PUBLIC

Code Assessment

of the Sky OApp OFT Smart Contracts

October 17, 2025

Produced for

Sky

by

CHAINSECURITY

Contents

1 Executive Summary

Dear all,

Thank you for trusting us to help Sky with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Sky OApp OFT according to Scope to support you in forming an opinion on their security risks.

Sky implements an OFT (Omnichain Fungible Token) adapter for the tokens of the Sky Ecosystem (SKY, USDS, SUSDS and SDAO tokens) using the LayerZero V2 stack to bridge tokens across chains in a standardized manner. Currently, implementations for EVM chains as well as Solana are provided.

The most critical subjects covered in our audit are functional correctness, correct integration with LayerZero, Denial-of-Service resilience. The general subjects covered are gas efficiency and trustworthiness.

Notably this token bridge has different security assumptions than the existing native bridges since it relies on the DVN network as oracle and features shared escrowing of funds for different chains. Some general considerations regarding this are provided, in particular:

- Denial of Service.
- Migration Considerations.
- Configuration Considerations.
- Configuration Ordering Considerations.
- LayerZero V2 Considerations.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	3
• Code Corrected	2
• Code Partially Corrected	1

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

In Version 1 the scope of the assessment was limited to the smart contracts of the LayerZero EVM Token Bridge:

```
contracts/
    MintAndBurnOFTAdapter.sol
    OFTAdapter.sol
    oft-dsrl/
        DoubleSidedRateLimiter.sol
        MABAOFTDSRLFee.sol
        OFTAdapterDoubleSidedRLFee.sol
```

In Version 2 the following file

```
contracts/oft-dsrl/OFTAdapterDoubleSidedRLFee.sol
```

due to refactoring, was replaced by the following files:

```
contracts/oft-dsrl/
    OFTAdapterDSRLFee.sol
    OFTAdapterDSRLFeeBase.sol
```

Version 4 adds the Solana OFT implementation to the scope:

```
programs/oft/
    build.rs
    src/
        compose_msg_codec.rs
        errors.rs
        events.rs
        instructions/
            init_oft.rs
            lz_receive.rs
            lz_receive_types.rs
            mod.rs
            quote_oft.rs
            quote_send.rs
            send.rs
            set_oft_config.rs
            set_pause.rs
            set_peer_config.rs
            withdraw_fee.rs
        lib.rs
        msg_codec.rs
        state/
```

```
            mod.rs
            oft.rs
            peer_config.rs
```

In Version 5 the EVM contracts have been refactored to the following files:

```
contracts/
    SkyOFTAdapterMintBurn.sol
    SkyOFTCore.sol
    SkyOFTAdapter.sol
    SkyRateLimiter.sol
    interfaces/
        IMintBurnVoidReturn.sol
        ISkyOFT.sol
        ISkyOFTAdapter.sol
        ISkyRateLimiter.sol
```

Note only the EVM Solidity contracts were reviewed in Version 5.

The assessment was performed on the source code files inside the Sky OApp OFT repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	05 May 2025	620bd8385c51b0769e08500559b3bd1488dd173a	Initial Version
2	21 May 2025	d47aeea481f513c22182d5cf8af312425041cf29	After Intermediate Report
3	03 June 2025	272bac2ac752ce793922d17e341f5eb3501b21c6	Pausable quoteSend
4	14 July 2025	c48c3c0ea62df8dbf348339cc0227bbf92dc3100	Solana OFT
5	19 Sep 2025	c4e4f8caba41dd987c0b8fe564790762679792df	OFT Updates
6	16 Oct 2025	5ad5cb6bbe624e2b1cb99acfe3e4140fa1c233b9	Final Version

For the solidity smart contracts, the compiler version `0.8.22` was chosen and `evm_version` was set to `shanghai`.

For the solana smart contracts, the rust version `v1.75.0`, the anchor version `v0.29` and the solana CLI version `v1.17.31` were chosen.

2.1.1 Excluded from scope

The migration process from the existing bridges to this new bridge is out of scope of this review.

LayerZero V2 itself is out of scope and assumed to function correctly as per its documentation.

The configurations of OApp and its settings (e.g. choice of libraries) on LayerZero V2 are out of scope. The DVN, executor, send and receive libraries are assumed to be properly configured.

Tests and imports, including the OFT-EVM contracts are out of scope and treated as external dependencies for this review. While the core OFT-EVM contracts are out of scope, we reviewed their implementation to understand their behavior and to confirm that the in scope contracts integrate with and use them correctly. The OFT/OApp smart contracts were previously reviewed by ChainSecurity (LayerZero OFT/OApp Audit). These contracts are particularly relevant, as the contracts in scope inherit from them.

Further, the Solana programs make use of the Anchor framework and integrate with SPL / SPL2022 programs, which are assumed to function properly and work as documented.

2.2 System Overview

This system overview describes the latest version of the contracts as defined in the Assessment Overview.

At the end of this report section, we have added a changelog subsection for each of the changes according to the versions.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Sky implements an OFT (Omnichain Fungible Token) adapter for the tokens of the Sky Ecosystem (SKY, USDS, SUSDS and SDAO tokens) using the LayerZero V2 stack to bridge tokens across chains in a standardized manner. Currently, implementations for EVM chains as well as Solana are provided.

2.2.1 OFT Overview

In general, two implementations with the following main differences are provided:

- *Escrow Adapter*: This implementation works by locking the tokens to be bridged into an escrow and releasing them when bridged back.
- *Mint/Burn Adapter*: This implementation works by burning tokens when bridged out and minting tokens when bridged in.

Integration and Interaction Flow of the OFT is outlined below:

1. *Source Chain User*. Instructs the source chain OFT to send tokens to a destination chain.
2. *Source Chain OFT*: Performs fee computation, slippage check, and rate limit checks. Then it takes the tokens from the user. Finally, it sends a message through the LayerZero V2 endpoint with the configured send library to the destination OFT. Note that the execution fee (denominated in native token or LZ token) is paid in this step.
3. *LayerZero V2*: Handles the message. Eventually, the message is verified on the LayerZero V2 endpoint, marking its validity. Note that this step includes on- and off-chain components.
4. *Destination Chain OFT*: `lzReceive()` is triggered and tokens are credited to the receiver. Note that this adheres to the receive library. If a compose message is attached, it will be registered in the endpoint.
5. *Compose Receiver*: If a compose is registered, the compose data can be consumed in a separate call by the designated receiver.

However, note that depending on the configuration and the chain the exact details may vary.

Rate Limit A double-sided rate limiting mechanism is implemented. The rate limit will decrease if consumed, and will gradually refill over time with a configured rate until it reaches its limit. It supports both net and gross accounting modes:

- Net accounting allows bidirectional traffic to offset each other, meaning tokens sent and received can reduce the effective in-flight amount of each other.
- Gross accounting counts the total traffic in one direction, without any offset, applying stricter limits.

Messaging Note that the messages passed between the OFT follow the generic format below:

sendTo	bytes32	Receiver of the bridged tokens
amountShared	u64	Bridged amount in shared decimals

composeFrom	bytes32	Initiator of the compose
composeMsg	Variable	Actual compose payload

For sending tokens to another chain, the following holds:

- The OFT is not paused and the slippage is respected.
- The peer is configured, namely there is a corresponding OFT deployed on the destination chain.
- The rate limit is respected and updated.

For receiving tokens from another chain, the following holds:

- The OFT is not paused.
- The sender matches the configured peer on the source chain.
- The rate limit is respected and updated.

Quote Before triggering a cross-chain transfer, one can use the following view functions to estimate the amounts being transferred and fees:

- *Quote OFT*: computes the `oftLimit`, the amount actually being sent and received based on an input amount, and the fee detail.
- *Quote Send*: computes the LayerZero bridging fee for the send operation given the send parameters including the potential options and compose messages.

Privileged Roles can configure the OFT as well as its configuration on the LayerZero V2 endpoint. Below, the generally available functionalities are listed.

The following general owner-privileged actions exist for all OApps:

- *Set Rate Limit*: Set the inbound or outbound rate limit configurations for an `Eid`.
- *Set Pauser*: Sets the privileged pauser or unpauser (if exists). Further the owner can pause / unpause the OFT.
- *Set Enforce Options*: Set enforced options that is attached with a `send` or `send_and_call`.
- *Set (Default) Fee Bps*: Set the OFT fees per `Eid` or default fees.
- *Set Delegate*: Set the delegate of the OFT on EndpointV2.
- *Set Peer*: Set the peer address (the corresponding OFT) of an `Eid`.
- *Withdraw Fees*: Withdraw the OFT fees to a designated receiver.

Below are the relevant actions on the LayerZero V2 endpoint for the delegate (and technically the OFT itself):

- *Burn*: Marks a nonce as non-executable and non-verifiable. The nonce can never be re-verified or executed.
- *Nilify*: Marks a packet as verified, but disallows execution until it is re-verified.
- *Clear*: Clears a message that is efficiently burnt.
- *Skip*: Skips the next nonce to prevent message verification.
- *Set Send Library*: Sets the send library.
- *Set Receive Library*: Sets the receive library.
- *Set Receive Library Timeout*: Sets the timeout for a receive library change during which the previous library could be used.
- *Set Config*: Sets an OApp's configuration on a registered message library.

In general, the OFT should be configured correctly, for more details please consult Configuration Considerations.

2.2.2 EVM OFT

The EVM OFT follows the general design. An OFT adapter contract needs to be deployed for each token to be bridged.

Sending: To initiate a cross-chain transfer on the source chain to a supported peer, `send()` is called on the OFT; after the fees are charged, a bridge message is constructed and eventually a call is made to `EndpointV2.send()`.

Receiving: once a cross-chain transfer is verified, it is ready to be executed by anyone with `lzReceive()` on the destination chain's EndPointV2 contract. The token is credited to the `toAddress`. The compose message will be stored in the endpoint if it is used and can be executed in a separate transaction with `EndpointV2.lzCompose()`.

Further the following features exist on EVM OFT:

- A getter `approvalRequired()` is provided to indicate whether user needs to approve the tokens to be transferred before calling `send()`.
- The owner further has privilege over `setMsgInspector()` and `setPreCrime()` to set an inspector or preCrime address.
- `migrateLockedTokens()` is only implemented in SkyOFTAdapter, the owner can transfer all the `innerToken` except fees to an address to facilitate a future upgrade.

Governance: The EVM OFT is expected to be governed by the respective governance contracts on Mainnet or governance relays on other EVM chains.

Note for *Mint/Burn Adapter*, the EVM OFT contract itself bears the minting and burning privileges.

2.2.3 Solana OFT

The Solana OFT follows the general design while multiple OFT instances (represented by distinctive OFTStore PDAs) can be created using the same program.

Namely, the instruction `oft::init_oft` is provided to create an OFT with a given `token_escrow` while setting other configurations. Note that this generates an `OFTStore` PDA that will be registered on the LayerZero V2 endpoint with `endpoint::register_oapp` to set its delegate and should be used on the source-chain as the peer address of the LayerZero V2 cross-chain message.

On Solana, some instructions should be to be called prior to the message reception, especially the delegate-only ones:

- `init_receive_library` / `init_config`: (delegate-only) init the receive library and relevant configurations.
- `set_receive_library` / `set_config`: (delegate or OApp) set the receive library and relevant configurations.
- `init_nonce`: (delegate-only) initialize the Nonce and PendingInboundNonce PDAs.
- `init_verify`: (permissionless) initialize the PayloadHash PDA for each message to be received.

Additionally, instruction `oft::lz_receive_types` is provided to help the executor to prepare the necessary account list for `oft::lz_receive`.

Sending: To initiate a cross-chain transfer on the source chain to a supported peer, `send()` is called on the OFT program with the specific OFTStore PDA attached; after the fees are charged, a bridge message is constructed and eventually a CPI is made to `EndpointV2::send()`.

Receiving: the verified message can be consumed by anyone (typically the paid executor) with `oft::lz_receive` which consumes the message on the endpoint with `endpoint::clear`. Then, the

program proceeds to perform a CPI to the spl-token program to credit the tokens by either transfer or mint. In case a compose message is attached, another CPI `EndpointV2::send_compose` will register the compose, which can be later cleared by the designated receiver with a CPI `EndpointV2::clear_compose`.

Note for *Mint/Burn Adapter*, the Solana OFTStore PDA is expected to be the spl-token's `mint_authority` or it is one of the signers of the spl-token multisig account, and the quorum is 1-of-n. Otherwise, it cannot mint or burn tokens.

Governance: The Solana OFT is expected to be governed by the respective `cpi_authority` of the Governance OApp.

For both EVM OFTs and Solana OFTStore PDAs, configurations need to be set for each OFT and there is no shared configurations.

2.2.4 Changelog

In Version 2, function `_debit()` will decrease the rate limit by `amountReceivedLD`, excluding the fee and dust.

In Version 3, function `_debitView()` is now guarded by the `whenNotPaused` modifier, hence view functions `quoteOFT()` and `quoteSend()` will revert if contract is paused.

In Version 4, Solana OFT has been added.

In Version 5, the EVM OFT has been refactored with the following changes:

- `quoteOFT()` has been overridden to return the `oftLimit` based on the rate limits and the actual fee charged in fee detail.
- The feeBalance is no longer accounted in the *Mint/Burn Adapter*, and all its underlying balance is regarded as fees.
- `migrateLockedTokens()` in *Escrow Adapter* migrates all underlying tokens except fees.

In Version 5, the default rate limit for a peer has been changed to 0 in Solana OFT.

2.3 Trust Model

Owner (EVM OFT) / Admin (Solana OFT). Set at deployment. Fully trusted. Can set rate limits, fees, and other OFT configurations (e.g. peers and enforced options). Is expected to perform these configuration changes correctly. In the worst case, the OFT could be maliciously configured to allow receiving malicious cross-chain messages. Additionally, it could DoS or censor the message relay or prevent the cross-chain token transfers. **Additionally**, in the OFTAdapter this role can execute `migrateLockedTokens()`, effectively transferring all locked tokens except fees (for the underlying token only) to any address.

Delegate of the OApp in LayerZero. Fully trusted. Initially set to the owner of the OApp who can reassign this role. Can configure the library and manipulate the message handling on the EndPointV2 contract. In the worst case, the OApp could be maliciously configured to allow receiving malicious cross-chain messages. Additionally, it could DoS or censor the message relay or prevent the execution of messages.

Pauser. Semi-trusted. In the worst case can temporarily DoS the OFT.

Unpauser. Semi-trusted. In the worst case can prevent pauser from pausing the OFT in emergency.

Upgrade Authority. Fully trusted if exists. On Solana, programs can have an upgrade authority (typically the account that originally deployed the program), which bears the privileges to upgrade the program code. If the OFT program is intended to be immutable, its upgrade authority should be removed.

LayerZero. LayerZero is out of scope for this review and is trusted to behave correctly and deliver messages to the correct destination. In general, the libraries are trusted, otherwise for instance, the

Solana OFT's send library may impersonate signers passed for privilege escalations. For more details please consult LayerZero V2 Considerations.

Tokens. Tokens are expected to be the Sky-ecosystem tokens: Sky, USDS, SUSDS, SDAO:

- Tokens without CpiGuard, otherwise the transfer CPI will fail.
- Tokens without malicious hooks.
- For Solana Native adapter, the OFTStore is expected to be the `mint_authority` or a signer of the 1-of-n multisig account for minting.

The configuration required for managing the LayerZero applications on multiple chains is considered out of scope for this review and should be performed by the delegate or its owner through the utility functions, please consult Configuration Considerations for more details.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

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

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Open Findings

In this section, we describe any open findings. Findings that have been resolved have been moved to the Resolved Findings section. The findings are split into these different categories:

- Design: Architectural shortcomings and design inefficiencies
- Correctness: Mismatches between specification and implementation

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	1

5.1 Inconsistent Rate Limiting Logic Between EVM and Solana OFT

Design Low Version 4 Code Partially Corrected

CS-SOA-001

The following discrepancies exist between the Solana and EVM OFT implementations:

- The EVM OFT's rate limit of a peer is 0 by default, limiting any inbound or outbound transfers. However the Solana OFT's rate limit, implemented as an option, is infinite by default.
- The EVM OFT's rate limit accounting type is either `Net` or `Gross` for both `Inbound` and `Outbound` rate limiter of a given peer. However on Solana OFT's, the `Inbound` and `Outbound` rate limiters for a given peer can have different accounting types.
- When the `capacity` of a rate limiter is set, the Solana implementation will reset the rate limiter whereas the EVM one will not.

Code partially corrected:

The Solana OFT implementation has been changed to prevent transfers if the corresponding rate limit is not configured yet.

Other inconsistencies still exit.

6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the Open Findings section.

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	2

- Incorrect Amount Passed in quote_send Code Corrected
- Unable to Receive Transfers to address(0) or Token Contract Address Code Corrected

Informational Findings	5

- Incorrect Comments Specification Changed
- Redundant Checks Code Corrected
- Change Event Emitted Even if No Change Code Corrected
- Direct Burn in Case of No Fees Code Corrected
- Incorrect Natspec and Redundant Logic _amountCanBeSent/_amountCanBeReceived Code Corrected

6.1 Incorrect Amount Passed in `quote_send`

Correctness Low Version 4 Code Corrected

CS-SOA-006

In the Solana OFT, instruction `quote_send` calls `EndpointV2::quote` with the encoded message to query the bridging fees. However, when encoding the message, the field `amount_sd` (amount in shared decimals) is incorrectly set as amount received in local decimals. In case the estimated fee is dependent on the amount in shared decimals, wrong quote results will be returned.

Code corrected:

The amount passed into quote has been corrected to be in shared decimals.

6.2 Unable to Receive Transfers to `address(0)` or Token Contract Address

Correctness Low Version 1 Code Corrected

CS-SOA-002

The Sky tokens this adapter is designed to work with do not support transfers or minting to `address(0)` or to the token contract address itself.

To illustrate, the USDS token implements the following check in its `transfer()` function:

```
function transfer(address to, uint256 value) external returns (bool) {
    require(to != address(0) && to != address(this), "Sky/invalid-address");
}
```

The same checks are present in `mint()`.

This causes any message attempting to bridge tokens to the `address(0)` or to the token contract address to fail in the OFTAdapter.

The `MABAOFTAdapter` which uses minting and burning instead of token locking partially mitigates this if the recipient is `address(0)`. In that specific case `address(0)` is translated to `0xDEAD`. However, it is still affected if the recipient is the token contract address.

Code corrected:

Logic has been implemented in both adapters to credit tokens to `address(0xdead)` in case the recipient is set to `address(0)` or the token contract address.

6.3 Incorrect Comments

Informational **Version 4** **Specification Changed**

CS-SOA-007

In `lz_receive_types`, multiple comments indicate that for the `clear` instruction, 9 accounts (`0..9`) are required. However, `get_accounts_for_clear()` returns 8 accounts.

Specifications changed:

The comments have been updated to reflect the correct amount of accounts.

6.4 Redundant Checks

Informational **Version 4** **Code Corrected**

CS-SOA-008

In Solana OFT, before doing the CPI to `endpoint::send` the `oft::send` instruction checks the `oft_store` matches the second account of the remaining accounts.

```
require!(
    ctx.accounts.oft_store.key() == ctx.remaining_accounts[1].key(),
    OFTError::InvalidSender
);
```

However, this is already checked in `oapp::endpoint_cpi::send`, hence the check above is redundant.

```
if sender != accounts[1].key() {
    return Err(ErrorCode::ConstraintAddress.into());
}
```

Code corrected:

The redundant check has been removed.

6.5 Change Event Emitted Even if No Change

Informational | Version 1 | Code Corrected

CS-SOA-003

Some events indicating a state update are emitted even if no state change occurred. This only applies to admin functions that update the configuration.

- `_setRateLimits()` with the `RateLimitsChanged` event
- `_resetRateLimits()` with the `RateLimitReset` event
- `_setRateLimitAccountingType()` with the `RateLimitAccountingTypeSet` event
- `setPauser()` with the `PauserStatusChange` event

Code corrected:

In Version 4 `setPauser()` performs an idempotency check to prevent unnecessary state changes. For other functions, the privileged role (Governance) is expected to call them only with meaningful changes.

6.6 Direct Burn in Case of No Fees

Informational | Version 1 | Code Corrected

CS-SOA-004

The mint and burn adapter first transfers tokens from the user to itself before burning them in a second step. This is required if fees are enabled since the fee amount remains in the contract and isn't burned. However, if the fee is disabled, the adapter could skip the transfer and instead burn the tokens directly from the caller using their allowance. This would save a notable amount of gas. The Sky tokens this adapter is intended to be used support burning tokens using allowances:

```
function burn(address from, uint256 value) external {
    uint256 balance = balanceOf[from];
    require(balance >= value, "Sky/insufficient-balance");

    if (from != msg.sender) {
        uint256 allowed = allowance[from][msg.sender];
        if (allowed != type(uint256).max) {
            require(allowed >= value, "Sky/insufficient-allowance");

            unchecked {
                allowance[from][msg.sender] = allowed - value;
            }
        }
        ...
```

Code corrected:

The adapter now directly burns tokens from the caller when fees are disabled avoiding an unnecessary transfer and thereby reducing gas used.

In Version 5, instead of transferring the fee, the `amountSentLD` is burned and the fee is minted if non-zero.

6.7 Incorrect Natspec and Redundant Logic _amountCanBeSent/_amountCanBeReceived

Informational Version 1 Code Corrected

CS-SOA-005

DoubleSideRateLimiter implements two separate functions `_amountCanBeSent` and `_amountCanBeReceived` to calculate the transferable amount.

The natspec of `_amountCanBeReceived()` incorrectly starts with `@notice Checks current amount in flight and amount that can be sent for a given rate limit window.` describing `send` instead of `receive`.

While separating send and receive functions may help keep the interface generic, both currently implement identical logic and could be unified into a single function to reduce redundancy.

Code corrected:

The natspec has been corrected. The separate functions have been kept since they were already audited and used in production. Moreover, in Version 2 the code has been refactored, the shared functionality of the Adapters has been moved into a base contract.

7 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

7.1 Bridge Amount Considerations

Note Version 1

Due to the OFT fee and potential transfer fees, the bridged amount may not match the input amount. Assuming 6 shared decimals:

EVM OFT: In both adapters, function `_debit` will compute the amount to be received by deducting fees and dust:

1. Any amount below 1e12 cannot be bridged.
2. The dust is also accumulated as fees.
3. The computation of the fee is slightly rounded down.

Solana OFT: In the `send` instruction, function `compute_fee_and_adjust_amount` will compute the amount to be received by deducting fees, token transfer fees and dust:

1. Any amount below 1e3 cannot be bridged.
2. The dust is not accumulated as fees.
3. The computation of the token transfer fee is rounded up while the OFT fee is rounded down.

7.2 CPI Depth Limitations

Note Version 4

In Solana, CPI (Cross-Program Invocation) has a depth limitation of 4. Hence, some operations may not be supported and revert. If users are using an executor program to trigger the Governance OApp's `lz_receive` and further call the EndpointV2 / OFT, there would be 3 CPIs left:

- For configuring OFT on EndpointV2, 3 CPIs are sufficient.
- For configuring OFT directly on the OFT itself, 3 CPIs are sufficient.
- For withdraw OFT fees, in case the token has a hook which further triggers another CPI, the transfer will revert due to exceeding 4 CPIs.
- For sending tokens with OFT, at least another 4 CPIs are needed due to the nested calls into OFT, EndpointV2, SendLibrary, and token. Hence, this will revert.

If users are triggering `lz_receive` directly without an executor contract, one more CPI will be available, and some operations above may become possible.

7.3 Configuration Considerations

Note Version 1

Both the EVM and SVM OFTs require correct configurations to work correctly. In general, this includes:

- Correctly setting the shared decimals.

- Correctly setting peers (and their configurations) on each chain.
- Correctly setting a DVN configuration, including optional settings such as block confirmations, security threshold, the Executor, max message size, and send/receive libraries. If no send and receive libraries are explicitly set, the Endpoint will fall back to the default settings set by LayerZero Labs. In case LayerZero Labs changes the default settings, the OApps will be impacted and use the new default settings which implies a trust in LayerZero Labs.
- Correctly setting the `enforcedOptions` to ensure users pay a predetermined amount of gas for delivery on the destination transaction. It should be computed such that messages sent from a source have sufficient gas to be executed on the destination chain. Setting a gas limit too small could mean that no executor has an incentive to pay for the delivery of the message at the destination, and the message should either be dropped by the admin, or some executor should execute it at a loss to resume message handling.

7.4 Configuration Ordering Considerations

Note Version 4

The Solana OFT is expected to be governed by the Solana Governance OApp, namely a *cpi-authority* is expected to be its admin and delegate. The *cpi-authority* is expected to be used as signer during the governance dictated CPI, and is responsible for configuring both the OFT and the OFT on the EndpointV2.

Since execution order on LayerZero V2 is not enforced, measures should be taken to ensure the necessary ordering of some configurations is respected.

For instance, when setting up a new peer and eid:

1. First several governance messages setup the send / receive library and the peer address of the eid.
2. Following governance messages setup the rate limits.

Since these actions are non-atomic, after Step 1 one can potentially already send or receive unlimited amounts, bypassing the intended limits set up in Step 2.

Alternatively, a multicall instruction can help to aggregate the configurations with ordering requirement, however:

- This adds up complexity to the encoded calls in the governance message.
- This consumes one more CPI call hence further restricting the CPIs available to the governance action.

To summarize, multiple governance messages may need to be relayed for OFT configurations, and measures should be taken to ensure the ordering dependency.

Since Version 5, the rate limit has been changed to be 0 by default, restricting transfers from / to peers with no rate limit configurations.

7.5 Default Library Can Be Updated by LayerZero

Note Version 1

The owner of LayerZero's MessageLibManager can register and update the default send or receive libraries. If the default library is implicitly used by the OApp, it may be subject to future changes by

LayerZero. The OApp needs to explicitly select the default library if an automatic update by LayerZero is not expected.

7.6 Denial of Service

Note Version 1

It is known and documented that denial of service scenarios against the bridge exist.

Inherent to rate limits, if active, they may be abused to deny service:

- When gross accounting is active, the limit can be consumed by repeated back and forth transfers.
- When net accounting is active, DoS is possible by bridging in circles using other methods to return the value to the origin chain.

Bridging cost (Bridge fee if active, LayerZero fee, and transaction fee) are comparatively low, making such attacks feasible if sufficient capital is available.

The SKY and SDAO tokens feature a special role as governance tokens used for voting. If large amounts of such governance tokens are on another chain, a DoS against the bridge may prevent them from bridging back and participating in votes. In case too many governance tokens are stuck in flight, governance attacks may become cheaper.

A mempool observer can see transactions and may attempt to block individual transactions by consuming the available limit.

Further reasons why users may be unable to use the bridge:

- Executor is down, hence the `lzReceive()` function needs to be triggered by someone else.
- Gas on the destination chain spikes too high, so the executor cannot execute it immediately with the gas reimbursed.

7.7 Discrepancies Between OApp and Existing Bridges

Note Version 1

In the existing Optimism bridge, the caller to `bridgeERC20()` is restricted to be an EOA to prevent user accidentally bridge from / to a smart wallet account since some smart contract wallets cannot be deployed at the same address on every blockchain.

Note different from the existing bridges, such check does not exist in the OApp anymore. In addition, the introduction of EIP-7702 enables setting code for an EOA, rendering the original check meaningless.

In addition, address translation (implemented in EVM OFT) does not exist in the existing Optimism and Arbitrum bridges.

The existing bridges use separate escrow contracts to lock funds separately, while:

- The EVM OFT locks all funds transferred to all supported destinations within the OFTAdapter contract itself.
- The Solana OFT locks the funds into a designated token escrow that may or may not be shared between different destinations.

7.8 EVM OFT Net Mode Considerations

Note Version 1

The rate limit accounting mode can be switched to `NET`, where change of one rate limit may influence its opposite one. For a flow A configured with a rate limit, if its opposite flow B is not configured:

1. B will not have any available capacity even if A and B are working in `NET` mode.
2. In `NET` mode the `lastUpdated` timestamp of B can be updated when rate limit of A is updated. Hence, the `lastUpdated` field should not be assumed to be zero for unconfigured flows.

7.9 LayerZero V2 Considerations

Note Version 1

OFTs and OFT integrators should be aware of the considerations below:

- *Execution Order*: According to the design of LayerZero V2, the delivery of messages on the destination is not guaranteed to be in the same order as they were dispatched on the source. Consequently, in case multiple sends are relayed to the same destination, they might be reordered and lead to unexpected results.
- *Censorship*: Denial-of-Service and censorship are possible since it is not guaranteed the DVN will verify the send messages in time.
- *Sandwiching*: The execution of receiving messages is permissionless. Hence, anyone can trigger the execution once the message is verified. Consequently, the reception can be sandwiched by an attacker with other operations for MEVs or attacks with flashloans in particular.
- *Refund*: When sending messages a refund address can be provided. This refund target should be able to receive the refund. For example, on the EVM OApp the refund address should be able to receive native token transfers (e.g. by implementing `receive()` or `fallback()` if it is a contract or an EOA using EIP-7702) and LayerZero token transfers (i.e. non-zero address). Otherwise, sending may revert if there is a refund.
- *Alternative Native Token*: LayerZero implements endpoints with alternative native tokens (e.g. EndPointV2Alt for EVM chains) where the native token has no significant value. Note that chains with such endpoints are unsupported.

7.10 Migration Considerations

Note Version 1

The existing Optimism and Arbitrum bridges lock funds into (release funds from) an escrow on mainnet and mint (burn) tokens on L2. Transfers in flight during migration might be stuck in the following scenarios:

- Assume a L1->L2 transfer is not yet finalized after the L2 bridge is migrated. Consequently, this transfer cannot be finalized anymore since L2 bridge already loses its minting rights.
- Assume a L2->L1 transfer is not yet finalized after the L1 escrow is migrated. Consequently, this transfer cannot be finalized anymore since insufficient funds can be transferred from the escrow.

The migration should be carefully planned and care should be taken to ensure there is no pending transfers blocked because of the bridge migration.

Note similar situations may also apply to the OFTAdapter. Once `migrateLockedTokens()` is called, in-flight cross-chain transfers may fail due to insufficient funds.

7.11 No Additional Minting Roles on L2

Note Version 1

Generally the OFTAdapter can only complete redemptions if it has the amount of tokens available. The underlying assumption is that only bridged tokens exist on L2, i.e. the bridge is the only minting role on L2.

This adapter is intended to replace the old canonical bridges. Assumption is that upon migration the token balances are transferred from the old escrows and starting balances are correct.

7.12 Pausing Outbound Flow Using High Fees

Note Version 1

The system allows accounts with pauser role (granted by owner) to pause all inbound / outbound transfers entirely. Note that in addition to the intended and documented pausing functionality the owner can also pause all outbound transfers by simply setting a high fee.

7.13 Rate Limit May Not Match Each Other

Note Version 1

To bridge funds from chain A and B, outbound limit on A and inbound limit on B should be configured.

In case the outbound limit on A is less than the inbound limit on B, there could be a "traffic jam" of inbound transfers on B, which can only be finalized later when the demand of this flow decreases.

7.14 Receive Flow Can Be Paused

Note Version 1

Unlike the existing native Optimism and Arbitrum bridges of Sky where outbound transfers can be paused but inbound finalizations cannot, this bridge allows pausing the receive flow on the destination chain. As a result finalization of in-flight cross-chain transfers can be blocked potentially leaving funds stuck (either held in the source chain's OApp or already burned).

This design is intentional due to different security assumptions. On L1, all funds are held in a single escrow, without distinction between destination chains. In the event of an oracle compromise, the ability to pause receive flows is a critical safety mechanism.

Users should be aware that paused receive flows may delay access to funds.

7.15 Send Context in Callbacks

Note Version 1

When initiating a `send()`, the EVM OApp eventually calls `EndpointV2.send()` which will cache the `dstEid` and `msg.sender` into the current context. If extra native token is provided, a refund will happen by a lowlevel call to the refund address.

Consequently, any sender can trigger a callback by passing a little bit more native token, during which the following view functions on EndpointV2 will return the current context:

- `isSendingMessage()`
- `getSendContext()`

Third party integrators should be aware of this feature to avoid unexpected usage.

7.16 Sharing DVN Makes Rate Limit Ineffective

Note Version 1

The OFT rate limit, by design, restricts the potential loss incurred by a compromised DVN. Namely, a compromised DVN can only bridge limited tokens by forging validated cross-chain messages.

However, if the OFT and the Governance OApp share the same DVN, in case of a compromise, the DVN can directly forge governance messages to configure everything on the OFT including the rate limit itself, effectively renders the rate limit meaningless.

7.17 Token With Transfer Fees

Note Version 4

The Solana OFT implementation supports tokens with transfer fees. However, the EVM OFT does not.

7.18 `lzReceive` May Fail on Solana OFT

Note Version 4

The EVM OFT adapters implement address translation where transfer or mint to certain addresses will be redirected to ensure it will succeed.

However, such address translation is not implemented on Solana OFT due to the way Solana SPL token works. Token transfer and minting may fail due to many reasons for instance:

- If CpiGuard is enabled on the token, the CPI call to transfer will revert.
- If the destination token account is frozen, no tokens can be transferred or minted to it.
- If the token uses hook extension, it may revert during the hook CPI.

To summarize, a cross-chain token transfer message to Solana OFT may be never consumed and stuck.

7.19 `quote_oft()` Does Not Consider Rate Limits

Note Version 1

The natspec of `struct OFT` states: “These amounts can change dynamically and are up to the specific OFT implementation”.

In Version 1 of the EVM OFT, `quoteOFT()` returns `0` and `token.totalSupply()` as limits without considering rate limits.

In Version 4 of the Solana OFT, `oft::quote_oft` returns `max(uint64)` as the upper limit and also ignores rate limiter configurations.

Rate limits are now considered in the both the EVM OFT (Version 5) and the Solana OFT (Version 6).

7.20 sendTo Address May Contain Garbage

Note Version 1

In function `_lzReceive()` of the EVM OFT, the `toAddress` is casted from `sendTo` field (`bytes32`), taking only the least significant 160 bits. If garbage exists at the 96 bits of prefix, it will be ignored. Consequently, different `sendTo` field may point to the same `toAddress`.