
# SMART CONTRACT AUDIT

ZOKYO.

Oct 22st, 2021 | v. 1.0

## PASS

Zokyo Security team has concluded that the given smart contracts passed security audit and are fully production-ready

# TECHNICAL SUMMARY

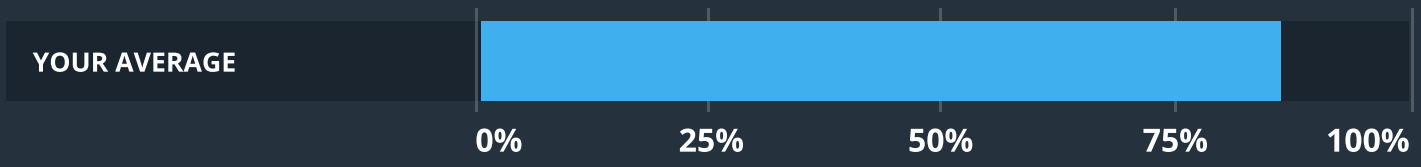
This document outlines the overall security of the Layer Zero smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Layer Zero 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 86.23%.

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 Layer Zero 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 . . . . . 3

Summary . . . . . 5

Structure and Organization of Document . . . . . 6

Complete Analysis . . . . . 7

Code Coverage and Test Results for all files . . . . . 18

- Tests written by Layer Zero team . . . . . 18
- Tests written by Zokyo Secured team . . . . . 24

# AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Layer Zero repository.

**Repository:**
https://github.com/ryanzarick/LayerZero/commit/a9af62a3d3f4c4b45e8ccce2dd2d067fe784350f

**Last commit:**
7ed3fd9ce07a6c0a78d621b016f16027ee58b62f

**Contracts under the scope:**

- RelayerToken;
- ChainlinkOracleClient;
- Treasury;
- Communicator;
- Network;
- Validator;
- Relayer;
- LayerZeroToken;
- RelayerStaking;
- ERC1363;
- IERC1363Receiver;
- IERC1363;
- IERC1363Spender;
- ILayerZeroEndpoint;
- ILayerZeroTreasury;
- ILayerZeroOracle;
- ILayerZeroValidationLibrary;
- ILayerZeroReceiver;
- ILayerZeroValidator;
- ILayerZeroRelayer;
- LayerZeroPacket;
- Buffer;
- EVMValidator;
- ECVerify;
- EthereumDecoder;
- MPT;
- RLPDecode;
- RLPEncode;
- Decoder.

**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 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 Truffle 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. | 3 | Testing contract logic against common and uncommon attack vectors. |
| 2 | Cross-comparison with other, similar smart contracts by industry leaders. | 4 | Thorough, manual review of the codebase, line-by-line. |

# SUMMARY

The Zokyo team has conducted a security audit of the given codebase. The contracts provided for an audit are well written and structured. All the findings within the auditing process are presented in this document.

There were no critical issues found during the auditing process.  High, medium,  low, and some informational issues were found. All of them were successfully resolved by the Layer Zero team. After a review of the fixes and comments from the Layer Zero team, we decided to mark some issues as not valid.

All the mentioned findings may have an effect only in the case of specific conditions performed by the contract owner. We can give the score of 98% to the provided codebase.

## 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

## unallocatedRewards value remains zeroed

**HIGH** | RESOLVED

In RelayerStaking.sol, unallocatedRewards which is important to trigger massUpdatePools(address _relayer) remains zero value for _relayer. Going through different scenarios, it does not seem to be one in which the map is going to take a non-zero value. onTransferReceived() is the function that updates this value and it is not triggered anywhere inside the project (unless this is intended to be called by an outside arbitrary caller to mimic a callback taking place on any transfer occurring on LayerZeroToken outside, is it ?).

This is part of the economic model of LayerZero which is not supposed to hinder the functional operation of cross-chain transactions but it is still significant in terms of funds that might be misallocated.

**Recommendation:**
onTransferReceived is the only function that updates the value of unallocatedRewards. As it is part of ERC1363, it should be called by the LayerZeroToken via transferAndCall or transferFromAndCall but contracts only call layerZeroToken.transfer() which does not trigger onTransferReceived.

## relayerPoolInfo amount value remains zeroed

HIGH | RESOLVED

In RelayerStaking.sol, rpi.amount which is an element of relayerPoolInfo normally starts as Zero but undesirably remains zero. The issue becomes critical in massUpdatePools(address _relayer) as it comprises this operation rpi.accLayerZeroPerShare = rpi.accLayerZeroPerShare.add(poolAmount.mul(1e12).div(rpi.amount)); which divides by zero.

In a test scenario like this:

```
it.only("pendingLayerZero() - runs properly and returns correct value", async function() {
    //TODO: requires a scenario with stake and unstake
    let pid = 0;
    let ONE_TOKEN = ONE_HUNDRED_TOKENS.div(new BN(100));
    await this.mockERC20.mint(this.bob, ONE_HUNDRED_TOKENS); // 100 ETH
    await this.mockERC20.approve(this.relayerStakingWithTokenZock.address, ONE_HUNDRED_TOKENS,
{from: this.bob}) ;
    await this.layerZeroToken.mint(this.relayerStakingWithTokenZock.address, ONE_HUNDRED_TOKENS); //
100 ETH
    await this.relayerStakingWithTokenZock.add(new BN(10), this.mockERC20.address );

    await this.layerZeroTokenZock.setRelayerStaking(this.relayerStakingWithTokenZock.address);
    let tx  = await this.layerZeroTokenZock.onTransferReceived(this.testAddr, this.testAddr, ONE_TOKEN,
this.relayer);

    expect(await this.relayerStakingWithTokenZock.unallocatedRewards(this.relayer)).to.be.bignumber.equal
(ONE_TOKEN);

    await this.relayerStakingWithTokenZock.stake(this.relayer, pid, ONE_TOKEN, {from: this.bob});
    expect(await this.mockERC20.balanceOf( this.relayerStakingWithTokenZock.address)).to.be.bignumber.
equal(ONE_TOKEN);

    let user = await this.relayerStakingWithTokenZock.userInfo(this.relayer, pid, this.bob);

    expect(user.amount).to.be.bignumber.equal(ONE_TOKEN);
    expect(user.rewardDebt).to.be.bignumber.equal(new BN(0));

    expect(await this.layerZeroToken.balanceOf(this.bob)).to.be.bignumber.equal(new BN(0));
```

```
    await this.relayerStakingWithTokenZock.stake(this.relayer, pid, ONE_TOKEN, {from: this.bob});
    expect(await this.mockERC20.balanceOf( this.relayerStakingWithTokenZock.address)).to.be.bignumber.
equal(ONE_TOKEN.mul(new BN(2)));
    // TODO: expect equal ONE_TOKEN
    console.log(await this.layerZeroToken.balanceOf(this.bob));
});
```

Given that relayerStakingWithTokenZock is just the relayerStaking Contract being fed the L0 Token mock implemented by Zokyo distinctively from L0 Token implemented by L0. Similarly is relayerStakingTokenZock.

The scenario:

- Starts by triggering a transfer of layerZeroTokenZock which triggers the transfer received inside relayerStaking which in turn assigns a value for unallocatedRewards;
- Make sure unallocatedRewards is assigned;
- Staking into relayerStakingWithTokenZock which triggers massUpdatePools that looks for the unallocatedRewards and updates;
- The procedure comes to this line which divides by rpi.amount hence doing the revert.

**Recommendation:**
We assume that there should be an update to relayerPoolInfo within the relayerStaking Contract hence updating the amount. It is noticed that accLayerZeroPerShare is getting updates to its value but not amount.

## State change before the transfer of the funds

**MEDIUM** | RESOLVED

In the RelayerStaking.sol file, in the massUpdatePools() function, update the state first before transferring the funds. That is place 156 lines before 154 lines.

**Recommendation:**
Update the state first before transferring the funds.

## Null returned from the function

MEDIUM | RESOLVED

In MPT.sol, the function sliceTransform returns null in tests since a return statement is missing from the body.

**Recommendation:**
Expectedly this function returns the value of newdata hence a return statement for that value is required or add the literal on the function header to serve the same purpose.

## Function not returning the expected output

MEDIUM | RESOLVED

In ECVerify.sol, the function ecverify should return the public address of the private key which signed the message that is sent along with the signature as an argument to this function. Despite that the logic seems correct but Geth prepends the message by \x19Ethereum Signed Message:\n<length of message> as shown in https://github.com/ethereum/go-ethereum/issues/3731.

**Recommendation:**
Prepend the message in the code body before applying ecrecover as follow (assuming size is always 32 bytes):

```
require(v == 27 || v == 28);

bytes memory prefix = "\x19Ethereum Signed Message:\n32";
bytes32 prefixedHash = keccak256(abi.encodePacked(prefix, hash));
signature_address = ecrecover(prefixedHash, v, r, s);

// ecrecover returns zero on error
require(signature_address != address(0x0), "ECVerify revert");
```

## Relayer passes through calls without verification

MEDIUM | NOT VALID

In RelayerStaking.sol, _relayer which refers to the Relayer Contract is passing through the functions without being verified. Inside massUpdatePools(address _relayer) there's an if statement that acts like a way to verify the relayer in a sense. But this is still not enough since _relayer is still referred to frequently in this contract in many functions while it can be entered by a user as an invalid address.

**Recommendation:**
Add required statements to verify relayer. In this project, an independent relayer (not owned by LayerZero) might still need to be registered by an authority in LayerZero like onlyOwner unless there is another solution to verify the _relayer without needing authoritative involvement from LayerZero which shall be better.

**Comment:**
Being a design choice by LayerZero team the issue is no longer relevant.

## Function call need a privileged sender

MEDIUM | NOT VALID

In Network.sol, attackers might find it easy to make the oracle do misinformed actions by calling notifyOracleOfBlock(uint16 _chainId, address _srcAddress) which does not require any certain privilege to be called.

**Recommendation:**
Caller applications might be needed to save into a mapping data structure from which this function requires the value to be called in order to be executed.

**Comment:**
Being a design choice by LayerZero team the issue is no longer relevant.

## Correcting the required statements to be more precise

LOW | RESOLVED

Since the transactions would fail if the required condition is not satisfied it is recommended to add a statement that justifies the reason for the failure of the transaction.

**Recommendation:**

1) In Treasury.sol file, in x`withdraw(), replace

```
require(success, "LayerZero Treasury: withdraw native failed")
```

with

```
require(success, "LayerZero Treasury: native asset withdraw failed")
```

2) In ChainlinkOracleClient.sol file, in withdraw(), replace

```
require(success, "Relayer: failed to withdraw")
```

with

```
require(success, "OracleClient: failed to withdraw")
```

3) In Relayer.sol file, in both getPrices() functions, the second argument userApplication is unused;

4) In RelayerStaking.sol file, in onTransferReceived() function, replace

```
require(msg.sender == address(layerZeroToken), "ERC1363Payable: acceptedToken is not message sender")
```

with

```
require(msg.sender == address(layerZeroToken), "RelayerStaking: acceptedToken is not message sender")
```

5) In RelayerStaking.sol file, line 34, spelling mistake in the comments.

## Lock pragma to a specific version

LOW | RESOLVED

Lock the pragma to a specific version, since not all the EVM compiler versions support all the features, especially the latest one's which are kind of beta versions, So the intended behavior written in code might not be executed as expected.

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs.

**Recommendation:**
Lock the pragma to a specific version.

## Stake does not check if the input _amount is nonzero

LOW | RESOLVED

In RelayerStaking.sol, inside stake(address _relayer, uint256 _pid, uint256 _amount) _amount is not verified if it is non-zero which might lead to unexpected errors.

**Recommendation:**
Requirement is needed for that.

## Several unused arguments

INFORMATIONAL | RESOLVED

In Several Contracts:

1) Treasury.sol: getNativeFee(uint) no need to take an argument, argument passed in Communicator.sol has no effect;

1) RelayerStaking.sol: set(): _withUpdate is unused variable;

3) Relayer.sol: getPrices(): userApplication and parameters not used in the function body. This is in the two implemented getPrices() functions in this contract.

**Recommendation:**
This issue might be of considerable importance if these variables are meant to provide more functionality, if that's the case then variables needed to be used in the function bodies.

**Comment:**
Partner stated that the first point is of no use right now, hence the issue is irrelevant. Regarding the second point, the partner resolved this one. To the third one partner stated that Relayer.sol is just a simple implementation of a relayer contract and becoming a reference to others who are implementing the relayer contract.

## Error Message unclear

INFORMATIOMAL | RESOLVED

In Network.sol, revert message inside require a statement of updateBlockHeader(uint16 _remoteChainId, address _oracle, bytes calldata _blockHash, uint _confirmations, bytes calldata _data) might be confusing.

**Recommendation:**
More descriptive and still short message for updateBlockHeader like `LayerZero: sender is not approved by oracle`.

## Check balance before withdrawing

INFORMATIONAL | NOT VALID

In Treasury.sol, withdrawNative() does not check the contract's balance before the withdrawal.

**Recommendation:**
This issue does not affect the operation as the transaction still fails because there's a required statement checking the success of the operation after the call. Checking the balance though is helpful as it provides information on why the call shall revert.

## One function implementation for what might better be two

INFORMATIONAL | NOT VALID

In Communicator.sol, writing two implementations in solidity for similar function calls is sometimes needed. In this case, being discussed here, the contract operation in this call will be saving gas if implemented as recommended. This snipped if(_zroPaymentAddress == address(0x0)) acts as a flag for send(uint16 _chainId, bytes calldata _destination, bytes calldata _payload, address payable _refundAddress, address _zroPaymentAddress, bytes calldata txParameters) which leads to two different calls in which one is payable and the other should not be payable.

**Recommendation:**
Throw the common logic into one private function and implement two functions for both cases in which one of them only is payable.

**Comment:**
Being a design choice by the LayerZero team the issue is no longer relevant.

| | | |
|---|---|---|
| | **RelayerToken;**<br>**ChainlinkOracleClient;**<br>**Treasury;**<br>**Communicator;**<br>**Network;**<br>**Validator;**<br>**Relayer;**<br>**LayerZeroToken;** | **RelayerStaking;**<br>**ERC1363;**<br>**IERC1363Receiver;**<br>**IERC1363;**<br>**IERC1363Spender;**<br>**ILayerZeroEndpoint;**<br>**ILayerZeroTreasury;**<br>**ILayerZeroOracle.** |
| Re-entrancy | Pass | Pass |
| Access Management Hierarchy | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass |
| Unexpected Ether | Pass | Pass |
| Delegatecall | Pass | Pass |
| Default Public Visibility | Pass | Pass |
| Hidden Malicious Code | Pass | Pass |
| Entropy Illusion (Lack of Randomness) | Pass | Pass |
| External Contract Referencing | Pass | Pass |
| Short Address/ Parameter Attack | Pass | Pass |
| Unchecked CALL Return Values | Pass | Pass |
| Race Conditions / Front Running | Pass | Pass |
| General Denial Of Service (DOS) | Pass | Pass |
| Uninitialized Storage Pointers | Pass | Pass |
| Floating Points and Precision | Pass | Pass |
| Tx.Origin Authentication | Pass | Pass |
| Signatures Replay | Pass | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass | Pass |

| | | |
|---|---|---|
| | **ILayerZeroValidationLibrary;**<br>**ILayerZeroReceiver;**<br>**ILayerZeroReceiver;**<br>**ILayerZeroValidator;**<br>**ILayerZeroRelayer;**<br>**LayerZeroPacket;**<br>**Buffer;** | **EVMValidator;**<br>**ECVerify;**<br>**EthereumDecoder;**<br>**MPT;**<br>**RLPDecode;**<br>**RLPEncode;**<br>**Decoder.** |
| Re-entrancy | Pass | Pass |
| Access Management Hierarchy | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass |
| Unexpected Ether | Pass | Pass |
| Delegatecall | Pass | Pass |
| Default Public Visibility | Pass | Pass |
| Hidden Malicious Code | Pass | Pass |
| Entropy Illusion (Lack of Randomness) | Pass | Pass |
| External Contract Referencing | Pass | Pass |
| Short Address/ Parameter Attack | Pass | Pass |
| Unchecked CALL Return Values | Pass | Pass |
| Race Conditions / Front Running | Pass | Pass |
| General Denial Of Service (DOS) | Pass | Pass |
| Uninitialized Storage Pointers | Pass | Pass |
| Floating Points and Precision | Pass | Pass |
| Tx.Origin Authentication | Pass | Pass |
| Signatures Replay | Pass | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass | Pass |

# CODE COVERAGE AND TEST RESULTS FOR ALL FILES

## Tests written by Layer Zero 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 |
|---|---|---|---|---|---|
| contracts/ | 87.01 | 63.85 | 82.35 | 86.67 | |
| ChainlinkOracleClient.sol | 35.00 | 33.33 | 50.00 | 33.33 | ... 62, 64, 56, 73 |
| Communicator.sol | 94.00 | 65.38 | 90.00 | 94.17 | ... 265, 273, 281 |
| LayerZeroToken.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| Network.sol | 100.00 | 75.00 | 100.00 | 100.00 | |
| Relayer.sol | 85.71 | 100.00 | 71.43 | 85.71 | 28 |
| RelayerStaking.sol | 97.54 | 86.84 | 100.00 | 97.56 | 186, 256, 295 |
| Treasury.sol | 40.00 | 25.00 | 62.50 | 40.00 | ... 40, 44, 45, 46 |
| Validator.sol | 65.63 | 31.82 | 81.82 | 64.71 | ... 4, 95, 96, 106 |
| contracts/ERC1363/ | 28.57 | 20.00 | 22.22 | 28.57 | |
| ERC1363.sol | 28.57 | 20.00 | 22.22 | 28.57 | ... 152, 154, 155 |
| IERC1363.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| IERC1363Receiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| IERC1363Spender.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| contracts/chainlink/ | 0.00 | 0.00 | 28.57 | 0.00 | |
| ERC677Receiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| MockLinkToken.sol | 0.00 | 0.00 | 20.00 | 0.00 | ... 48, 49, 50, 56 |
| MockOracle.sol | 100.00 | 100.00 | 50.00 | 100.00 | |

| FILE | % STMTS | % BRANCH | % FUNCS | % LINES | UNCOVERED LINES |
|---|---|---|---|---|---|
| contracts/interfaces/ | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroEndpoint.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroOracle.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroReceiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroRelayer.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroTreasury.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroValidationLibrary.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroValidator.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| contracts/libraries/ | 31.25 | 33.33 | 34.78 | 31.52 | |
| Buffer.sol | 46.30 | 50.00 | 44.44 | 44.62 | ... 495, 506, 527 |
| EVMValidator.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 46, 47, 51, 52 |
| LayerZeroPacket.sol | 0.00 | 100.00 | 0.00 | 0.00 | ... 52, 53, 54, 55 |
| contracts/proof/lib/ | 0.00 | 0.00 | 0.00 | 0.00 | |
| ECVerify.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 25, 26, 29, 31 |
| EthereumDecoder.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 249, 254, 259 |
| MPT.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 228, 229, 231 |
| RLPDecode.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 333, 337, 338 |
| RLPEncode.sol | 0.00 | 0.00 | 0.00 | 0.00 | ... 223, 225, 264 |
| contracts/proof/utils/ | 0.00 | 100.00 | 0.00 | 100.00 | |
| Decoder.sol | 0.00 | 100.00 | 0.00 | 0.00 | 12, 16, 20 |
| **All files** | **36.11** | **26.88** | **46.07** | **38.89** | |

## Test Results

**ChainlinkOracleClient**

- ✓ setLayerZero as non owner reverts (65ms)
- ✓ setLayerZero as owner
- ✓ setLayerZero as owner, reverts if already has been set (40ms)
- ✓ getPrice
- ✓ setApprovedAddress reverts for non owner
- ✓ setApprovedAddress and confirm for owner
- ✓ isApproved for approved and non approved addresses

**Communicator**

- ✓ is created properly
- ✓ reverts when non owner sets: layer zero fee
- ✓ setLayerZeroFee / get layerZeroFee
- ✓ reverts when non owner sets: relayer staking contract
- ✓ setRelayerStakingContract / get relayerStakingContract
- ✓ reverts when non owner sets: treasury contract
- ✓ setTreasuryContract / get treasuryContract
- ✓ reverts when non owner sets: default config for chainId
- ✓ setLibraryVersion reverts when greater than existing max version
- ✓ setLibraryVersion / getLibraryVersion
- ✓ setRelayer / getRelater
- ✓ setOracle / getOracle
- ✓ setOracle emits AppConfigUpdated
- ✓ setRelayer emits AppConfigUpdated
- ✓ setBlockConfirmations / getBlockConfirmations
- ✓ setBlockConfirmations emits AppConfigUpdated
- ✓ setLibraryVersion emits AppConfigUpdated
- ✓ send(), reverts if relayer fee not paid (native fee)
- ✓ send(), paying with native (55ms)
- ✓ estimateNativeFees(), paying with ZRO
- ✓ estimateNativeFees(), paying with Native
- ✓ send(), paying with ZRO (62ms)
- ✓ getAppConfig defaults
- ✓ getAppConfig non defaults (61ms)
- ✓ receiveAndForward() event DestinationFailed emitted
- ✓ resendAndClearStoredPayload() event StoredPayloadCleared emitted

Outgoing
- ✓ receiveAndForward() success
- ✓ receiveAndForward() event StoredPayloadSaved emitted
- ✓ receiveAndForward() event DestinationFailed emitted
- ✓ resendAndClearStoredPayload() event StoredPayloadCleared emitted (54ms)

**LayerZeroToken**
- ✓ token symbol
- ✓ token name
- ✓ token decimals

**Network**
- ✓ is created for local chain id
- ✓ setValidator reverts for non owner
- ✓ setValidator by owner and confirm it
- ✓ setCommunicator reverts for non owner
- ✓ setCommunicator by owner and confirm it
- ✓ setCommunicator 2  by owner and confirm it
- ✓ setEndpoint defaults to no bytes (its not set to anything) on deployment
- ✓ setEndpoint reverts for non owner
- ✓ setEndpoint by owner and dont allow setting more than once (39ms)
- ✓ updateBlockHeader - oracle address isnt an oracle
- ✓ updateBlockHeader - called by non approved oracle
- ✓ updateBlockHeader and getBlockHeaderData (39ms)
- ✓ getBlockHeaderDataHash (38ms)
- ✓ getBlockHeaderConfirmations (51ms)
- ✓ getApplicationConfiguration succeeds
- ✓ getApplicationConfiguration relayer, oracle, confirms, library
- ✓ notifyOracleOfBlock [example] reverts if communicator is not a contract
- ✓ notifyOracleOfBlock can be called by any owner
- ✓ notifyOracleOfBlock can be called by any eoa (56ms)

**Relayer**
- ✓ created and is approved
- ✓ reverts when non owner sets approval
- ✓ owner sets approval
- ✓ reverts when non owner sets tx fees (40ms)
- ✓ owner sets tx fees - todo

**RelayerStaking**

- ✓ created with layerzero token
- ✓ onTransferReceived requires msg.sender is layerZeroToken
- ✓ onTransferReceived emits TokensReceived() - todo
- ✓ add() reverts for non owner (46ms)
- ✓ add() emits PoolState updated event
- ✓ set() emits PoolState updated event
- ✓ add() adds a pool with the given alloc points and address
- ✓ add some pools and change them and ensure totalAllocPoints is accurate (123ms)
- ✓ set() reverts for non owner
- ✓ setCooldownTime() reverts for non owner
- ✓ initial cooldown time > 0
- ✓ setCooldownTime() and read it
- ✓ setCooldownTime() emits CooldownUpdated event
- ✓ pauseRelayerWithdrawal() reverts for non owner
- ✓ pauseRelayerWithdrawal() emits event (46ms)
- ✓ cannot stake (reverts) if relayer is paused (79ms)
- ✓ slashRelayer() reverts for non owner
- ✓ slashRelayer() cannot slash amount of 0 (44ms)
- ✓ pendingLayerZero(address _relayer, uint _pid, address _user) (62ms)
- ✓ massUpdatePools() any address can call (39ms)
- ✓ harvest() anyone can call harvest

**RelayerStaking [w/ starting 3 pools]**

- ✓ stake pid 0 emits RelayerStake
- ✓ unstake pid 0 emits RelayerUnstake (67ms)
- ✓ unstake pid 0 reverts if too much allocation points are unstaked
- ✓ claim reverts when paused
- ✓ claim with no tokens to claim reverts
- ✓ claim during cooldown reverts (88ms)
- ✓ claim emits event (83ms)
- ✓ emergency unstake pid 0 emits event (53ms)
- ✓ alice and bob stake pid 0 for equal amounts (432ms)
- ✓ alice and bob stake, unstake, and claim (211ms)
- ✓ slashRelayer emits event (214ms)
- ✓ dao relayer slashing (142ms)

**Treasury**

- ✓ setNativeFee() reverts for non owner

✓ approveTokenSpender() reverts for non owner (65ms)
✓ withdrawToken() reverts for non owner
✓ withdrawNative() reverts for non owner
✓ setNativeFee() and getNativeFee()

**Validator**
✓ created for network
✓ setCommunicator - reverts as non owner (56ms)
✓ setCommunicator - and confirm it, as owner
✓ setChainAddressSize - reverts as non owner (38ms)
✓ setChainAddressSize - and confirm it, as owner
✓ setDefaultLibraryForChain - reverts as non owner (38ms)
✓ getDefaultLibraryForChain - is 0x0 before ever being set
✓ setDefaultLibraryForChain - and confirm it, as owner
✓ addValidationLibraryForChain - reverts as non owner (40ms)
✓ addValidationLibraryForChain - and confirm it, owner (50ms)
✓ maxValidationLibrary defaults to 0
✓ maxValidationLibrary is incremented after library added
✓ notifyRelayer reverts if called by non network
✓ notifyRelayer emits HeaderReceived - todo
✓ send reverts when sent by non communicator

**Validator (w/ network mock)**
✓ send() emits Packet event (88ms)

**Validator - validateTransactionProof()**
✓ validateTransactionProof - todo

119 passing (32s)

## Tests written by Zokyo Security team

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

Tests were based on the functionality of the code, as well as a review of the Layer Zero 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 |
|---|---|---|---|---|---|
| contracts/ | 98.70 | 84.62 | 96.47 | 98.73 | |
| ChainlinkOracleClient.sol | 95.00 | 83.33 | 91.67 | 95.24 | 73 |
| Communicator.sol | 100.00 | 75.00 | 100.00 | 100.00 | |
| LayerZeroToken.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| Network.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| Relayer.sol | 100.00 | 100.00 | 85.71 | 100.00 | |
| RelayerStaking.sol | 97.54 | 86.84 | 100.00 | 97.56 | 186, 256, 295 |
| Treasury.sol | 100.00 | 100.00 | 87.50 | 100.00 | |
| Validator.sol | 100.00 | 95.45 | 100.00 | 100.00 | |
| contracts/ERC1363/ | 100.00 | 90.00 | 100.00 | 100.00 | |
| ERC1363.sol | 100.00 | 90.00 | 100.00 | 100.00 | |
| IERC1363.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| IERC1363Receiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| IERC1363Spender.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| contracts/chainlink/ | 100.00 | 50.00 | 85.71 | 100.00 | |

| FILE | % STMTS | % BRANCH | % FUNCS | % LINES | UNCOVERED LINES |
|---|---|---|---|---|---|
| ERC677Receiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| MockLinkToken.sol | 100.00 | 50.00 | 100.00 | 100.00 | |
| MockOracle.sol | 100.00 | 100.00 | 50.00 | 100.00 | |
| contracts/interfaces/ | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroEndpoint.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroOracle.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroReceiver.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroRelayer.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroTreasury.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroValidationLibrary.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| ILayerZeroValidator.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| contracts/libraries/ | 96.25 | 75.00 | 95.65 | 96.74 | |
| Buffer.sol | 94.44 | 81.25 | 94.44 | 95.38 | 97, 210, 349 |
| EVMValidator.sol | 100.00 | 62.50 | 100.00 | 100.00 | |
| LayerZeroPacket.sol | 100.00 | 100.00 | 100.00 | 100.00 | |
| contracts/proof/lib/ | 73.89 | 56.11 | 78.00 | 80.00 | |
| ECVerify.sol | 100.00 | 75.00 | 100.00 | 100.00 | 22 |
| EthereumDecoder.sol | 65.97 | 58.57 | 63.64 | 72.29 | ... 249, 254, 259 |
| MPT.sol | 64.21 | 40.00 | 100.00 | 70.24 | ... 169, 170, 174 |
| RLPDecode.sol | 81.83 | 59.09 | 77.78 | 83.81 | ... 216, 217, 221 |
| RLPEncode.sol | 91.53 | 100.00 | 75.00 | 92.65 | 48, 82, 91, 92, 93 |
| contracts/proof/utils/ | 66.67 | 100.00 | 75.00 | 66.67 | |
| Decoder.sol | 66.67 | 100.00 | 75.00 | 66.67 | 20 |
| **All files** | **86.23** | **69.08** | **90.45** | **90.15** | |

## Test Results

**Buffer Wrapper**

- ✓ Deployed properly
- ✓ fromBytes() - (60ms)
- ✓ truncate() - Makes buffer a Zero Buffer (48ms)
- ✓ append() - append whole length of data (65ms)
- ✓ append() - append part of length of data (51ms)
- ✓ append() - reverts because append with invalid length of data (71ms)
- ✓ writeUint8() - writes uint8 inplace according to offset (58ms)
- ✓ appendUint8() - appends uint8 after end of buffer (39ms)
- ✓ writeBytes20() - writes 20 bytes (typically address) into buffer inplace according of offset (63ms)
- ✓ appendBytes32() - appends 32 bytes into buffer (42ms)
- ✓ appendInt() - appends 32 bytes uint into buffer

**ChainlinkOracleClient**

- ✓ setLayerZero() - reverts when called by notOwner (93ms)
- ✓ setLayerZero() - Called by owner (123ms)
- ✓ setLayerZero() - Called by owner but reverts since it only sets once (176ms)
- ✓ getPrice() - (261ms)
- ✓ setApprovedAddress() - reverts when called by notOwner (139ms)
- ✓ setApprovedAddress() - Called by owner, should set true approvedAddresses for that address (232ms)
- ✓ isApproved() - for approved and non approved addresses (289ms)
- ✓ approveToken() - Called by Owner successfully (1133ms)
- ✓ approveToken() - reverts when called by notOwner (333ms)
- ✓ setJob() - Called by Owner successfully and new value for Job added (246ms)
- ✓ setJob() - Reverts when called by notOwner (90ms)
- ✓ notifyOracleOfBlock() (1154ms)
- ✓ notifyOracleOfBlock() - Reverts when called by an unapproved caller (198ms)

**Communicator**

- ✓ Deployed properly (84ms)
- ✓ setLayerZeroFee() - reverts when called by notOwner (121ms)
- ✓ setLayerZeroFee() - Called successfully by owner, value of fee checked to be changed (232ms)
- ✓ setRelayerStakingContract() - reverts when called by non owner - sets: relayer staking contract (90ms)
- ✓ setRelayerStakingContract() - called successfully by owner (428ms)

- ✓ setTreasuryContract() - reverts when called by notOwner (72ms)
- ✓ setTreasuryContract() - called successfully by owner (284ms)
- ✓ setDefaultConfigForChainId() - reverts when called by notOwner (168ms)
- ✓ setDefaultConfigForChainId() - called successfully by Owner (461ms)
- ✓ setLibraryVersion() - reverts when added version not compatible with maxVersion (145ms)
- ✓ setLibraryVersion() - Called successfully (534ms)
- ✓ setBlockConfirmations() - Called successfully (259ms)
- ✓ setBlockConfirmations() - should emit event AppConfigUpdated (261ms)
- ✓ setRelayer() - along with getRelayer() (533ms)
- ✓ setOracle() - along with getOracle() (511ms)
- ✓ setLibraryVersion emits AppConfigUpdated (303ms)
- ✓ setDefaultTxParametersforChainId() - Called successfully (229ms)
- ✓ send(), paying with native successfully: contracts receive funds / user refunded extra amount (5530ms)
- ✓ send(), paying with native unsuccessfully: not enough value payment (985ms)
- ✓ send(), paying with native unsuccessfully: very low value payment (1466ms)
- ✓ estimateNativeFees() - paying with ZRO (65ms)
- ✓ estimateNativeFees() - paying with Native (163ms)
- ✓ send() - paying with ZRO (4861ms)
- ✓ getAppConfig() - defaults (75ms)
- ✓ getAppConfig() - non defaults (974ms)

Outgoing

- ✓ receiveAndForward() - Called successfully (354ms)
- ✓ receiveAndForward() - event StoredPayloadSaved emitted (274ms)
- ✓ receiveAndForward() - event DestinationFailed emitted (892ms)
- ✓ resendAndClearStoredPayload() - event StoredPayloadCleared emitted (815ms)

**Default configs for relayer, oracle, libraryVersion, blockConfirmations**

- ✓ getLibraryVersion() - return default config (84ms)
- ✓ getBlockConfirmations() - return default config (81ms)
- ✓ getOracle() - return default config (150ms)
- ✓ getRelayer() - return default config (123ms)

**ECVerify Wrapper**

- ✓ Deployed properly
- ✓ ecverify() - verify that signed message is called by the verified signer (157ms)
- ✓ ecverify() - same test with v=25 (101ms)

**ERC1363**

- ✓ supportsInterface() - (108ms)
- ✓ supportsInterface() - Unsupported Interface (should return false) (95ms)
- ✓ transferAndCall() - Check ERC1363Receiver callback is successfully called (284ms)
- ✓ transferFromAndCall() - Check ERC1363Receiver callback is successfully called (720ms)
- ✓ transferAndCall() - Check ERC1363Receiver callback is successfully called with Data (646ms)
- ✓ transferFromAndCall() - Check ERC1363Receiver callback is successfully called with Data (796ms)
- ✓ approveAndCall() - Check ERC1363Receiver callback is successfully called (243ms)
- ✓ approveAndCall() - Check ERC1363Receiver callback is successfully called with Data (293ms)
- ✓ approveAndCall() - Reverts because receiver is not a contract (97ms)
- ✓ transferAndCall() - Reverts because receiver is not a contract (191ms)

**EVMValidator**

- ✓ network() / validator() - network and validator addresses are correct (72ms)
- ✓ verifyReceipt() - testing that tx receipt is properly verified (1299ms)

Invalid packets - intended to pump up coverage

- ✓ validateProof() - emitting event1 (754ms)
- ✓ validateProof() - emitting event2 (643ms)
- ✓ validateProof() - emitting event3 (553ms)

**LayerZeroToken**

- ✓ token symbol (63ms)
- ✓ token name (50ms)
- ✓ token decimals (50ms)

**MPT Wrapper for expandKeyEven & expandKeyOdd**

- ✓ Deployed properly
- ✓ expandKeyEven() - adding zeros to the left of each nibble (98ms)
- ✓ expandKeyOdd() - adding zeros to the left of each nibble (for odd number of nibbles) (52ms)
- ✓ sliceTransform() - 0 (45ms)
- ✓ sliceTransform() - 1 (41ms)
- ✓ sliceTransform() - 2 (58ms)
- ✓ sliceTransform() - 3 (38ms)
- ✓ sliceTransform() - 4 (52ms)
- ✓ sliceTransform() - 5 (51ms)

**Network**

- ✓ Deployed successfully with correct local chain id (67ms)
- ✓ setValidator() - reverts when called by notOwner (86ms)

- ✓ setValidator() - Reverts for setting to an invalid value (121ms)
- ✓ setValidator() - Called by owner and confirm it (291ms)
- ✓ setCommunicator() - reverts when called by notOwner (118ms)
- ✓ setCommunicator() - Called by owner and confirm it (241ms)
- ✓ setCommunicator() -2 - Called by owner and confirm it (210ms)
- ✓ setCommunicator() - Reverts for setting to an invalid value (67ms)
- ✓ setEndpoint() - defaults to no bytes (its not set to anything) on deployment (166ms)
- ✓ setEndpoint() - reverts when called by notOwner (87ms)
- ✓ setEndpoint() - Called by owner successfully - Shall not allow setting more than once (439ms)
- ✓ updateBlockHeader() - Reverts as oracle address isnt an oracle (95ms)
- ✓ updateBlockHeader() - Reverts since it is called by non approved oracle (110ms)
- ✓ updateBlockHeader() / getBlockHeaderData() (674ms)
- ✓ getBlockHeaderDataHash() - (662ms)
- ✓ getBlockHeaderConfirmations() - (665ms)
- ✓ getApplicationConfiguration() - Called successfully (451ms)
- ✓ getApplicationConfiguration() - relayer, oracle, confirms, library (220ms)
- ✓ notifyOracleOfBlock() - reverts when communicator is not a contract (273ms)
- ✓ notifyOracleOfBlock can be called by any owner (446ms)
- ✓ notifyOracleOfBlock can be called by any eoa (593ms)

**Contract: EthereumClient**

- ✓ deploy (932ms)
- ✓ receipt logs RLP encoding/decoding (303ms)
- ✓ client - adding blocks (11566ms)

receipt trie

- ✓ receipt trie 0 (472ms)
- ✓ receipt trie 1 (590ms)
- ✓ receipt trie 2 (584ms)
- ✓ receipt trie 3 (561ms)
- ✓ receipt trie 4 (644ms)

**Relayer**

- ✓ created and is approved
- ✓ reverts when non owner sets approval (64ms)
- ✓ owner sets approval (137ms)
- ✓ reverts when non owner sets tx fees (57ms)
- ✓ setTransactionFees() - owner sets tx fees (174ms)
- ✓ setTransactionFees() - owner sets tx fees (227ms)
- ✓ getPrices() - (142ms)

**RelayerStaking**

- ✓ created with layerzero token (58ms)
- ✓ onTransferReceived requires msg.sender is layerZeroToken (76ms)
- ✓ onTransferReceived emits TokensReceived() - todo
- ✓ add() reverts for non owner (100ms)
- ✓ add() emits PoolState updated event (202ms)
- ✓ set() emits PoolState updated event (245ms)
- ✓ add() adds a pool with the given alloc points and address (161ms)
- ✓ add some pools and change them and ensure totalAllocPoints is accurate (816ms)
- ✓ set() reverts for non owner (84ms)
- ✓ setCooldownTime() reverts for non owner (64ms)
- ✓ initial cooldown time > 0
- ✓ setCooldownTime() and read it (163ms)
- ✓ setCooldownTime() emits CooldownUpdated event (148ms)
- ✓ pauseRelayerWithdrawal() reverts for non owner (101ms)
- ✓ pauseRelayerWithdrawal() emits event (178ms)
- ✓ cannot stake (reverts) if relayer is paused (271ms)
- ✓ slashRelayer() reverts for non owner (92ms)
- ✓ slashRelayer() cannot slash amount of 0 (349ms)
- ✓ pendingLayerZero(address _relayer, uint _pid, address _user) (275ms)
- ✓ massUpdatePools() any address can call (623ms)
- ✓ harvest() anyone can call harvest (668ms)

**RelayerStaking [w/ starting 3 pools]**

- ✓ stake pid 0 emits RelayerStake (383ms)
- ✓ unstack pid 0 emits RelayerUnstake (587ms)
- ✓ unstack pid 0 reverts if too many allocation points are unstacked (482ms)
- ✓ claim reverts when paused (342ms)
- ✓ claim with no tokens to claim reverts (122ms)
- ✓ claim during cooldown reverts (1057ms)
- ✓ claim emits an event (1069ms)
- ✓ emergency unstack pid 0 emits an event (741ms)
- ✓ alice and bob stake pid 0 for equal amounts (2828ms)
- ✓ alice and bob stake, unstack, and claim (2393ms)
- ✓ slashRelayer emits an event (1193ms)
- ✓ dao relayer slashing (1249ms)

**Treasury**

- ✓ setNativeFee - and confirm it, as owner (374ms)

✓ setNativeFee - reverts as non owner (393ms)
✓ payWithNative - reverts as value is not equal native fee (504ms)
✓ payWithNative - passes as value is the equal native fee (577ms)
✓ approveTokenSpender - approve amount for a spender (698ms)
✓ approveTokenSpender - revert as the method is called by nonowner
✓ withdrawToken - withdraw token from the treasury (509ms)
✓ withdrawToken - revert as the method is called by nonowner

**Treasury - Dealing with Native**

✓ withdrawNative - withdraw native from treasury to Alice (1067ms)
✓ receive() - contract receives native coins successfully (299ms)
✓ withdrawNative - revert for fail native coin transfer (430ms)
✓ withdrawNative - revert as method is called by non owner (266ms)

**Validator**

✓ created for network (55ms)
✓ setCommunicator - reverts as non owner (95ms)
✓ setCommunicator - reverts for invalid address (75ms)
✓ setCommunicator - and confirm it, as owner (219ms)
✓ setChainAddressSize - reverts as non owner (180ms)
✓ setChainAddressSize - and confirm it, as owner (190ms)
✓ setDefaultLibraryForChain - reverts as non owner (153ms)
✓ setDefaultLibraryForChain - reverts for invalid address (54ms)
✓ getDefaultLibraryForChain - is 0x0 before ever being set (47ms)
✓ setDefaultLibraryForChain - and confirm it, as owner (242ms)
✓ addValidationLibraryForChain - reverts as non owner (83ms)
✓ addValidationLibraryForChain - reverts for invalid address (99ms)
✓ addValidationLibraryForChain - and confirm it, owner (429ms)
✓ maxValidationLibrary defaults to 0 (57ms)
✓ maxValidationLibrary is incremented after library added (231ms)
✓ notifyRelayer reverts if called by non network (97ms)
✓ notifyRelayer emits HeaderReceived (483ms)
✓ send reverts when sent by non communicator (75ms)

**Validator (w/ network mock)**

✓ send() emits Packet event (622ms)

**Validator - validateTransactionProof()**

✓ validateTransactionProof (948ms)

```
✓ validateTransactionProof reverts for false nonce (779ms)
✓ validateTransactionProof reverts for not enough blockConfirmations (1218ms)
✓ validateTransactionProof reverts unset validation library (803ms)
✓ validateTransactionProof reverts since _packet.dstAddress != _dstAddress (1368ms)
✓ validateTransactionProof reverts for false Relayer::isApproved (769ms)

193 passing (9m)
```

We are grateful to have been given the opportunity to work with the Layer Zero 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 Layer Zero team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.