



CERTIK

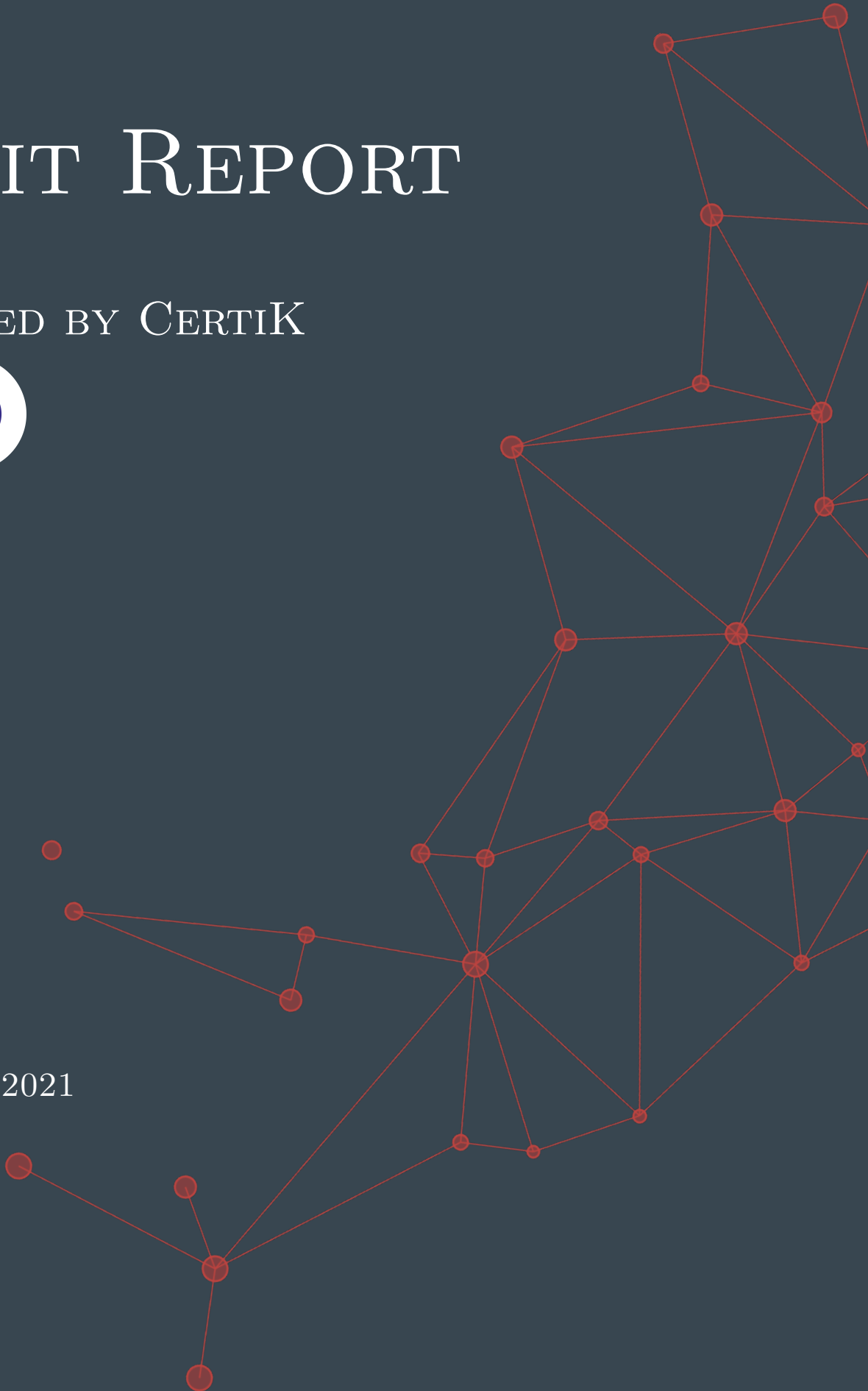
AUDIT REPORT

PRODUCED BY CERTIK

FOR



MARCH 23, 2021



CERTIK AUDIT REPORT FOR CONVERGENCE FINANCE



Request Date: 2021-03-23
Revision Date: 2021-03-23
Platform Name: Ethereum



Contents

Disclaimer	1
About CertiK	2
Executive Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Review Notes	6
Static Analysis Results	7
Formal Verification Results	8
How to read	8
Source Code with CertiK Labels	30

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 Teller. 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 Convergence Finance to discover issues and vulnerabilities in the source code of their ConvergenceToken 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.

Mar 23, 2021



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.	0	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. msg.sender instead.	Use 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

- **ConvergenceToken.sol**¹
d9d086932b36df750cf8cbcbc728ccfe7d15fca441998d58472103956ef69bf7

Summary

CertiK team is invited by The ConvergenceToken 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 ConvergenceTokenToken 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 ConvergenceToken contract follows good practices, with reasonable amount of features on top of the ERC20 related to administrative 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.

ConvergenceToken.sol

- **INFO** Literals with many digits are difficult to read and review. Consider using scientific notation or ether suffix for variable `TOTAL_SUPPLY`.

¹<https://etherscan.io/address/0xc834fa996fa3bec7aad3693af486ae53d8aa8b50>

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 9 in File ConvergenceToken.sol

```
9 pragma solidity >=0.6.0 <0.8.0;
```



 Only these compiler versions are safe to compile your code: 0.7.4

Formal Verification Results

How to read


Detail for Request 1

transferFrom to same address

Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

CERTIK label location	Line 30-34 in File howtoread.sol
CERTIK label	<pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from] [msg.sender] == 34 */ </pre>


Raw code location	Line 35-41 in File howtoread.sol
Raw code	<pre> 35 function transferFrom(address from, address to) { 36 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 } </pre>

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

Formal Verification Request 1

Context `_msgSender`

 23, Mar 2021

 20.18 ms

Line 22-25 in File `ConvergenceToken.sol`

```
22  /*@CTK "Context _msgSender"
23     @tag assume_completion
24     @post __return == msg.sender
25  */
```

Line 27-29 in File `ConvergenceToken.sol`


```
27  function _msgSender() internal view virtual returns (address payable) {
28      return msg.sender;
29  }
```

 The code meets the specification.

Formal Verification Request 2

Context `_msgData`

 23, Mar 2021

 5.48 ms

Line 31-34 in File `ConvergenceToken.sol`

```
31  /*@CTK "Context _msgData"
32     @tag assume_completion
33     @post __return == msg.data
34  */
```

Line 36-39 in File `ConvergenceToken.sol`


```
36  function _msgData() internal view virtual returns (bytes memory) {
37      this; // silence state mutability warning without generating
↪  bytecode - see https://github.com/ethereum/solidity/issues/2691
38      return msg.data;
39  }
```

 The code meets the specification.

Formal Verification Request 3

SafeMath `tryAdd`

 23, Mar 2021

 38.0 ms

Line 147-153 in File `ConvergenceToken.sol`

```
147  /*@CTK "SafeMath tryAdd"
148      @tag spec
149      @tag is_pure
150      @post a + b < a || a + b < b -> __has_overflow
151      @post a + b < a -> __return == false && __return1 == 0
152      @post a + b >= a -> __return == true && __return1 == a + b
153  */
```

Line 155-159 in File ConvergenceToken.sol


```
155  function tryAdd(uint256 a, uint256 b) internal pure returns (bool,
↪  uint256) {
156      uint256 c = a + b;
157      if (c < a) return (false, 0);
158      return (true, c);
159  }
```

 The code meets the specification.

Formal Verification Request 4

SafeMath trySub

 23, Mar 2021

 11.55 ms

Line 167-172 in File ConvergenceToken.sol

```
167  /*@CTK "SafeMath trySub"
168      @tag spec
169      @tag is_pure
170      @post b > a -> __return == false && __return1 == 0
171      @post b <= a -> __return == true && __return1 == a - b
172  */
```

Line 174-177 in File ConvergenceToken.sol


```
174  function trySub(uint256 a, uint256 b) internal pure returns (bool,
↪  uint256) {
175      if (b > a) return (false, 0);
176      return (true, a - b);
177  }
```

 The code meets the specification.

Formal Verification Request 5

SafeMath tryMul

 23, Mar 2021

 95.86 ms

Line 185-191 in File ConvergenceToken.sol

```

185     /*@CTK "SafeMath tryMul"
186         @tag spec
187         @tag is_pure
188         @post a == 0 -> __return == true &&& __return1 == 0
189         @post a != 0 &&& (a * b) / a != b -> __return == false &&& __return1
↳ == 0
190         @post a != 0 &&& (a * b) / a == b -> __return == true &&& __return1 ==
↳ a * b
191     */

```

Line 193-201 in File ConvergenceToken.sol

```

193     function tryMul(uint256 a, uint256 b) internal pure returns (bool,
↳ uint256) {
194         // Gas optimization: this is cheaper than requiring 'a' not being
↳ zero, but the
195         // benefit is lost if 'b' is also tested.
196         // See:
↳ https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
197         if (a == 0) return (true, 0);
198         uint256 c = a * b;
199         if (c / a != b) return (false, 0);
200         return (true, c);
201     }


```

✔ The code meets the specification.

Formal Verification Request 6

SafeMath tryDiv

 23, Mar 2021

 12.81 ms

Line 209-214 in File ConvergenceToken.sol

```

209     /*@CTK "SafeMath tryDiv"
210         @tag spec
211         @tag is_pure
212         @post b == 0 -> __return == false &&& __return1 == 0
213         @post b != 0 -> __return == true &&& __return1 == a / b
214     */

```

Line 216-219 in File ConvergenceToken.sol

```

216     function tryDiv(uint256 a, uint256 b) internal pure returns (bool,
↳ uint256) {
217         if (b == 0) return (false, 0);
218         return (true, a / b);
219     }


```

✔ The code meets the specification.

Formal Verification Request 7

SafeMath tryMod

 23, Mar 2021

 14.44 ms

Line 227-232 in File ConvergenceToken.sol

```
227  /*@CTK "SafeMath tryMod"
228     @tag spec
229     @tag is_pure
230     @post b == 0 -> __return == false && __return1 == 0
231     @post b != 0 -> __return == true && __return1 == a % b
232  */
```

Line 234-237 in File ConvergenceToken.sol


```
234  function tryMod(uint256 a, uint256 b) internal pure returns (bool,
↪  uint256) {
235      if (b == 0) return (false, 0);
236      return (true, a % b);
237  }
```

 The code meets the specification.

Formal Verification Request 8

If method completes, integer overflow would not happen.

 23, Mar 2021

 18.96 ms

Line 250 in File ConvergenceToken.sol

```
250  //@CTK NO_OVERFLOW
```

Line 261-265 in File ConvergenceToken.sol


```
261  function add(uint256 a, uint256 b) internal pure returns (uint256) {
262      uint256 c = a + b;
263      require(c >= a, "SafeMath: addition overflow");
264      return c;
265  }
```

 The code meets the specification.

Formal Verification Request 9

SafeMath add

 23, Mar 2021

 4.15 ms

Line 251-259 in File ConvergenceToken.sol

```
251  /*@CTK "SafeMath add"
252     @tag spec
253     @tag is_pure
254     @post (a + b < a || a + b < b) == __reverted
255     @post !__reverted -> __return == a + b
256     @post !__reverted -> !__has_overflow
257     @post !__reverted -> !__has_assertion_failure
258     @post !(__has_buf_overflow)
259  */
```

Line 261-265 in File ConvergenceToken.sol


```
261  function add(uint256 a, uint256 b) internal pure returns (uint256) {
262      uint256 c = a + b;
263      require(c >= a, "SafeMath: addition overflow");
264      return c;
265  }
```

✔ The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

 23, Mar 2021

 17.51 ms

Line 278 in File ConvergenceToken.sol

```
278  //@CTK NO_OVERFLOW
```

Line 290-293 in File ConvergenceToken.sol


```
290  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
291      require(b <= a, "SafeMath: subtraction overflow");
292      return a - b;
293  }
```

✔ The code meets the specification.

Formal Verification Request 11

SafeMath sub

 23, Mar 2021

 2.27 ms

Line 279-288 in File ConvergenceToken.sol

```
279  /*@CTK "SafeMath sub"
280     @tag spec
281     @tag is_pure
```



```

282     @pre b <= a
283     @post (a < b) == __reverted
284     @post !__reverted -> __return == a - b
285     @post !__reverted -> !__has_overflow
286     @post !__reverted -> !__has_assertion_failure
287     @post !(__has_buf_overflow)
288     */

```

Line 290-293 in File ConvergenceToken.sol

```

290     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
291         require(b <= a, "SafeMath: subtraction overflow");
292         return a - b;
293     }

```

✔ The code meets the specification.

Formal Verification Request 12

If method completes, integer overflow would not happen.

23, Mar 2021

55.72 ms

Line 306 in File ConvergenceToken.sol

```

306     /*@CTK NO_OVERFLOW

```

Line 318-323 in File ConvergenceToken.sol

```

318     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
319         if (a == 0) return 0;
320         uint256 c = a * b;
321         require(c / a == b, "SafeMath: multiplication overflow");
322         return c;
323     }

```

✔ The code meets the specification.

Formal Verification Request 13

SafeMath mul

23, Mar 2021

132.1 ms

Line 307-316 in File ConvergenceToken.sol

```

307     /*@CTK "SafeMath mul"
308         @tag spec
309         @tag is_pure
310         @tag assume_completion

```

```

311     @post (((a) > (0)) ES (((a) * (b)) / (a)) != (b))) == (__reverted)
312     @post !__reverted -> __return == a * b
313     @post !__reverted == !__has_overflow
314     @post !__reverted -> !__has_assertion_failure
315     @post !(__has_buf_overflow)
316 */

```

Line 318-323 in File ConvergenceToken.sol

```

318     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
319         if (a == 0) return 0;
320         uint256 c = a * b;
321         require(c / a == b, "SafeMath: multiplication overflow");
322         return c;
323     }


```

✔ The code meets the specification.

Formal Verification Request 14

If method completes, integer overflow would not happen.

 23, Mar 2021

 13.43 ms

Line 338 in File ConvergenceToken.sol

```

338     //@CTK NO_OVERFLOW

```

Line 350-353 in File ConvergenceToken.sol

```

350     function div(uint256 a, uint256 b) internal pure returns (uint256) {
351         require(b > 0, "SafeMath: division by zero");
352         return a / b;
353     }


```

✔ The code meets the specification.

Formal Verification Request 15

SafeMath div

 23, Mar 2021

 2.04 ms

Line 339-348 in File ConvergenceToken.sol

```

339     /*@CTK "SafeMath div"
340         @tag spec
341         @tag is_pure
342         @tag assume_completion
343         @post (b <= 0) == __reverted

```

```

344     @post !__reverted -> __return == a / b
345     @post !__reverted -> !__has_overflow
346     @post !__reverted -> !__has_assertion_failure
347     @post !(__has_buf_overflow)
348 */

```

Line 350-353 in File ConvergenceToken.sol

```

350     function div(uint256 a, uint256 b) internal pure returns (uint256) {
351         require(b > 0, "SafeMath: division by zero");
352         return a / b;
353     }


```

 The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.

 23, Mar 2021

 14.42 ms

Line 368 in File ConvergenceToken.sol

```

368     //@CTK NO_OVERFLOW

```

Line 380-383 in File ConvergenceToken.sol

```

380     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
381         require(b > 0, "SafeMath: modulo by zero");
382         return a % b;
383     }


```

 The code meets the specification.

Formal Verification Request 17

SafeMath mod

 23, Mar 2021

 1.85 ms

Line 369-378 in File ConvergenceToken.sol

```

369     /*@CTK "SafeMath mod"
370         @tag spec
371         @tag is_pure
372         @tag assume_completion
373         @post b != 0 -> !__reverted
374         @post !__reverted -> __return == a % b
375         @post !__reverted -> !__has_overflow
376         @post !(__has_buf_overflow)
377         @post !(__has_assertion_failure)
378     */

```

Line 380-383 in File ConvergenceToken.sol


```
380     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
381         require(b > 0, "SafeMath: modulo by zero");
382         return a % b;
383     }
```

✔ The code meets the specification.

Formal Verification Request 18

If method completes, integer overflow would not happen.

 23, Mar 2021

 13.94 ms

Line 399 in File ConvergenceToken.sol

```
399     //@CTK NO_OVERFLOW
```

Line 411-418 in File ConvergenceToken.sol


```
411     function sub(
412         uint256 a,
413         uint256 b,
414         string memory errorMessage
415     ) internal pure returns (uint256) {
416         require(b <= a, errorMessage);
417         return a - b;
418     }
```

✔ The code meets the specification.

Formal Verification Request 19

SafeMath sub_with_err

 23, Mar 2021

 1.89 ms

Line 400-409 in File ConvergenceToken.sol

```
400     /*@CTK "SafeMath sub_with_err"
401         @tag spec
402         @tag is_pure
403         @pre b <= a
404         @post (a < b) == __reverted
405         @post !__reverted -> __return == a - b
406         @post !__reverted -> !__has_overflow
407         @post !__reverted -> !__has_assertion_failure
408         @post !(__has_buf_overflow)
409     */
```

Line 411-418 in File ConvergenceToken.sol

```

411     function sub(
412         uint256 a,
413         uint256 b,
414         string memory errorMessage
415     ) internal pure returns (uint256) {
416         require(b <= a, errorMessage);
417         return a - b;
418     }


```

✔ The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

 23, Mar 2021

 14.03 ms

Line 436 in File ConvergenceToken.sol

```

436     //@CTK NO_OVERFLOW

```

Line 448-455 in File ConvergenceToken.sol

```

448     function div(
449         uint256 a,
450         uint256 b,
451         string memory errorMessage
452     ) internal pure returns (uint256) {
453         require(b > 0, errorMessage);
454         return a / b;
455     }


```

✔ The code meets the specification.

Formal Verification Request 21

SafeMath div_with_err

 23, Mar 2021

 2.19 ms

Line 437-446 in File ConvergenceToken.sol

```

437     /*@CTK "SafeMath div_with_err"
438         @tag spec
439         @tag is_pure
440         @tag assume_completion
441         @post (b <= 0) == __reverted
442         @post !__reverted -> __return == a / b

```

```
443     @post !__reverted -> !__has_overflow
444     @post !__reverted -> !__has_assertion_failure
445     @post !(__has_buf_overflow)
446 */
```

Line 448-455 in File ConvergenceToken.sol


```
448     function div(
449         uint256 a,
450         uint256 b,
451         string memory errorMessage
452     ) internal pure returns (uint256) {
453         require(b > 0, errorMessage);
454         return a / b;
455     }
```

✔ The code meets the specification.

Formal Verification Request 22

If method completes, integer overflow would not happen.

 23, Mar 2021

 12.32 ms

Line 473 in File ConvergenceToken.sol

```
473     //@CTK_NO_OVERFLOW
```

Line 485-492 in File ConvergenceToken.sol


```
485     function mod(
486         uint256 a,
487         uint256 b,
488         string memory errorMessage
489     ) internal pure returns (uint256) {
490         require(b > 0, errorMessage);
491         return a % b;
492     }
```

✔ The code meets the specification.

Formal Verification Request 23

SafeMath mod_with_err

 23, Mar 2021

 1.81 ms

Line 474-483 in File ConvergenceToken.sol

```

474  /*@CTK "SafeMath mod_with_err"
475      @tag spec
476      @tag is_pure
477      @tag assume_completion
478      @post b != 0 -> !__reverted
479      @post !__reverted -> __return == a % b
480      @post !__reverted -> !__has_overflow
481      @post !(__has_buf_overflow)
482      @post !(__has_assertion_failure)
483  */

```

Line 485-492 in File ConvergenceToken.sol

```

485  function mod(
486      uint256 a,
487      uint256 b,
488      string memory errorMessage
489  ) internal pure returns (uint256) {
490      require(b > 0, errorMessage);
491      return a % b;
492  }


```

 The code meets the specification.

Formal Verification Request 24

ERC20 constructor

 23, Mar 2021

 16.85 ms

Line 546-551 in File ConvergenceToken.sol

```

546  /*@CTK "ERC20 constructor"
547      @tag assume_completion
548      @post __post._name == name_
549      @post __post._symbol == symbol_
550      @post __post._decimals == 18
551  */

```

Line 552-556 in File ConvergenceToken.sol

```

552  constructor(string memory name_, string memory symbol_) public {
553      _name = name_;
554      _symbol = symbol_;
555      _decimals = 18;
556  }


```

 The code meets the specification.

Formal Verification Request 25

ERC20 name

 23, Mar 2021

 5.71 ms

Line 562-565 in File ConvergenceToken.sol

```
562  /*@CTK "ERC20 name"
563     @tag assume_completion
564     @post __return == _name
565  */
```

Line 566-568 in File ConvergenceToken.sol


```
566  function name() public view virtual returns (string memory) {
567      return _name;
568  }
```

 The code meets the specification.

Formal Verification Request 26

ERC20 symbol

 23, Mar 2021

 7.15 ms

Line 574-577 in File ConvergenceToken.sol

```
574  /*@CTK "ERC20 symbol"
575     @tag assume_completion
576     @post __return == _symbol
577  */
```

Line 578-580 in File ConvergenceToken.sol


```
578  function symbol() public view virtual returns (string memory) {
579      return _symbol;
580  }
```

 The code meets the specification.

Formal Verification Request 27

ERC20 decimals

 23, Mar 2021

 4.6 ms

Line 596-599 in File ConvergenceToken.sol


```

596  /*@CTK "ERC20 decimals"
597      @tag assume_completion
598      @post __return == _decimals
599  */

```

Line 600-602 in File ConvergenceToken.sol

```

600  function decimals() public view virtual returns (uint8) {
601      return _decimals;
602  }

```

✔ The code meets the specification.

Formal Verification Request 28

ERC20 totalSupply

📅 23, Mar 2021

🕒 6.38 ms

Line 607-610 in File ConvergenceToken.sol

```

607  /*@CTK "ERC20 totalSupply"
608      @tag assume_completion
609      @post __return == _totalSupply
610  */

```

Line 611-613 in File ConvergenceToken.sol

```

611  function totalSupply() public view virtual override returns (uint256) {
612      return _totalSupply;
613  }

```

✔ The code meets the specification.

Formal Verification Request 29

ERC20 balanceOf

📅 23, Mar 2021

🕒 7.22 ms

Line 619-622 in File ConvergenceToken.sol

```

619  /*@CTK "ERC20 balanceOf"
620      @tag assume_completion
621      @post __return == _balances[account]
622  */

```

Line 623-625 in File ConvergenceToken.sol

```

623  function balanceOf(address account) public view virtual override returns
↪  (uint256) {
624      return _balances[account];
625  }


```

✔ The code meets the specification.

Formal Verification Request 30

ERC20 transfer

 23, Mar 2021

 202.25 ms

Line 636-643 in File ConvergenceToken.sol

```

636     /*@CTK "ERC20 transfer"
637         @tag assume_completion
638         @pre recipient != address(0)
639         @pre amount <= _balances[msg.sender]
640         @post msg.sender == recipient -> _balances[msg.sender] ==
↪     __post._balances[msg.sender]
641         @post msg.sender != recipient -> __post._balances[msg.sender] ==
↪     _balances[msg.sender] - amount
642         @post msg.sender != recipient -> __post._balances[recipient] ==
↪     _balances[recipient] + amount
643     */

```

Line 644-647 in File ConvergenceToken.sol

```

644     function transfer(address recipient, uint256 amount) public virtual
↪     override returns (bool) {
645         _transfer(_msgSender(), recipient, amount);
646         return true;
647     }


```

 The code meets the specification.

Formal Verification Request 31

ERC20 allowance

 23, Mar 2021

 5.11 ms

Line 653-656 in File ConvergenceToken.sol

```

653     /*@CTK "ERC20 allowance"
654         @tag assume_completion
655         @post __return == _allowances[owner][spender]
656     */

```

Line 657-659 in File ConvergenceToken.sol

```

657     function allowance(address owner, address spender) public view virtual
↪     override returns (uint256) {
658         return _allowances[owner][spender];
659     }


```

 The code meets the specification.

Formal Verification Request 32

ERC20 approve

 23, Mar 2021

 70.08 ms

Line 669-673 in File ConvergenceToken.sol

```

669  /*@CTK "ERC20 approve"
670     @tag assume_completion
671     @pre spender != address(0)
672     @post __post._allowances[msg.sender][spender] == amount
673  */

```

Line 674-677 in File ConvergenceToken.sol

```

674  function approve(address spender, uint256 amount) public virtual override
↪  returns (bool) {
675      _approve(_msgSender(), spender, amount);
676      return true;
677  }


```

 The code meets the specification.

Formal Verification Request 33

ERC20 transferFrom

 23, Mar 2021

 364.76 ms

Line 693-703 in File ConvergenceToken.sol

```

693  /*@CTK "ERC20 transferFrom"
694     @tag assume_completion
695     @pre sender != address(0)
696     @pre recipient != address(0)
697     @pre amount <= _balances[sender] && amount <=
↪  _allowances[sender][msg.sender]
698     @post sender == recipient -> _balances[sender] ==
↪  __post._balances[sender]
699     @post sender != recipient -> __post._balances[sender] ==
↪  _balances[sender] - amount
700     @post sender != recipient -> __post._balances[recipient] ==
↪  _balances[recipient] + amount
701     @post __post._allowances[sender][msg.sender] ==
↪  _allowances[sender][msg.sender] - amount
702     @post __return ==true
703  */

```

Line 704-716 in File ConvergenceToken.sol

```

704     function transferFrom(
705         address sender,
706         address recipient,
707         uint256 amount
708     ) public virtual override returns (bool) {
709         _transfer(sender, recipient, amount);
710         _approve(
711             sender,
712             _msgSender(),
713             _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer
↪ amount exceeds allowance")
714         );
715         return true;
716     }

```

✔ The code meets the specification.

Formal Verification Request 34

ERC20 increaseAllowance

23, Mar 2021

54.18 ms

Line 731-735 in File ConvergenceToken.sol

```

731     /*@CTK "ERC20 increaseAllowance"
732         @tag assume_completion
733         @pre spender != address(0)
734         @post __post._allowances[msg.sender][spender] ==
↪ _allowances[msg.sender][spender] + addedValue
735     */

```

Line 736-739 in File ConvergenceToken.sol

```

736     function increaseAllowance(address spender, uint256 addedValue) public
↪ virtual returns (bool) {
737         _approve(_msgSender(), spender,
↪ _allowances[_msgSender()][spender].add(addedValue));
738         return true;
739     }

```

✔ The code meets the specification.

Formal Verification Request 35

ERC20 decreaseAllowance

23, Mar 2021

66.76 ms

Line 757-762 in File ConvergenceToken.sol

```

757     /*@CTK "ERC20 decreaseAllowance"
758         @tag assume_completion
759         @pre spender != address(0)
760         @pre subtractedValue < _allowances[msg.sender][spender]
761         @post __post._allowances[msg.sender][spender] ==
↪     _allowances[msg.sender][spender] - subtractedValue
762     */

```

Line 763-770 in File ConvergenceToken.sol

```

763     function decreaseAllowance(address spender, uint256 subtractedValue)
↪     public virtual returns (bool) {
764         _approve(
765             _msgSender(),
766             spender,
767             _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20:
↪     decreased allowance below zero")
768         );
769         return true;
770     }

```

✔ The code meets the specification.

Formal Verification Request 36

ERC20 _transfer

📅 23, Mar 2021

🕒 75.47 ms

Line 788-796 in File ConvergenceToken.sol

```

788     /*@CTK "ERC20 _transfer"
789         @tag assume_completion
790         @pre sender != address(0)
791         @pre recipient != address(0)
792         @pre amount <= _balances[sender]
793         @post sender == recipient -> _balances[sender] ==
↪     __post._balances[sender]
794         @post sender != recipient -> __post._balances[sender] ==
↪     _balances[sender] - amount
795         @post sender != recipient -> __post._balances[recipient] ==
↪     _balances[recipient] + amount
796     */

```

Line 797-810 in File ConvergenceToken.sol

```

797     function _transfer(
798         address sender,
799         address recipient,
800         uint256 amount

```

```

801     ) internal virtual {
802         require(sender != address(0), "ERC20: transfer from the zero
↪ address");
803         require(recipient != address(0), "ERC20: transfer to the zero
↪ address");
804
805         _beforeTokenTransfer(sender, recipient, amount);
806
807         _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer
↪ amount exceeds balance");
808         _balances[recipient] = _balances[recipient].add(amount);
809         emit Transfer(sender, recipient, amount);
810     }

```

✔ The code meets the specification.

Formal Verification Request 37

ERC20 `_mint`

📅 23, Mar 2021

🕒 88.21 ms

Line 822-827 in File ConvergenceToken.sol

```

822     /*@CTK "ERC20 _mint"
823         @tag assume_completion
824         @pre account != address(0)
825         @post __post._totalSupply == _totalSupply + amount
826         @post __post._balances[account] == _balances[account] + amount
827     */

```

Line 828-836 in File ConvergenceToken.sol

```

828     function _mint(address account, uint256 amount) internal virtual {
829         require(account != address(0), "ERC20: mint to the zero address");
830
831         _beforeTokenTransfer(address(0), account, amount);
832
833         _totalSupply = _totalSupply.add(amount);
834         _balances[account] = _balances[account].add(amount);
835         emit Transfer(address(0), account, amount);
836     }

```

✔ The code meets the specification.

Formal Verification Request 38

ERC20 `_burn`

📅 23, Mar 2021

🕒 202.18 ms

Line 850-856 in File ConvergenceToken.sol

```

850  /*@CTK "ERC20 _burn"
851     @tag assume_completion
852     @pre account != address(0)
853     @pre amount <= _balances[account] && amount <= _totalSupply
854     @post __post._totalSupply == _totalSupply - amount
855     @post __post._balances[account] == _balances[account] - amount
856  */

```

Line 857-865 in File ConvergenceToken.sol

```

857  function _burn(address account, uint256 amount) internal virtual {
858      require(account != address(0), "ERC20: burn from the zero address");
859
860      _beforeTokenTransfer(account, address(0), amount);
861
862      _balances[account] = _balances[account].sub(amount, "ERC20: burn
↪ amount exceeds balance");
863      _totalSupply = _totalSupply.sub(amount);
864      emit Transfer(account, address(0), amount);
865  }


```

✔ The code meets the specification.

Formal Verification Request 39

ERC20 _approve

 23, Mar 2021

 3.02 ms

Line 881-886 in File ConvergenceToken.sol

```

881  /*@CTK "ERC20 _approve"
882     @tag assume_completion
883     @pre owner != address(0)
884     @pre spender != address(0)
885     @post __post._allowances[owner][spender] == amount
886  */

```

Line 887-897 in File ConvergenceToken.sol

```

887  function _approve(
888      address owner,
889      address spender,
890      uint256 amount
891  ) internal virtual {
892      require(owner != address(0), "ERC20: approve from the zero address");
893      require(spender != address(0), "ERC20: approve to the zero address");

```

```

894
895     _allowances[owner][spender] = amount;
896     emit Approval(owner, spender, amount);
897 }

```

✔ The code meets the specification.

Formal Verification Request 40

ERC20 _setupDecimals

📅 23, Mar 2021

🕒 6.67 ms

Line 907-910 in File ConvergenceToken.sol

```

907     /*@CTK "ERC20 _setupDecimals"
908         @tag assume_completion
909         @post __post._decimals == decimals_
910     */

```

Line 911-913 in File ConvergenceToken.sol

```

911     function _setupDecimals(uint8 decimals_) internal virtual {
912         _decimals = decimals_;
913     }

```

✔ The code meets the specification.

Formal Verification Request 41

ConvergenceToken constructor

📅 23, Mar 2021

🕒 152.77 ms

Line 949-954 in File ConvergenceToken.sol

```

949     /*@CTK "ConvergenceToken constructor"
950         @tag assume_completion
951         @pre genesis_holder != address(0)
952         @post __post._totalSupply == _totalSupply + TOTAL_SUPPLY
953         @post __post._balances[genesis_holder] == _balances[genesis_holder]
↪ + TOTAL_SUPPLY
954     */

```

Line 955-958 in File ConvergenceToken.sol

```

955     constructor(address genesis_holder) {
956         require(genesis_holder != address(0), "ConvergenceToken: zero
↪ address");
957         _mint(genesis_holder, TOTAL_SUPPLY);
958     }

```

✔ The code meets the specification.

Source Code with CertiK Labels

ConvergenceToken.sol

```

1  /**
2   *Submitted for verification at Etherscan.io on 2021-03-22
3   */
4
5  // SPDX-License-Identifier: MIT
6
7  // File @openzeppelin/contracts/utils/Context.sol@v3.4.1
8
9  pragma solidity >=0.6.0 <0.8.0;
10
11 /*
12  * @dev Provides information about the current execution context, including
13  * ↪ the
14  * ↪ sender of the transaction and its data. While these are generally
15  * ↪ available
16  * ↪ via msg.sender and msg.data, they should not be accessed in such a direct
17  * ↪ manner, since when dealing with GSN meta-transactions the account sending
18  * ↪ and
19  * ↪ paying for execution may not be the actual sender (as far as an
20  * ↪ application
21  * ↪ is concerned).
22  *
23  * This contract is only required for intermediate, library-like contracts.
24  */
25 abstract contract Context {
26     /*@CTK "Context _msgSender"
27     @tag assume_completion
28     @post __return == msg.sender
29     */
30     /* CertiK Smart Labelling, for more details visit: https://certik.org */
31     function _msgSender() internal view virtual returns (address payable) {
32         return msg.sender;
33     }
34
35     /*@CTK "Context _msgData"
36     @tag assume_completion
37     @post __return == msg.data
38     */
39     /* CertiK Smart Labelling, for more details visit: https://certik.org */
40     function _msgData() internal view virtual returns (bytes memory) {
41         this; // silence state mutability warning without generating
42         ↪ bytecode - see https://github.com/ethereum/solidity/issues/2691
43         return msg.data;
44     }

```

```

40 }
41
42 // File @openzeppelin/contracts/token/ERC20/IERC20.sol@v3.4.1
43
44
45
46 /**
47  * @title ConvergenceToken
48  *
49  * @dev A minimal ERC20 token contract for the Convergence token.
50  */
51 contract ConvergenceToken is ERC20("Convergence", "CONV") {
52     uint256 private constant TOTAL_SUPPLY = 10000000000e18;
53
54
55     /*@CTK "ConvergenceToken constructor"
56      @tag assume_completion
57      @pre genesis_holder != address(0)
58      @post __post._totalSupply == _totalSupply + TOTAL_SUPPLY
59      @post __post._balances[genesis_holder] == _balances[genesis_holder]
60      + TOTAL_SUPPLY
61      */
62     constructor(address genesis_holder) {
63         require(genesis_holder != address(0), "ConvergenceToken: zero
64         address");
65         _mint(genesis_holder, TOTAL_SUPPLY);
66     }
67 }

```

