Unfrozen(_target);
691     }
692     /*@CTK "E2P isFrozen"
693       @tag assume_completion
694       @post __return == freezes[_target]
695     */
696     function isFrozen(address _target) public view returns (bool) {
697         return freezes[_target];
698     }
699     /*@CTK "E2P transfer"
700       @tag assume_completion
701       @post !paused
702       @post !freezes[msg.sender]
703     */
704     function transfer(
705         address _to,
706         uint256 _value
707     )
708       public
709       whenNotFrozen
710       whenNotPaused
711       returns (bool)
712     {
713         return super.transfer(_to, _value);
714     }
715     /*@CTK "E2P transferFrom"
716       @tag assume_completion
717       @post !paused
718       @post !freezes[_from]
719     */
720     function transferFrom(
```

```
721         address _from,
722         address _to,
723         uint256 _value
724     )
725       public
726       whenNotPaused
727       returns (bool)
728     {
729         require(!freezes[_from], "From account is locked.");
730         return super.transferFrom(_from, _to, _value);
731     }
732 
733     //mintable
734     event Mint(address indexed to, uint256 amount);
735     /*@CTK "E2P mint"
736       @tag assume_completion
737       @pre _to != address(0)
738       @pre msg.sender == owner
739       @post __post._totalSupply == _totalSupply + _amount
740       @post __post._balances[_to] == _balances[_to] + _amount
741       @post __return == true
742     */
743     function mint(
744         address _to,
745         uint256 _amount
746     )
747       public
748       onlyOwner
749       returns (bool)
750     {
751         super._mint(_to, _amount);
752         emit Mint(_to, _amount);
753         return true;
754     }
755 
756     //burnable
757     event Burn(address indexed burner, uint256 value);
758     /*@CTK "E2P burn"
759       @tag assume_completion
760       @pre _who != address(0)
761       @pre msg.sender == owner
762       @pre _value <= _balances[_who]
763       @post __post._totalSupply == _totalSupply - _value
764       @post __post._balances[_who] == _balances[_who] - _value
765     */
766     function burn(address _who, uint256 _value) public onlyOwner {
767         require(_value <= super.balanceOf(_who), "Balance is too small.");
```

```
768
769         _burn(_who, _value);
770         emit Burn(_who, _value);
771     }
772
773     //lockable
774     struct LockInfo {
775         uint256 releaseTime;
776         uint256 balance;
777     }
778     mapping(address => LockInfo[]) internal lockInfo;
779
780     event Lock(address indexed holder, uint256 value, uint256 releaseTime);
781     event Unlock(address indexed holder, uint256 value);
782     //@CTK NO_ASF
783     function balanceOf(address _holder) public view returns (uint256
    ↪  balance) {
784         uint256 lockedBalance = 0;
785         /*@CTK "E2P balanceOf_loop"
786           @inv _holder == _holder__pre
787           @inv i <= this.lockInfo[_holder].length
788           @inv lockedBalance >= lockedBalance
789           @post i == this.lockInfo[_holder].length
790           @post !__should_return
791         */
792         for(uint256 i = 0; i < lockInfo[_holder].length ; i++ ) {
793             lockedBalance = lockedBalance.add(lockInfo[_holder][i].balance);
794         }
795         return super.balanceOf(_holder).add(lockedBalance);
796     }
797     //$CTK NO_ASF
798     function releaseLock(address _holder) internal {
799      /*$CTK "E2P releaseLock_loop"
800       @inv _holder == _holder__pre
801       @inv i <= lockInfo[_holder].length.length
802       @post lockInfo[_holder][i].releaseTime <= now ->
    ↪  __post.lockInfo[_holder].length == lockInfo[_holder].length
803       @post i == lockInfo[_holder].length.length
804       @post !__should_return
805      */
806         for(uint256 i = 0; i < lockInfo[_holder].length ; i++ ) {
807             if (lockInfo[_holder][i].releaseTime <= now) {
808                 _balances[_holder] =
    ↪  _balances[_holder].add(lockInfo[_holder][i].balance);
809                 emit Unlock(_holder, lockInfo[_holder][i].balance);
810                 lockInfo[_holder][i].balance = 0;
811
```

```
812                 if (i != lockInfo[_holder].length - 1) {
813                     lockInfo[_holder][i] =
    ↪ lockInfo[_holder][lockInfo[_holder].length - 1];
814                     i--;
815                 }
816                 lockInfo[_holder].length--;
817
818             }
819         }
820     }
821     /*@CTK "E2P lockCount"
822       @tag assume_completion
823       @post __return == lockInfo[_holder].length
824     */
825     function lockCount(address _holder) public view returns (uint256) {
826         return lockInfo[_holder].length;
827     }
828     /*@CTK "E2P lockState"
829       @tag assume_completion
830       @post __return == lockInfo[_holder][_idx].releaseTime
831       @post __return1 == lockInfo[_holder][_idx].balance
832     */
833     function lockState(address _holder, uint256 _idx) public view returns
    ↪ (uint256, uint256) {
834         return (lockInfo[_holder][_idx].releaseTime,
    ↪ lockInfo[_holder][_idx].balance);
835     }
836     /*@CTK "E2P lock"
837       @tag assume_completion
838       @pre msg.sender == owner
839       @post (_balances[_holder] < _amount) -> __reverted == true
840       @post __post._balances[_holder] == _balances[_holder] - _amount
841       @post __post.lockInfo[_holder].length == lockInfo[_holder].length + 1
842       @post __post.lockInfo[_holder][lockInfo[_holder].length].releaseTime
    ↪ == _releaseTime
843       @post __post.lockInfo[_holder][lockInfo[_holder].length].balance ==
    ↪ _amount
844     */
845     function lock(address _holder, uint256 _amount, uint256 _releaseTime)
    ↪ public onlyOwner {
846         require(super.balanceOf(_holder) >= _amount, "Balance is too
    ↪ small.");
847         _balances[_holder] = _balances[_holder].sub(_amount);
848         lockInfo[_holder].push(
849             LockInfo(_releaseTime, _amount)
850         );
851         emit Lock(_holder, _amount, _releaseTime);
```

```
852     }
853     /*@CTK "E2P lockAfter"
854       @tag assume_completion
855       @pre msg.sender == owner
856       @post (_balances[_holder] < _amount) -> __reverted == true
857       @post __post._balances[_holder] == _balances[_holder] - _amount
858       @post __post.lockInfo[_holder].length == lockInfo[_holder].length + 1
859       @post __post.lockInfo[_holder][lockInfo[_holder].length].releaseTime
    ↪  == now + _afterTime
860       @post __post.lockInfo[_holder][lockInfo[_holder].length].balance ==
    ↪  _amount
861     */
862     function lockAfter(address _holder, uint256 _amount, uint256 _afterTime)
    ↪  public onlyOwner {
863         require(super.balanceOf(_holder) >= _amount, "Balance is too
    ↪  small.");
864         _balances[_holder] = _balances[_holder].sub(_amount);
865         lockInfo[_holder].push(
866             LockInfo(now + _afterTime, _amount)
867         );
868         emit Lock(_holder, _amount, now + _afterTime);
869     }
870     /*@CTK "E2P unlock"
871       @tag assume_completion
872       @post msg.sender == owner
873       @post (i >= lockInfo[_holder].length) -> __reverted == true
874       @post __post._balances[_holder] == _balances[_holder] +
    ↪  lockInfo[_holder][i].balance
875       @post (i == lockInfo[_holder].length - 1) ->
    ↪  __post.lockInfo[_holder][i].balance == 0
876       @post (i != lockInfo[_holder].length - 1) ->
    ↪  __post.lockInfo[_holder][i] ==
    ↪  lockInfo[_holder][lockInfo[_holder].length - 1]
877       @post __post.lockInfo[_holder].length == lockInfo[_holder].length - 1
878     */
879     function unlock(address _holder, uint256 i) public onlyOwner {
880         require(i < lockInfo[_holder].length, "No lock information.");
881 
882         _balances[_holder] =
    ↪  _balances[_holder].add(lockInfo[_holder][i].balance);
883         emit Unlock(_holder, lockInfo[_holder][i].balance);
884         lockInfo[_holder][i].balance = 0;
885 
886         if (i != lockInfo[_holder].length - 1) {
887             lockInfo[_holder][i] = lockInfo[_holder][lockInfo[_holder].length
    ↪  - 1];
888         }
```

```
889         lockInfo[_holder].length--;
890     }
891     /*@CTK "E2P transferWithLock"
892       @tag assume_completion
893       @post msg.sender == owner
894       @post (_to == address(0)) -> __reverted == true
895       @post (_value > _balances[owner]) -> __reverted == true
896       @post __post._balances[owner] == _balances[owner] - _value
897       @post __post.lockInfo[_to].length == lockInfo[_to].length + 1
898       @post __post.lockInfo[_to][lockInfo[_to].length].releaseTime ==
    ↪  _releaseTime
899       @post __post.lockInfo[_to][lockInfo[_to].length].balance == _value
900       @post __return == true
901     */
902     function transferWithLock(address _to, uint256 _value, uint256
    ↪  _releaseTime) public onlyOwner returns (bool) {
903         require(_to != address(0), "wrong address");
904         require(_value <= super.balanceOf(owner), "Not enough balance");
905 
906         _balances[owner] = _balances[owner].sub(_value);
907         lockInfo[_to].push(
908             LockInfo(_releaseTime, _value)
909         );
910         emit Transfer(owner, _to, _value);
911         emit Lock(_to, _value, _releaseTime);
912 
913         return true;
914     }
915     /*@CTK "E2P transferWithLockAfter"
916       @tag assume_completion
917       @post msg.sender == owner
918       @post (_to == address(0)) -> __reverted == true
919       @post (_value > _balances[owner]) -> __reverted == true
920       @post __post._balances[owner] == _balances[owner] - _value
921       @post __post.lockInfo[_to].length == lockInfo[_to].length + 1
922       @post __post.lockInfo[_to][lockInfo[_to].length].releaseTime == now +
    ↪  _afterTime
923       @post __post.lockInfo[_to][lockInfo[_to].length].balance == _value
924       @post __return == true
925     */
926     function transferWithLockAfter(address _to, uint256 _value, uint256
    ↪  _afterTime) public onlyOwner returns (bool) {
927         require(_to != address(0), "wrong address");
928         require(_value <= super.balanceOf(owner), "Not enough balance");
929 
930         _balances[owner] = _balances[owner].sub(_value);
931         lockInfo[_to].push(
```

```
            LockInfo(now + _afterTime, _value)
        );
        emit Transfer(owner, _to, _value);
        emit Lock(_to, _value, now + _afterTime);

        return true;
    }
    /*@CTK "E2P currentTime"
      @post __return == now
    */
    function currentTime() public view returns (uint256) {
        return now;
    }
    /*@CTK "E2P afterTime"
      @post __return == now + _value
    */
    function afterTime(uint256 _value) public view returns (uint256) {
        return now + _value;
    }

    //airdrop
    mapping (address => uint256) public airDropHistory;
    event AirDrop(address _receiver, uint256 _amount);
   //$CTK NO_ASF
   /*$CTK "E2P dropToken"
    @tag assume_completion
    @post owner == msg.sender
    @post receivers.length != 0
    @post receivers.length == values.length
    */
    function dropToken(address[] memory receivers, uint256[] memory values)
↪   onlyOwner public {
    require(receivers.length != 0);
    require(receivers.length == values.length);
    /*$CTK "E2P dropToken_loop"
      @inv i <= receivers.length
      @inv __post.airDropHistory[receiver] == airDropHistory[receiver] +
↪   amount
      @post i == receivers.length
      @post !__should_return
     */
    for (uint256 i = 0; i < receivers.length; i++) {
      address receiver = receivers[i];
      uint256 amount = values[i];

      transfer(receiver, amount);
      airDropHistory[receiver] += amount;
```

```
977
978         emit AirDrop(receiver, amount);
979       }
980     }
981   }
```