

Report

v. 1.0

Customer

zkLink



# Smart Contract Audit Era Update

27th March 2024

# Contents

<b>1 Changelog</b>	<b>4</b>
<b>2 Introduction</b>	<b>5</b>
<b>3 Project scope</b>	<b>6</b>
<b>4 Methodology</b>	<b>8</b>
<b>5 Our findings</b>	<b>9</b>
<b>6 Major Issues</b>	<b>10</b>
CVF-2. FIXED . . . . .	10
CVF-3. FIXED . . . . .	10
CVF-4. FIXED . . . . .	10
CVF-5. FIXED . . . . .	11
<b>7 Moderate Issues</b>	<b>12</b>
CVF-1. INFO . . . . .	12
CVF-6. INFO . . . . .	12
<b>8 Minor Issues</b>	<b>13</b>
CVF-7. FIXED . . . . .	13
CVF-8. FIXED . . . . .	13
CVF-9. FIXED . . . . .	13
CVF-10. INFO . . . . .	13
CVF-11. INFO . . . . .	14
CVF-12. FIXED . . . . .	14
CVF-13. FIXED . . . . .	14
CVF-14. FIXED . . . . .	15
CVF-15. INFO . . . . .	15
CVF-16. INFO . . . . .	15
CVF-17. FIXED . . . . .	15
CVF-18. FIXED . . . . .	16
CVF-19. INFO . . . . .	16
CVF-20. FIXED . . . . .	16
CVF-21. INFO . . . . .	16
CVF-22. INFO . . . . .	17
CVF-23. FIXED . . . . .	17
CVF-24. INFO . . . . .	17
CVF-25. INFO . . . . .	17
CVF-26. INFO . . . . .	18
CVF-27. INFO . . . . .	18
CVF-28. INFO . . . . .	18
CVF-29. INFO . . . . .	18

CVF-30. INFO	19
CVF-31. FIXED	19
CVF-32. INFO	19
CVF-33. INFO	19
CVF-34. INFO	20
CVF-35. INFO	20
CVF-36. INFO	20
CVF-37. FIXED	20
CVF-38. INFO	21
CVF-39. FIXED	21
CVF-40. INFO	21
CVF-41. INFO	22
CVF-42. INFO	22
CVF-43. INFO	22
CVF-44. INFO	23
CVF-45. INFO	23
CVF-46. FIXED	23
CVF-47. INFO	24
CVF-48. FIXED	24
CVF-49. INFO	24
CVF-50. INFO	25
CVF-51. INFO	25
CVF-52. FIXED	25
CVF-53. INFO	26
CVF-54. INFO	26

# 1 Changelog

#	Date	Author	Description
0.1	27.03.24	A. Zveryanskaya	Initial Draft
0.2	27.03.24	A. Zveryanskaya	Minor revision
1.0	27.03.24	A. Zveryanskaya	Release

## 2 Introduction

All modifications to this document are prohibited. Violators will be prosecuted to the full extent of the U.S. law.

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

zkLink is a trading-focused multi-chain L2 network with unified liquidity secured by ZK-Rollups.



# 3 Project scope

We were asked to review difference from zksync-era:

- Compared code
- Code with Fixes

Files:

<b>era/</b>		
Storage.sol	ValidatorTimelock.sol	
<b>era/facets/</b>		
Admin.sol	Base.sol	Executor.sol
Getters.sol	Mailbox.sol	
<b>era/interfaces/</b>		
IAdmin.sol	IExecutor.sol	IGetters.sol
IL2Gateway.sol	IMailbox.sol	IZkLink.sol
<b>era/libraries/</b>		
PriorityQueue.sol		
<b>evm/gateway/arbitrum/</b>		
ArbitrumL1Gateway.sol	ArbitrumL2Gateway.sol	
<b>evm/gateway/ethereum/</b>		
EthereumGateway.sol		
<b>evm/gateway/linea/</b>		
LineaGateway.sol	LineaL1Gateway.sol	LineaL2Gateway.sol
<b>evm/gateway/optimism/</b>		
OptimismGateway.sol	OptimismL1Gateway.sol	OptimismL2Gateway.sol
<b>evm/gateway/scroll/</b>		
ScrollGateway.sol	ScrollL1Gateway.sol	ScrollL2Gateway.sol

**evm/gateway/zkpolygon/**

ZkPolygonL1 Gateway.sol	ZkPolygonL2 Gateway.sol
----------------------------	----------------------------

**evm/gateway/zksync/**

ZkSyncL1Gateway.sol	ZkSyncL2Gateway.sol
---------------------	---------------------

**evm/gateway/**

BaseGateway.sol	L1BaseGateway.sol	L2BaseGateway.sol
-----------------	-------------------	-------------------

**evm/interfaces/**

IArbitrator.sol	IGateway.sol	IL1Gateway.sol
IL2Gateway.sol	IMessageClaimer.sol	IZkLink.sol

**evm/interfaces/linea/**

IL2MessageService.sol	IMessageService.sol
-----------------------	---------------------

**evm/interfaces/optimism/**

IOptimismMessenger.sol
------------------------

**evm/interfaces/scroll/**

IScrollMessenger.sol
----------------------

**evm/interfaces/zkpolygon/**

IBridgeMessage Receiver.sol	IZkPolygon.sol
--------------------------------	----------------

**evm/interfaces/zksync/**

IL2ETHToken.sol	IL2Messenger.sol	IZkSyncL1Gateway.sol
-----------------	------------------	----------------------



# 4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment.** The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis.** Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis.** For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis.** The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

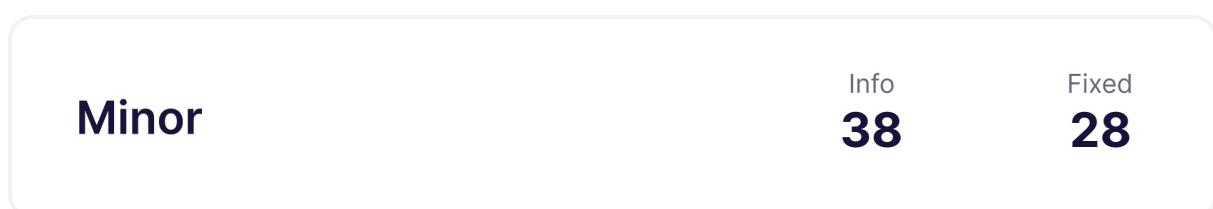
We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.



# 5 Our findings

We found 8 major, and a few less important issues. All identified Major issues have been fixed.



Fixed 37 out of 78 issues

# 6 Major Issues

## CVF-2. FIXED

- **Category** Suboptimal
- **Source** ZkPolygonL1Gateway.sol

**Recommendation** A bit mask would be more efficient if message numbers go in sequence.

```
23 23    /// @dev A mapping L2 batch number => message number => flag
24 24    /// @dev Used to indicate that zkSync L2 -> L1 message was already
25 25      ↪ processed
mapping(uint256 => mapping(uint256 => bool)) public
      ↪ isMessageFinalized;
```

## CVF-3. FIXED

- **Category** Unclear behavior
- **Source** BaseGateway.sol

**Recommendation** As zero address has the special meaning “not set”, consider explicitly requiring “\_remoteGateway” to be not zero.

```
41 41    require(remoteGateway == address(0), "Duplicate_init_remote_gateway"
      ↪ );
42 42    remoteGateway = _remoteGateway;
```

## CVF-4. FIXED

- **Category** Unclear behavior
- **Source** Admin.sol

**Recommendation** As zero address has a special meaning “not set”, there should be a check to ensure “\_gateway” isn’t zero.

```
24 +require(address(s.gateway) == address(0), "g9");
25 +s.gateway = _gateway;
```

## CVF-5. FIXED

- **Category** Suboptimal
- **Source** Mailbox.sol

**Description** Here all fields of the “SecondaryChain” structure are written into the storage, while only one field was actually changed.

**Recommendation** Consider writing only the changed field.

185    +s.secondaryChains[\_secondaryChainGateway] = secondaryChain;

345    + s.secondaryChains[\_request.gateway] = secondaryChain;



# 7 Moderate Issues

## CVF-1. INFO

- **Category** Suboptimal
- **Source** ZkSyncL1Gateway.sol

**Recommendation** A bit mask would be more efficient.

**Client Comment** *This contract was deployed on mainnet and it's hard for us to migrate to the recommend structure, so we do not plan to modify it.*

```
19 19    /// @dev A mapping L2 batch number => message number => flag
20 20    /// @dev Used to indicate that zkSync L2 -> L1 message was already
         ↪ processed
21 21    mapping(uint256 => mapping(uint256 => bool)) public
         ↪ isMessageFinalized;
```

## CVF-6. INFO

- **Category** Unclear behavior
- **Source** LineaL2Gateway.sol

**Description** Exact equality check could cause problems in case the fee changes often.

**Recommendation** Consider allowing “msg.value” to be higher than the sum of “\_value” and “coinbaseFee”, and refund the surplus.

**Client Comment** *The msg value send to sendMessage is provided by the validator. The validator will estimate the transaction in advance to ensure that the coinbase fee provided to the linea canonical bridge is correct.*

```
24 24    require(msg.value == _value + coinbaseFee, "Invalid\u00b3value");
```

# 8 Minor Issues

## CVF-7. FIXED

- **Category** Documentation
- **Source** IL2Messenger.sol

**Description** The semantics of the returned value is unclear.

**Recommendation** Consider documenting.

7 7 `function sendToL1(bytes memory _message) external returns (bytes32);`

## CVF-8. FIXED

- **Category** Bad datatype
- **Source** IZkPolygon.sol

**Recommendation** The type for these fields should be “IERC20”.

9 9 `address token,`

27 27 `address originTokenAddress,`

## CVF-9. FIXED

- **Category** Documentation
- **Source** IL2MessageService.sol

**Description** This comment is confusing.

**Recommendation** Consider rephrasing it and adding more details.

7 7 `/// @notice Returns coinbase fee when sendMessage`

## CVF-10. INFO

- **Category** Bad datatype
- **Source** IGateway.sol

**Recommendation** The return type should be more specific.

**Client Comment** *The remote gateway only used in encoding as an address and does not use its interface, so we do not plan to modify it.*

6 6 `function getRemoteGateway() external view returns (address);`



## CVF-11. INFO

- **Category** Procedural
- **Source** ZkSyncL1Gateway.sol

**Description** We didn't review these files.

```
4 4 import {IMailbox} from "../../zksync/l1-contracts/zksync/interfaces/
  ↪ IMailbox.sol";
5 5 import {IGetters} from "../../zksync/l1-contracts/zksync/interfaces/
  ↪ IGetters.sol";

11 11 import {L2Message} from "../../zksync/l1-contracts/zksync/Storage.
  ↪ sol";
12 12 import {UnsafeBytes} from "../../zksync/l1-contracts/common/
  ↪ libraries/UnsafeBytes.sol";
13 13 import {L2_ETH_TOKEN_SYSTEM_CONTRACT_ADDR} from "../../zksync/l1-
  ↪ contracts/common/L2ContractAddresses.sol";
```

## CVF-12. FIXED

- **Category** Procedural
- **Source** ZkSyncL1Gateway.sol

**Recommendation** It is a good practice to put a comment into a empty block to explain why the block is empty.

```
29 29 receive() external payable {}
```

## CVF-13. FIXED

- **Category** Documentation
- **Source** ZkSyncL1Gateway.sol

**Recommendation** Consider adding a comment next to this argument with the argument name to clarify why the transaction origin address is used here.

```
50 50 // solhint-disable-next-line avoid-tx-origin
51 51 tx.origin
```

## CVF-14. FIXED

- **Category** Bad datatype
- **Source** ZkSyncL1Gateway.sol

**Recommendation** The value “108” should be a named constant.

111 111

```
require(_message.length >= 108, "Incorrect_message_length");
```

## CVF-15. INFO

- **Category** Procedural
- **Source** ZkSyncL2Gateway.sol

**Description** We didn’t review this file.

7 7

```
import {AddressAliasHelper} from "../../zksync/l1-contracts/vendor/
    ↪ AddressAliasHelper.sol";
```

## CVF-16. INFO

- **Category** Bad datatype
- **Source** ZkSyncL2Gateway.sol

**Recommendation** The type for the “\_zkLink” argument should be more specific.

**Client Comment** *The ‘\_zkLink’ only used to check whether the caller is permitted and does not use its interface, so we do not plan to modify it.*

26 26

```
constructor(address _zkLink) L2BaseGateway(_zkLink) {
```

## CVF-17. FIXED

- **Category** Documentation
- **Source** ZkPolygonL1Gateway.sol

**Description** It is unclear what “default” means for a constant.

**Recommendation** Consider rephrasing or adding more details.

15 15  
16 16

```
// Default to true
bool public constant FORCE_UPDATE_GLOBAL_EXIT_ROOT = true;
```

## CVF-18. FIXED

- **Category** Documentation
- **Source** ZkPolygonL2Gateway.sol

**Description** It is unclear what “default” means for a constant.

**Recommendation** Consider rephrasing or adding more details.

```
14 14 // Default to true
15 15 bool public constant FORCE_UPDATE_GLOBAL_EXIT_ROOT = true;
```

## CVF-19. INFO

- **Category** Bad datatype
- **Source** ZkPolygonL2Gateway.sol

**Recommendation** The type for the “\_zkLink” argument should be more specific.

**Client Comment** Same as CVF-16.

```
23 23 constructor(address _zkLink, IZkPolygon _messageService)
    ↪ L2BaseGateway(_zkLink) {
```

## CVF-20. FIXED

- **Category** Documentation
- **Source** ScrollL1Gateway.sol

**Recommendation** Consider adding a comment next to this argument with the argument name, otherwise it is unclear why the transaction origin address is used here.

```
36 36 tx.origin
```

## CVF-21. INFO

- **Category** Bad datatype
- **Source** ScrollL2Gateway.sol

**Recommendation** The type for this argument should be more specific.

**Client Comment** Same as CVF-16.

```
11 11 address _zkLink,
```

**CVF-22.INFO**

- **Category** Bad datatype
  - **Source** OptimismL2Gateway.sol

**Recommendation** The type for this argument should be more specific.

**Client Comment** Same as CVF-16.

11 11

**address\_zkLink**

CVF-23. FIXED

- **Category** Procedural
  - **Source** OptimismL2Gateway.sol

**Recommendation** The address of the optimism messenger should be a named constant or a constructor argument.

12 12

CVF-24.INFO

- **Category** Bad datatype
  - **Source** LinealL2Gateway.sol

**Recommendation** The type for this argument should be more specific.

**Client Comment** Same as CVF-16.

11

**address** zklink.

CVF-25.INFO

- **Category** Bad datatype
  - **Source** EthereumGateway.sol

**Recommendation** The type for the “`zklLink`” argument should be more specific.

**Client Comment** Same as CVF-16

10

```
constructor(IArbitrator _arbitrator, address _zkLink) L1BaseGateway(  
    ↴ _arbitrator) L2BaseGateway(_zkLink) {
```



## CVF-26. INFO

- **Category** Procedural
- **Source** ArbitrumL1Gateway.sol

**Description** We didn't review these files.

```
4 4 import {Inbox, IBridge} from "@arbitrum/nitro-contracts/src/bridge/
  ↪ Inbox.sol";
5 5 import {IOutbox} from "@arbitrum/nitro-contracts/src/bridge/Outbox.
  ↪ sol";
```

## CVF-27. INFO

- **Category** Procedural
- **Source** ArbitrumL2Gateway.sol

**Description** We didn't review this file.

```
4 4 import {ArbSys} from "@arbitrum/nitro-contracts/src/precompiles/
  ↪ ArbSys.sol";
```

## CVF-28. INFO

- **Category** Procedural
- **Source** ArbitrumL2Gateway.sol

**Description** We didn't review this file.

```
6 6 import {AddressAliasHelper} from "../../zksync/l1-contracts/vendor/
  ↪ AddressAliasHelper.sol";
```

## CVF-29. INFO

- **Category** Bad datatype
- **Source** ArbitrumL2Gateway.sol

**Recommendation** The argument type should be more specific.

**Client Comment** Same as CVF-16.

```
20 20 constructor(address _zkLink) L2BaseGateway(_zkLink) {
```

## CVF-30. INFO

- **Category** Bad datatype
- **Source** BaseGateway.sol

**Recommendation** The type for this variable should be more specific.

**Client Comment** Same as CVF-10.

11 11

```
address internal remoteGateway;
```

## CVF-31. FIXED

- **Category** Bad naming
- **Source** BaseGateway.sol

**Recommendation** Events are usually named via nouns, such as “RemoteGateway”.

20 20

```
event SetRemoteGateway(address remoteGateWay);
```

## CVF-32. INFO

- **Category** Bad datatype
- **Source** BaseGateway.sol

**Recommendation** The parameter type should be more specific.

**Client Comment** Same as CVF-10.

20 20

```
event SetRemoteGateway(address remoteGateWay);
```

## CVF-33. INFO

- **Category** Bad datatype
- **Source** BaseGateway.sol

**Recommendation** The return type should be more specific.

**Client Comment** Same as CVF-10.

34 34

```
function getRemoteGateway() external view returns (address) {
```



## CVF-34. INFO

- **Category** Bad datatype
- **Source** BaseGateway.sol

**Recommendation** The argument type should be more specific.

**Client Comment** Same as CVF-10.

40 40 `function setRemoteGateway(address _remoteGateway) external onlyOwner { ↪ }`

## CVF-35. INFO

- **Category** Bad datatype
- **Source** L2BaseGateway.sol

**Recommendation** The type for this variable should be more specific.

**Client Comment** Same as CVF-16.

8 8 `address public immutable ZKLINK;`

## CVF-36. INFO

- **Category** Bad naming
- **Source** L2BaseGateway.sol

**Recommendation** Events are usually named via nouns, such as “SentL2GatewayMessage”.

**Client Comment** May affect third-party indexing services, we do not plan to modify it.

17 17 `event L2GatewayMessageSent(uint256 value, bytes callData);`

## CVF-37. FIXED

- **Category** Unclear behavior
- **Source** L2BaseGateway.sol

**Recommendation** This event is never emitted in this contract. Probably it would be declared elsewhere, e.g. in the “IL2Gateway” interface or in the contract that actually emits it.

17 17 `event L2GatewayMessageSent(uint256 value, bytes callData);`



## CVF-38. INFO

- **Category** Bad datatype
- **Source** L2BaseGateway.sol

**Recommendation** The argument type should be more specific.

**Client Comment** Same as CVF-16.

25 25

```
constructor(address _zkLink) {
```

## CVF-39. FIXED

- **Category** Suboptimal
- **Source** PriorityQueue.sol

**Description** The expression “\_queue.head + i” is calculated twice.

**Recommendation** Consider calculating once and reusing.

65   

```
+require(_queue.head + i < _queue.tail, "D"); // invalid index
```

  
67   

```
+return _queue.data[_queue.head + i];
```

## CVF-40. INFO

- **Category** Procedural
- **Source** IL2Gateway.sol

**Description** Specifying a particular compiler version makes it harder to upgrade to newer versions.

**Recommendation** Consider specifying as “^0.8.0”. Also relevant for: IZkLink.sol

**Client Comment** We do not plan to modify it and keep it consistent with the zksync era code.

2   

```
+pragma solidity 0.8.19;
```

## CVF-41. INFO

- **Category** Bad datatype
- **Source** IAdmin.sol

**Recommendation** The type for the “\_gateway” argument should be more specific.

**Client Comment** *The ‘\_gateway’ is only used as an address and does not use its interface, so we do not plan to modify it.*

```
21 +function setSecondaryChainGateway(address _gateway, bool _active)
    ↪ external;
73 +event SecondaryChainStatusUpdate(address indexed gateway, bool
    ↪ isActive);
```

## CVF-42. INFO

- **Category** Bad datatype
- **Source** IGetters.sol

**Recommendation** The type of the “gateway” arguments should be “IL2Gateway”.

**Client Comment** *Same as CVF-41.*

```
22 +function getSecondaryChain(address gateway) external view returns (
    ↪ SecondaryChain memory);
26 + address gateway,
```

## CVF-43. INFO

- **Category** Bad datatype
- **Source** IMailbox.sol

**Recommendation** The type for the secondary chain gateway arguments should be more specific.

**Client Comment** *Same as CVF-41.*

```
107 +address gateway;
230 +address _secondaryChainGateway,
241 +address _secondaryChainGateway,
```



## CVF-44. INFO

- **Category** Bad datatype
- **Source** IMailbox.sol

**Recommendation** The type for the secondary chain gateway parameters should be more specific.

**Client Comment** Same as CVF-41.

```
275 + address secondaryChainGateway,  
285 +event SyncBatchRoot(address secondaryChainGateway, uint256  
    ↴ batchNumber, uint256 forwardEthAmount);
```

## CVF-45. INFO

- **Category** Bad datatype
- **Source** Admin.sol

**Recommendation** The type for the “\_gateway” argument should be more specific.

**Client Comment** Same as CVF-41.

```
30 +function setSecondaryChainGateway(address _gateway, bool _active)  
    ↴ external onlyGateway {
```

## CVF-46. FIXED

- **Category** Unclear behavior
- **Source** Admin.sol

**Description** This event is emitted even if nothing actually changed.

```
32 +emit SecondaryChainStatusUpdate(_gateway, _active);
```



## CVF-47. INFO

- **Category** Bad datatype
- **Source** Getters.sol

**Recommendation** The type for the “gateway” arguments should be more “IL2Gateway”.

**Client Comment** Same as CVF-41.

```
37 +function getSecondaryChain(address gateway) external view returns (
    ↪ SecondaryChain memory) {
43     + address gateway,
```

## CVF-48. FIXED

- **Category** Procedural
- **Source** Mailbox.sol

**Recommendation** Consider specifying the constant value as a hash expression rather than as a hardcoded hash value. Solidity compiler is smart enough to precompute hash expressions.

```
35 //+ keccak256("ForwardL2Request(address gateway,bool isContractCall,
    ↪ address sender,uint256 txId,address contractAddressL2,uint256
    ↪ l2Value,bytes32 l2CallDataHash,uint256 l2GasLimit,uint256
    ↪ l2GasPricePerPubdata,bytes32 factoryDepsHash,address
    ↪ refundRecipient)")
36 +bytes32 public constant FORWARD_REQUEST_TYPE_HASH =
37 +0xe0aaca1722ef50bb0c9b032e5b16ce2b79fa9f23638835456b27fd6894f8292c;
```

## CVF-49. INFO

- **Category** Bad datatype
- **Source** Mailbox.sol

**Recommendation** The type for these arguments should be more specific.

**Client Comment** Same as CVF-41.

```
151 +address _secondaryChainGateway,
191 +address _secondaryChainGateway,
```



## CVF-50. INFO

- **Category** Suboptimal
- **Source** Mailbox.sol

**Description** The expression “`s.secondaryChains[_secondaryChainGateway]`” is calculated twice.

**Recommendation** Consider calculating once and reusing.

**Client Comment** *We do not plan to modify it.*

```
157 +SecondaryChain memory secondaryChain = s.secondaryChains[  
    ↵ _secondaryChainGateway];
```

```
185 +s.secondaryChains[_secondaryChainGateway] = secondaryChain;
```

## CVF-51. INFO

- **Category** Suboptimal
- **Source** Mailbox.sol

**Description** The expression “`s.secondaryChains[_l2WithdrawReceiver]`” is calculated several times.

**Recommendation** Consider calculating once and reusing.

**Client Comment** *We do not plan to modify it.*

```
276 +if (s.secondaryChains[_l1WithdrawReceiver].valid) {  
277     +    s.secondaryChains[_l1WithdrawReceiver].totalPendingWithdraw =  
278         +            s.secondaryChains[_l1WithdrawReceiver].totalPendingWithdraw  
    ↵     +
```

## CVF-52. FIXED

- **Category** Suboptimal
- **Source** Mailbox.sol

**Description** The storage slot address of “`s.secondaryChains[_l1WithdrawReceiver].totalPendingWithdraw`” is calculated twice.

**Recommendation** Consider using the “`+=`” operator to make the slot address to be reused.

```
277 +s.secondaryChains[_l1WithdrawReceiver].totalPendingWithdraw =  
278     +    s.secondaryChains[_l1WithdrawReceiver].totalPendingWithdraw +
```



## CVF-53. INFO

- **Category** Bad datatype
- **Source** Storage.sol

**Recommendation** The first key type for these mappings should be more specific.

**Client Comment** Same as CVF-41.

```
137 +mapping(address secondaryChainGateway => SecondaryChain)
     ↵ secondaryChains;
138 +mapping(address secondaryChainGateway => mapping(uint256
     ↵ secondaryChainPriorityOpId => SecondaryChainSyncStatus
     ↵ syncStatus)) secondaryChainSyncStatus;
```

## CVF-54. INFO

- **Category** Suboptimal
- **Source** Storage.sol

**Recommendation** It would be more efficient to merge these two mappings into a single mapping whose keys are secondary chain gateways and values are structs of two fields encapsulating values of the original mappings.

**Client Comment** We do not plan to modify it.

```
137 +mapping(address secondaryChainGateway => SecondaryChain)
     ↵ secondaryChains;
138 +mapping(address secondaryChainGateway => mapping(uint256
     ↵ secondaryChainPriorityOpId => SecondaryChainSyncStatus
     ↵ syncStatus)) secondaryChainSyncStatus;
```



# ABDK Consulting

## About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

## Contact

### Email

[dmitry@abdkconsulting.com](mailto:dmitry@abdkconsulting.com)

### Website

[abdk.consulting](http://abdk.consulting)

### Twitter

[twitter.com/ABDKconsulting](https://twitter.com/ABDKconsulting)

### LinkedIn

[linkedin.com/company/abdk-consulting](https://linkedin.com/company/abdk-consulting)