Report

v. 1.0

Customer

zkLink

Circuits audit

zkLink Protocol Phase II

9th August 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog **3**

2 Introduction **4**

3 Project scope **5**

4 Methodology **6**

5 Our findings **7**

6 Moderate Issues **8**
CVF-1. INFO 8

7 Minor Issues **9**
CVF-2. INFO 9
CVF-3. INFO 9
CVF-4. INFO 9
CVF-5. INFO 10
CVF-6. FIXED 10
CVF-7. FIXED 10
CVF-8. FIXED 10
CVF-9. FIXED 11
CVF-10. FIXED 11
CVF-11. FIXED 11
CVF-12. FIXED 11
CVF-13. FIXED 12
CVF-14. INFO 12
CVF-15. INFO 12
CVF-16. FIXED 13
CVF-17. INFO 13

1 Changelog

#	Date	Author	Description
0.1	09.08.23	A. Zveryanskaya	Initial Draft
0.2	09.08.23	A. Zveryanskaya	Minor revision
1.0	09.08.23	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.

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

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	account.rs	allocated_structures.rs	circuit.rs
	element.rs	exit_circuit.rs	operation.rs
	serialization.rs		
op_circuit/			
	change_pubkey _offchain.rs	deposit.rs	forced_exit.rs
	full_exit.rs	order_matching.rs	transfer_to_new.rs
	transfer.rs	withdraw.rs	
witness/			
	change_pubkey _offchain.rs	deposit.rs	forced_exit.rs
	full_exit.rs	mod.rs	noop.rs
	order_matching.rs	transfer_to_new.rs	transfer.rs
	utils.rs	withdraw.rs	

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 1 Moderate, and a few less important issues.

	Active	Fixed
Moderate	1	0
Minor	7	9

Fixed 9 out of 17 issues

6 Moderate Issues

CVF-1. INFO

- **Category** Documentation
- **Source** withdraw.rs

Description It is unclear why this check was removed.

Recommendation Consider explaining a balance cannot overflow during update.

Client Comment *This restriction is redundant, because the CircuitElement::conditionally_select_with_number_strict function does the length bits constraint every time the balance is updated.*

```
192 -cur.balance
193 -    .enforce_length(cs.namespace(|| "mutated balance is still
    ↪ correct length"))?;
```

7 Minor Issues

CVF-2. INFO

- **Category** Documentation
- **Source** utils.rs

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
261 +) -> (Vec<Option<Fr>>, Vec<Option<Fr>>, Vec<Option<Fr>>, Vec<Option
    ↪ <Fr>>) {
```

CVF-3. INFO

- **Category** Documentation
- **Source** utils.rs

Description The semantics of the first argument of “Fb” is unclear.

Recommendation Consider documenting.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
304 +pub fn apply_leaf_operation<Fa: Fn(&mut CircuitAccount<Bn256>), Fb:
    ↪  Fn(&mut Fr, &mut Balance<Bn256>), Fc: Fn(&mut
    ↪ CircuitTidyOrder<Bn256>)>(
```

CVF-4. INFO

- **Category** Documentation
- **Source** utils.rs

Description The semantics of this additional returned value is unclear.

Recommendation Consider documenting.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
314 +Fr, Fr,
```

CVF-5. INFO

- **Category** Documentation
- **Source** utils.rs

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
370 +) -> (AccountWitness<Bn256>, Fr, Fr, CircuitTidyOrder<Bn256>) {
```

CVF-6. FIXED

- **Category** Suboptimal
- **Source** utils.rs

Recommendation Should be a: [0, 1, 1, 0, 0], b: [0, 1, 1, 1, 1], c: [0, 1, 0, 1, 1].

```
593 +/// block ops composition a: [0,1,1,0,0], support ops composition a
    ↪ : [0,1,1,1,1], support ops composition b: [0,1,0,1,1]
594 +/// Here a is a subset of b, but not a subset of c
```

CVF-7. FIXED

- **Category** Suboptimal
- **Source** utils.rs

Recommendation It would be more correct to do it as: “num & EXEC_ALL_OPS_COMPOSITION_NUMBER == num”. Just in case “EXEC_ALL_OPS_COMPOSITION_NUMBER” would have gaps between bits.

```
613 +assert!(num <= EXEC_ALL_OPS_COMPOSITION_NUMBER);
```

CVF-8. FIXED

- **Category** Documentation
- **Source** forced_exit.rs

Description It is not clear what the role of ‘USD_TOKEN_ID‘ is here.

Recommendation Consider documenting.

```
281 +token: Some(Fr::from_u64(USD_TOKEN_ID as u64)), // Not actually
    ↪ involved in circuit operation, just to open the account
    ↪ commitment
```

CVF-9. FIXED

- **Category** Bad datatype
- **Source** withdraw.rs

Recommendation 8 should be a named constant.

```
117 +8,
```

CVF-10. FIXED

- **Category** Bad datatype
- **Source** order_matching.rs

Recommendation 8 should be a named constant.

```
42 +pubdata_bits.extend(op_data[OrderMatchingArgs::MakerIsSell].clone()
   ↪ .into_padded_be_bits(8));
```

CVF-11. FIXED

- **Category** Documentation
- **Source** account.rs

Description It is unclear how many bits are used from each word.

Recommendation Consider documenting.

```
42 +pub high: Option<E::Fr>,
43 +pub low: Option<E::Fr>
```

CVF-12. FIXED

- **Category** Bad datatype
- **Source** account.rs

Recommendation This value should be a separate named constant.

```
113 +LAYER1_ADDR_BIT_WIDTH / 2,
```

```
119 +LAYER1_ADDR_BIT_WIDTH / 2,
```

CVF-13. FIXED

- **Category** Bad datatype
- **Source** account.rs

Recommendation This variable should have type ‘AllocatedAddress’

Client Comment *Fixed. If you pass in AllocatedAddress directly, you need to clone the two Args, which has some performance loss, so I added AsRef to make it possible to have both reference and ownership.*

```
142 +    updated_address: (&CircuitElement<E>, &CircuitElement<E>),
```

```
164 +) -> Result<Boolean, SynthesisError> {
```

CVF-14. INFO

- **Category** Suboptimal
- **Source** allocated_structures.rs

Description Defining several variables with the same name makes code harder to read.

Recommendation Consider refactoring.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
196 +let sub_account_root = allocate_merkle_root(
```

```
204 +let sub_account_root = CircuitElement::from_number(cs.namespace(||
        ↪ "order_subtree_root_ce"), sub_account_root)?;
205 +let sub_account_root = calc_account_state_tree_root(
```

CVF-15. INFO

- **Category** Suboptimal
- **Source** circuit.rs

Description This value, which is actually a “multi_and” output, is later used as a “multi_and” argument. This is suboptimal.

Recommendation Consider refactoring to avoid nested “multi_and” components.

Client Comment *The constraints are the same for two multi_and’s and one multi_and’s, and to ensure the completeness of the equals function, I don’t think it’s necessary here.*

```
1089 +let is_address_zero = branch.account.address.equals(
```

CVF-16. FIXED

- **Category** Bad naming
- **Source** element.rs

Description The variable name is confusing as actually it's value is "x > y".

Recommendation Consider renaming.

```
398 +let is_lt = Self::less_than_fixed(
```

CVF-17. INFO

- **Category** Documentation
- **Source** exit_circuit.rs

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

Client Comment *This part of the code has been refactored in our new version of the derivative branch, and this lack of clarity will no longer occur in the next audit.*

```
239 +) -> (Vec<Vec<AccountWitness<Bn256>>>,(Vec<Vec<Fr>>,(Vec<Vec<Fr>>,
    ↪ Vec<Vec<CircuitTidyOrder<Bn256>>>)){
```

```
276 +) -> (Vec<Vec<Vec<Option<Fr>>>>,(Vec<Vec<Vec<Option<Fr>>>>,(Vec<Vec
    ↪ <Vec<Option<Fr>>>>,Vec<Vec<Vec<Option<Fr>>>>)){
```

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