SMART CONTRACT AUDIT

ZOKYO.

Oct 18th, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

TECHNICAL SUMMARY

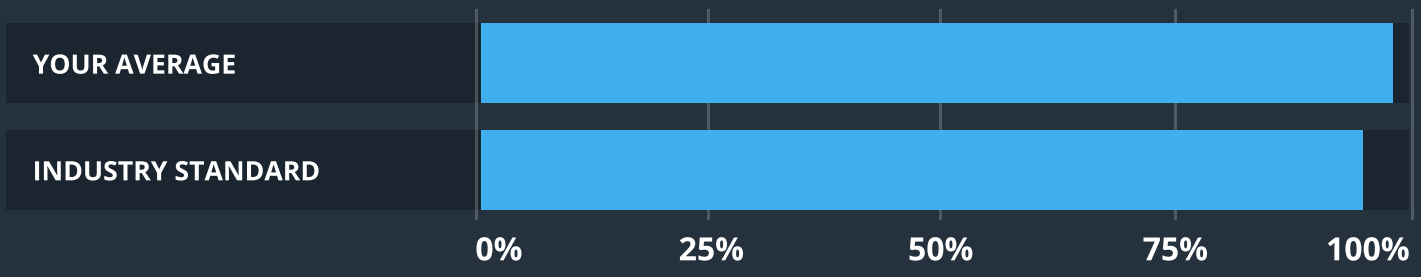
This document outlines the overall security of the DeBridge Finance smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the DeBridge Finance smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical issues found during the audit.

Testable Code



The testable code is 98.74%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the DeBridge Finance team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the DeBridge Finance repository.

Repository:
https://github.com/debridge-finance/debridge-contracts-v1

Last commit:
5cc5e2b3c88f2ffaa6f8ffda676c82f3b956df7a

Contracts:

- AggregatorBase;
- ConfirmationAggregator;
- DeBridgeGate;
- SignatureAggregator;
- SignatureVerifier;
- CallProxy;
- WrappedAsset;
- SignatureUtil.

Requirements:
https://docs.debridge.finance/

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of DeBridge Finance smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Hardhat testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.
2	Cross-comparison with other, similar smart contracts by industry leaders.
3	Testing contract logic against common and uncommon attack vectors.
4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

Zokyo team has conducted a security audit. During the audit we found 3 informational issues and 3 issues with low severity. All of them are successfully resolved by DeBridge team.

The contracts are well written and structured. The overall score for the codebase is set to 99.

Based on the re-audit and the test results, we can state that the findings have no impact anymore on contract performance or security. Hence, the contract is fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

LOW | RESOLVED

Duplicating code leaves the codebase prone to the introduction of errors. Errors can inadvertently be introduced when functionality changes are not replicated across all instances of code that should be identical. Logic and storage duplication between ConfirmationAggregator and SignatureVerifier, the logic in deployAsset function is duplicated and also the storage slots of getDeployInfo are duplicated on ConfirmationAggregator, SignatureAggregator, and SignatureVerifier, this can run on synchronization problems between storages.

Recommendation:
Implement a common module to store the data and inherit that module by the other contracts, and also a common module for the duplicated logic should be added.

LOW | RESOLVED

Lack of input validation in functions updateChainSupport, updateAssetFixedFees, updateExcessConfirmations, setChainSupport, setCallProxy, setAggregator, setSignatureVerifier, setDefiController, setFeeProxy, setWeth, updateFlashFee, updateCollectRewardBps, updateFeeDiscount, updateTreasury from DeBridgeGate contract, these are all functions that take input from “trusted” parties, but those inputs must be checked too because “trusted” parties can make mistakes too, for example, you should check if the value you want to send is equal with the value that is already stored, to prevent unnecessary gas used, if a require would detect that the values are the same, then the gas used would be reverted, or to check if an address or uint256 is equal to 0.

Recommendation:
Add input validation to do sanity checks.

LOW | RESOLVED

Many of the event definitions are lacking indexed parameters. Some examples including, all events in IAggregatorBase, all events in IConfirmationAggregator, all events in IDeBridgeGate, all events in ISignatureAggregator, all events in DelegateStaking.

Recommendation:
Index the parameters, as another best practice recommendation is that both the new value set and the old value set should be added in the event.

INFORMATIONAL | RESOLVED

Specifying a pragma version with the caret symbol (^) upfront which tells the compiler to use any version of solidity bigger than specified considered not a good practice. Since there could be major changes between versions that would make your code unstable. The latest known versions with bugs are 0.8.3 and 0.8.4.

Recommendation:
Set the latest version without the caret. (The latest version that is also known as bug-free is 0.8.7).

INFORMATIONAL | RESOLVED

In the DeBridgeGate contract, Initialize function, the ChainSupportInfo data is provided, later in the internal _send function, the contract is using that data, for example, the fixedNativeFee field, if that field has the value zero, the execution it is stopped because of the next require line (line 864), because there is no sanitization on the initialization in the Initialize function.

Recommendation:
Put some sanitization requirements in the Initialize function for the data structures. We know that function can only be called by the team, but mistakes can be made internally too, that's why this is classified just as informational and not as a medium problem.

INFORMATIONAL | RESOLVED

In the DeBridgeGate contract, Initialize function, it would be useful to add in the for loop condition an OR (“ || “) logical operator to check for the gas left, this way if the _supportedChainIds array is too large, the execution will not stop with “out of gas” and block it.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by DeBridge Finance team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
libraries/	88.89	50.00	100.00	100.00	
Flags.sol	100.00	100.00	100.00	100.00	
SignatureUtil.sol	85.71	50.00	100.00	100.00	
periphery/	95.00	68.18	91.67	93.48	
CallProxy.sol	90.91	75.00	83.33	88.46	48, 54, 120
DeBridgeToken.sol	100	50.00	100.00	100.00	
transfers/	91.40	76.06	84.06	91.14	
AggregatorBase.sol	95.24	73.53	100.00	95.35	118, 131
ConfirmationAggregator.sol	90.70	77.78	77.78	90.70	40, 41, 66, 149
DeBridgeGate.sol	91.56	77.68	83.72	90.99	... 840, 841, 949
SignatureVerfier.sol	87.18	70.83	75.00	88.10	... 121, 122, 154
All files	**91.71**	**74.31**	**86.05**	**91.61**	

Test Results

Contract: ConfirmationAggregator

Your project has Truffle migrations, which have to be turned into a fixture to run your tests with Hardhat

- ✓ should have correct initial values

Test setting configurations by different users

- ✓ should set min confirmations if called by the admin (40ms)
- ✓ should set excessConfirmations if called by the admin (41ms)
- ✓ should set confirmationThreshold if called by the admin (49ms)
- ✓ should revert adding new oracle if minConfirmations will be too low (60ms)
- ✓ should increase minConfirmations before adding new oracle (45ms)
- ✓ should add new oracle if called by the admin (115ms)
- ✓ should revert decreasing minConfirmations if value is too low
- ✓ should updateOracle oracle (disable) if called by the admin (108ms)
- ✓ should decrease minConfirmations after disabling oracle
- ✓ should revert enabling oracle if minConfirmations will be too low
- ✓ should reject setting min confirmations if called by the non-admin
- ✓ should reject setting excessConfirmations if called by the non-admin
- ✓ should reject setting confirmationThreshold if called by the non-admin
- ✓ should reject adding the new oracle if called by the non-admin
- ✓ should reject removing the new oracle if called by the non-admin

Test data submission

- ✓ should submit mint identifier by the oracle (46ms)
- ✓ should submit burnt identifier by the oracle
- ✓ should submit mint identifier by the second oracle (49ms)
- ✓ should submit burnt identifier by the second oracle (56ms)
- ✓ should submit mint identifier by the extra oracle (66ms)
- ✓ should submit burnt identifier by the extra oracle (49ms)
- ✓ should reject submission of mint identifier if called by the non-admin
- ✓ should reject submission of burnt identifier if called by the non-admin (39ms)
- ✓ should reject submission of duplicate mint identifiers with the same id by the same oracle
- ✓ should reject submission of duplicate burnt identifiers with the same id by the same oracle

CallProxy

Direct interaction

```
    plain calls
      ✓ when receiver is EOA - ETH goes to receiver 54ms)
      ✓ when receiver is reverting contract - ETH goes to reserve (84ms)
      when receiver contract is executed - ETH stays on receiver
        ✓ plain ETH to payable receive() (123ms)
        ✓ plain ETH to payable fallback() (136ms)
        ✓ single-argument uint256 call (152ms)
        ✓ call with array (234ms)
      when receiver is uniswap
        ✓ successful swapExactETHForTokens (78ms)
        ✓ reverting swapExactETHForTokens
        ✓ successful swapETHForExactTokens (76ms)
        ✓ if swapExactETHForTokens reverts - ETH go to reserve (43ms)
    ERC20 calls
      ✓ reverts if _token fails (82ms)
      ✓ when receiving contract doesn't pull tokens - they transfer to reserve (106ms)
      ✓ when receiver is EOA - tokens transfer to reserve (104ms)
      ✓ when receiver is reverting contract - tokens go to reserve (97ms)
      when receiver is token-pulling contract
        ✓ receiver got data and pulled tokens from proxy (127ms)
      when receiver is uniswap
      ✓ swapTokensForExactTokens (99ms)
      ✓ swapTokensForExactETH (112ms)
      ✓ swapExactTokensForETH (115ms)
  Cross-contract interaction
    plain calls
      ✓ when receiver is EOA
      ✓ when receiver is reverting contract (52ms)
      when receiver is ordinary contract
        ✓  single-argument uint256 call (70ms)
        ✓ call with array (115ms)
      when receiver is uniswap
        ✓ successful swapExactETHForTokens (71ms)
        ✓ reverting swapExactETHForTokens
        ✓ swapETHForExactTokens (69ms)
        ✓ reverting swapExactETHForTokens (42ms)
    ERC20 calls
```

✓ reverts if _token fails (60ms)
✓ when receiving contract doesn't pull tokens - they transfer to reserve (111ms)
✓ when receiver is EOA - tokens transfer to reserve (128ms)
✓ when receiver is reverting contract (194ms)
when receiver is ordinary contract
✓ single-argument uint256 call and pulled tokens (82ms)
✓ receiver got data and pulled tokens (132ms)
when receiver is uniswap
✓ swapTokensForExactTokens (95ms)
✓ swapTokensForExactETH (129ms)
✓ swapExactTokensForETH (109ms)

Contract: DeBridgeGate full mode
✓ Check init params
✓ should update excessConfirmations if called by the admin
Role-based security checks for setters
✓ should set aggregator if called by the admin
✓ should set free proxy if called by the admin
✓ should set defi controller if called by the admin
✓ should update Chain Support if called by the admin and emits ChainsSupportUpdated event
✓ should set Chain Supporting if called by the admin and emits ChainSupportUpdated event (57ms)
✓ should set CallProxy if called by the admin and emits CallProxyUpdated
✓ should update flash fee if called by the admin
✓ should reject setting aggregator if called by the non-admin
✓ should reject setting free proxy if called by the non-admin
✓ should reject setting defi controller if called by the non-admin
✓ should reject setting flash fee if called by the non-admin
✓ should reject updating Chain Support if called by the non-admin
✓ should reject updating Asset Sixed Fees if called by the non-admin
✓ should reject setting Chain Id Support if called by the non-admin
✓ should reject setting CallProxy if called by the non-admin
✓ should reject stopping (pausing) all transfers if called buy the non-gov-monitoring
✓ should reject allowing (uppausing) all transfers if called buy the non-admin
✓ should reject setting amount flashFeeBps if called by the non-admin
with LINK and DBR assets
✓ getBalance returns a zero balance eth
✓ getBalance returns a zero balance token
with flash contract

✓ flash increases balances and counters of received funds (111ms)
✓ flash reverts if not profitable (52ms)
with uniswap periphery
with feeProxy
with confirmation aggregator
✓ debridge and aggregator are linked together
with oracles
✓ should confirm new asset if called by the oracles (268ms)
Test mint method
✓ check confirmation without DSRM confirmation
✓ should reject native token without DSRM confirmation (54ms)
✓ debridge and aggregator are linked together
✓ debridge and aggregator are linked together
confirm by required oracle
✓ check confirmation with DSRM confirmation
✓ should reject exceed amount
✓ should reject withdrawing too many fees
✓ should reject withdrawing fees if the token not from current chain
update reduce ExcessConfirmations
✓ should reject when the submission is blocked (41ms)
✓ should unblock the submission by admin
✓ should reject minting with unconfirmed submission
✓ should reject funding treasury if the token not from current chain
should mint when the submission is approved
✓ should reject minting twice
Test burn method discount: 0%
✓ should burning with permit when the amount is sufficient (181ms)
✓ should burning when the amount is sufficient(_useAssetFee=true) (185ms)
Test burn method discount: 50%
✓ should burning with permit when the amount is sufficient (162ms)
✓ should burning when the amount is sufficient(_useAssetFee=true) (184ms)
Test burn method discount: 100%
✓ should burning with permit when the amount is sufficient (219ms)
✓ should burning when the amount is sufficient(_useAssetFee=true) (181ms)
After oracle submitted
✓ check confirmation without DSRM confirmation (181ms)
✓ should reject native token without DSRM confirmation (38ms)
after confirmation by required oracle

✓ should reject when the submission is blocked (54ms)
✓ check confirmation with DSRM confirmation

should unblock the submission by admin

✓ getBalance returns balance eth that went into functions send + fee
✓ getBalance returns token that went into functions send
✓ should claim ERC20 when the submission is approved (69ms)
✓ should reject claiming with unconfirmed submission
✓ should reject claiming the token from outside chain

After claim native token

✓ should reject claiming twice

Test send method. discount: discount 0%

✓ should send native tokens from the current chain (89ms)
✓ should send ERC20 tokens from the current chain (132ms)
✓ should send ERC20 tokens from the current chain (_useAssetFee=true) (340ms)
✓ should reverts if amount more than maxAmount (84ms)
✓ should reject sending too mismatched amount of native tokens (44ms)
✓ should reject sending tokens to unsupported chain

When transfers are stopped (pause)

✓ should rejects if transfers were stopped by admin

Test send method. discount: discount 50%

✓ should send native tokens from the current chain (106ms)
✓ should send ERC20 tokens from the current chain (138ms)
✓ should send ERC20 tokens from the current chain (_useAssetFee=true) (144ms)
✓ should reverts if amount more than maxAmount (71ms)
✓ should reject sending too mismatched amount of native tokens (56ms)
✓ should reject sending tokens to unsupported chain (47ms)

When transfers are stopped (pause)

✓ should rejects if transfers were stopped by admin

Test send method. discount: discount 100%

✓ should send native tokens from the current chain (105ms)
✓ should send ERC20 tokens from the current chain (137ms)
✓ should send ERC20 tokens from the current chain (_useAssetFee=true) (134ms)
✓ should reverts if amount more than maxAmount (46ms)
✓ should reverts if amount more than maxAmount (46ms)
✓ should reject sending too mismatched amount of native tokens
✓ should reject sending tokens to unsupported chain

When transfers are stopped (pause)

```
    ✓ should rejects if transfers were stopped by admin

  Contract: DeBridgeGate light mode
minConfirmations: 7
confirmationThreshold: 5
excessConfirmations: 7
initialOracles.length: 10
    Test setting configurations by different users
    ✓ should set Verifier if called by the admin
    ✓ should set defi controller if called by the admin
    ✓ should reject setting Verifier if called by the non-admin
    ✓ should reject setting defi controller if called by the non-admin
  Test managing assets
    ✓ should add external asset if called by the admin (665ms)
    ✓ should reject add external asset without DSRM confirmation (153ms)
    ✓ should reject add external asset without -1 confirmation (98ms)
  Test send method
    ✓ should send native tokens from the current chain 76ms)
    ✓ should send ERC20 tokens from the current chain (128ms)
    ✓ should reject sending the too mismatched amount of native tokens
    ✓ should reject sending tokens to unsupported chain
  Test mint method
    ✓ should mint when the submission is approved (141ms)
    ✓ should reject minting with unconfirmed submission
    ✓ should reject minting with error signature (86ms)
    ✓ should reject minting twice (86ms)
  Test burn method
    ✓ should burning when the amount is sufficient (223ms)
  Test claim method
    ✓ should reject native token without DSRM confirmation (98ms)
    ✓ should claim native token when the submission is approved (134ms)
    ✓ should claim ERC20 when the submission is approved (143ms)
    ✓ should reject claiming with unconfirmed submission (75ms)
    ✓ should reject claiming twice (84ms)

Contract: DeBridgeGate full with auto
  Test setting configurations by different users
    ✓ should set aggregator if called by the admin
```

```
    ✓ should set fee proxy if called by the admin
    ✓ should set defi controller if called by the admin
    ✓ should reject setting aggregator if called by the non-admin
    ✓ should reject setting fee proxy if called by the non-admin
    ✓ should reject setting defi controller if called by the non-admin
  Test managing assets
    ✓ should add external asset if called by the admin (242ms)
  Test send method
    ✓ should send native tokens from the current chain (66ms)
    ✓ should send ERC20 tokens from the current chain (138ms)
    ✓ should reject sending too mismatched amount of native tokens (75ms)
    ✓ should reject sending tokens to unsupported chain
  Test mint method
    ✓ should mint when the submission is approved (78ms)
    ✓ should reject minting with unconfirmed submission (78ms)
    ✓ should reject minting twice (62ms)
  Test burn method
    ✓ should burning when the amount is sufficient (262ms)
  Test claim method
    ✓ should claim native token when the submission is approved (93ms)
    ✓ should claim ERC20 when the submission is approved (99ms)
    ✓ should reject claiming with unconfirmed submission
    ✓ should reject claiming twice

Contract: DeBridgeGate real pipeline mode
  Configure contracts
    ✓ Check init contract params (86ms)
    ✓  Initialize oracles (248ms)
    ✓ Update fixed fee for WETH (77ms)
  Test setting configurations by different users
    ✓ should set aggregator if called by the admin
    ✓ should set debridgeGate to fee proxy if called by the admin (54ms)
    ✓ should set fee proxy if called by the admin (39ms)
    ✓ should set defi controller if called by the admin (41ms)
    ✓ should reject setting aggregator if called by the non-admin
    ✓ should reject setting fee proxy if called by the non-admin
    ✓ should reject setting defi controller if called by the non-admin
  Test managing assets
```

- ✓ should revert deploying new asset without confirm (130ms)
- ✓ should confirm new asset if called by the oracles (1808ms)
- ✓ should deploy new asset (525ms)
- ✓ should reject deploy new asset twice
- ✓ should have same output for getDebridgeId/getbDebridgeId (43ms)

Test send method from ETH to BSC. discount: 0%

- ✓ set discount 0% fee for customer alice
- ✓ should send native tokens 93ms)
- ✓ should send ERC20 tokens without permit (157ms)
- ✓ should send ERC20 tokens with permit (170ms)
- ✓ should reject sending too mismatched amount of native tokens
- ✓ should reject sending tokens to unsupported chain
- ✓ should support non EVM receiver parameter (162ms)

Test send method from ETH to BSC. discount: 50%

- ✓ set discount 50% fee for customer alice
- ✓ should send native tokens (103ms)
- ✓ should send ERC20 tokens without permit (151ms)
- ✓ should send ERC20 tokens with permit (161ms)
- ✓ should reject sending too mismatched amount of native tokens
- ✓ should reject sending tokens to unsupported chain
- ✓ should support non EVM receiver parameter (121ms)

Test send method from ETH to BSC. discount: 100%

- ✓ set discount 100% fee for customer alice
- ✓ should send native tokens (85ms)
- ✓ should send ERC20 tokens without permit (149ms)
- ✓ should send ERC20 tokens with permit (177ms)
- ✓ should reject sending too mismatched amount of native tokens
- ✓ should reject sending tokens to unsupported chain
- ✓ should support non EVM receiver parameter (148ms)

Test mint method (BSC network)

- ✓ Oracles confirm transfers (without required oracle) (1464ms)
- ✓ heck confirmation without required oracle
- ✓ should reject native token without confirmation from required oracle
- ✓ confirm by required oracle (41ms)
- ✓ check confirmations
- ✓ update reduce ExcessConfirmations if called by the admin
- ✓ should reject when the submission is blocked (49ms)
- ✓ should unblock the submission by admin

- ✓ should mint (deETH) when the submission is approved (93ms)
- ✓ should mint (deLink) when the submission is approved (90ms)
- ✓ should reject minting with unconfirmed submission
- ✓ should reject minting twice

Test burn method (BSC network) discount: 0%

- ✓ set discount 0% fee for customer bob
- ✓ should burning (deETH, deLink) when the amount is sufficient (364ms)
- ✓ should support non EVM receiver parameter (479ms)

Test burn method (BSC network) discount: 50%

- ✓ set discount 0% fee for customer bob
- ✓ should burning (deETH, deLink) when the amount is sufficient (416ms)
- ✓ should support non EVM receiver parameter (416ms)

Test burn method (BSC network) discount: 100%

- ✓ set discount 100% fee for customer bob
- ✓ should burning (deETH, deLink) when the amount is sufficient (405ms)
- ✓ should support non EVM receiver parameter (359ms)

Test claim method (ETH network)

- ✓ check view method is valid signature
- ✓ should reject when the submission is blocked
- ✓ should unblock the submission by admin
- ✓ should reject when exist duplicate signatures (49ms)
- ✓ should claim native token when the submission is approved (109ms)
- ✓ should claim ERC20 when the submission is approved (97ms)
- ✓ should reject claiming with unconfirmed submission (59ms)
- ✓ should reject claiming twice (68ms)

Test transfer between BSC to HECO

- ✓ should send native tokens (from BSC to HECO) (73ms)
- ✓ should send ERC20 (Cake) tokens (from BSC to HECO) (133ms)
- ✓ Oracles confirm transfers (337ms)
- ✓ should mint (deBSC) when the submission is approved (93ms)
- ✓ should mint (deCake) when the submission is approved (75ms)
- ✓ should burn (deCake in HECO network) (157ms)

Test autosubmissions

- ✓ should have same submissionId for same autoparams (81ms)

Test submission flags

Test UNWRAP_ETH flag

- ✓ should burn/claim with UNWRAP_ETH flag without auto call data (227ms)
- ✓ should burn/claim with UNWRAP_ETH flag with auto call data (321ms)

Test PROXY_WITH_SENDER flag
- ✓ should set callProxy with sender (238ms)
- ✓ should use different proxy with PROXY_WITH_SENDER flag (280ms)

Test REVERT_IF_EXTERNAL_FAIL flag
- ✓ should revert if externall call fails with REVERT_IF_EXTERNAL_FAIL flag (368ms)

Collect fee management
- ✓ FeeProxy should set FeeProxyAddress it is called by the admin (204ms)
- ✓ should withdraw fee of ERC20 token (BSC network, deLink) if it is called by the worker (189ms)
- ✓ should auto claim fee transaction (burn event deLink from BSC to ETH) (124ms)
- ✓ should withdraw fee of ERC20 token (HECO network, deCake) if it is called by the worker (224ms)
- ✓ should auto claim fee transaction (burn event deCake from HECO to BSC) (566ms)
- ✓ should withdraw fee of ERC20 token (ETH network, Link) if it is called by the worker (139ms)
- ✓ should reject withdrawing fee by non-worker (47ms)

PriceConsumer
- ✓ deployer is owner
- ✓ non-owner is unable to add priceFeed

using aggregator as price feed
- ✓ token price is available

Contract: SignatureAggregator
- ✓ should have correct initial values

Test setting configurations by different users
- ✓ should set min confirmations if called by the admin
- ✓ should add a new oracle if called by the admin
- ✓ should updateOracle oracle (disable) if called by the admin
- ✓ should reject setting min confirmations if called by the non-admin
- ✓ should reject adding the new oracle if called by the non-admin
- ✓ should reject removing the new oracle if called by the non-admin

Test submission
- ✓ ✓ should submit identifier by the oracle
- ✓ should submit identifier by the second oracle 38ms)
- ✓ should submit identifier by the extra oracle (46ms)
- ✓ should reject submission of identifier if called by the non-admin
- ✓ should reject submition of duplicated identifiers with the same id by the same oracle

Test managing assets
- ✓ should add new asset if called by the oracle (66ms)

```
    ✓ should reject if called by the non-oracle (38ms)
    ✓ should reject if called by the same oracle again (42ms)

  274 passing (3m)
```

Tests written by Zokyo Security team

As part of our work assisting DeBridge Finance in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Hardhat testing framework.

Tests were based on the functionality of the code, as well as a review of the DeBridge Finance contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
libraries/	100.00	100.00	100.00	100.00	
Flags.sol	100.00	100.00	100.00	100.00	
SignatureUtil.sol	100.00	100.00	100.00	100.00	
periphery/	100.00	95.45	100.00	97.83	
CallProxy.sol	100.00	93.75	100.00	96.15	83
DeBridgeToken.sol	100.00	100.00	100.00	100.00	
transfers/	98.57	93.62	100.00	98.34	
AggregatorBase.sol	100.00	100.00	100.00	100.00	
ConfirmationAggregator.sol	100.00	100.00	100.00	100.00	
DeBridgeGate.sol	98.22	91.07	100.00	97.85	... 718, 719, 916
SignatureVerfier.sol	97.44	91.67	100.00	97.62	77
All files	**98.74**	**94.04**	**100.00**	**98.32**	

✓ should call internal claim function, execution fee bigger then 0 (244ms)
✓ should call internal claim function, execution fee bigger then 0, data length 0 (265ms)
✓ should call internal send, permit length 0, is native false, fail wrong target chain (159ms)
✓ should call internal send, permit length 0, is native false, fail transfer amount mismatch (78ms)
✓ should call internal send, permit length 0, is native false, fail amount mismatch (45ms)
✓ should call internal send, permit length 0, is native false, msg.value equal amount, fail not supported fix fee (48ms)
✓ should call internal add asset, should fail zero address (95ms)
✓ should call internal add asset, should fail zero address (52ms)
✓ should call internal add asset, should work
✓ should call internal check confirmations, fail not confirmed (72ms)
✓ should call internal check confirmations, amount smaller then amount threshold (96ms)
✓ should call internal check confirmations, amount bigger or equal with amount threshold, fail not confirmed (82ms)
✓ should call internal check confirmations, amount bigger or equal with amount threshold, submission blocked (119ms)
✓ should deploy new asset,signatures lenght bigger then 0, should work (427ms)
✓ should deploy new asset,signatures lenght bigger then 0, should fail, asset already exist (179ms)
✓ should deploy new asset,signatures lenght bigger equal 0, should fail, asset not confirmed (120ms)
✓ should get defi available reserves
✓ should updateFeeDiscount, should fail, wrong arguments
✓ should updateFeeDiscount, should work
✓ should set chain support
✓ should call internal validate auto params
✓ block submissions with false is blocked
✓ update flash fee, should fail, wrong arguments
✓ should set signature verifier
✓ should set defi controller
✓ should get native token info
✓ should get version
✓ should revert call with fee proxy bad role
✓ should revert call with gov monitoring bad role

Contract: **ConfirmationAggregator**

Your project has Truffle migrations, which have to be turned into a fixture to run your tests with Hardhat

✓ should submit many, by oracle, should succeed
✓ should set min confirmations, as admin, should succeed
✓ should set min confirmations, not as admin, should fail

- ✓ should update oracle, fail not exist
- ✓ should confirm new asset, fail deployed already (47ms)
- ✓ should confirm new asset, oracle required true (67ms)
- ✓ should confirm new asset, oracle required false and min confirmation 0 (86ms)
- ✓ should call internal submit function if path (53ms)
- ✓ should update oracle, should fail, oracle not found (38ms)
- ✓ should update oracle
- ✓ should update oracle admin, fail only callable by admin
- ✓ should set min confirmations, fail should low min confirmations
- ✓ should update oracle, fail wrong arguments
- ✓ should update oracle, should succeed, oracle required true and internal required false (44ms)
- ✓ should update oracle, should succeed, oracle required false and internal required false (44ms)
- ✓ should set confirmation threshold
- ✓ should set confirmation threshold, fail wrong arguments
- ✓ should set excess confirmations
- ✓ should set excess confirmations, fail low min confirmations
- ✓ should call submit as not an oracle, fail bad role
- ✓ should call submit, as oracle, fail submitted already
- ✓ should call submit, as oracle, required true, go inside block confirmations condition (90ms)
- ✓ should call submit, as oracle, required true, not go inside block confirmations condition (56ms)
- ✓ should call submit, as oracle, current block same value (59ms)
- ✓ should call submit, as oracle, fail submitted already (70ms)
- ✓ should get submission confirmations
- ✓ should call getConfirmedDeployId
- ✓ should call version
- ✓ should add oracle, fail wrong arguments
- ✓ should add oracle, fail low min confirmations
- ✓ should add oracle, fail already exist
- ✓ should add oracle, should work required true
- ✓ should update oracle, oracle info is valid true and is valid local false, oracle on index (49ms)
- ✓ should update oracle, oracle info is valid true and is valid local false, oracle not on index (57ms)
- ✓ should update oracle, oracle info is valid false and is valid local true, fail min confirmations (57ms)
- ✓ should initialize new aggregator, fail low min confirmations

Contract: SignatureUtil

- ✓ should split bytes signature, fail invalid signature length
- ✓ should parse bytes signature, fail signature invalid
- ✓ should parse bytes signature, fail wrong arguments

WrappedAsset

- ✓ should call decimals and verify
- ✓ should call permit, fail permit expired (48ms)
- ✓ should mint assets, as minter, and burn them
- ✓ should call mint function without minter role
- ✓ should call permit with valid signature
- ✓ should call permit with invalid signature (40ms)

Contract: SignatureVerifier

- ✓ should set debridge address
- ✓ should set threshold fail wrong arguments (53ms)
- ✓ should set threshold, work
- ✓ should call version
- ✓ should call submit, fail not confirmed (74ms)
- ✓ should call submit, oracle is valid false, submision has verified false (51ms)
- ✓ should call submit, fail not confirmed by required oracle (97ms)
- ✓ should call submit, oracle required true (86ms)
- ✓ should call submit, oracle is valid true, block is confirmed (73ms)
- ✓ should call submit, fail not confirmed threshold (97ms)
- ✓ should call submit, work, global excess confirmations smaller then confirmations (99ms)
- ✓ should call submit, fail debridge gate bad role
- ✓ should call is valid signature

Contract: CallProxy

- ✓ should call version

BigNumber.toString does not accept any parameters; base-10 is assumed

- ✓ should use call function, fail, external call failed, flags 0 (143ms)
- ✓ should use call function, should work (124ms)
- ✓ should use call function, should work, external call result true (110ms)
- ✓ should use call function, should work, flag proxy sender (143ms)
- ✓ should use call function, should fail call failed (158ms)
- ✓ should use callERC20, fail gate role
- ✓ should use callERC20, fail external call failed (46ms)

127 passing (47s)

We are grateful to have been given the opportunity to work with the DeBridge Finance team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the DeBridge Finance team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.