ABDK CONSULTING

SMART CONTRACT AUDIT

ZkSwap

Rust And Solidity v.2

abdk.consulting

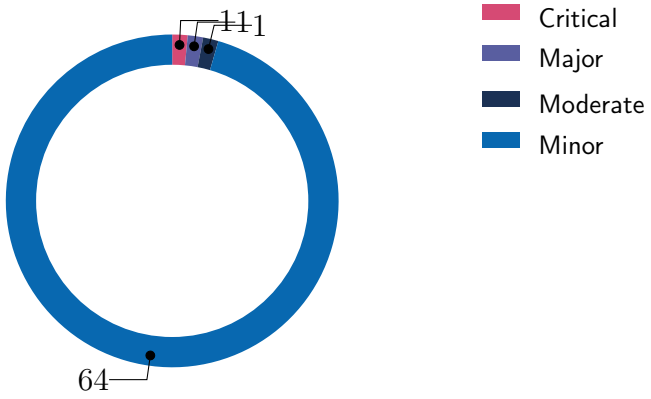
SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
1st July 2021

This is a report on the security of certain smart contracts and circuits that comprise the ZkSwap protocol. Previous reports covered:

1. Some Solidity smart contracts for v.0.9;
2. The same Solidity smart contracts for v.0.9.8;
3. Some circuit files for v.0.9 (part 1, part 2, part 3);
4. The same circuit files for v.0.9.8;
5. The remaining circuits and contracts for v.0.9.8;
6. All contracts and circuits updated for v.1.0.3

This report covers the smart contracts and circuits updated for the version v.2.0. We have reported a number of issues to the authors, and all the major ones were fixed in subsequent releases.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Bad naming	Info
CVF-2	Minor	Bad naming	Info
CVF-3	Minor	Bad naming	Info
CVF-4	Minor	Procedural	Info
CVF-5	Minor	Procedural	Info
CVF-6	Minor	Unclear behavior	Info
CVF-7	Minor	Documentation	Info
CVF-8	Minor	Procedural	Info
CVF-9	Minor	Bad naming	Info
CVF-10	Minor	Procedural	Info
CVF-11	Minor	Bad naming	Info
CVF-12	Critical	Flaw	Fixed
CVF-13	Minor	Unclear behavior	Info
CVF-14	Minor	Procedural	Info
CVF-15	Minor	Flaw	Info
CVF-16	Minor	Procedural	Info
CVF-17	Minor	Procedural	Info
CVF-18	Minor	Documentation	Info
CVF-19	Minor	Suboptimal	Info
CVF-20	Minor	Suboptimal	Info
CVF-21	Minor	Procedural	Info
CVF-22	Minor	Bad datatype	Info
CVF-23	Minor	Documentation	Info
CVF-24	Moderate	Suboptimal	Info
CVF-25	Minor	Bad naming	Info
CVF-26	Minor	Documentation	Info
CVF-27	Minor	Bad naming	Info

ID	Severity	Category	Status
CVF-28	Minor	Suboptimal	Info
CVF-29	Minor	Unclear behavior	Info
CVF-30	Minor	Flaw	Info
CVF-31	Minor	Documentation	Info
CVF-32	Minor	Bad datatype	Info
CVF-33	Minor	Suboptimal	Info
CVF-34	Minor	Suboptimal	Info
CVF-35	Minor	Readability	Info
CVF-36	Minor	Documentation	Info
CVF-37	Minor	Procedural	Info
CVF-38	Minor	Procedural	Info
CVF-39	Minor	Procedural	Info
CVF-40	Minor	Procedural	Info
CVF-41	Minor	Bad naming	Info
CVF-42	Minor	Procedural	Info
CVF-43	Minor	Unclear behavior	Info
CVF-44	Minor	Bad naming	Info
CVF-45	Minor	Suboptimal	Info
CVF-46	Minor	Suboptimal	Info
CVF-47	Minor	Suboptimal	Info
CVF-48	Minor	Suboptimal	Info
CVF-49	Minor	Bad naming	Info
CVF-50	Minor	Suboptimal	Info
CVF-51	Minor	Suboptimal	Info
CVF-52	Minor	Suboptimal	Info
CVF-53	Minor	Procedural	Info
CVF-54	Minor	Suboptimal	Info
CVF-55	Minor	Procedural	Info
CVF-56	Minor	Procedural	Info
CVF-57	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-58	Minor	Unclear behavior	Info
CVF-59	Major	Flaw	Fixed
CVF-60	Minor	Procedural	Info
CVF-61	Minor	Readability	Info
CVF-62	Minor	Readability	Info
CVF-63	Minor	Suboptimal	Info
CVF-64	Minor	Bad naming	Info
CVF-65	Minor	Suboptimal	Info
CVF-66	Minor	Procedural	Info
CVF-67	Minor	Procedural	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	June 18, 2021	D. Khovratovich	Initial Draft
0.2	June 18, 2021	D. Khovratovich	Minor revision
1.0	June 19, 2021	D. Khovratovich	Release
1.1	June 30, 2021	D. Khovratovich	Add client comment
2.0	July 1, 2021	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting 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.

We were given access to the ZkSwap repo, commit 72ffa2. These files are written in Rust language:

- allocated_structures.rs;
- circuit.rs;
- exit_circuit.rs;
- lp_exit_circuit.rs;
- operation.rs;
- witness/add_liquidity.rs;
- witness/change_pubkey_offchain.rs;
- witness/close_account.rs;
- witness/create_pair.rs;
- witness/deposit.rs;
- witness/full_exit.rs;
- witness/noop.rs;
- witness/remove_liquidity.rs;
- witness/swap.rs;
- witness/tests/mod.rs;
- witness/transfer.rs;
- witness/transfer_to_new.rs;
- witness/withdraw.rs

The other files are written in Solidity language:

- Config.sol;
- DeployFactory.sol;
- Events.sol;
- Governance.sol;

- Operations.sol;
- PairTokenManager.sol;
- PlonkAggCore.sol;
- Storage.sol;
- Verifier.sol;
- ZkSync.sol;
- ZkSyncCommitBlock.sol.

2.1 About ABDK

ABDK Consulting, 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.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best 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 and that 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 is 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 that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check 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.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** operation.rs

Recommendation Consider renaming "balance" to "balance1" for consistency with "balance2".
Client Comment Will not fix.

Listing 1:

```
18  pub balance_value: Option<E::Fr>,
    pub balance_subtree_path: Vec<Option<E::Fr>>,

23  pub balance2_value: Option<E::Fr>,
    pub balance2_subtree_path: Vec<Option<E::Fr>>,
```

3.2 CVF-2

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** operation.rs

Recommendation Consider renaming "token" to "token1" for consistency with "token2".
Client Comment Will not fix.

Listing 2:

```
31  pub token: Option<E::Fr>,
    pub token2: Option<E::Fr>,
```

3.3 CVF-3

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** operation.rs

Description The fields are named "token0" and "token1", while comment says "tokenA" and "tokenB". This is very confusing, especially taking into account that in other places there are "token" and "token2".
Recommendation Consider using consistent naming across the code.
Client Comment Will not fix.

Listing 3:

```
62  pub token0: Option<E::Fr>,    // tokenA
    pub token1: Option<E::Fr>,    // tokenB
```

3.4 CVF-4

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** lp_exit_circuit.rs

Recommendation Consider making this argument 'Optional' to avoid dummy values.
Client Comment Will not fix.

Listing 4:

```
83 &mut dummy_root,
```

```
91 &mut dummy_root,
```

```
99 &mut dummy_root,
```

3.5 CVF-5

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** exit_circuit.rs

Recommendation Consider making this parameter 'Optional' to avoid dummy values.
Client Comment Will not fix.

Listing 5:

```
69 &mut dummy_root,
```

3.6 CVF-6

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** swap.rs

Description Conversion through string looks weird.
Recommendation Consider implementing an utility function for this.
Client Comment Will not fix.

Listing 6:

```
83 let fee_fe = Fr::from_str(&data.fee.to_string()).unwrap();
```

3.7 CVF-7

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** swap.rs

Recommendation Should be "floating point".
Client Comment Will not fix.

Listing 7:

```
85 // float point repr of amount_b_min
```

3.8 CVF-8

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** swap.rs

Recommendation There should be an assert that 'fee_token' is either token A or token B, and the circuit verifies this.
Client Comment Will not fix.

Listing 8:

```
97 ** fee_a = if token_a == fee_token { fee } else { 0 }
   ** fee_b = if token_b == fee_token { fee } else { 0 }
```

3.9 CVF-9

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** swap.rs

Description The "_fe" suffix in "amount_b_fe" is confusing, as it looks like B token treatment is different from the A token treatment.
Recommendation Remove the suffix.
Client Comment Will not fix.

Listing 9:

```
166 ** pair_account.reserveB = args.b1 - amount_b_fe
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** swap.rs

Recommendation The number here should not be hardcoded, but should refer to named constants. Otherwise, changing the constants would make the comment incorrect.
Client Comment Will not fix.

Listing 10:

```
310 // TODO: add assertion that amount_b = rb*amount_a*9975/(ra
    ↪ *10000+amount_a*9975)
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** circuit.rs

Description The name "last_token" is not accurate anymore.
Recommendation Consider renaming to "fee_token" or "last_fee_token" everywhere.
Client Comment Will not fix.

Listing 11:

```
925  last_token_id: &mut AllocatedNum<E>,

1349 let new_last_token_id = AllocatedNum::conditionally_select(

1356     &last_token_id,

1359 *last_token_id = new_last_token_id.clone();

1436         &new_last_token_id,
```

3.12 CVF-12

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** circuit.rs

Recommendation There should be similar check for 'fee_token' here.

Listing 12:

```
1075 cs.namespace(|| "is token0 equal to previous"),

1080 cs.namespace(|| "is token1 equal to previous"),

1085 cs.namespace(|| "is lp_token equal to previous"),
```

3.13 CVF-13

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** circuit.rs

Description Why an extra bit is needed if it is always false?
Client Comment Will not fix.

Listing 13:

```
1124 Boolean::enforce_equal(
        cs.namespace(|| "0 <= fee_token < 2^(FEE_TOKEN_BIT_WIDTH-1)
            ↪ "),
        &op_data.fee_token.get_bits_le()[FEE_TOKEN_BIT_WIDTH - 1],
        &Boolean::constant(false),
    )?;
```

3.14 CVF-14

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation Probably only one of these lines should stay.
Client Comment Will not fix.

Listing 14:

```
1709 //     from.balance[token] -= full_amount + fee
1710 //     from.balance[fee_token] -= full_amount + fee
```

3.15 CVF-15

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** circuit.rs

Description Only the check is different between the cases, as the subtraction can be done in two steps in both cases.
Client Comment Will not fix.

Listing 15:

```
2454 //     from.balance[token] -= full_amount + fee
     //     from.balance[fee_token] -= full_amount + fee

2459 //     from.balance[token] -= full_amount
2460 //     from.balance[fee_token] -= fee
```

3.16 CVF-16

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Description As long as plain transfer to pair accounts are forbidden, it should not be possible for the balances to differ from the reserve amounts.
Recommendation Consider removing all the logic responsible for handling the case when reserves and corresponding balances are different.
Client Comment Will not fix.

Listing 16:

```
3581 // a0 = balance0 - r0 + amountA (it's possible that balance0 >
     ↪ r0)
     // a1 = balance1 - r1 + amountB (it's possible that balance1 >
     ↪ r1)
```

3.17 CVF-17

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation The minimum liquidity values shouldn't be hardcoded, even in string literals and comments, as changing this value would make such literals and comments inaccurate.
Client Comment Will not fix.

Listing 17:

```
3663 let liq_minimum = alloc_const(cs.namespace(|| "minimum_liquidity
     ↪ "), &Some(liq_min_fr))?;

3666     cs.namespace(|| "initial_liquidity >= 1000"),
```

3.18 CVF-18

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Recommendation Should be 'third chunk'.
Client Comment Will not fix.

Listing 18:

```
3847 cs.namespace(|| "is_fourth_chunk"),
```

3.19 CVF-19

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Recommendation This check is redundant, as it is anyway checked that the LP amount doesn't exceed the user's balance of LP tokens.
Client Comment Will not fix.

Listing 19:

```
4243 let is_lp_amount_valid = geq(
         cs.namespace(|| "total_supply >= lp_amount"),
         &cur.pair_account.total_supply.get_number(),
         &op_data.lp_token_amount.get_number(),
         BALANCE_BIT_WIDTH,
     )?;

4263 chunk1_valid_flags.push(is_lp_amount_valid.clone());
```

3.20 CVF-20

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Recommendation Should be ‘is chunk2 valid‘.
Client Comment Will not fix.

Listing 20:

```
4420 let is_chunk2_valid = multi_and(cs.namespace(|| "is chunk4 valid
     ↪ "), &chunk2_valid_flags)?;
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** circuit.rs

Recommendation The constant values shouldn’t be hardcoded, even in comments and string literals. In case the constants will be changed, such hardcoded values will become inaccurate.
Client Comment Will not fix.

Listing 21:

```
4578 // amountInWithFee = amountIn*9975

4586         cs.namespace(|| "amountIn*9975"),

4615         cs.namespace(|| "reserveIn = r0*10000"),

4627 // amountOut = r1*(amountIn*9975)/(r0*10000+amountIn*9975)
```

3.22 CVF-22

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** circuit.rs

Recommendation 16 should be a named constant.
Client Comment Will not fix.

Listing 22:

```
4641 params::BALANCE_BIT_WIDTH + 16, // sum's bit_length <=
     ↪ BALANCE_BIT_WIDTH+16
```

3.23 CVF-23

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Recommendation Consider adding an assert that the dropped bit is zero.
Client Comment Will not fix.

Listing 23:

```
4665 let fee_token_bits = op_data.fee_token.get_bits_be()[1..].to_vec
     ↪ ();
```

3.24 CVF-24

- **Severity** Moderate
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Description Swap fee has to be paid in one of the swapped tokens, but at the same time fee could be paid only in one of the first 32 registered tokens. Thus, pairs whose both tokens are outside the first 32 tokens are useless.
Recommendation Consider forbidding creating such pairs at circuit level.
Client Comment The fee token is checked in L1's smart contract. It's NOT necessary to protect that in the circuit. For the case, do NOT want to fix.

Listing 24:

```
5233 let is_token_not_valid = multi_or(
         cs.namespace(|| "is token all same"),
         &[
             is_token_same.clone(),
             is_lp_same_with_token0.clone(),
             is_lp_same_with_token1.clone(),
         ],
5240 )?;
```

3.25 CVF-25

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** circuit.rs

Description In the "allocate_account_leaf_bits" function this parameter is named "balance2_updated_root".
Recommendation Consider using consistent naming.
Client Comment Will not fix.

Listing 25:

```
5375 balance_updated_root: &mut AllocatedNum<E>,
```

```
5384             balance_updated_root,
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** circuit.rs

Recommendation The semantics of these two parameters is non-trivial and should be documented.
Client Comment Will not fix.

Listing 26:

```
5472 balance2_updated_root: &mut AllocatedNum<E>,
     is_read: bool,
```

3.27 CVF-27

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** circuit.rs

Description The name is very confusing. Actually, this is the root after the first token balance was updated but before the second token balance update.
Recommendation Consider renaming.
Client Comment Will not fix.

Listing 27:

```
5472 balance2_updated_root: &mut AllocatedNum<E>,
```

3.28 CVF-28

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** circuit.rs

Recommendation There code blocks are very similar. In order to avoid code duplication, consider allocating balance2 Merkle tree only one: before the conditional statement.
Client Comment Will not fix.

Listing 28:

```
5493 *balance2_updated_root = allocate_merkle_root(
         cs.namespace(|| "balance_updated_root"),
         balance2_data,
         &branch.token2.get_bits_le(),
         &branch.balance2_audit_path,
         params::balance_tree_depth(),
         params,
5500 )?;
     balances_root = balance_root.clone();

5504 let balance2_root = allocate_merkle_root(
         cs.namespace(|| "balance2_subtree_root"),
         balance2_data,
         &branch.token2.get_bits_le(),
         &branch.balance2_audit_path,
         params::balance_tree_depth(),
5510     params,
     )?;
     balances_root = balance2_root.clone();
```

3.29 CVF-29

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** create_pair.rs

Description This doesn't seem possible as transfers to empty accounts are forbidden. However, in case it is possible, there could be some troubles with zero balances in operation branches below.
Client Comment Will not fix.

Listing 29:

```
209 // TODO: what if someone transfer some token to this account
      ↪ before CreatePair?
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** create_pair.rs

Description In case it is possible to transfer some funds to a zero account, these balances should not be hardcoded to zero, but should use the actual amount of plain ether at the pair account ID.
Client Comment Will not fix.

Listing 30:

```
232  balance_value: zero,
235 balance2_value: zero,
247  balance_value: zero,
250 balance2_value: zero,
```

3.31 CVF-31

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** allocated_structures.rs

Description IT is unclear what this comment refers to. Below the comment that are both, circuit elements, that do have bit representations, and allocated numbers, that don't have them.
Recommendation Consider rearranging the comment or removing it.
Client Comment Will not fix.

Listing 31:

```
31 //we do not need their bit representations
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** allocated_structures.rs

Recommendation Consider renaming the first balance and the first token to "balance1" and "token1" for consistency with "balance2" and "token".
Client Comment Will not fix.

Listing 32:

```
33  pub balance: CircuitElement<E>,
    pub balance_audit_path: Vec<AllocatedNum<E>>,
pub balance2: CircuitElement<E>,
pub balance2_audit_path: Vec<AllocatedNum<E>>,
    pub token: CircuitElement<E>,
pub token2: CircuitElement<E>,
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** allocated_structures.rs

Description These code blocks are very similar to what is done for the first token.
Recommendation Consider extracting utility functions or macros to avoid code duplication.
Client Comment Will not fix.

Listing 33:

```
74  let balance2 = CircuitElement::from_fe_with_known_length(
        cs.namespace(|| "balance2"),
        || Ok(operation_branch.witness.balance2_value.grab()?),
        franklin_constants::BALANCE_BIT_WIDTH,
    )?;

87  let token2 = CircuitElement::from_fe_with_known_length(
        cs.namespace(|| "token2"),
        || Ok(operation_branch.token2.grab()?),
90      franklin_constants::balance_tree_depth(),
    )?;
    let token2 = token2.pad(franklin_constants::TOKEN_BIT_WIDTH);

98  let balance2_audit_path = utils::allocate_numbers_vec(
        cs.namespace(|| "balance2_audit_path"),
100     &operation_branch.witness.balance2_subtree_path,
    )?;
```

3.34 CVF-34

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** allocated_structures.rs

Description As long as plain transfers to pair accounts are forbidden, there is no need to store reserve amounts along with token balances, as they are guaranteed to be always the same.
Client Comment Will not fix.

Listing 34:

```
164 pub r0: CircuitElement<E>,                    // pair.reserve0
    pub r1: CircuitElement<E>,                    // pair.reserve1
```

3.35 CVF-35

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** allocated_structures.rs

Description The field orders in the struct definition and initialization are different. Also, variables, corresponding to the fields are declared and assign in yet another order.
Recommendation Consider using the same order everywhere. This would improve code readability and make the code less error-prone.
Client Comment Will not fix.

Listing 35:

```
409 Ok(AllocatedOperationData {

605 Ok(AllocatedOperationData {
```

3.36 CVF-36

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** transfer_to_new.rs

Recommendation Consider documenting what this first part consist of, i.e. that it actually subtract transferred value and fee from the origin account (probably in different tokens).
Client Comment Will not fix.

Listing 36:

```
297 //applying first transfer part
```

3.37 CVF-37

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** add_liquidity.rs

Recommendation This commented code should be removed.
Client Comment Will not fix.

Listing 37:

```
204 // append_be_fixed_width(
    //     &mut pubdata_bits,
    //     &self.args.token0.unwrap(),
    //     TOKEN_BIT_WIDTH,
    // );
    // append_be_fixed_width(
210 //     &mut pubdata_bits,
    //     &self.args.token1.unwrap(),
    //     TOKEN_BIT_WIDTH,
    // );
    // append_be_fixed_width(
    //     &mut pubdata_bits,
    //     &self.args.lp_token.unwrap(),
    //     TOKEN_BIT_WIDTH,
    // );
```

3.38 CVF-38

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** add_liquidity.rs

Recommendation This commented line should be removed.
Client Comment Will not fix.

Listing 38:

```
670 // args.amount2_packed = Some(a1_encoded);
```

3.39 CVF-39

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Config.sol

Recommendation If these constants are related, one of them could be derived from the other.
Client Comment Will not fix.

Listing 39:

```
36 uint16 constant MAX_AMOUNT_OF_REGISTERED_FEE_TOKENS = 32 - 1;

39 uint16 constant USER_TOKENS_START_ID = 32;
```

3.40 CVF-40

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Config.sol

Description No access level specified for these constants, so internal access will be used by default.
Recommendation Consider specifying access level explicitly.
Client Comment Will not fix.

Listing 40:

```
36 uint16 constant MAX_AMOUNT_OF_REGISTERED_FEE_TOKENS = 32 - 1;

39 uint16 constant USER_TOKENS_START_ID = 32;

42 uint16 constant MAX_AMOUNT_OF_REGISTERED_USER_TOKENS = 16352;

45 uint16 constant MAX_AMOUNT_OF_REGISTERED_TOKENS = 16384 - 1;

49 uint32 constant MAX_ACCOUNT_ID = (2 ** 28) - 1;

64 uint256 constant CREATE_PAIR_BYTES = 3 * CHUNK_BYTES;
   uint256 constant DEPOSIT_BYTES = 4 * CHUNK_BYTES;
   uint256 constant TRANSFER_TO_NEW_BYTES = 4 * CHUNK_BYTES;
   uint256 constant PARTIAL_EXIT_BYTES = 5 * CHUNK_BYTES;

71 uint256 constant UNISWAP_ADD_LIQ_BYTES = 3 * CHUNK_BYTES;
   uint256 constant UNISWAP_RM_LIQ_BYTES = 3 * CHUNK_BYTES;
   uint256 constant UNISWAP_SWAP_BYTES = 2 * CHUNK_BYTES;

77 uint256 constant FULL_EXIT_BYTES = 4 * CHUNK_BYTES;

84 uint256 constant CHANGE_PUBKEY_BYTES = 5 * CHUNK_BYTES;
```

3.41 CVF-41

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** Config.sol

Description The word "amount" is confusing here, as it usually has different meaning for tokens.
Recommendation Consider using the word "number" or "count" instead.
Client Comment Will not fix.

Listing 41:

```
36 uint16 constant MAX_AMOUNT_OF_REGISTERED_FEE_TOKENS = 32 - 1;

42 uint16 constant MAX_AMOUNT_OF_REGISTERED_USER_TOKENS = 16352;

45 uint16 constant MAX_AMOUNT_OF_REGISTERED_TOKENS = 16384 - 1;
```

3.42 CVF-42

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Config.sol

Recommendation This value should probably be derived from other constants.
Client Comment Will not fix.

Listing 42:

```
42 uint16 constant MAX_AMOUNT_OF_REGISTERED_USER_TOKENS = 16352;
```

3.43 CVF-43

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Config.sol

Description Is '28' here actually the account tree height? If so, it should probably be a named constant by itself.
Client Comment Will not fix.

Listing 43:

```
49 uint32 constant MAX_ACCOUNT_ID = (2 ** 28) - 1;
```

3.44 CVF-44

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** PairTokenManager.sol

Description The word "amount" here is confusing, as it usually has different meaning for tokens.
Recommendation Consider using the word "number" or "count" instead.
Client Comment Will not fix.

Listing 44:

```
7 uint16 constant MAX_AMOUNT_OF_PAIR_TOKENS = 49152;
```

3.45 CVF-45

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** PairTokenManager.sol

Recommendation This condition is redundant as it always holds.
Client Comment Will not fix.

Listing 45:

```
46 require(tokenId <= (PAIR_TOKEN_START_ID -1 +
   ↪ MAX_AMOUNT_OF_PAIR_TOKENS), "pms4");
```

3.46 CVF-46

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSyncCommitBlock.sol

Recommendation The "_blockNumberFrom" parameter is redundant as it could be derived from the "totalBlocksChecked" storage variable.
Client Comment Will not fix.

Listing 46:

```
202 == totalBlocksChecked + 1, "cw2");
```

3.47 CVF-47

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSyncCommitBlock.sol

Recommendation It is possible to pack all the bytes counts into a single word and then extract them from there via shift and bitwise "and". This would be cheaper than a series of conditional operations.
Client Comment Will not fix.

Listing 47:

```
367   if (opType == Operations.OpType.Transfer) {
          pubDataPtr += TRANSFER_BYTES;
      } else if (opType == Operations.OpType.Noop) {
370       pubDataPtr += NOOP_BYTES;
      } else if (opType == Operations.OpType.TransferToNew) {
          pubDataPtr += TRANSFER_TO_NEW_BYTES;
      } else if (opType == Operations.OpType.AddLiquidity || opType ==
        ↪  Operations.OpType.RemoveLiquidity) {
          pubDataPtr += UNISWAP_ADD_RM_LIQ_BYTES;
      } else if (opType == Operations.OpType.AddLiquidity) {
          pubDataPtr += UNISWAP_ADD_LIQ_BYTES;
      } else if (opType == Operations.OpType.RemoveLiquidity) {
          pubDataPtr += UNISWAP_RM_LIQ_BYTES;
      } else if (opType == Operations.OpType.Swap) {
380       pubDataPtr += UNISWAP_SWAP_BYTES;
```

3.48 CVF-48

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSyncCommitBlock.sol

Description This code duplicates the code in the "collectOnchainOps" function.
Recommendation Consider extracting into into a separate function.
Client Comment Will not fix.

Listing 48:

```
504  if (opType == Operations.OpType.Transfer) {
         pubDataPtr += TRANSFER_BYTES;
     } else if (opType == Operations.OpType.Noop) {
         pubDataPtr += NOOP_BYTES;
     } else if (opType == Operations.OpType.TransferToNew) {
         pubDataPtr += TRANSFER_TO_NEW_BYTES;
510 } else if (opType == Operations.OpType.AddLiquidity || opType ==
    ↪  Operations.OpType.RemoveLiquidity) {
        pubDataPtr += UNISWAP_ADD_RM_LIQ_BYTES;
    } else if (opType == Operations.OpType.AddLiquidity)  {
        pubDataPtr += UNISWAP_ADD_LIQ_BYTES;
    } else if (opType == Operations.OpType.RemoveLiquidity) {
        pubDataPtr += UNISWAP_RM_LIQ_BYTES;
     } else if (opType == Operations.OpType.Swap) {
         pubDataPtr += UNISWAP_SWAP_BYTES;
```

3.49 CVF-49

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** ZkSync.sol

Recommendation "ZKSync" should be replaced with "ZKSwap" everywhere. Probably a credit to "ZKSync" should be placed somewhere in comments.
Client Comment Will not fix.

Listing 49:

```
21 /// @title zkSync main contract

24 contract ZkSync is PairTokenManager, UpgradeableMaster, Storage,
    ↪  Config, Events, ReentrancyGuard {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSync.sol

Description Events with token contract addresses are less useful.
Recommendation Consider logging token contract addresses along with token IDs.
Client Comment Will not fix.

Listing 50:

```
115 emit OnchainCreatePair(_tokenAID, _tokenBID, _tokenPair, _pair);
```

3.51 CVF-51

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSync.sol

Description This function has much in common with the "withdrawETHWithAddress" function.
Recommendation Consider implementing one function on top of the other.
Client Comment Will not fix.

Listing 51:

```
316 function withdrawETH(uint128 _amount) external nonReentrant {
```

3.52 CVF-52

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSync.sol

Recommendation These checks look redundant. IT is anyway possible to withdraw funds to a dead address.
Client Comment Will not fix.

Listing 52:

```
328 require(_to != address(0), "ipa11");

398 require(_to != address(0), "ipa12");
```

3.53 CVF-53

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ZkSync.sol

Recommendation This empty comment should be removed.
Client Comment Will not fix.

Listing 53:

```
360 //
```

3.54 CVF-54

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ZkSync.sol

Description This function has very much in common with the 'withdrawERC20WithAddress' function.
Recommendation Consider implementing one function on top of the other.
Client Comment Will not fix.

Listing 54:

```
378 function withdrawERC20(IERC20 _token, uint128 _amount) external
    ↪ nonReentrant {
```

3.55 CVF-55

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Verifier.sol

Recommendation All the dummy functionality should be removed from the production code. Just inherit a new contract named "DummyVerifier" from the "Verifier" smart contract, and override necessary functions with dummy versions.
Client Comment Will not fix.

Listing 55:

```
 9 bool constant DUMMY_VERIFIER = false;
10

20     if (DUMMY_VERIFIER) {
           return true;
       } else {
           return isBlockSizeSupportedInternal(_size);
       }

34     if (DUMMY_VERIFIER) {
           uint oldGasValue = gasleft();
           uint tmp;
           while (gasleft() + 500000 > oldGasValue) {
               tmp += 1;
           }
40         return true;
       }
```

3.56 CVF-56

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** PlonkAggCore.sol

Recommendation This commented line should be removed.
Client Comment Will not fix.

Listing 56:

```
680 //              assert(input < PairingsBn254.r_mod);
```

3.57 CVF-57

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Governance.sol

Recommendation Local variables are not needed here. Just assign to the storage variables directly: (networkGovernor, tokenLister) = ...;
Client Comment Will not fix.

Listing 57:

```
60 (address _networkGovernor, address _tokenLister) = abi.decode(
   ↪ initializationParameters, (address, address));

62  networkGovernor = _networkGovernor;
tokenLister = _tokenLister;
```

3.58 CVF-58

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Governance.sol

Description Why to forbid this? The governor may want to temporary disable token listings by setting token lister to zero address. Teh governer may anyway set token lister to some dead address.
Client Comment Will not fix.

Listing 58:

```
85 require(_newTokenLister != address(0), "zero address is passed
   ↪ as _newTokenLister");
```

3.59 CVF-59

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** Governance.sol

Recommendation A different event should be logged here to distinguish token lister changes from governor changes.

Listing 59:

```
88 emit NewGovernor(_newTokenLister);
```

3.60 CVF-60

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Governance.sol

Recommendation These check should be done earlier, before expensive checks that read from the storage.
Client Comment Will not fix.

Listing 60:

```
98
        _token != address(0), "address cannot be zero"
100 );

118  require(
         _token != address(0), "address cannot be zero"
120  );
```

3.61 CVF-61

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** Governance.sol

Recommendation These two lines could be merged into one: 'uint16 newTokenId = ++totalFeeTokens;'
Client Comment Will not fix.

Listing 61:

```
102 totalFeeTokens++;
    uint16 newTokenId = totalFeeTokens; // it is not `totalTokens -
       ↪ 1` because tokenId = 0 is reserved for eth
```

3.62 CVF-62

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** Governance.sol

Recommendation These two lines could be merged into one: 'uint16 newTokenId = USER_TOKENS_START_ID + totalUserTokens++;'
Client Comment Will not fix.

Listing 62:

```
124 uint16 newTokenId = USER_TOKENS_START_ID + totalUserTokens;
    totalUserTokens++;
```

3.63 CVF-63

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Governance.sol

Recommendation This check is redundant, as the condition may never be false.
Client Comment Will not fix.

Listing 63:

```
176 <= MAX_AMOUNT_OF_REGISTERED_TOKENS, "gvs12");
```

3.64 CVF-64

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** DeployFactory.sol

Recommendation Consider adding parameter names in comments near "this" values to explain why the same value is passed twice.
Client Comment Will not fix.

Listing 64:

```
53 Proxy governance = new Proxy(address(_governanceTarget), abi.
   ↪ encode(this, this));
```

3.65 CVF-65

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** DeployFactory.sol

Description Calling a function just once a contract lifetime is a waste of gas.
Recommendation Consider passing the token addresses just in calldata.
Client Comment Will not fix.

Listing 65:

```
90 address[] memory tokens = getTokens();
```

3.66 CVF-66

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** transfer.rs

Recommendation These commented lines should be removed.
Client Comment Will not fix.

Listing 66:

```
57 // pub from_after: OperationBranch<E>,

61 // pub to_after: OperationBranch<E>,
```

3.67 CVF-67

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** transfer.rs

Recommendation This commented code should be removed.
Client Comment Will not fix.

Listing 67:

```
425 // from_after: OperationBranch {
    //     address: Some(account_address_from_fe),
    //     token: Some(token_fe),
    //     witness: OperationBranchWitness {
    //         account_witness: account_witness_from_intermediate,
430 //         account_path: audit_path_from_after,
    //         balance_value: Some(balance_from_intermediate),
    //         balance_subtree_path: audit_balance_path_from_after,
    //         pair_account_witness: PairAccountWitness::empty(),
    //     },
    // },

476 // to_after: OperationBranch {
    //     address: Some(account_address_to_fe),
    //     token: Some(token_fe),
    //     witness: OperationBranchWitness {
480 //         account_witness: account_witness_to_after,
    //         account_path: audit_path_to_after,
    //         balance_value: Some(balance_to_after),
    //         balance_subtree_path: audit_balance_path_to_after,
    //         pair_account_witness: pair_witness_to_after,
    //     },
    // },
```