Report

v. 1.0

Customer

Term Structure

Smart Contract Audit

Term Structure

29th August 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	29.08.23	A. Zveryanskaya	Initial Draft
0.2	29.08.23	A. Zveryanskaya	Minor revision
1.0	29.08.23	A. Zveryanskaya	Release

2 Introduction

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.

Term Structure is a decentralized bond protocol powered by zkTrue-up, a platform that enables peer-to-peer lending and borrowing with fixed interest rates.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	EvacuationVerifier.sol	Governance.sol	Rolluper.sol
	TsbFactory.sol	TsbToken.sol	Verifier.sol
	Viewer.sol	ZkTrueUp.sol	
/lib			
	Utils.sol	Struct.sol	Storage.sol
	Operations.sol	CustomError.sol	Config.sol
	Checker.sol	Bytes.sol	
interfaces/			
	IZkTrueUp.sol	IWETH.sol	IVerifier.sol
	ITsbToken.sol	ITsbFactory.sol	IPoseidonUnit2.sol
	IGovernance.sol	IFlashLoanReceiver.sol	AggregatorV3Interface.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 23 major, and a few less important issues. All identified Major issues have been fixed.

	Info	Fixed
Major	0	23
Moderate	4	29
Minor	11	140

Fixed 192 out of 207 issues

6 Major Issues

CVF-2. FIXED

- **Category** Suboptimal
- **Source** Bytes.sol

Recommendation It would probably be cheaper to implement this function via the ”Identity” precompile. This would also make the code much simpler.

Client Comment *Implemented a customized slice function for each OpType.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Bytes.sol*
- *comment: implemented customized slice function for each chunk size*

```
10 function slice(
```

CVF-5. FIXED

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

Description These checks actually ensure consistency of the code, not data. Doing them at run time is waste of gas.

Recommendation Consider either removing them of replacing with asserts.

Client Comment *Removed them.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Operations.sol*
- *line: function 147, 153, 160, 167, 174, 183, 190, 196*

```
213 if (offset != REGISTER_PUBDATA_BYTES) revert
    ↪ ReadRegisterPubDataError();

221 if (offset != DEPOSIT_PUBDATA_BYTES) revert ReadDepositPubDataError
    ↪ ();

229 if (offset != WITHDRAW_PUBDATA_BYTES) revert
    ↪ ReadWithdrawPubDataError();

237 if (offset != FORCE_WITHDRAW_PUBDATA_BYTES) revert
    ↪ ReadForceWithdrawPubDataError();

247 if (offset != AUCTION_END_PUBDATA_BYTES) revert
    ↪ ReadAuctionEndPubDataError();

255 if (offset != CREATE_TS_BOND_TOKEN_PUBDATA_BYTES) revert
    ↪ ReadCreateTsbTokenPubDataError();

262 if (offset != WITHDRAW_FEE_PUBDATA_BYTES) revert
    ↪ ReadWithdrawFeePubDataError();

270 if (offset != EVACUATION_PUBDATA_BYTES) revert
    ↪ ReadEvacuationPubDataError();
```

CVF-8. FIXED

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

Description Using block numbers to measure time is unreliable. Average block time could change in the future.

Recommendation Consider using block timestamps instead.

Client Comment *Switched to using EXPIRATION_PERIOD = 14 days for calculate expiration time.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Config.sol*
- *line: 38*

```
47 uint256 internal constant BLOCK_PERIOD = 15 seconds;
```

CVF-9. FIXED

- **Category** Procedural
- **Source** Config.sol

Description This constant is field-dependent.

Recommendation Consider making it explicit

Client Comment *Switched to using the constant value SCALAR_FIELD_SIZE to represent the bn254 field prime.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Config.sol*
- *line: 87*

```
88 uint256 internal constant INPUT_MASK = (type(uint256).max >> 3);
```

CVF-10. FIXED

- **Category** Procedural
- **Source** IFlashLoanReceiver.sol

Recommendation A good practice is to revert in case on an unsuccessful operation, rather than return false. Returning without revert doesn't guarantee that the state wasn't changed.

Client Comment *Removed return value and switched to using try/catch in flashloan function.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/interfaces/IFlashLoanReceiver.sol*
- *line: 20*

```
21 /// @return Boolean indicating if the operation was successful
```

CVF-12. FIXED

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

Description In case p.X >= q, this code return a G1Point instance whose "x" coordinate equals to p.X, i.e. >= q.

Recommendation Consider using p.X % q instead.

Client Comment *Used the new version (0.7.0) of snarkJS to check the field of proof.*
Diamond:

- *status: fixed*

```
63 return G1Point(p.X, q - (p.Y % q));
```

CVF-16. FIXED

- **Category** Flaw
- **Source** EvacuationVerifier.sol

Description A bug in ‘negate‘ function allows invalid inputs to the verify function.

Recommendation Consider checking the proof structure elements to be in the proper range.

Client Comment *Used the new version (0.7.0) of snarkJS to check the field of proof.*
Diamond:

- *status: fixed*

```
276 Pairing.negate(proof.A),
```

CVF-17. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description Unchained initializers should be used to prevent double initialization of common base contracts.

Client Comment *Used unchained initializers.*
Diamond:

- *status: no existed*

```
38 __UUPSUpgradeable_init();
   __AccessControl_init();
```

CVF-18. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The number of committed blocks is read from the storage twice.

Recommendation Consider reading once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 396, 416*

```
45 if (_getStoredBlockHash(Utils.getCommittedBlockNum()) != keccak256(
      ↪ abi.encode(lastCommittedBlock)))
```

```
62 ZkTrueUpStorage.getStorage().committedBlockNum += uint32(newBlocks.
      ↪ length);
```

CVF-19. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The expression ”ZkTrueUpStorage.getStorage()” is calculated multiple times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 396, 415, 416*

```
52    ZkTrueUpStorage.getStorage().storedBlockHashes[
          ↪ lastCommittedBlock.blockNumber] = keccak256(
```

```
61 ZkTrueUpStorage.getStorage().committedL1RequestNum =
       ↪ committedL1RequestNum;
   ZkTrueUpStorage.getStorage().committedBlockNum += uint32(newBlocks.
       ↪ length);
```

CVF-20. FIXED

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

Description There is no check to ensure that the ”committedBlocks” and ”proof” arrays are of the same length. In case the ”proof” array longer, extra element are silently ignored.

Recommendation Consider adding appropriate length check.

Client Comment *Used a new struct to wrap them.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 66*

```
68 function verifyBlocks(RolluperStruct.StoredBlock[] memory
       ↪ committedBlocks, RolluperStruct.Proof[] memory proof)
```

CVF-21. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description Updating the same storage variable on every loop iteration is suboptimal.

Recommendation Consider updating once after the loop.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 700*

```
91 ZkTrueUpStorage.getStorage().executedL1RequestNum += pendingBlocks[i
   ↪ ].storedBlock.l1RequestNum;
```

CVF-22. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The expression ”ZkTrueUpStorage.getStorage()” is calculated multiple times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 699, 700*

```
91     ZkTrueUpStorage.getStorage().executedL1RequestNum +=
          ↪ pendingBlocks[i].storedBlock.l1RequestNum;
```

```
97 ZkTrueUpStorage.getStorage().executedBlockNum = executedBlockNum;
```

CVF-23. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The expression ”ZkTrueUpStorage.getStorage()” is calculated multiple times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 107, 108, 109*

```
115     delete ZkTrueUpStorage.getStorage().storedBlockHashes[
            ↪ committedBlockNum];
```

```
120 ZkTrueUpStorage.getStorage().committedBlockNum = committedBlockNum;
    ZkTrueUpStorage.getStorage().committedL1RequestNum -=
        ↪ revertedL1RequestNum;
```

```
123     ZkTrueUpStorage.getStorage().verifiedBlockNum =
            ↪ committedBlockNum;
```

CVF-24. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Recommendation Consider replacing the ”publicDataOffsets” array is an array of public data chunk IDs. This would make the numbers smaller and would allow simplifying the code.

Client Comment *Used chunkIdDeltas array to replace publicDataOffset.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 441, 442*

```
219 if (offset % Config.CHUNK_BYTES != 0) revert RolluperError.
        ↪ InvalidOffset(offset);
220 uint256 chunkId = offset / Config.CHUNK_BYTES;
```

CVF-25. FIXED

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

Description EncodePacking variable-length data structures is not injective and may lead to collisions, particularly as the input is under user control.

Recommendation Consider using regular ‘encode‘

Client Comment *Switched to using encode.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 408, 637, 730*

```
278         processableRollupTxHash = keccak256(abi.encodePacked(
               ↪ processableRollupTxHash, pubData));
```

```
292 bytes memory pubData = abi.encodePacked(commitmentOffset, newBlock.
       ↪ publicData);
```

```
371     pendingRollupTxHash = keccak256(abi.encodePacked(
           ↪ pendingRollupTxHash, pubData));
```

CVF-32. FIXED

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

Description Unchained initializers should be used here to prevent double initialization of base contracts.

Client Comment *Used unchained initalizers.*
Diamond:

- *status: no existed*

```
49 __UUPSUpgradeable_init();
50 __AccessControl_init();
   __ReentrancyGuard_init();
```

CVF-34. FIXED

- **Category** Flaw
- **Source** Verifier.sol

Description In case p.X >= q, this code return a G1Point instance whose ”x” coordinate equals to p.X, i.e. >= q.

Recommendation Consider using p.X % q instead.

Client Comment *Switched to the new version (0.7.0) of snarkJS to check the field of proof. Diamond:*

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
63 return G1Point(p.X, q - (p.Y % q));
```

CVF-38. FIXED

- **Category** Flaw
- **Source** Verifier.sol

Recommendation Proof elements should be checked to be valid curve points.

Client Comment *Switched to the new version (0.7.0) of snarkJS to check the field of proof. Diamond:*

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
296 Proof memory proof;
```

CVF-39. FIXED

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

Description Unchained initializers should be used to prevent double initialization of base contracts.

Client Comment *Used unchained initializers. Diamond:*

- *status: no existed*

```
53 __UUPSUpgradeable_init();
   __AccessControl_init();
   __ReentrancyGuard_init();
```

CVF-40. FIXED

- **Category** Flaw
- **Source** ZkTrueUp.sol

Recommendation These values should be checked to be a valid curve point, otherwise collisions are possible when calculating the TS address.

Client Comment *Added the babyJubjub library to validate curve points.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountFacet.sol*
- *line: 175*

```
108 uint256 tsPubKeyX,
    uint256 tsPubKeyY,
```

CVF-41. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description Here implicit underflow check introduced by compiler is used to enforce an important business-level constraint. Such approach is a bad practice as it makes code harder to read and more error-prone.

Recommendation Consider adding an explicit “require” statement to ensure the collateral amount is sufficient.

Client Comment *Added underflow check.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanLib.sol*
- *line: 49, 62, 63*

```
246 loan.collateralAmt -= amount;
```

```
274 loan.collateralAmt -= collateralAmt;
```

CVF-42. FIXED

- **Category** Overflow/Underflow
- **Source** ZkTrueUp.sol

Description The multiplication here is performed in 128 bits, which could lead to overflow.

Recommendation Consider multiplying in 256 bits.

Client Comment *Switched to uint256.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/flashLoan/FlashLoanFacet.sol*
- *line: 47*

```
356 premiums[i] = (amounts[i] * flashLoanPremium) / Config.
    ↪ FLASH_LOAN_PREMIUM_BASE;
```

7 Moderate Issues

CVF-3. FIXED

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

Description In Circom code this is know as ”req type”.

Recommendation Consider using consistent naming.

Client Comment *Used OpType to replace ReqType in circom.*
Diamond:

- *status: fixed*

```
16 enum OpType {
```

```
56 uint8 internal constant OP_TYPE_BYTES = 1;
```

CVF-4. FIXED

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

Description These operations are named differently in Circom code.

Recommendation Consider using consistent naming.

Client Comment *Revised ”CREATE_TS_BOND_TOKEN” to ”CREATE_TSB_TOKEN”. Corresponding revisions have already been implemented on the circuit side.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Operations.sol*
- *line: 17, 35, 36, 38*

```
17 NOOP,
```

```
35 ADMIN_CANCEL_ORDER,
   USER_CANCEL_ORDER,
```

```
38 CREATE_TS_BOND_TOKEN,
```

CVF-6. FIXED

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

Description Solidity allows hash expressions in constant values.

Recommendation Consider replacing hardcoded values with a hash expressions currently specified in comments.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
13 // bytes32(uint256(keccak256("governance.storage")) - 1)
   bytes32 private constant STORAGE_SLOT =
   0x1c03ec2fe6acf7b94b95c87bd1c750db913cc1fec10e1e766e5eb2c5f8b774f7;
```

```
69 /// @dev STORAGE_SLOT = bytes32(uint256(keccak256("zktrueup.storage
      ↪ ")) - 1)
70 bytes32 private constant STORAGE_SLOT =
   0xecdc3de15f62012ce8edb075e3ee0791091ec6b6b4f92c08b9f351e31a4a4ee0;
```

```
139 // bytes32(uint256(keccak256("tsbfactory.storage")) - 1)
140 bytes32 private constant STORAGE_SLOT =
    0xfa1372ca0be757cf045d81950fdc803be135140e80156400b9c6ec5a1960c006;
```

CVF-7. FIXED

- **Category** Suboptimal
- **Source** Struct.sol

Description Storing offset deltas instead of offsets would allow using a narrower type.

Client Comment *in CVF-24 change to use chunkIdDeltas to replace this and also change to use uint16 because the max chunk number is 1152.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupStorage.sol*
- *line: 30*

```
52 uint32[] publicDataOffsets;
```

CVF-11. FIXED

- **Category** Procedural
- **Source** EvacuationVerifier.sol

Description This compiler version requirement is inconsistent and even incompatible with compiler version requirements in other files.

Recommendation Consider using consistent compiler version requirements across the code base.

Client Comment *Aixed to ^0.8.17.*
Diamond:

- *status: fixed*

```
14 pragma solidity ^0.6.11;
```

CVF-13. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Used the new version (0.7.0) of snarkJS.*
Diamond:

- *status: fixed*

```
83 require(success, "pairing-add-failed");
```

CVF-14. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Used the new version (0.7.0) of snarkJS.*
Diamond:

- *status: fixed*

```
103 require(success, "pairing-mul-failed");
```

CVF-15. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Used the new version (0.7.0) of snarkJS.*
Diamond:

- *status: fixed*

```
134 require(success, "pairing-opcode-failed");
```

CVF-27. INFO

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

Description This formula assumes that the withdrawal amount fits into uint128, which could be wrong for very cheap tokens, or token with very many decimals.

Recommendation Consider not relying on such assumptions.

Client Comment *We have switched to using uint256 for a portion of amount, while maintaining uint128 for collateral and debt amounts. This approach is aimed at packing storage efficiently and saving gas.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 755, 779, 801, 807, 862*

```
350         amount = SafeCast.toUint128(((withdrawReq.amount * (10**
               ↪ decimals)) / (10**Config.SYSTEM_DECIMALS)));
```

```
355         amount = SafeCast.toUint128(
                ((forceWithdrawReq.amount * (10**decimals)) / (10**
                   ↪ Config.SYSTEM_DECIMALS))
```

```
490 uint128 increaseDebtAmt = SafeCast.toUint128((auctionEnd.debtAmt *
       ↪ 10**decimals) / 10**Config.SYSTEM_DECIMALS);
```

```
492 uint128 increaseCollateralAmt = SafeCast.toUint128(
        (auctionEnd.collateralAmt * 10**decimals) / 10**Config.
           ↪ SYSTEM_DECIMALS
```

```
522 uint128 l1Amt = SafeCast.toUint128(
        ((withdrawFee.amount * (10**assetConfig.decimals)) / (10**Config
           ↪ .SYSTEM_DECIMALS))
    );
```

```
548 uint128 l1Amt = SafeCast.toUint128(
        ((evacuation.amount * (10**assetConfig.decimals)) / (10**Config.
           ↪ SYSTEM_DECIMALS))
550 );
```

CVF-28. FIXED

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

Description The function actually never returns false.

Recommendation Consider either returning nothing or changing the code to actually return false for non-existing requests.

Client Comment *Refactored the function to return either true or false.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 302*

```
381 /// @return isExisted Return true is the request is existed in the
    ↪ L1 request queue, else return false
```

CVF-29. FIXED

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

Description The function actually never returns false.

Recommendation Consider either returning nothing or changing the code to actually return false for non-existing requests.

Client Comment *Refactored the function to return either true or false.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 315*

```
397 /// @return isExisted Return true is the request is existed in the
    ↪ L1 request queue, else return false
```

CVF-30. FIXED

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

Description The function actually never returns false.

Recommendation Consider either returning nothing or changing the code to actually return false for non-existing requests.

Client Comment *Refactored the function to return either true or false.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 328*

```
413 /// @return isExisted Return true is the request is existed in the
       ↪ L1 request queue, else return false
```

CVF-31. FIXED

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

Description The function actually never returns false.

Recommendation Consider either returning nothing or changing the code to actually return false for non-existing requests.

Client Comment *Refactored the function to return either true or false.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 341*

```
429 /// @return isExisted Return true is the request is existed in the
       ↪ L1 request queue, else return false
```

CVF-33. FIXED

- **Category** Procedural
- **Source** Verifier.sol

Description This compiler version requirement is inconsistent and even incompatible with compiler version requirements in other files.

Recommendation Consider using consistent compiler version requirements across the code base.

Client Comment *Switched to Solidity ^0.8.17.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
14 pragma solidity ^0.6.11;
```

CVF-35. FIXED

- **Category** Suboptimal
- **Source** Verifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
83 require(success, "pairing-add-failed");
```

CVF-36. FIXED

- **Category** Suboptimal
- **Source** Verifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
103 require(success, "pairing-mul-failed");
```

CVF-37. FIXED

- **Category** Suboptimal
- **Source** Verifier.sol

Description This check never fails as it duplicates another check performed inside assembly block.

Recommendation Consider refactoring.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
134 require(success, "pairing-opcode-failed");
```

CVF-43. FIXED

- **Category** Procedural
- **Source** ZkTrueUp.sol

Description The original error message is lost here.

Recommendation Consider including it into the error as a parameter.

Client Comment *Incorporated returned data into the error log.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Utils.sol*
- *line: 41*

```
621 if (!success) revert ZkTrueUpError.TransferFailed();
```

CVF-44. INFO

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Recommendation The “10**decimals” value could be precomputed and stored in the asset config.

Client Comment *Used decimals to enhance the intuitiveness and readability.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 364, 374*

```
698 repayAmt = (normalizedCollateralPrice * loan.collateralAmt) / 10**
      ↪ collateralAsset.decimals <

704 uint256 normalizedRepayValue = (normalizedDebtPrice * repayAmt) /
      ↪ 10**debtAsset.decimals;

712 uint256 repayToCollateralRatio = (Config.LTV_BASE *
      ↪ normalizedRepayValue * 10**collateralAsset.decimals) /

733         10**collateralAsset.decimals) /

752     (liquidationFactor.protocolPenalty * normalizedRepayValue * 10**
          ↪ collateralAsset.decimals) /

859     (ltvThreshold * normalizedCollateralPrice * loan.collateralAmt *
          ↪  10**debtAsset.decimals) /

861     10**collateralAsset.decimals;

898 return uint256(price) * 10**(18 - decimals);
```

CVF-45. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description The expression “10**collateralAsset.decimals” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 394, 421*

```
698 repayAmt = (normalizedCollateralPrice * loan.collateralAmt) / 10**
      ↪ collateralAsset.decimals <

712 uint256 repayToCollateralRatio = (Config.LTV_BASE *
      ↪ normalizedRepayValue * 10**collateralAsset.decimals) /

733         10**collateralAsset.decimals) /

752     (liquidationFactor.protocolPenalty * normalizedRepayValue * 10**
          ↪ collateralAsset.decimals) /
```

CVF-46. INFO

- **Category** Procedural
- **Source** ZkTrueUp.sol

Description In ERC-20 the “decimals” property is used by UI to render token amounts in a human-friendly way. Using this property in smart contracts is discouraged.

Recommendation Consider treating all token amounts as integers.

Client Comment *During the liquidation process, liquidators repay the debt tokens and receive collateral tokens of equivalent value. Consequently, the converted amount must be calculated using decimals.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 364, 374*

```
698 repayAmt = (normalizedCollateralPrice * loan.collateralAmt) / 10**
      ↪ collateralAsset.decimals <

704 uint256 normalizedRepayValue = (normalizedDebtPrice * repayAmt) /
      ↪ 10**debtAsset.decimals;

712 uint256 repayToCollateralRatio = (Config.LTV_BASE *
      ↪ normalizedRepayValue * 10**collateralAsset.decimals) /

733         10**collateralAsset.decimals) /

752     (liquidationFactor.protocolPenalty * normalizedRepayValue * 10**
          ↪ collateralAsset.decimals) /

793 uint128 l2Amt = SafeCast.toUint128((amount * 10**Config.
      ↪ SYSTEM_DECIMALS) / 10**decimals);

859     (ltvThreshold * normalizedCollateralPrice * loan.collateralAmt *
          ↪  10**debtAsset.decimals) /

861     10**collateralAsset.decimals;

895 uint8 decimals = AggregatorV3Interface(priceFeed).decimals();

898 return uint256(price) * 10**(18 - decimals);
```

CVF-47. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Recommendation Using the free memory pointer is redundant, as this function never returns control. Just use memory starting from zero address instead.

Client Comment *Removed the free memory pointer.*
Diamond:

- *status: no existed*

```
936 let ptr := mload(0x40)
```

CVF-68. FIXED

- **Category** Procedural
- **Source** Operations.sol

Recommendation In Circom code time is 64 bits. Consider using consistent time format.

Client Comment *Switched to uint32 (bytes4) as the consistent time format.*
Diamond:

- *status: no existed*

```
61 uint8 internal constant TIME_BYTES = 4;
```

CVF-69. FIXED

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

Description There is no such function in the WETH contract.

Client Comment *Removed this funciton.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/interfaces/IWETH.sol*

```
25 function mint(address to, uint256 amount) external;
```

CVF-70. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Description In case ”p.X” is not zero and ”p.Y % q” is zero, this code returns ”G1Point(p.X, q)” which is not a valid point.

Recommendation Consider returning ”G1Point(p.X, 0)” in such a case.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/EvacuVerifier.sol*

```
62 if (p.X == 0 && p.Y == 0) return G1Point(0, 0);
   return G1Point(p.X, q - (p.Y % q));
```

CVF-71. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description Governance address is read from the storage twice.

Recommendation Consider reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 600, 601, 606*

```
238 IGovernance governance = IGovernance(Utils.getGovernanceAddr());
```

```
244 GovernanceStruct.AssetConfig memory baseTokenConfig = IGovernance(
       ↪ Utils.getGovernanceAddr())
```

CVF-72. INFO

- **Category** Suboptimal
- **Source** Rolluper.sol

Description This logic looks very suspicient as it allows evacuating the same account multiple times.

Client Comment *This function needs to be used with system recovery function. In the future, we will add the function for ending evacation mode and restarting the contract. The indicated code snippet can only be executed after the evacuation mode has been triggered and all L1 requests have been consumed during this mode. It is necessary to update the status as false after we consume all requests, allowing us to restart the system. If needed, users will be able to trigger the evacuation mode again after the system restarts. Diamond:*

- *status: info*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 724-726*

```
365 } else if (opType == Operations.OpType.EVACUATION) {
        Operations.Evacuation memory evacuation = Operations.
            ↪ readEvacuationPubdata(pubData);
        ZkTrueUpStorage.getStorage().evacuated[evacuation.accountId][
            ↪ evacuation.tokenId] = false;
```

CVF-73. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description Performing several token transfer in a single transaction is a bad idea. If one of the transfers will fail for whatever reason, the other transfers will get reverted.

Recommendation Consider increasing pending balances instead.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 827, 833, 839*

```
535 _transfer(assetConfig.tokenAddr, payable(treasuryAddr), treasuryAmt)
       ↪ ;
    _transfer(assetConfig.tokenAddr, payable(insuranceAddr),
       ↪ insuranceAmt);
    _transfer(assetConfig.tokenAddr, payable(vaultAddr), vaultAmt);
```

CVF-74. FIXED

- **Category** Suboptimal
- **Source** Verifier.sol

Description In case ”p.X” is not zero and ”p.Y % q” is zero, this code returns ”G1Point(p.X, q)” which is not a valid point.

Recommendation Consider returning ”G1Point(p.X, 0)” in such a case.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/verifier/Verifier.sol*

```
62 if (p.X == 0 && p.Y == 0) return G1Point(0, 0);
```

CVF-75. FIXED

- **Category** Procedural
- **Source** Viewer.sol

Description The code checks if the id is greater or equal rather than ‘greater’.

Recommendation Consider making the comment consistent with the code.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupLib.sol*
- *line: 199*

```
68 /// @notice Check whether the request id is greater than the current
      ↪  request number
```

CVF-76. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description There is no way for a liquidator to specify the maximum “repayAmt” value, the minimum “liquidatorRewardAmt” value, and the minimum ratio between these two values. Thus, liquidation outcome could appear to be much worse than the liquidator expected.

Recommendation Consider adding extra arguments for liquidation bounding conditions.

Client Comment *Updated the liquidate function to include repayAmt as a parameter. This gives the liquidator the flexibility to choose their repayment amount, as long as it doesn’t surpass the maxRepayAmt.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 105*

```
325 _transferFrom(debtAsset.tokenAddr, sender, repayAmt, msg.value);
```

```
329 _transfer(collateralAsset.tokenAddr, payable(sender),
      ↪ liquidatorRewardAmt);
```

CVF-77. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description This also covers the case when collateral value cannot cover repay value. In such case liquidation is not profitable. However, for the protocol, liquidation with a discount (i.e. when liquidator repays part of the debt and received full collateral) could be better than no liquidation.

Recommendation Consider covering this case as well.

Client Comment *Enhanced the function, now enabling liquidators to determine the repayAmt if 0 < repayAmt ≤ maxRepayAmt.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 366*

```
716 // case1: if collateral value cannot cover protocol penalty and full
    ↪  liquidator reward
    // in this case, liquidator reward = all collateral, and protocol
    ↪ penalty = 0
    // liquidatorRewardAmt = totalCollateralAmt, and protocolPenaltyAmt
    ↪ = 0
    if (repayToCollateralRatio + liquidationFactor.liquidatorIncentive >
    ↪  maxLtvRatio)
720     return (repayAmt, loan.collateralAmt, 0);
```

CVF-78. FIXED

- **Category** Overflow/Underflow
- **Source** ZkTrueUp.sol

Description Overflow is possible when converting to “uint128”.

Recommendation Consider using safe conversion.

Client Comment *Added SafeCast to check if the value has overflow issue.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 408, 417*

```
730 liquidatorRewardAmt = uint128(
```

```
751     : protocolPenaltyAmt = uint128(
```

8 Minor Issues

CVF-96. INFO

- **Category** Procedural
- **Source** Bytes.sol

Recommendation Consider specifying as ”^0.8.0”, unless there is something special about this particular version. Also relevan for: Operations.sol, Storage.sol, CustomError.sol, Checker.sol, Struct.sol, Utils.sol, Config.sol, IZkTrueUp.sol, IFlashLoanReceiver.sol, IGovernance.sol, IPoseidonUnit2.sol, ITsbFactory.sol, ITsbToken.sol, IVerifier.sol, IWETH.sol, TsbToken.sol, Governance.sol, Rolluper.sol, TsbFactory.sol, Viewer.sol.

Client Comment *Some OZ’s dependencies are over 0.8.0 like:*

- *@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol (^0.8.1)*
- *@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol (^0.8.2)*
- *@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol (^0.8.2)*

Diamond:

- status: info

```
3 pragma solidity ^0.8.17;
```

CVF-97. FIXED

- **Category** Procedural
- **Source** Bytes.sol

Description These comments are redundant, as potential overflow would be caught by Solidity compiler.

Recommendation Consider removing these comments.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Bytes.sol*

```
38 // NOTE: theoretically possible overflow of (_offset + 4)
44 // NOTE: theoretically possible overflow of (_start + 0x4)
53 // NOTE: theoretically possible overflow of (_offset + 20)
61 // NOTE: theoretically possible overflow of (_start + 20)
69 // NOTE: theoretically possible overflow of (_offset + 2)
76 // NOTE: theoretically possible overflow of (_start + 0x2)
85 // NOTE: theoretically possible overflow of (_start + 0x10)
94 // NOTE: theoretically possible overflow of (_offset + 16)
```

CVF-98. FIXED

- **Category** Procedural
- **Source** Storage.sol

Recommendation This library should be moved into a separate file named ”GovernanceStorage.sol”.

Client Comment *Moved the library to a separate file.*
Diamond:

- *status: no existed*

```
12 library GovernanceStorage {
```

CVF-99. FIXED

- **Category** Documentation
- **Source** Storage.sol

Description The number format of these fields is unclear.

Recommendation Consider documenting.

Client Comment *fixed and remove maxLtvRatio, healthFactorThreshold because of they are constant and defined in Config.sol.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Config.sol*
- *line: 20, 26*
- *path: contracts/zkTrueUp/flashLoan/FlashLoanStorage.sol*
- *line: 12, 13*
- *path: contracts/zkTrueUp/loan/LoanStorage.sol*
- *line: 54, 55*

```
28 uint16 healthFactorThreshold;

30 uint16 halfLiquidationThreshold;

32 uint16 maxLtvRatio;

34 uint16 flashLoanPremium;
```

CVF-100. FIXED

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

Recommendation The key type should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/TokenStorage.sol*
- *line: 28, 30*

```
42 mapping(address => uint16) toTokenId;
```

```
44 mapping(address => bool) isPaused;
```

CVF-101. FIXED

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

Description Asserting a constant value looks weird.

Recommendation Consider removing this line.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
51 assert(STORAGE_SLOT == bytes32(uint256(keccak256("governance.storage
   ↪ ")) - 1));
```

CVF-102. FIXED

- **Category** Procedural
- **Source** Storage.sol

Recommendation This library should be moved into a separate file named ”ZkTrueUpStorage.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
68 library ZkTrueUpStorage {
```

CVF-103. FIXED

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

Recommendation The type of this field should be ”IGovernance”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
74 address governanceAddr;
```

CVF-104. FIXED

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

Recommendation The type of this field should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
76 address tsbFactoryAddr;
```

CVF-105. FIXED

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

Recommendation The type of this field should be ”IWETH9”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressStorage.sol*
- *line: 18*

```
78 address wETHAddr;
```

CVF-106. FIXED

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

Recommendation The type of this field should be IPoseidonUnit2”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressStorage.sol*
- *line: 20*

```
80 address poseidonUnit2Addr;
```

CVF-107. FIXED

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

Recommendation The type of this field should be ”IVerifier”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressStorage.sol*
- *line: 22*

```
82 address verifierAddr;
```

CVF-108. FIXED

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

Recommendation The type of this field should be ”EvacuationVerifier” or an interface extracted from it.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressStorage.sol*
- *line: 24*

```
84 address evacuationVerifierAddr;
```

CVF-109. FIXED

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

Recommendation The type of this field should be ”Rolluper” or an interface extracted from it.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
86 address rolluperAddr;
```

CVF-110. FIXED

- **Category** Procedural
- **Source** Storage.sol

Recommendation This library should be moved into a separate file named ”TsbFactoryStorage.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
138 library TsbFactoryStorage {
```

CVF-111. FIXED

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

Recommendation The type of this field should be ”IGovernance”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
144 address governanceAddr;
```

CVF-112. FIXED

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

Recommendation The type of this field should be ”IZkTrueUp”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
146 address zkTrueUpAddr;
```

CVF-113. FIXED

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

Recommendation The values type should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbStorage.sol*
- *line: 16*

```
149 mapping(uint48 => address) tsbTokens;
```

CVF-114. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Recommendation This library should be moved into a separate file named ”GovernanceError.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
12 library GovernanceError {
```

CVF-115. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Description These libraries contain only errors.

Recommendation Consider removing the libraries and moving the errors to the top level.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
12 library GovernanceError {
36 library ZkTrueUpError {
88 library RolluperError {
150 library TsbFactoryError {
168 library TsbTokenError {
```

CVF-116. FIXED

- **Category** Suboptimal
- **Source** CustomError.sol

Recommendation These errors could be make more useful by adding the problematic values as error parameters.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *comment: move to corresponding file*

```
14  error InvalidFundWeight();
16  error InvalidHalfLiquidationThreshold();
18  error InvalidLiquidationFactor();
20  error InvalidZeroAddr();
52  error LoanIsNotExist();
54  error InvalidZeroAddr();
104 error InvalidZeroAddr();
106 error InvalidAccountId();
120 error BaseTokenAddrIsNotMatched();
122 error MaturityTimeIsNotMatched();
136 error PendingRollupTxHashIsNotMatched();
160 error InvalidZeroAddr();
```

CVF-117. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Recommendation This library should be moved into a separate file named ”ZkTrueUpError.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
36 library ZkTrueUpError {
```

CVF-118. FIXED

- **Category** Suboptimal
- **Source** CustomError.sol

Recommendation These errors could be made more useful by adding as error arguments the original return data of the failed transaction.

Client Comment *Switched to using try/catch and renamed Error name to ExecuteOperationFailedLogString and ExecuteOperationFailedLogBytes.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/flashLoan/IFlashLoanFacet.sol*
- *line: 14, 16*
- *path: contracts/zkTrueUp/libraries/Utils.sol*
- *line: 26*

```
64 error FlashLoanExecuteFailed();
```

```
66 error TransferFailed();
```

CVF-119. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Recommendation This library should be moved into a separate file named ”RolluperError.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
88 library RolluperError {
```

CVF-120. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Recommendation This library should be moved into a separate file named ”TsbFactoryError.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
150 library TsbFactoryError {
```

CVF-121. FIXED

- **Category** Procedural
- **Source** CustomError.sol

Recommendation This library should be moved into a separate file named ”TsbTokenError.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
168 library TsbTokenError {
```

CVF-122. FIXED

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

Recommendation Should be ”nonZeroAddr” or ”notZeroAddr” but not ”noneZeroAddr”.

Client Comment *Switched to notZeroAddr.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/Utils.sol*
- *line: 78*

```
38 function noneZeroAddr(address addr) internal pure {
```

CVF-123. FIXED

- **Category** Procedural
- **Source** Struct.sol

Recommendation This library should be moved into a separate file named ”ZkTrueUp-Struct.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
11 library ZkTrueUpStruct {
```

CVF-124. FIXED

- **Category** Procedural
- **Source** Struct.sol

Description These libraries contain only structs.

Recommendation Consider removing the libraries and moving the struct to the top level.

Client Comment *Resolved.*
Diamond:

- *status: no existede*

```
11 library ZkTrueUpStruct {
```

```
35 library RolluperStruct {
```

```
76 library GovernanceStruct {
```

CVF-125. FIXED

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

Description It is unclear whether this is the lender or the debtor account.

Recommendation Consider renaming.

Client Comment *Removed this in the Loan struct. The reason is that the loan is mapped by the key of loanId, which is packed by accountId, maturityTime, debtTokenId, and collateralTokenId.*
Diamond:

- *status: no existed*

```
21 uint32 accountId;
```

CVF-126. FIXED

- **Category** Procedural
- **Source** Struct.sol

Recommendation This library should be moved into a separate file named ”Rolluper-Struct.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
35 library RolluperStruct {
```

CVF-127. FIXED

- **Category** Procedural
- **Source** Struct.sol

Recommendation This library should be moved into a separate file named ”GovernanceS-truct.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
76 library GovernanceStruct {
```

CVF-128. FIXED

- **Category** Documentation
- **Source** Struct.sol

Description The number format of these fields is unclear.

Recommendation Consider documenting.

Client Comment *Added comments.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/protocolParams/ProtocolParamsFacet.sol*
- *line: 11-15*
- *path: contracts/zkTrueUp/laon/LoanStorage.sol*
- *line: 6-15*

```
79 uint16 treasury;
80 uint16 insurance;
   uint16 vault;
```

```
96 uint16 ltvThreshold;
   uint16 liquidatorIncentive;
   uint16 protocolPenalty;
```

CVF-129. FIXED

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

Recommendation The type of this field should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/TokenFacet.sol*
- *line: 13*

```
90 address tokenAddr;
```

CVF-130. FIXED

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

Recommendation The type of this field should be ”AggregatorV3Interface”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/TokenFacet.sol*
- *line: 14*

```
91 address priceFeed;
```

CVF-131. FIXED

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

Recommendation The return type should be ”IGovernance”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
16 function getGovernanceAddr() internal view returns (address
   ↪ governanceAddr) {
```

CVF-132. FIXED

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

Recommendation The return type should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
22 function getTsbFactoryAddr() internal view returns (address
     ↪ tsbFactoryAddr) {
```

CVF-133. FIXED

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

Recommendation The return type should be ”IWETH9”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressFacet.sol*
- *line: 48*

```
28 function getWETHAddr() internal view returns (address wETHAddr) {
```

CVF-134. FIXED

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

Recommendation The return type should be ”IVerifier”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressFacet.sol*
- *line: 62*

```
34 function getVerifierAddr() internal view returns (address
     ↪ verifierAddr) {
```

CVF-135. FIXED

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

Recommendation The return type should be ”EvacuationVerifier” or an interface extracted from it.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressFacet.sol*
- *line: 69*

```
40 function getEvacuationVerifierAddr() internal view returns (address
     ↪ verifierAddr) {
```

CVF-136. FIXED

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

Recommendation The return type should be ”IPoseidonUnit2”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/address/AddressFacet.sol*
- *line: 56*

```
46 function getPoseidonUnit2Addr() internal view returns (address
      ↪ poseidonUint2Addr) {
```

CVF-137. INFO

- **Category** Readability
- **Source** Config.sol

Description Using different denominators for fixed point numbers of the same width in the same code base makes code harder to read.

Recommendation Consider using the same denominator, namely 10000.

Client Comment *Each denominator represents a unique meaning in the respective variables.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/libraries/Config.sol*
- *line: 15, 20, 23, 26, 29*

```
15 uint16 internal constant FUND_WEIGHT_BASE = 10000;

20 uint16 internal constant HEALTH_FACTOR_THRESHOLD = 1000;

23 uint16 internal constant LTV_BASE = 1000;

26 uint16 internal constant MAX_LTV_RATIO = 1000; // 100%

29 uint16 internal constant FLASH_LOAN_PREMIUM_BASE = 10000;
```

CVF-138. FIXED

- **Category** Procedural
- **Source** Config.sol

Description Solidity compiler is smart enough to calculate hash expression an compile time.

Recommendation Consider using hash expressions rather than hardcoded hash values when initializing constants.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
58 /// @notice Hash of empty string = keccak256("")
   bytes32 internal constant EMPTY_STRING_KECCAK = 0
      ↪ xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470
      ↪ ;
```

CVF-139. FIXED

- **Category** Procedural
- **Source** Config.sol

Recommendation This library should be moved into a separate file named ”InitialConfig.sol”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitiaConfig.sol*

```
111 library InitialConfig {
```

CVF-140. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.5e4” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 12*

```
113 uint16 internal constant INIT_TREASURY_WEIGHT = 5000; // 50%
```

CVF-141. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.1e4” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 15*

```
116 uint16 internal constant INIT_INSURANCE_WEIGHT = 1000; // 10%
```

CVF-142. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.4e4” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 18*

```
119 uint16 internal constant INIT_VAULT_WEIGHT = 4000; // 40%
```

CVF-143. INFO

- **Category** Suboptimal
- **Source** Config.sol

Recommendation USD amounts are usually represented as a number of cents. I.e. the value should be ”10000e2”.

Client Comment *The token prices from Chainlink are based on US dollars. Therefore, it is more convenient to use US dollar here, and it is not necessary to use such a small unit in this variable.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 22*

```
123 uint16 internal constant INIT_HALF_LIQUIDATION_THRESHOLD = 10000; //
    ↪  10000 USD
```

CVF-144. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.8e3” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 25*

```
126 uint16 internal constant INIT_LTV_THRESHOLD = 800; // 80%
```

CVF-145. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.925e3” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 28*

```
129 uint16 internal constant INIT_STABLECOIN_PAIR_LTV_THRESHOLD = 925;
    ↪ // 92.5%
```

CVF-146. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.05e3” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 31, 37*

```
132 uint16 internal constant INIT_LIQUIDATOR_INCENTIVE = 50; // 5%
```

```
138 uint16 internal constant INIT_PROTOCOL_PENALTY = 50; // 5%
```

CVF-147. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.03e3” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 34*

```
135 uint16 internal constant INIT_STABLECOIN_PAIR_LIQUIDATOR_INCENTIVE =
    ↪  30; // 3%
```

CVF-148. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.015e3” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 40*

```
141 uint16 internal constant INIT_STABLECOIN_PAIR_PROTOCOL_PENALTY = 15;
      ↪  // 1.5%
```

CVF-149. FIXED

- **Category** Readability
- **Source** Config.sol

Recommendation ”0.0003e4” would be more readable.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/libraries/InitialConfig.sol*
- *line: 43*

```
144 uint16 internal constant INIT_FLASH_LOAN_PREMIUM = 3; // 0.03%
```

CVF-150. FIXED

- **Category** Procedural
- **Source** AggregatorV3Interface.sol

Description This interface was copied from ChainLink code base.

Recommendation Consider importing it from ChainLink via NPM.

Client Comment *Resolved.*
Diamond:

- *status: fixed*

```
8 interface AggregatorV3Interface {
```

CVF-151. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Recommendation This interface should be moved into a separate file named ”RolluperEvent.sol”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
14 interface RolluperEvent {
```

CVF-152. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Description Unlike names of the other interfaces, this name doesn’t have the ”I” prefix.

Recommendation Consider using a consistent naming schema.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
14 interface RolluperEvent {
```

CVF-153. FIXED

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

Recommendation Events are usually named via nouns, such as ”Commit”, ”Verification”, ”Execution” etc.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/IRollupFacet.sol*
- *line: 61, 65, 69, 73, 107*

```
18 event BlockCommitted(uint32 blockNumber, bytes32 commitment);

22 event BlockVerified(uint32 blockNumber);

26 event BlockExecuted(uint32 blockNumber);

30 event BlockReverted(uint32 blockNumber);

47 event UpdateLoan(
```

CVF-154. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Recommendation The ”blockNumber” parameter should be indexed.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/IRollupFacet.sol*
- *line: 61, 65, 69, 73*

```
18 event BlockCommitted(uint32 blockNumber, bytes32 commitment);

22 event BlockVerified(uint32 blockNumber);

26 event BlockExecuted(uint32 blockNumber);

30 event BlockReverted(uint32 blockNumber);
```

CVF-155. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Recommendation The ”accountId” and ”tokenId” parameters should be indexed.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/IRollupFacet.sol*
- *line: 83, 85*

```
37 event Evacuation(address indexed accountAddr, uint32 accountId,
     ↪ uint16 tokenId, uint128 amount);
```

CVF-156. FIXED

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

Description The word ”New” in the name is redundant.

Recommendation Consider renaming to just ”L1Request”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupLib.sol*
- *line: 39*

```
63 event NewL1Request(
```

CVF-157. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Recommendation Events are usually named via nouns, such as ”Registration”, ”Withdrawal”, etc.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountLib.sol*
- *line: 37, 51, 58, 66*
- *path: contracts/zkTrueUp/loan/ILoanFacet.sol*
- *line: 46, 58, 73, 105*
- *path: contracts/zkTrueUp/rollup/IRollupFacet.sol*
- *line: 81, 90*

```
84  event Register(address indexed accountAddr, uint32 accountId,
        ↪ uint256 tsPubX, uint256 tsPubY, bytes20 tsAddr);

98  event Withdraw(address indexed accountAddr, uint32 accountId, uint16
        ↪  tokenId, uint128 amount);

104 event ForceWithdraw(address indexed accountAddr, uint32 accountId,
        ↪ uint16 tokenId);

109 event UpdateTsbFactory(address indexed oldTsbFactoryAddr, address
        ↪ indexed newTsbFactoryAddr);

116 event Liquidate(

129 event Redeem(

142 event AddCollateral(

154 event RemoveCollateral(

169 event Repay(

194 event EvacuationActivated();
```

CVF-158. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Recommendation The ”accountId” parameter should be indexed.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountLib.sol*
- *line: 37, 51, 58, 66*

```
84 event Register(address indexed accountAddr, uint32 accountId,
      ↪ uint256 tsPubX, uint256 tsPubY, bytes20 tsAddr);

91 event Deposit(address indexed accountAddr, uint32 accountId, uint16
      ↪ tokenId, uint128 amount);

98 event Withdraw(address indexed accountAddr, uint32 accountId, uint16
      ↪  tokenId, uint128 amount);

104 event ForceWithdraw(address indexed accountAddr, uint32 accountId,
      ↪ uint16 tokenId);
```

CVF-159. FIXED

- **Category** Suboptimal
- **Source** IZkTrueUp.sol

Recommendation The ”oldTsbFactoryAddr” is redundant, as its value could be derived from the previous events.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
109 event UpdateTsbFactory(address indexed oldTsbFactoryAddr, address
      ↪ indexed newTsbFactoryAddr);
```

CVF-160. FIXED

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

Recommendation The type of parameters should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
109 event UpdateTsbFactory(address indexed oldTsbFactoryAddr, address
      ↪ indexed newTsbFactoryAddr);
```

CVF-161. FIXED

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

Recommendation The type of this parameter should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 49*

```
132 address underlyingAssetAddr,
```

CVF-162. FIXED

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

Recommendation The type of these parameters should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountFacet.sol*
- *line: 45, 64*

```
204 address tokenAddr,
```

```
214 address tokenAddr,
```

CVF-163. FIXED

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

Recommendation The type of the ”tokenAddr” parameters should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountFacet.sol*
- *line: 86, 109*

```
221 function withdraw(address tokenAddr, uint128 amount) external;
```

```
227 function forceWithdraw(address tokenAddr) external;
```

CVF-164. FIXED

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

Recommendation The type of this argument should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 54*

```
239 address tsbTokenAddr,
```

CVF-165. FIXED

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

Recommendation The argument type should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
297 function setTsbFactory(address tsbFactoryAddr) external;
```

CVF-166. FIXED

- **Category** Procedural
- **Source** IZkTrueUp.sol

Description This function should be declared in an interface.

Recommendation Consider removing it.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
377 receive() external payable;
```

CVF-167. FIXED

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

Recommendation The type of this argument should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/interfaces/IFlashLoanReceiver.sol*
- *line: 22*

```
24 address[] calldata assets,
```

CVF-168. INFO

- **Category** Suboptimal
- **Source** IFlashLoanReceiver.sol

Recommendation It would be more efficient to pass a single array of structs with three fields, rather than three parallel arrays. This would also make the length check unnecessary.

Client Comment *We adopted the practice of separate array usage from other protocols like AAVE to enhance simplicity and user convenience.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/interfaces/IFlashLoanReceiver.sol*
- *line: 22-24*

```
24 address[] calldata assets,
   uint128[] calldata amounts,
   uint128[] calldata premiums,
```

CVF-169. INFO

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

Recommendation Events are usually named via nouns, such as ”BaseTokenWhitelisting”, ”TsbTokenWhitelisting”, ”TreasuryAddr”, etc.

Client Comment *Switched to using BaseTokenWhitelisted, TsbTokenWhitelisted, TreasuryAddr, InsuranceAddr, VaultAddr. Other event names like Set... remain the same as they are intuitive.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 22, 27, 32, 37, 43, 50*
- *path: contracts/zkTrueUp/protocolParams/IProtocolParamsFacet.sol*
- *line: 25, 29, 33, 37*

```
16 event WhitelistBaseToken(

27 event WhitelistTsbToken(

36 event SetTreasuryAddr(address indexed treasuryAddr);

40 event SetInsuranceAddr(address indexed insuranceAddr);

44 event SetVaultAddr(address indexed vaultAddr);

49 event SetIsStableCoin(address indexed tokenAddr, bool indexed
   ↪ isStableCoin);

54 event SetPaused(address indexed tokenAddr, bool indexed isPaused);

59 event SetPriceFeed(address indexed tokenAddr, address indexed
   ↪ priceFeed);

64 event SetMinDepositAmt(address indexed tokenAddr, uint128 indexed
   ↪ minDepositAmt);

68 event SetFlashLoanPremium(uint16 indexed flashLoanPremium);

72 event SetFundWeight(GovernanceStruct.FundWeight indexed fundWeight);
```

(76, 81)

CVF-170. FIXED

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

Recommendation The type of the ”tokenAddr” parameters should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 22, 27, 32, 37, 43*

```
17      address indexed tokenAddr,
```

```
28      address indexed tokenAddr,
```

```
49  event SetIsStableCoin(address indexed tokenAddr, bool indexed
        ↪ isStableCoin);
```

```
54  event SetPaused(address indexed tokenAddr, bool indexed isPaused);
```

```
59  event SetPriceFeed(address indexed tokenAddr, address indexed
        ↪ priceFeed);
```

```
64  event SetMinDepositAmt(address indexed tokenAddr, uint128 indexed
        ↪ minDepositAmt);
```

CVF-171. FIXED

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

Recommendation The type of the ”priceFeed” parameter should be ”AggregatorV3Interface”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 32*

```
59 event SetPriceFeed(address indexed tokenAddr, address indexed
      ↪ priceFeed);
```

CVF-172. FIXED

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

Recommendation The type of the ”tokenAddr” arguments should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 64, 69, 74, 79, 88*

```
93  function setPaused(address tokenAddr, bool isPaused) external;

98  function setPriceFeed(address tokenAddr, address priceFeed) external
        ↪ ;

103 function setIsStableCoin(address tokenAddr, bool isStableCoin)
        ↪ external;

108 function setMinDepositAmt(address tokenAddr, uint128 minDepositAmt)
        ↪ external;

146 function getValidToken(address tokenAddr)

154 function getTokenId(address tokenAddr) external view returns (uint16
        ↪  tokenId);

169 function getAssetConfig(address tokenAddr)
```

CVF-173. FIXED

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

Recommendation The type of the ”priceFeed” argument should be ”AggregatorV3Interface”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 69*

```
98 function setPriceFeed(address tokenAddr, address priceFeed) external
     ↪ ;
```

CVF-174. FIXED

- **Category** Documentation
- **Source** IGovernance.sol

Description The number format of the ”uint16” values is unclear.

Recommendation Consider documenting.

Client Comment *Added comments.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/flashLoan/IFlashLoanFacet.sol*
- *line: 49-51*
- *path: contracts/zkTrueUp/loan/ILoanFacet.sol*
- *line: 164-165*
- *path: contracts/zkTrueUp/loan/ILoanFacet.sol*
- *line: 184*

```
112 function setFlashLoanPremium(uint16 flashLoanPremium) external;
```

```
121 function setHalfLiquidationThreshold(uint16 halfLiquidationThreshold
    ↪ ) external;
```

```
177 function getLiquidationConfig() external view returns (uint16
    ↪ maxLtvRatio, uint16 halfLiquidationThreshold);
```

CVF-175. FIXED

- **Category** Documentation
- **Source** IGovernance.sol

Description The number format of the returned values is unclear.

Recommendation Consider documenting.

Client Comment *Added comments.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/ILoanFacet.sol*
- *line: 184*
- *path: contracts/zkTrueUp/flashLoan/IFlashLoanFacet.sol*
- *line: 55-56*

```
177 function getLiquidationConfig() external view returns (uint16
    ↪ maxLtvRatio, uint16 halfLiquidationThreshold);

181 function getHalfLiquidationThreshold() external view returns (uint16
    ↪  halfLiquidationThreshold);

193 function getFlashLoanPremium() external view returns (uint16
    ↪ flashLoanPremium);
```

CVF-176. FIXED

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

Recommendation The type of the ”isbTokenAddr” parameters should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/ITsbFacet.sol*
- *line: 27*
- *path: contracts/zkTrueUp/tsb/TsbLib.sol*
- *line: 19, 25*

```
14 event TsbTokenCreated(address indexed tsbTokenAddr, uint16
      ↪ underlyingTokenId, uint32 maturity);

20 event TsbTokenMinted(address indexed tsbTokenAddr, address indexed
      ↪ accountAddr, uint256 amount);

26 event TsbTokenBurned(address indexed tsbTokenAddr, address indexed
      ↪ accountAddr, uint256 amount);
```

CVF-177. INFO

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

Recommendation Events are usually named via nouns, such as ”TsbToken”, ”Mint”, or ”Burn”.

Client Comment *TsbToken will conflict with the TsbToken contract name. TsbTokenMinted, and TsbTokenBurn are used to distinguish mint and burn event in general ERC20 smart contract.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/tsb/ITsbFacet.sol*
- *line: 27*
- *path: contracts/zkTrueUp/tsb/TsbLib.sol*
- *line: 19, 25*

```
14 event TsbTokenCreated(address indexed tsbTokenAddr, uint16
      ↪ underlyingTokenId, uint32 maturity);

20 event TsbTokenMinted(address indexed tsbTokenAddr, address indexed
      ↪ accountAddr, uint256 amount);

26 event TsbTokenBurned(address indexed tsbTokenAddr, address indexed
      ↪ accountAddr, uint256 amount);
```

CVF-178. FIXED

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

Recommendation The return type should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/ITsbFacet.sol*
- *line: 66*

```
39 ) external returns (address tsbTokenAddr);
```

```
87 function getTsbTokenAddr(uint16 underlyingTokenId, uint32 maturity)
       ↪ external view returns (address tsbTokenAddr);
```

CVF-179. FIXED

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

Recommendation The type of the ”tsbTokenAddr” arguments should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/ITsbFacet.sol*
- *line: 72, 79, 89, 94*

```
46    address tsbTokenAddr,
```

```
56    address tsbTokenAddr,
```

```
65 function balanceOf(address account, address tsbTokenAddr) external
   ↪ view returns (uint256);
```

```
75    address tsbTokenAddr
```

```
81 function activeSupply(address tsbTokenAddr) external view returns (
   ↪ uint256);
```

```
92 function getUnderlyingAsset(address tsbTokenAddr) external view
   ↪ returns (address underlyingAsset);
```

```
97 function getMaturityTime(address tsbTokenAddr) external view returns
   ↪  (uint32 maturityTime);
```

CVF-180. FIXED

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

Recommendation The return type should be more specific.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/ITsbFacet.sol*
- *line: 89*

```
92 function getUnderlyingAsset(address tsbTokenAddr) external view
   ↪ returns (address underlyingAsset);
```

CVF-181. FIXED

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

Recommendation The return type should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
25 function tsbFactory() external view returns (address);
```

CVF-182. INFO

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

Recommendation The return type should be more specific.

Client Comment *Removed this function. The underlying asset can be obtain from tokenInfo.*
Diamond:

- *status: no existed*

```
28 function underlyingAsset() external view returns (address);
```

CVF-183. FIXED

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

Recommendation The type of the ”_underlyingAsset” should be more specific.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/interfaces/ITsbToken.sol*
- *line: 31*

```
39 function tokenInfo() external view returns (address _underlyingAsset
   ↪ , uint32 _maturityTime);
```

CVF-184. FIXED

- **Category** Procedural
- **Source** IWETH.sol

Description These functions are common for all ERC20 tokens.

Recommendation Consider inheriting from IERC20 instead of declaring these functions here.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/interfaces/IWETH.sol*
- *line: 12*

```
15 function approve(address guy, uint256 wad) external returns (bool);
```

```
17 function transferFrom(
```

```
23 function balanceOf(address guy) external returns (uint256);
```

CVF-185. FIXED

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

Recommendation The type of this variable should be ”ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
16 address public immutable tsbFactory;
```

CVF-186. FIXED

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

Recommendation The type of this variable should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbToken.sol*
- *line: 24*

```
18 address public immutable underlyingAsset;
```

CVF-187. FIXED

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

Recommendation The type of this argument should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbToken.sol*
- *line: 36*

```
30 address underlyingAsset_,
```

CVF-188. FIXED

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

Recommendation The type of the ”_underlyigAsset” return value should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbToken.sol*
- *line: 75*

```
69 function tokenInfo() external view returns (address _underlyingAsset
   ↪ , uint32 _maturityTime) {
```

CVF-189. FIXED

- **Category** Procedural
- **Source** EvacuationVerifier.sol

Recommendation This library should be moved into a separate file named ”Pairing.sol”.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

```
16 library Pairing {
```

CVF-190. FIXED

- **Category** Procedural
- **Source** EvacuationVerifier.sol

Recommendation By convention, field names start with lower case letters.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

18
```
uint256 X;
uint256 Y;
```

23
```
uint256[2] X;
uint256[2] Y;
```

CVF-191. FIXED

- **Category** Procedural
- **Source** EvacuationVerifier.sol

Recommendation This commented code should be removed.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

-

```
47      /*
        // Changed by Jordi point
        return G2Point(
50          [10857046999023057135944570762232829481370756359578518086990519993285655852781,
            ↪
            11559732032986387107991004021392285783925812861821192530917403151452391805634],
            [84956539231234314176049732474892724384181905872636001487702806493069581019304,
            ↪
            40823678758634336813322034031454355683168513275934012081057410762141200935310*/
            ↪
            );
*/
```

CVF-192. FIXED

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

Recommendation This value should be a named constant.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

```
61  uint256 q =
    ↪ 21888242871839275222246405745257275088696311157297823662689037894645226208583;
    ↪
```

CVF-193. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Description The only part of this contract that is specific to the evacuation feature is the verification key.

Recommendation Consider extracting generic logic from this contract into a separate abstract base contract.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

199
```
contract EvacuationVerifier {
```

CVF-194. FIXED

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

Recommendation This value should be a named constant.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

264
```
uint256 snark_scalar_field =
    ↪ 21888242871839275222246405745257275088548364400416034343698204186
    ↪
575808495617;
```

CVF-195. FIXED

- **Category** Readability
- **Source** EvacuationVerifier.sol

Recommendation Should be ”else return”.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

286
```
return 0;
```

CVF-196. FIXED

- **Category** Suboptimal
- **Source** EvacuationVerifier.sol

Recommendation This could be simplified to: return verify(inputValues, proof) == 0;

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: fixed*

304
```
if (verify(inputValues, proof) == 0) {
    return true;
} else {
    return false;
}
```

CVF-197. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description The functions ”_setupRole” and ”_grantRole” do exactly the same.

Recommendation Consider using the same function for both roles.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/zkTrueUpInit.sol*
- *line: 56-61*

```
55 _setupRole(DEFAULT_ADMIN_ROLE, adminAddr);
   _grantRole(Config.OPERATOR_ROLE, operatorAddr);
```

CVF-198. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description The expression ”GovernanceStorage.getStorage()” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/zkTrueUpInit.sol*
- *line: 81-90, 93-108, 123-136*

```
58  GovernanceStorage.getStorage().healthFactorThreshold = Config.
       ↪ HEALTH_FACTOR_THRESHOLD;

65  GovernanceStorage.getStorage().fundWeight = initFundWeight;

68  GovernanceStorage.getStorage().halfLiquidationThreshold =
       ↪ InitialConfig.INIT_HALF_LIQUIDATION_THRESHOLD;

71  GovernanceStorage.getStorage().flashLoanPremium = InitialConfig.
       ↪ INIT_FLASH_LOAN_PREMIUM;

79  GovernanceStorage.getStorage().liquidationFactor =
       ↪ initLiquidationFactor;

88  GovernanceStorage.getStorage().stableCoinPairLiquidationFactor =
       ↪ initStableCoinPairLiquidationFactor;

91  GovernanceStorage.getStorage().treasuryAddr = treasuryAddr;

93  GovernanceStorage.getStorage().insuranceAddr = insuranceAddr;

95  GovernanceStorage.getStorage().vaultAddr = vaultAddr;

99  GovernanceStorage.getStorage().tokenNum = newTokenId;
100 GovernanceStorage.getStorage().toTokenId[Config.ETH_ADDRESS] =
       ↪ newTokenId;
    GovernanceStorage.getStorage().assetConfigs[newTokenId] =
       ↪ GovernanceStruct.AssetConfig({

112     GovernanceStorage.getStorage().assetConfigs[newTokenId]
```

CVF-199. FIXED

- **Category** Documentation
- **Source** Governance.sol

Recommendation Consider explaining why overflow is impossible here.

Client Comment *Added comments.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/zkTrueUpInit.sol*
- *line: 124*

```
98 uint16 newTokenId = getTokenNum() + 1;
```

CVF-200. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description Here a value just written into the storage, is read back from the storage.

Recommendation Consider reusing the written value.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/zkTrueUpInit.sol*
- *line: 129*

```
112 GovernanceStorage.getStorage().assetConfigs[newTokenId]
```

CVF-201. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description The expression ”GovernanceStorage.getStorage()” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/zkTrueUpInit.sol*
- *line: 126, 128, 129*

```
124 GovernanceStorage.getStorage().tokenNum = newTokenId;
    GovernanceStorage.getStorage().toTokenId[tokenAddr] = newTokenId;
    GovernanceStorage.getStorage().assetConfigs[newTokenId] =
       ↪ assetConfig;
```

CVF-202. FIXED

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

Recommendation The type of the ”tokenAddr” arguments should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/TokenFacet.sol*
- *line: 51, 61, 72, 82, 101, 115*
- *path: contracts/zkTrueUp/token/TokenLib.sol*
- *line: 50, 65, 86, 96, 104*

```
138 function setPaused(address tokenAddr, bool isPaused) external
    ↪ onlyRole(DEFAULT_ADMIN_ROLE) {

147 function setPriceFeed(address tokenAddr, address priceFeed) external
    ↪  onlyRole(DEFAULT_ADMIN_ROLE) {

157 function setIsStableCoin(address tokenAddr, bool isStableCoin)
    ↪ external onlyRole(DEFAULT_ADMIN_ROLE) {

166 function setMinDepositAmt(address tokenAddr, uint128 minDepositAmt)
    ↪ external onlyRole(DEFAULT_ADMIN_ROLE) {

246 function getValidToken(address tokenAddr)

268 function getAssetConfig(address tokenAddr)

346 function getTokenId(address tokenAddr) public view returns (uint16
    ↪ tokenId) {

367 function _getValidTokenId(address tokenAddr) internal view returns (
    ↪ uint16 tokenId) {
```

CVF-203. INFO

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

Description These events are emitted even if nothing actually changed.

Client Comment *They are admin function and no critical effect although no change. Diamond:*

- *status: info*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 22, 27, 32, 37*
- *path: contracts/zkTrueUp/protocolParams/IProtocolParamsFacet.sol*
- *line: 25, 29, 33, 37*
- *path: contracts/zkTrueUp/loan/ILoanFacet.sol*
- *line: 115, 120*

```
141 emit SetPaused(tokenAddr, isPaused);
151 emit SetPriceFeed(tokenAddr, priceFeed);
160 emit SetIsStableCoin(tokenAddr, isStableCoin);
169 emit SetMinDepositAmt(tokenAddr, minDepositAmt);
176 emit SetFlashLoanPremium(flashLoanPremium);
185 emit SetFundWeight(fundWeight);
194 emit SetHalfLiquidationThreshold(halfLiquidationThreshold);
212 emit SetLiquidationFactor(liquidationFactor, isStableCoinPair);
220 emit SetTreasuryAddr(treasuryAddr);
228 emit SetInsuranceAddr(insuranceAddr);
236 emit SetVaultAddr(vaultAddr);
```

CVF-204. FIXED

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

Recommendation The type of the ”priceFeed” argument should be ”AggregatorV3Interface”.

Client Comment *Removed this as it is a constant value in the config file.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/token/ITokenFacet.sol*
- *line: 69*

```
147 function setPriceFeed(address tokenAddr, address priceFeed) external
      ↪  onlyRole(DEFAULT_ADMIN_ROLE) {
```

CVF-205. FIXED

- **Category** Suboptimal
- **Source** Governance.sol

Description This is basically a constant value.

Recommendation Consider not returning it.

Client Comment *Removed this as it is a constant value in the config file.*
Diamond:

- *status: fixed*

```
278 /// @return maxLtvRatio The maximum LTV ratio
```

CVF-206. FIXED

- **Category** Procedural
- **Source** Governance.sol

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

Client Comment *Added comments.*
Diamond:

- *status: no existed*

```
373 function _authorizeUpgrade(address) internal view override onlyRole(
      ↪ DEFAULT_ADMIN_ROLE) {}
```

CVF-207. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *Wrapped the data to struct VerifyingBlock.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 66*

```
68 function verifyBlocks(RolluperStruct.StoredBlock[] memory
     ↪ committedBlocks, RolluperStruct.Proof[] memory proof)
```

CVF-208. FIXED

- **Category** Procedural
- **Source** Rolluper.sol

Recommendation This check should be done before the loop, as the final value of the ”verifiedBlockNum” variable could be calculated in advance.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 648*

```
79 if (verifiedBlockNum > Utils.getCommittedBlockNum())
```

CVF-209. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The expression ”pendingBlocks[i]” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 687, 694, 696, 697*

```
90 _executeOneBlock(pendingBlocks[i], i);
ZkTrueUpStorage.getStorage().executedL1RequestNum += pendingBlocks[i
   ↪ ].storedBlock.l1RequestNum;
emit BlockExecuted(pendingBlocks[i].storedBlock.blockNumber);
```

CVF-210. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description The expression ”pendingBlocks[i].storedBlock” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 696, 697*

```
91 ZkTrueUpStorage.getStorage().executedL1RequestNum += pendingBlocks[i
       ↪ ].storedBlock.l1RequestNum;
   emit BlockExecuted(pendingBlocks[i].storedBlock.blockNumber);
```

CVF-211. FIXED

- **Category** Procedural
- **Source** Rolluper.sol

Recommendation This check should be done before the loop, as the final value of the ”executedBlockNum” variable could be calculated in advance.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 682*

```
95 if (executedBlockNum > Utils.getVerifiedBlockNum())
```

CVF-212. FIXED

- **Category** Documentation
- **Source** Rolluper.sol

Description These returned values are not documented.

Recommendation Consider documenting.

Client Comment *Added comments.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 872*

```
175 ) internal view returns (RolluperStruct.StoredBlock memory
      ↪ storedNewBlock) {
```

```
206         bytes32 processableRollupTxHash,
            uint64 processedL1RequestNum,
            bytes memory commitmentOffset
```

```
291 ) internal pure returns (bytes32 commitment) {
```

CVF-213. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Recommendation Consider replacing this array with a bit mask.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 451*

```
208 bytes memory commitmentOffset
```

CVF-214. FIXED

- **Category** Procedural
- **Source** Rolluper.sol

Recommendation Variable name usually starts with a lower case letter.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 599*

```
237 Operations.CreateTsbToken memory CreateTsbTokenReq = Operations.
    ↪ readCreateTsbTokenPubData(rollupData);
```

CVF-215. FIXED

- **Category** Suboptimal
- **Source** Rolluper.sol

Description Here ”pubdta” is packed only to be immediately repacked into a larger array.

Recommendation Consider packing all at once.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: function 873*

```
292 bytes memory pubData = abi.encodePacked(commitmentOffset, newBlock.
    ↪ publicData);
```

```
295     abi.encodePacked(
            previousBlock.stateRoot,
            newBlock.newStateRoot,
            newBlock.newTsRoot,
            newBlock.timestamp,
300         pubData
        )
```

CVF-216. FIXED

- **Category** Procedural
- **Source** Rolluper.sol

Description These functions are very similar.

Recommendation Consider merging them into one.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 222, 656, 671*

```
308 function _verifyOneBlock(RolluperStruct.StoredBlock memory
    ↪ committedBlock, RolluperStruct.Proof memory proof)
```

```
322 function _verifyEvacuationBlock(bytes32 commitment, RolluperStruct.
    ↪ Proof memory proof) internal view {
```

CVF-217. FIXED

- **Category** Readability
- **Source** Rolluper.sol

Recommendation Consider explicitly requiring that the commitment is a field element.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 667*

```
312 if (proof.commitment[0] & Config.INPUT_MASK != uint256(
    ↪ committedBlock.commitment) & Config.INPUT_MASK)
```

```
323 if (proof.commitment[0] & Config.INPUT_MASK != uint256(commitment) &
    ↪  Config.INPUT_MASK)
```

CVF-218. FIXED

- **Category** Procedural
- **Source** Rolluper.sol

Recommendation This check should be moved to the calling code to avoid reading multiple times the current number of executed blocks.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/rollup/RollupFacet.sol*
- *line: 691*

```
337 if (executeBlock.storedBlock.blockNumber != Utils.
    ↪ getExecutedBlockNum() + blockNum + 1)
     revert RolluperError.InvalidExecutedBlockNum(executeBlock.
        ↪ storedBlock.blockNumber);
```

CVF-219. FIXED

- **Category** Suboptimal
- **Source** TsbFactory.sol

Description The functions "_setupRole" and "_grantRole" basically do the same.

Recommendation Consider using once of these two functions.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
62 _setupRole(DEFAULT_ADMIN_ROLE, adminAddr);
_grantRole(OPERATOR_ROLE, operatorAddr);
_grantRole(ZKTRUEUP_ROLE, zkTrueUpAddr);
```

CVF-220. FIXED

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

Recommendation The return type should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
83 ) external virtual onlyRole(OPERATOR_ROLE) returns (address
      ↪ tsbTokenAddr) {
```

CVF-221. INFO

- **Category** Suboptimal
- **Source** TsbFactory.sol

Description Deploying a separate contract for each token is suboptimal.

Recommendation Consider deploying a minimal proxy for a shared implementation as described in ERC-1167 (https://eips.ethereum.org/EIPS/eip-1167).

Client Comment *Based on our business logic, deploying contracts separately could be more efficient due to the following reasons: 1. The contracts are relatively simple, meaning that cost reduction would be minimal. Direct deployment can reduce gas fees for future calls. 2. Unlike the Uniswap Pair contract, which needs to manage an unlimited number of cloned contracts, we maintain a limited number of child contracts.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 57*

```
90 tsbTokenAddr = address(new TsbToken(name, symbol,
      ↪ underlyingAssetAddr, maturityTime));
```

CVF-222. FIXED

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

Recommendation The type of the ”tsbTokenAddr” arguments should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 110, 118, 125, 132, 139, 147*

```
102     address tsbTokenAddr,
```

```
116     address tsbTokenAddr,
```

```
130 function balanceOf(address account, address tsbTokenAddr) external
    ↪ view returns (uint256 balance) {
```

```
142     address tsbTokenAddr
```

```
150 function activeSupply(address tsbTokenAddr) external view returns (
    ↪ uint256 totalSupply) {
```

```
157 function getUnderlyingAsset(address tsbTokenAddr) external view
    ↪ returns (address underlyingAsset) {
```

```
164 function getMaturityTime(address tsbTokenAddr) external view returns
    ↪  (uint32 maturityTime) {
```

CVF-223. INFO

- **Category** Suboptimal
- **Source** TsbFactory.sol

Description These events duplicate events emitted by TSB tokens.

Recommendation Consider not emitting them.

Client Comment *It make our server can subscribe this factory contract only instead of each token contract.*
Diamond:

- *status: info*
- *path: contracts/zkTrueUp/tsb/TsbLib.sol*
- *line: 42, 52*

```
107 emit TsbTokenMinted(tsbTokenAddr, to, amount);
```

```
121 emit TsbTokenBurned(tsbTokenAddr, from, amount);
```

CVF-224. FIXED

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

Recommendation The return type should be ”IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 139*

```
157 function getUnderlyingAsset(address tsbTokenAddr) external view
    ↪ returns (address underlyingAsset) {
```

CVF-225. FIXED

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

Recommendation The return type should be ”ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 110*

```
174 function getTsbTokenAddr(uint16 underlyingTokenId, uint32 maturity)
      ↪ public view returns (address tsbTokenAddr) {
```

CVF-226. FIXED

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

Recommendation The return type should be ”IGovernance”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
188 function _getGovernanceAddr() internal view returns (address
      ↪ governanceAddr) {
```

CVF-227. FIXED

- **Category** Procedural
- **Source** TsbFactory.sol

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

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
200 function _authorizeUpgrade(address) internal view override onlyRole(
        ↪ DEFAULT_ADMIN_ROLE) {}
```

CVF-228. FIXED

- **Category** Procedural
- **Source** Verifier.sol

Recommendation This library should be moved into a separate file named ”Pairing.sol”.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
16 library Pairing {
```

CVF-229. FIXED

- **Category** Procedural
- **Source** Verifier.sol

Recommendation By convention, field names start with lower case letters.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
18 uint256 X;
   uint256 Y;
```

CVF-230. FIXED

- **Category** Procedural
- **Source** Verifier.sol

Recommendation This commented code should be removed.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
47      /*
       // Changed by Jordi point
        return G2Point(
50          [10857046999023057135944570762232829481370756359578518086990
               ↪
            19993285655852781,
             11559732032986387107991004021392285783925812861821192530917
                ↪
             03151452391805634],
            [84956539231234314176049732474892724384181905872636001487702
               ↪
            0649306958101930,
            40823678758634336813322034031454355683168513275934012081057
               ↪
            076214120093531]
            );
*/
```

CVF-231. FIXED

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

Recommendation This value should be a named constant.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
61  uint256 q =
       ↪ 21888242871839275222246405745257275088696311157297823662689037
       ↪
    94645226208583;
```

CVF-232. FIXED

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

Recommendation This value should be a named constant.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
264 uint256 snark_scalar_field =
    ↪ 21888242871839275222246405745257275088548364400416034343698204186575808495617;
```

CVF-233. FIXED

- **Category** Readability
- **Source** Verifier.sol

Recommendation Should be ”else return”.

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

```
286 return 0;
```

CVF-234. FIXED

- **Category** Suboptimal
- **Source** Verifier.sol

Recommendation ”This could be simplified to: return verify(inputValues, proof) == 0;”

Client Comment *Switched to snarkJS version 0.7.0; this resolved the issue.*
Diamond:

- *status: no existed*

304
```
if (verify(inputValues, proof) == 0) {
    return true;
} else {
    return false;
}
```

CVF-235. FIXED

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

Recommendation The argument type should be ”IZkTrueUp”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

19
```
constructor(address payable _zkTrueUp) {
```

CVF-236. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description The functions “_setupRole” and “_grantRole” basically do the same.

Recommendation Consider using once of these functions.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/ZkTrueUpInit.sol*
- *line: 57-61*

```
77 _setupRole(DEFAULT_ADMIN_ROLE, adminAddr);
   _grantRole(Config.COMMITTER_ROLE, operatorAddr);
   _grantRole(Config.VERIFIER_ROLE, operatorAddr);
80 _grantRole(Config.EXECUTER_ROLE, operatorAddr);
```

CVF-237. FIXED

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Description The expression “ZkTrueUpStorage.getStorage()” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/initializer/ZkTrueUpInit.sol*
- *line: 66, 70-74, 120*

```
82 ZkTrueUpStorage.getStorage().governanceAddr = governanceAddr;
   ZkTrueUpStorage.getStorage().wETHAddr = wETHAddr;
   ZkTrueUpStorage.getStorage().poseidonUnit2Addr = poseidonUnit2Addr;
   ZkTrueUpStorage.getStorage().verifierAddr = verifierAddr;
   ZkTrueUpStorage.getStorage().evacuationVerifierAddr =
      ↪ evacuationVerifierAddr;
   ZkTrueUpStorage.getStorage().rolluperAddr = rolluperAddr;

89 ZkTrueUpStorage.getStorage().accountNum = Config.
      ↪ NUM_RESERVED_ACCOUNTS;

98 ZkTrueUpStorage.getStorage().storedBlockHashes[0] = keccak256(abi.
      ↪ encode(genesisBlock));
```

CVF-238. FIXED

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

Recommendation The type of the “tokenAddr” arguments should be “IERC20”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/account/AccountFacet.sol*
- *line: 45, 64, 86, 109*

```
110     address tokenAddr,
```

```
130     address tokenAddr,
```

```
144 function withdraw(address tokenAddr, uint128 amount) external
    ↪ virtual nonReentrant {
```

```
158 function forceWithdraw(address tokenAddr) external {
```

```
471 function getPendingBalances(address accountAddr, address tokenAddr)
    ↪ external view returns (uint128 pendingBalance) {
```

```
596     address tokenAddr,
```

```
614     address tokenAddr,
```

```
634     address tokenAddr,
```

```
915 function _getValidToken(address tokenAddr)
```

CVF-239. FIXED

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

Recommendation The type of this argument should be “ITsbToken”.

Client Comment *Resolved.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/tsb/TsbFacet.sol*
- *line: 73*

```
195 address tsbTokenAddr,
```

CVF-240. INFO

- **Category** Suboptimal
- **Source** ZkTrueUp.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *We adopted the practice of separate array usage from other protocols like AAVE to enhance simplicity and user convenience.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/flashLoan/FlashLoanFacet.sol*
- *line: 38, 39*

```
345 address[] calldata assets,
    uint128[] calldata amounts,
```

CVF-241. FIXED

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

Recommendation The argument type should be “ITsbFactory”.

Client Comment *Resolved.*
Diamond:

- *status: no existed*

```
373 function setTsbFactory(address tsbFactoryAddr) external onlyRole(
      ↪ DEFAULT_ADMIN_ROLE) {
```

CVF-242. FIXED

- **Category** Procedural
- **Source** ZkTrueUp.sol

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

Client Comment *Resolved.*
Diamond:

- *status: no exsited*

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

CVF-243. FIXED

- **Category** Overflow/Underflow
- **Source** ZkTrueUp.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, but some intermediary calculation overflows.

Recommendation Consider using the “mulDiv” function.

Client Comment *Switched to ’mulDiv’.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanFacet.sol*
- *line: 374, 375, 394, 409, 421*

```
698 repayAmt = (normalizedCollateralPrice * loan.collateralAmt) / 10**
      ↪ collateralAsset.decimals <
```

```
704 uint256 normalizedRepayValue = (normalizedDebtPrice * repayAmt) /
      ↪ 10**debtAsset.decimals;
```

```
712 uint256 repayToCollateralRatio = (Config.LTV_BASE *
      ↪ normalizedRepayValue * 10**collateralAsset.decimals) /
       normalizedCollateralPrice /
       loan.collateralAmt;
```

```
731     ((maxLtvRatio + liquidationFactor.liquidatorIncentive) *
            normalizedRepayValue *
            10**collateralAsset.decimals) /
            Config.LTV_BASE /
            normalizedCollateralPrice
```

```
752     (liquidationFactor.protocolPenalty * normalizedRepayValue * 10**
          ↪ collateralAsset.decimals) /
            Config.LTV_BASE /
            normalizedCollateralPrice
```

CVF-244. FIXED

- **Category** Overflow/Underflow
- **Source** ZkTrueUp.sol

Description Phantom overflow is possible here.

Recommendation Consider using the “mulDiv” function.

Client Comment *Switched to ’mulDiv’.*
Diamond:

- *status: fixed*
- *path: contracts/zkTrueUp/loan/LoanLib.sol*
- *line: 107, 111, 112*

```
859 (ltvThreshold * normalizedCollateralPrice * loan.collateralAmt *
       ↪ 10**debtAsset.decimals) /
860 (normalizedDebtPrice * loan.debtAmt) /
    10**collateralAsset.decimals;
```

CVF-245. FIXED

- **Category** Procedural
- **Source** ZkTrueUp.sol

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

Client Comment *Added comments.*
Diamond:

- *status: no existed*

```
927 function _authorizeUpgrade(address) internal view override onlyRole(
       ↪ DEFAULT_ADMIN_ROLE) {}
```

CVF-246. FIXED

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

Description This function delegates to a “rolluper”, not “rollup”.

Recommendation Consider renaming.

Client Comment *Renamed to deledateCall and added an address parameter. Moved FlashLoan and Rolluper to the other contracts for delegate call.*
Diamond:

- *status: no existed*

```
931 function _delegateRollup() internal {
```

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

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting