Security Assessment & Formal Verification Agora Dollar Core Final Report

July 2024

Prepared for
Agora Ledger Corp.

certora AGORA

Table of content

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Agora Dollar Core	https://github.com/agora-finance/agora-dollar-evm-rc/	20795bd	EVM/Solidity 0.8

Project Overview

This document describes the specification and verification of Agora Dollar using the Certora Prover and manual code review findings. The work was undertaken from July 7, 2024 to July 29, 2024.

Agora creates a USD-pegged stablecoin which is meant to be traded over an EVM-compatible blockchain. Agora tokens are minted and burned by those who have the respective roles. Those who have the appropriate roles can also freeze the activity of a certain user, pause several functions, and upgrade the protocol's functions.

The following contract list is included in our scope:

agora-dollar-evm

```
contracts/AgoraDollar.sol
contracts/AgoraDollarCore.sol
contracts/AgoraDollarAccessControl.sol
contracts/Eip3009.sol
contracts/Eip712.sol
contracts/Erc20Core.sol
contracts/Erc20Privileged.sol
contracts/Erc2612.sol
```

The Certora Prover demonstrated that the implementation of the Solidity contracts above is correct with respect to the formal rules written by the Certora team. In addition, the team performed a manual audit of all the Solidity contracts. During the verification process and the

manual audit, the Certora team discovered bugs in the Solidity contracts code, as listed on the following page.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	0		
High	0		
Medium	0		
Low	0		
Informational	2		
Total	**2**		

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
I-01	IAgoraDollar.ConstructorParams mismatches the used ContructorParams struct in the AgoraDollar contact	Informational	
I-02	IAgoraDollar.ConstructorParams mismatches the used ContructorParams struct in the AgoraDollar contact	Informational	

Informational Severity Issues

I-01. Message sender restriction on the transferWithAuthorization function could be bypassed

Description: In the AgoraDollarErc1967Proxy contract, a frozen user should not be able to call the transferWithAuthorization function if the _isMsgSenderFrozenCheckEnabled flag is true. However, this restriction could easily be bypassed just by calling this function using a different address. While this check could still make sense under some assumptions, it is important to note that it is not a real restriction on the user.

Customer's response: Acknowledged

I-02. IAgoraDollar.ConstructorParams mismatches the used ConstructorParams struct in the AgoraDollar contract

Description: In IAgoraDollar.ConstructorParams, the field address proxyAddress is missing, making the interface incorrect:

In IAgoraDollar.sol:

```
struct ConstructorParams {
    string name;
    string symbol;
    string eip712Name;
    string eip712Version;
}
```

In AgoraDollarCore.sol:

```
struct ConstructorParams {
    string name;
    string symbol;
    string eip712Name;
    string eip712Version;
    address proxyAddress;
}
```

Customer's response: fixed in commit TODO

Formal Verification

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counterexample exists that violates one of the assertions of the rule.

Formal Verification Properties

AgoraDollar.sol

Module General Assumptions

- Loop iterations: Any loop was unrolled at most 4 times (iterations)
- While technically possible (see I-01), we assume that balances do not overflow on transfer or mint operations.
- Additionally we implement the methods transfer(), transferFrom(), transferWithAuthorization() and receiveWithAuthorization() as non-upgraded versions of the implementations in AgoraDollarProxyERC1967.sol to ensure correctness of parameterized rules and invariants.
- Signature Check summarization

Contract Properties

P-01. Correctness of ERC20 functions.	
Status: Verified	

Rule Name	Status	Description	Link to rule report
approveIntegrity	Verified	*Calling approve changes the allowance correctly.*	*Report*
approveDoesNotAffectThirdParty	Verified	*Calls to approve do change any third party allowances.*	
approveRevertingConditions	Verified	*Calls to approve revert if and only if* *- msg.value != 0*	
burnIntegrity	Verified	*Calling burn decreases the balance and the totalSupply correctly, and can only be called by the burnerRole.*	

burnDoesNotAffectThirdParty	Verified	*Calls to batchBurnFrom do not change any third party balances.*
burnRevertingConditions	Verified	*Calls to batchBurnFrom revert if and only if* - *msg.value != 0* - *Msg.sender != burnerAddress* - *burnFrom is paused* - *Not enough balance of from address* - *Underflow: burned amount > totalSupply()*
mintIntegrity	Verified	*Calling batchMint increases the balance and the totalSupply correctly, and can only be called by the minterRole.*
mintDoesNotAffectThirdParty	Verified	*Calls to batchMint do not change any third party balances.*
mintRevertingConditions	Verified	*Calls to batchMint revert if and only if* - *msg.value != 0* - *Msg.sender != minterAddress* - *mint is paused* - *Minted amount > max_uint248 (technically possible, see I-01)* - *Account minted to is the zero address* - *Overflow: minted amount + totalSupply() > max_uint256 or balanceOf(to) + amount > max_uint248*
transferIntegrity	Verified	*Calling transfer increases and decreases the according balances correctly*
transferDoesNotAffectThirdParty	Verified	*Calls to transfer do not change any third party balances.*

P-02. Valid State Changes and Authorized Calls

Status: Verified

Rule Name	Status	Description	Link to rule report
onlyBatchMethodsChangeMoreThanTwoBalances	Verified	*Only calls to batchBurn() and batchMint() can change more than two balances at once.*	*Report*
onlyAllowedMethodsMayChangeAllowance	Verified	*Only approve and permit can increase allowance. Only approve, permit and transferFrom can decrease allowance*	
onlyAllowedMethodsMayChangeBalance	Verified	*Only batchMint, transfer, transferFrom, transferWithAuthorization, receiveWithAuthorization can increase balance. Only batchBurn, transfer, transferFrom, transferWithAuthorization, receiveWithAuthorization can decrease balance.*	

onlyAllowedMethodsMayChangeTotalSupply	Verified	*Only batchMint and batchBurn can increase and decrease totalSupply respectively.*	
onlyAuthorizedCanTransfer	Verified	*Only the token holder or an approved third party can reduce an account's balance.*	
onlyHolderOfSpenderCanChangeAllowance	Verified	*Only the token holder (or a permit) can increase allowance. The spender can decrease it by using it*	
onlyAllowedMethodsMayChangeFrozenBalance	Verified	*Only batchMint can increase a frozen account's balance, only batchBurn can decrease a frozen account's balance*	
onlyAllowedMethodsMayChangeBalanceWhenTransferPaused	Verified	*Only batchMint can increase an account's balance, only batchBurn can decrease an account's balance when transfer is paused*	
noBalanceChangeOnFullPause	Verified	*No balance can change when mint, burn and transfer is paused*	

P-03. Correctness of Contract Specific Methods.

Status: Verified

Rule Name	Status	Description	Link to rule report
freezeIntegrity	*Verified*	*Only freezerAddress can freeze an account*	*Report*
freezeRevertCo	*Verified*	*Calls to freeze revert if and only if*	

nditions		- *msg.value != 0* - *Msg.sender != freezerRole* - *Freezing is paused*	
unfreezeIntegrity	*Verified*	*Only freezerAddress can unfreeze an account*	
unfreezeRevertConditions	*Verified*	*Calls to unfreeze revert if and only if* - *msg.value != 0* - *Msg.sender != freezerRole* - *Freezing is paused*	
implementationAddressNeverChanges	*Verified*	*No calls to AgoraDollar change the implementationAddress field of the contractData in storage*	*Report*

P-04. Signature Checks

Status: Verified

Rule Name	Status	Description	Link to rule report
NoDiffBetweenPermitImplementations	*Verified*	*Both permit implementations behave identically*	*Report*
NoDiffBetweenReceiveImplementations	*Verified*	*Both receiveWithAuthorization implementations behave the same way*	
NoDiffBetweenTransferImplementations	*Verified*	*Both transferWithAuthorization implementations behave the same way*	

transferWithAuthorizationRevertCondition	*Verified*	*Calls to transferWithAuthorization revert if and only if* - *Transfer is paused* - *owner account does not have enough funds* - *msgSender is frozen and frozen check is enabled* - *From or to account is frozen* - *Validity expired* - *Overflow of spender balance + transferred amount* - *nonce not authorized* - *Signature verification is paused* - *msgSender is not recipient* - *Signature not valid*
receiveWithAuthorizationRevertCondition	*Verified*	*Calls to receiveWithAuthorization revert if and only if* - *Transfer is paused* - *owner account does not have enough funds* - *msgSender is frozen and frozen check is enabled* - *From or to account is frozen* - *Validity expired* - *Overflow of spender balance + transferred amount* - *nonce not authorized* - *Signature verification is paused* - *msgSender is not recipient* - *Signature not valid*
permitRevertConditions	*Verified*	*Calls to receiveWithAuthorization revert if and only if* - *msgValue != 0* - *Deadline expired* - *Signature verification paused* - *Signature not valid*
signatureIsUniquePerHolder	*Verified*	*Signature is unique between two accounts*

signatureIsUniqueePerHolderV2	*Verified*	*Signature is unique between two accounts even when spender address, amounts and deadlines vary*	

P-04. Reentrancy

Status: Verified

Rule Name	Status	Description	Link to rule report
reentrancySafety	Verified	*Verifies that all external calls are either first or last operation of a transaction thus ensuring reentrancy safety*	*Report*

AgoraDollarERC1967Proxy.sol

Module General Assumptions

- Loop iterations: Any loop was unrolled at most 4 times (iterations)
- We recheck non-upgraded paths of ERC20 implementations on the proxy and that DELEGATECALL opcode is executed on upgrades.
- SignatureCheck: we assume a deterministic version of isValidSignatureNow() implemented purely in CVL respecting the axiomatization of ecrecover()

Contract Properties

P-01. ERC20 Implementations

Status: Verified

Rule Name	Status	Description	Link to rule report
transferIntegrityNotUpgraded	Verified	*Calling transfer increases and decreases the according balances correctly*	*Report*
transferDoesNotAffectThirdPartyNotUpgraded	Verified	*Calls to transfer do not change any third party balances.*	
transferRevertingConditionsNotUpgraded	Verified	*Calls to transfer revert if and only if* - *msg.value != 0* - *msgSender or to are frozen accounts* - *Transfer is paused* - *msgSender does not have enough funds* *(Note while an overflow of amount + balance(to) is technically possible, an overflow here will not revert due to unchecked blocks, see I-01)*	
transferIsOneWayAdditiveNotUpgraded	Verified	*Splitting an amount into multiple transfers works the same as transferring one big amount*	

transferFromIntegrityNotUpgraded	Verified	*Calling transfer increases and decreases the according balances correctly*
transferFromDoesNotAffectThirdPartyNotUpgraded	Verified	*Calls to transfer do not change any third party balances.*
transferFromRevertingConditionsNotUpgraded	Verified	*Calls to transfer revert if and only if* - *msg.value != 0* - *from or to are frozen accounts* - *Transfer is paused* - *owner account does not have enough funds* - *Allowance not enough* - *msgSender is frozen and frozen check is enabled*
transferFromIsOneWayAdditiveNotUpgraded	Verified	*Splitting an amount into multiple transfers works the same as transferring one big amount*
onlyAuthorizedCanTransferIntegrityNotUpgraded	Verified	*only authorized can reduce an account's balance via calling transferWithAuthorization*
authorizedTransferDoesNotAffectThirdPartyNotUpgraded	Verified	*transferWithAuthorization does not affect third party balances*
onlyAuthorizedCanReceiveIntegrityNotUpgraded	Verified	*only authorized can receive an account's balance via calling receiveWithAuthorization*
authorizedReceiveDoesNotAffectThirdPartyNotUpgraded	Verified	*receiveWithAuthorization does not affect third party balances*

delegateCallOnlyUpgradeable	Verified	*This rule ensures twofold that if a delegatecall is executed an upgradeable method or the fallback was invoked. If an upgradeable method is upgraded and invoked, a delegatecall must have been executed.*	

P-02. The stored implementationAddress can only change when fallback() is invoked

Status: Verified

Rule Name	Status	Description	Link to rule report
implementationAddressChangnesOnlyOnUpgrade	Verified	*All calls to AgoraDollarErc1967Proxy except for the fallback() function leave the implementationAddress field of contractData storage slot unchanged.*	*Report*

P-03. Signature Checks

Status: Verified

Rule Name	Status	Description	Link to rule report
NoDiffBetweenReceiveImplementationsNotUpgraded	*Verified*	*Both receiveWithAuthorization implementations behave the same way*	*Report*
NoDiffBetweenTransferImplementationsNotUpgraded	*Verified*	*Both transferWithAuthorization implementations behave the same way*	

transferWithAuthorizationRevertConditionNotUpgraded	*Verified*	*Calls to transferWithAuthorization revert if and only if* - *Transfer is paused* - *owner account does not have enough funds* - *msgSender is frozen and frozen check is enabled* - *From or to account is frozen* - *Validity expired* - *Overflow of spender balance + transferred amount* - *nonce not authorized* - *Signature verification is paused* - *msgSender is not recipient* - *Signature not valid*	
receiveWithAuthorizationRevertConditionNotUpgraded	*Verified*	*Calls to receiveWithAuthorization revert if and only if* - *Transfer is paused* - *owner account does not have enough funds* - *msgSender is frozen and frozen check is enabled* - *From or to account is frozen* - *Validity expired* - *Overflow of spender balance + transferred amount* - *nonce not authorized* - *Signature verification is paused* - *msgSender is not recipient* - *Signature not valid*	

P-04. Reentrancy

Status: Verified

Rule Name	Status	Description	Link to rule report
reentrancySafety	Verified	*Verifies that all external calls are either first or last operation of a transaction thus ensuring reentrancy safety*	*Report*

Disclaimer

The Certora Prover takes a contract and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification and the Certora Prover does not check any cases not covered by the specification.

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.