ABDK CONSULTING

CIRCUIT
AUDIT

EDGESWAP

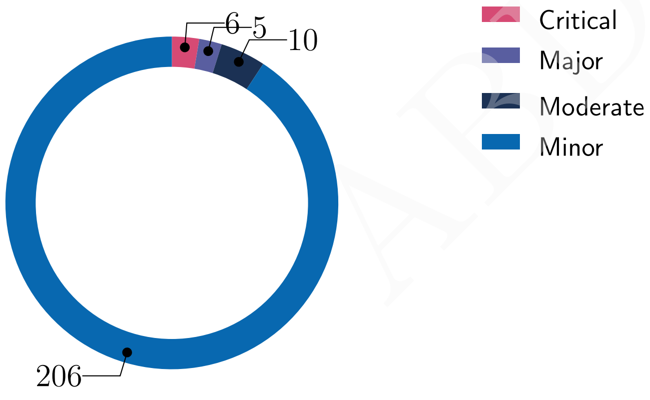
Circuits

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
15th December 2021

We've been asked to review the 14 files in a Github repo. We found 6 critical, 5 major, and a few less important issues. All critical issues were fixed.



Findings

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

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

ID	Severity	Category	Status
CVF-58	Minor	Bad naming	Info
CVF-59	Minor	Suboptimal	Info
CVF-60	Critical	Flaw	Fixed
CVF-61	Minor	Bad datatype	Info
CVF-62	Minor	Bad datatype	Info
CVF-63	Minor	Suboptimal	Info
CVF-64	Moderate	Flaw	Info
CVF-65	Critical	Flaw	Fixed
CVF-66	Minor	Bad naming	Info
CVF-67	Minor	Flaw	Info
CVF-68	Minor	Suboptimal	Info
CVF-69	Minor	Suboptimal	Info
CVF-70	Minor	Flaw	Info
CVF-71	Minor	Procedural	Info
CVF-72	Minor	Suboptimal	Info
CVF-73	Minor	Suboptimal	Info
CVF-74	Minor	Suboptimal	Info
CVF-75	Minor	Suboptimal	Info
CVF-76	Minor	Suboptimal	Info
CVF-77	Minor	Suboptimal	Info
CVF-78	Minor	Suboptimal	Info
CVF-79	Minor	Procedural	Info
CVF-80	Minor	Bad naming	Info
CVF-81	Minor	Procedural	Info
CVF-82	Moderate	Unclear behavior	Info
CVF-83	Minor	Documentation	Fixed
CVF-84	Minor	Unclear behavior	Info
CVF-85	Minor	Suboptimal	Info
CVF-86	Minor	Bad naming	Info
CVF-87	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Info
CVF-89	Minor	Documentation	Info
CVF-90	Minor	Suboptimal	Info
CVF-91	Minor	Suboptimal	Info
CVF-92	Minor	Bad naming	Info
CVF-93	Minor	Readability	Info
CVF-94	Minor	Suboptimal	Info
CVF-95	Minor	Overflow/Underflow	Info
CVF-96	Minor	Procedural	Info
CVF-97	Minor	Procedural	Info
CVF-98	Minor	Suboptimal	Info
CVF-99	Minor	Suboptimal	Info
CVF-100	Minor	Suboptimal	Info
CVF-101	Minor	Suboptimal	Info
CVF-102	Minor	Suboptimal	Info
CVF-103	Major	Suboptimal	Info
CVF-104	Minor	Bad naming	Info
CVF-105	Minor	Bad naming	Info
CVF-106	Minor	Overflow/Underflow	Info
CVF-107	Minor	Suboptimal	Info
CVF-108	Minor	Suboptimal	Info
CVF-109	Minor	Suboptimal	Info
CVF-110	Moderate	Procedural	Info
CVF-111	Minor	Bad naming	Info
CVF-112	Minor	Bad naming	Info
CVF-113	Minor	Bad naming	Info
CVF-114	Minor	Bad naming	Info
CVF-115	Minor	Bad naming	Info
CVF-116	Minor	Bad naming	Info
CVF-117	Minor	Bad naming	Info

ID	Severity	Category	Status
CVF-118	Minor	Bad naming	Info
CVF-119	Minor	Procedural	Info
CVF-120	Minor	Procedural	Info
CVF-121	Moderate	Unclear behavior	Info
CVF-122	Moderate	Unclear behavior	Info
CVF-123	Minor	Procedural	Info
CVF-124	Minor	Suboptimal	Info
CVF-125	Minor	Bad naming	Info
CVF-126	Minor	Bad naming	Info
CVF-127	Minor	Bad naming	Info
CVF-128	Minor	Bad naming	Info
CVF-129	Minor	Bad naming	Info
CVF-130	Minor	Suboptimal	Info
CVF-131	Minor	Bad datatype	Info
CVF-132	Minor	Bad datatype	Info
CVF-133	Minor	Procedural	Fixed
CVF-134	Minor	Suboptimal	Info
CVF-135	Minor	Suboptimal	Info
CVF-136	Minor	Readability	Info
CVF-137	Minor	Suboptimal	Info
CVF-138	Minor	Bad datatype	Info
CVF-139	Minor	Bad datatype	Info
CVF-140	Minor	Unclear behavior	Info
CVF-141	Minor	Suboptimal	Info
CVF-142	Minor	Procedural	Info
CVF-143	Minor	Readability	Info
CVF-144	Minor	Suboptimal	Info
CVF-145	Minor	Procedural	Info
CVF-146	Minor	Suboptimal	Info
CVF-147	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-148	Minor	Bad datatype	Info
CVF-149	Minor	Bad datatype	Info
CVF-150	Minor	Procedural	Fixed
CVF-151	Minor	Readability	Info
CVF-152	Minor	Procedural	Info
CVF-153	Minor	Procedural	Info
CVF-154	Minor	Suboptimal	Info
CVF-155	Minor	Suboptimal	Info
CVF-156	Minor	Bad datatype	Info
CVF-157	Minor	Readability	Info
CVF-158	Minor	Suboptimal	Info
CVF-159	Minor	Suboptimal	Info
CVF-160	Minor	Suboptimal	Info
CVF-161	Minor	Suboptimal	Info
CVF-162	Minor	Suboptimal	Info
CVF-163	Minor	Suboptimal	Info
CVF-164	Minor	Suboptimal	Info
CVF-165	Minor	Bad naming	Info
CVF-166	Minor	Suboptimal	Info
CVF-167	Minor	Procedural	Info
CVF-168	Minor	Suboptimal	Info
CVF-169	Minor	Suboptimal	Info
CVF-170	Minor	Suboptimal	Info
CVF-171	Minor	Bad datatype	Info
CVF-172	Minor	Suboptimal	Info
CVF-173	Minor	Procedural	Info
CVF-174	Minor	Suboptimal	Info
CVF-175	Minor	Suboptimal	Info
CVF-176	Minor	Procedural	Info
CVF-177	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-178	Minor	Suboptimal	Info
CVF-179	Minor	Readability	Info
CVF-180	Minor	Suboptimal	Fixed
CVF-181	Minor	Suboptimal	Info
CVF-182	Minor	Bad datatype	Info
CVF-183	Minor	Procedural	Info
CVF-184	Minor	Bad datatype	Info
CVF-185	Minor	Suboptimal	Info
CVF-186	Minor	Bad datatype	Info
CVF-187	Minor	Suboptimal	Info
CVF-188	Major	Flaw	Fixed
CVF-189	Minor	Suboptimal	Info
CVF-190	Minor	Bad naming	Info
CVF-191	Minor	Readability	Info
CVF-192	Minor	Suboptimal	Info
CVF-193	Minor	Procedural	Info
CVF-194	Minor	Suboptimal	Info
CVF-195	Minor	Suboptimal	Info
CVF-196	Minor	Readability	Info
CVF-197	Minor	Suboptimal	Info
CVF-198	Minor	Bad datatype	Info
CVF-199	Minor	Procedural	Info
CVF-200	Minor	Readability	Info
CVF-201	Minor	Suboptimal	Info
CVF-202	Moderate	Unclear behavior	Info
CVF-203	Minor	Readability	Info
CVF-204	Minor	Suboptimal	Info
CVF-205	Minor	Suboptimal	Info
CVF-206	Minor	Readability	Info
CVF-207	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-208	Minor	Bad naming	Info
CVF-209	Minor	Bad naming	Info
CVF-210	Minor	Procedural	Info
CVF-211	Minor	Unclear behavior	Info
CVF-212	Minor	Bad naming	Info
CVF-213	Minor	Bad naming	Info
CVF-214	Minor	Bad naming	Info
CVF-215	Minor	Bad naming	Info
CVF-216	Minor	Bad naming	Info
CVF-217	Minor	Bad naming	Info
CVF-218	Minor	Bad datatype	Info
CVF-219	Minor	Suboptimal	Info
CVF-220	Minor	Suboptimal	Info
CVF-221	Minor	Bad naming	Info
CVF-222	Minor	Bad datatype	Info
CVF-223	Minor	Bad datatype	Info
CVF-224	Major	Flaw	Fixed
CVF-225	Minor	Suboptimal	Info
CVF-226	Minor	Procedural	Info
CVF-227	Minor	Procedural	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	December 15, 2021	D. Khovratovich	Initial Draft
0.2	December 15, 2021	D. Khovratovich	Minor revision
1.0	December 15, 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 have reviewed the contracts at Github repo:

- witness/add_liquidity.rs
- witness/claim_bonus.rs
- witness/create_pair.rs
- witness/deposit.rs
- witness/full_exit.rs
- witness/mining_maintenance.rs
- witness/remove_liquidity.rs
- witness/swap.rs
- account.rs
- witness/withdraw.rs
- allocated_structures.rs
- circuit.rs
- exit_circuit.rs
- exit_lp_circuit.rs

The fixes were provided in the repository.

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

Description Mentioning "ZkSync" in the name of the circuit seems odd, as this circuit implements a protocol that is different from the ZkSync protocol.
Recommendation Consider renaming.

Listing 1:

```
pub struct ZkSyncCircuit<'a, E: RescueEngine + JubjubEngine> {
```

3.2 CVF-2

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

Description This variable name is ambigious.
Recommendation Consider documenting it.

Listing 2:

```
pub constant_number: ConstantNumber<E>,
```

3.3 CVF-3

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

Recommendation explicit_0, explicit_1000, explicit_10000 etc.' would be more readable.

Listing 3:

```
pub explicit_zero: CircuitElement<E>,
pub one_thousand: CircuitElement<E>,
pub ten_thousand: CircuitElement<E>,
pub ninety_ninety_five: CircuitElement<E>,
pub nine_hundred_ninety_seven: CircuitElement<E>,
```

3.4 CVF-4

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

Recommendation This implementation could be automatically derived.

Listing 4:

```
impl<'a, E: RescueEngine + JubjubEngine> std::clone::Clone for
    ↪ ZkSyncCircuit<'a, E> {
```

3.5 CVF-5

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

Recommendation This function is way too large. It consists of several stages that could be extracted into separate functions.

Listing 5:

```
fn synthesize<CS: ConstraintSystem<E>>(self, cs: &mut CS) ->
    ↪ Result<(), SynthesisError> {
```

3.6 CVF-6

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

Recommendation This should be extracted into a function implemented for "ConstantNumbers".

Listing 6:

```
//some constant numbers
```

3.7 CVF-7

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

Recommendation There should be a utility function to convert a constant into a circuit element.

Listing 7:

```
154 let one_thousand_ce = {

171 let nine_hundred_ninety_seven_ce = {

188 let ten_thousand_ce = {

206 let ninety_ninety_five_ce = {
```

3.8 CVF-8

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

Recommendation These constants should be named.

Listing 8:

```
157         Ok(E::Fr::from_str("1000").unwrap())

161 cs.namespace(|| "enforce one_thousand equal to 1000"),

168 10,

174         Ok(E::Fr::from_str("997").unwrap())

179 &E::Fr::from_str("997").unwrap(),

185 10,

196 &E::Fr::from_str("10000").unwrap(),

202 14,
```

3.9 CVF-9

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 9:

```
157                     Ok(E::Fr::from_str("1000").unwrap())

162             &E::Fr::from_str("1000").unwrap(),

174                     Ok(E::Fr::from_str("997").unwrap())

179             &E::Fr::from_str("997").unwrap(),

191                     Ok(E::Fr::from_str("10000").unwrap())

196             &E::Fr::from_str("10000").unwrap(),

209                     Ok(E::Fr::from_str("9995").unwrap())

214             &E::Fr::from_str("9995").unwrap(),

846       || E::Fr::from_str(&CreatePairOp::OP_CODE.to_string()).grab(),

1778          Expression::constant::<CS>(E::Fr::from_str(&i.to_string()).unwrap()),

2573      || Ok(E::Fr::from_str(&TransferOp::OP_CODE.to_string()).unwrap()),

2580      &E::Fr::from_str(&TransferOp::OP_CODE.to_string()).unwrap(),

5460 let x = E::Fr::from_str(&op_type.to_string()).unwrap();
     let y = E::Fr::from_str(&(op_chunks - 1).to_string()).unwrap();
```

3.10 CVF-10

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

Description Deriving circuit elements from constants is suboptimal.
Recommendation Just construct an array of bits using explicit zero and explicit one signals as many times as needed.

Listing 10:

```
    CircuitElement::from_number_with_known_length(
        cs.namespace(|| "circuit_element from 1000"),

    CircuitElement::from_number_with_known_length(
        cs.namespace(|| "circuit_element from 997"),

    CircuitElement::from_number_with_known_length(
        cs.namespace(|| "circuit_element from 10000"),

    CircuitElement::from_number_with_known_length(
        cs.namespace(|| "circuit_element from 9995"),

let lp_token_id_start = CircuitElement::
  ↪ from_expression_known_max_length(
    cs.namespace(|| "lp_token_id_start"),

let bonus_scaling_factor = CircuitElement::
  ↪ from_expression_known_max_length(
    cs.namespace(|| "bonus_scaling_factor"),
```

3.11 CVF-11

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

Recommendation It should be named ‘ninety_nine_ninety_five‘.

Listing 11:

```
let ninety_ninety_five_ce = {

        cs.namespace(|| "enforce ten_thousand equal to 9995"),
```

3.12 CVF-12

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

Description This is redundant in general, as may parts of pubdata are derived from op_data fields that are independently checked to have to same values in all chunks of a transaction.
Recommendation Consider checking only those values, that are not checked elsewhere.

Listing 12:

```
// when operation allocated pubdata from witness (that happens
    ↪ every chunk!)
// we check that this pubdata is equal to pubdata if the
    ↪ previous chunk in the same
// operation. So, we only allow these values to change if it's a
    ↪  first chunk in the operation
```

3.13 CVF-13

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

Recommendation The loop body should be extracted into a function.

Listing 13:

```
for (i, operation) in self.operations.iter().enumerate() {
```

3.14 CVF-14

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

Recommendation This should be a named constant.

Listing 14:

```
.extend_from_slice(&vec![Boolean::constant(false); 7]);
```

3.15 CVF-15

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

Description It is redundant in general to have two branches in every chunk, as only one of these branches is actually verified against the tree.
Recommendation Consider having only the current branch with audit path per chunk.

Listing 15:

```
424 let lhs =

426 let rhs =

428 let mut current_branch = self.select_branch(
```

3.16 CVF-16

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

Description This variable is called ‘cur‘ in other functions.
Recommendation Consider using consistent naming,

Listing 16:

```
428 let mut current_branch = self.select_branch(
```

3.17 CVF-17

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

Description Only one branch for each chunk is verified so only one is trusted. However, the operation code treats both branches equally.
Recommendation It would be safer to name the trusted branch differently and do not trust the account data from the other one.
Client Comment All the fields in untrusted branch that we use haved been verified.

Listing 17:

```
437 let (state_root, is_account_empty, _) = check_account_data(

439     &current_branch,

455     &lhs,
        &rhs,
```

3.18 CVF-18

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

Description Passing audit paths here is clearly redundant since they are not checked inside operations.
Recommendation It is more readable to pass accounts with explicit comment which side is pair and which is user.

Listing 18:

```
455 &lhs,
    &rhs,
```

3.19 CVF-19

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

Description Names are confusing.
Recommendation Consider using ‘validatorAccountId‘ and ‘validatorAccountData‘ instead

Listing 19:

```
490 let validator_address_padded = CircuitElement::
      ↪ from_fe_with_known_length(

508 let validator_account = AccountContent::from_witness(
```

3.20 CVF-20

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

Description Using both names ‘validator‘ and ‘operator‘ is confusing.
Recommendation Consider using only one of them.

Listing 20:

```
548 // calculate operator's balance_tree root hash from sub tree
      ↪ representation
```

3.21 CVF-21

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

Description This variable has been already declared.

Listing 21:

```
620 let mut operator_account_data = vec![];
```

3.22 CVF-22

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

Recommendation The numbers should be named constants.

Listing 22:

```
661 initial_hash_data.extend(block_number.into_padded_be_bits(256));
    initial_hash_data.extend(validator_address_padded.
       ↪ into_padded_be_bits(256));
    assert_eq!(initial_hash_data.len(), 512);

680     assert_eq!(old_root_be_bits.len(), 256);

697     assert_eq!(final_root_be_bits.len(), 256);

705 pack_bits.extend(global_variables.block_timestamp.
       ↪ into_padded_be_bits(256));
    assert_eq!(pack_bits.len(), 512);
```

3.23 CVF-23

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

Recommendation It would be more efficient to first pack the data into a single array, ensure it is properly domain-separated, and then call SHA-256 once.

Listing 23:

```
701 hash_block = sha256::sha256(cs.namespace(|| "hash with new_root
    ↪ "), &pack_bits)?;

708 hash_block = sha256::sha256(cs.namespace(|| "hash with timestamp
    ↪ "), &pack_bits)?;

715 hash_block = sha256::sha256(cs.namespace(|| "final hash public")
    ↪ , &pack_bits)?;
```

3.24 CVF-24

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

Description Doing this on every chunk is suboptimal.
Recommendation Consider doing once.

Listing 24:

```
745 let max_chunks_powers = generate_powers(

751 let max_chunks_last_coeffs = generate_maxchunk_polynomial::<E>()
    ↪ ;
```

3.25 CVF-25

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

Description This logic is difficult to read and verify.
Recommendation It would be more readable to have a flag in every operation, or a collection of these flags, that determine which branches to use.

Listing 25:

```
//check if type > create_pair
let is_type_ge_create_pair = CircuitElement::less_than_fixed(
    cs.namespace(|| "is_type_ge_create_pair"),
    &create_pair_type,
    &chunk_data.tx_type,
)?;

// check chunk number is 2 or 4
let is_third_or_fifth_chunk = boolean_or(
    cs.namespace(|| "is_third_or_fifth_chunk"),
    &chunk_data.is_chunk_third,
    &chunk_data.is_chunk_fifth,
)?;
```

3.26 CVF-26

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

Recommendation This logic differs from the other assignments and should be documented.

Listing 26:

```
account_audit_path: select_vec_ifeq(
    cs.namespace(|| "account_audit_path"),
    left_side.clone(),
    &cur_side,
    &first.account_audit_path,
    &second.account_audit_path,
```

3.27 CVF-27

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

Description This function is way too large and consists of several stages, that could be extracted into separate functions.
Recommendation Consider extracting stages into separate functions.

Listing 27:

```
fn execute_op<CS: ConstraintSystem<E>>(
```

3.28 CVF-28

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

Recommendation This code should be a designated macro defined near the definition of 'AllocatedOperationData'. Otherwise it is easy to miss some field checks.

Listing 28:

```
let op_data =
    AllocatedOperationData::from_witness(cs.namespace(|| "
      ↪ allocated_operation_data"), op)?;
// ensure op_data is equal to previous
{
```

3.29 CVF-29

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

Description We do not ensure that amount_unpacked, sig_msg fields and some other are equal, is it OK? If those fields are used only in the first chunk it should be both documented and asserted in the code.

Listing 29:

```
AllocatedOperationData::from_witness(cs.namespace(|| "
  ↪ allocated_operation_data"), op)?;
```

3.30 CVF-30

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

Recommendation This should be extracted to a function inside 'AllocatedOperationData'.

Listing 30:

```
// ensure op_data is equal to previous
```

3.31 CVF-31

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

Description Most of the flags computed below are valid only for certain chunks where the branches they are taken from are verified.
Recommendation Consider name or group them accordingly.

Listing 31:

```
// Begin : common constraints before entering the 12 transaction
↪   types
```

3.32 CVF-32

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

Recommendation The logic of building common contracts should be extracted to a function.

Listing 32:

```
// Begin : common constraints before entering the 12 transaction
↪   types
```

3.33 CVF-33

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

Description This number is called 'ordinatoryTokens' in another file.
Recommendation Consider using a consistent notation without redundancy.

Listing 33:

```
&global_variables.constant_number.lp_token_id_start,
```

3.34 CVF-34

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

Recommendation Once deposits and transfers to pair account are forbidden, the token reserves of a pair may not become different from the token balances of this pair, so no need to store the reserves separately from the balances.

Listing 34:

```
// deposit and transfer account can't be pair account
let is_not_pair_account = CircuitElement::equals(
```

3.35 CVF-35

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

Recommendation This code should be moved to the swap operation.

Listing 35:

```
let is_swap = Boolean::from(Expression::equals(
    cs.namespace(|| "is_swap"),
   &global_variables.chunk_data.tx_type.get_number(),
    Expression::u64::<CS>(u64::from(SwapOp::OP_CODE)),
)?);

{
    // swap: op_data.fee = 0;
    // others: op_data.swap_fee = 0
    let swap_fee_is_zero = Boolean::from(AllocatedNum::equals(
        cs.namespace(|| "swap fee is zero"),
       &op_data.swap_fee.get_number(),
       &global_variables.constant_number.explicit_zero.
           ↪ get_number(),
    )?);
    let first_condition = Boolean::and(
        cs.namespace(|| "swap && fee_is_zero"),
       &is_swap,
       &fee_is_zero,
    )?;
    let second_condition = Boolean::and(
        cs.namespace(|| "!swap && swap_fee_is_zero"),
       &is_swap.not(),
       &swap_fee_is_zero,
    )?;
    let is_fee_valid = boolean_or(
        cs.namespace(|| "is_fee_valid"),
       &first_condition,
       &second_condition,
    )?;

    Boolean::enforce_equal(
        cs.namespace(|| "is_fee_valid true"),
       &is_fee_valid,
       &Boolean::constant(true),
    )?;
}
```

3.36 CVF-36

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

Recommendation Most of these flags require only a single constraint and for readability can be computed within the operations where they are used, even if this slightly increases the total number of constraints.

Listing 36:

```
let op_check = OpCheck {
```

3.37 CVF-37

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

Recommendation This is not readable. Use explicit field names.

Listing 37:

```
op_flags.push(deposit.0);
update_pair_bonus_flags.push(deposit.1);
update_user_bonus_flags.push(deposit.2 .0);
update_user_balance_flags.push(deposit.2 .1);
```

3.38 CVF-38

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

Recommendation These commands (and other similar ones) are operation-specific and should be done in the functions responsible for these operations.

Listing 38:

```
op_flags.push(transfer_to_new.0);
update_nonce_flags.push(transfer_to_new.1);
```

3.39 CVF-39

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

Description It is odd that 'full_exit' does not update the nonce.

Listing 39:

```
let full_exit = self.full_exit(
```

3.40 CVF-40

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

Description Including the 'pubdata_properly_copied' flag into the base flags will include it into the 'first_valid_flags' collection, which is redundant, as for the first flag the pubdata doesn't have to be copied.
Recommendation Consider just checking the 'is_equal_pubdata' flag for all chunks except the first one.

Listing 40:

```
    base_valid_flags.push(pubdata_properly_copied);

    base_valid_flags.push(pubdata_is_properly_copied);

    base_valid_flags.push(pubdata_properly_copied);

base_valid_flags.push(pubdata_properly_copied);

base_valid_flags.push(pubdata_properly_copied);

base_valid_flags.push(pubdata_properly_copied);

base_valid_flags.push(pubdata_properly_copied);

base_valid_flags.push(pubdata_properly_copied);
```

3.41 CVF-41

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

Recommendation This variable is used only once and can be removed.

Listing 41:

```
let updated_balance = Expression::constant::<CS>(E::Fr::zero());
```

3.42 CVF-42

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

Description This update is probably redundant since the user will have 0 tokens after the operation.

Listing 42:

```
let should_update_user_bonus = Boolean::and(
```

3.43 CVF-43

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

Description This comment is incorrect: this is the L1 address rather than L2 pubkey (hash).

Listing 43:

```
// update pub_key
```

3.44 CVF-44

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

Description This field is taken from an untrusted branch since rhs is not verified for the first chunk. Therefore it is possible to supply a signature where the recipient ETH address does not match the ETH address in the actual accountId in the tree, i.e. effectively transfer funds not to the intended address.
Recommendation It is recommended to insert all fields from untrusted branches into pubdata which ensures they do not change over chunks.

Listing 44:

```
serialized_tx_bits.extend(rhs.account.address.get_bits_be());
  ↪ //160
```

3.45 CVF-45

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

Recommendation This variable is used only once and can be removed for readability.

Listing 45:

```
let updated_balance = Expression::from(&cur.balance.
  ↪ balance_value.get_number())
   + Expression::from(&op_data.amount_unpacked.get_number());

     updated_balance,
```

3.46 CVF-46

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

Description These checks are expensive.
Recommendation Consider using a,b,c,d variables for mining parameters and use a>b checks from the higher level.
Client Comment a,b,c,d variables have more bits. If we use, extra checks would be needed.

Listing 46:

```
let is_start_block_valid = CircuitElement::less_than_fixed(

let is_start_less_end_block = CircuitElement::less_than_fixed(
```

3.47 CVF-47

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

Description The ‘cur.account‘ field is not authorized since its accountId is not signed. It is possible to supply a malicious pairId which will pass the signature check.

Listing 47:

```
3001 cur.account.pair_info.bonus_info.bonus_token_id = CircuitElement
       ↪ ::conditionally_select(

3009 cur.account.pair_info.bonus_info.mining_start_block_num =

3017 cur.account.pair_info.bonus_info.mining_end_block_num =

3025 cur.account.pair_info.bonus_info.bonus_amount_per_block =

3039 serialized_tx_bits.extend(lhs.account.address.get_bits_be());
       ↪ //160
```

3.48 CVF-48

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

Description The signature is checked for the second chunk only and pairId is not signed. It is thus possible to supply a malicious pairId so that the signature is verified.

Listing 48:

```
3298 serialized_tx_bits.extend(lhs.account.address.get_bits_be());

3821 second_valid_flags.push(op_check.is_sig_verified.clone());
```

3.49 CVF-49

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

Recommendation This check is expensive and can be probably replaced by a>b check.

Listing 49:

```
3355 let is_fee_correctly = CircuitElement::less_than_fixed(
```

3.50 CVF-50

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

Recommendation The ‘lp_reserve_zero‘ would be more readable.

Listing 50:

```
3375 let is_initial_add_liquidity = CircuitElement::equals(
```

3.51 CVF-51

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

Recommendation This should be called 'a1optimal'.

Listing 51:

```
3405 let amount_one_from_desired_amount_zero = AllocatedNum::alloc(
```

3.52 CVF-52

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

Recommendation This should be called 'a0optimal'.

Listing 52:

```
3461 let amount_zero_from_desired_amount_one = AllocatedNum::alloc(
```

3.53 CVF-53

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

Description This code section is too big to compute a single variable.
Recommendation Consider splitting it into subroutines.

Listing 53:

```
3653 let is_not_first_add_correctly = {

3780 };
```

3.54 CVF-54

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

Description The 'tokenLp' supplies and reserves are mixed in the code whereas they seem to be the same thing.
Recommendation Consider refactoring.

Listing 54:

```
3727                 amount_one * token_lp_supply / token_one_reserve

3744 &cur.account.pair_info.token_lp_reserve,
```

3.55 CVF-55

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

Description The updated reserve is calculated from the current balance, rather than from the current reserve. Note, that in witness generator the logic is different and the updates reserve there is generated from the current reserve.
Client Comment Current balance and current reserve are always the same.

Listing 55:

```
let updated_token_zero_reserve = Expression::from(&cur.balance.balance_value.get_number())
  + Expression::from(&op_data.amount_zero.get_number());

cur.account.pair_info.token_zero_reserve =

            updated_token_zero_reserve,

let updated_token_one_reserve = Expression::from(&cur.balance.balance_value.get_number())
  + Expression::from(&op_data.amount_one.get_number());

cur.account.pair_info.token_one_reserve =

            updated_token_one_reserve,

let updated_token_lp_reserve = Expression::from(&cur.balance.balance_value.get_number())
  + Expression::from(&op_data.full_amount.get_number());

let updated_token_lp_reserve_initial = updated_token_lp_reserve.clone()
  + Expression::from(&global_variables.constant_number.one_thousand.get_number());

let updated_token_lp_reserve = Expression::conditionally_select(

      updated_token_lp_reserve_initial,

cur.account.pair_info.token_lp_reserve =

            updated_token_lp_reserve,
```

3.56 CVF-56

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

Description The pair 'accountId' is not signed so it is possible to modify it in the transaction and thus remove liquidity from the wrong account.

Listing 56:

```
pubdata_bits.extend(lhs.account_id.get_bits_be()); // 32

serialized_tx_bits.extend(lhs.account.address.get_bits_be());
    ↪ //160
```

3.57 CVF-57

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

Description The new reserve value is based on the current balance rather than on the current reserve. The logic in the witness generator is different: there the new reserve is based on the current reserve.
Client Comment Current balance and current reserve are always the same.

Listing 57:

```
let updated_token_zero_reserve = Expression::from(&cur.balance.balance_value.get_number())
  - Expression::from(&op_data.amount_zero.get_number());

cur.account.pair_info.token_zero_reserve =

            updated_token_zero_reserve,

let updated_token_one_reserve = Expression::from(&cur.balance.balance_value.get_number())
  - Expression::from(&op_data.amount_one.get_number());

cur.account.pair_info.token_one_reserve =

            updated_token_one_reserve,

let updated_token_lp_reserve = Expression::from(&cur.balance.balance_value.get_number())
  - Expression::from(&op_data.amount_unpacked.get_number());

cur.account.pair_info.token_lp_reserve =

            updated_token_lp_reserve,
```

3.58 CVF-58

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

Description This name is confusing as it is unclear what correctness means.

Listing 58:

```
third_valid_flags.push(op_check.is_d_correct.clone());
```

3.59 CVF-59

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

Recommendation This check could have been handled by a and b variables.

Listing 59:

```
4207 let is_token_lp_enough_in_pair = CircuitElement::less_than_fixed
     ↪ (
```

3.60 CVF-60

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

Description The pair 'accountId' is not signed so it is possible to change it in the transaction and attempt to swap with a wrong pair.

Listing 60:

```
4322 pubdata_bits.extend(lhs.account_id.get_bits_be()); // 32

4349 serialized_tx_bits.extend(lhs.account.address.get_bits_be());
     ↪ //160
```

3.61 CVF-61

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

Recommendation These should be named constants.

Listing 61:

```
4400 let result = amount_in * 9995usize / 10000usize;
```

3.62 CVF-62

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

Recommendation This should be a named constant.

Listing 62:

```
4515 E::Fr::CAPACITY as usize - 10 - params::VALID_AMOUNT_BIT_WIDTH,
```

3.63 CVF-63

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

Description This flag has been already computed as 'is_token_one_slippage_valid'.

Listing 63:

```
let is_slippage_valid = CircuitElement::less_than_fixed(
     cs.namespace(|| "amount_out>=amount_out_min"),
     &op_data.amount_one,
     &op_data.amount_one_min_unpacked,
)?
.not();
```

3.64 CVF-64

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

Description The new reserve values are based on the current balance rather than on the current reserve. In the witness the logic is different. There the new reserve is based on the current reserve.
Client Comment Current balance and current reserve are always the same.

Listing 64:

```
let updated_token_out_reserve = Expression::from(&cur.balance.
    ↪ balance_value.get_number())
     - Expression::from(&op_data.amount_one.get_number());

cur.account.pair_info.token_zero_reserve =

            updated_token_out_reserve.clone(),

cur.account.pair_info.token_one_reserve =

            updated_token_out_reserve,

let updated_token_in_reserve = Expression::from(&cur.balance.
    ↪ balance_value.get_number())
     + Expression::from(&amount_in_pool_ce.get_number());

cur.account.pair_info.token_zero_reserve =

            updated_token_in_reserve.clone(),

cur.account.pair_info.token_one_reserve =

            updated_token_in_reserve,
```

3.65 CVF-65

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

Description The 'account_id' field is not signed and can not be trusted. A malicious relayer may replace it with a wrong ID.

Listing 65:

```
pubdata_bits.extend(lhs.account_id.get_bits_be());

serialized_tx_bits.extend(lhs.account.address.get_bits_be());
serialized_tx_bits.extend(lhs.account.pair_info.token_lp.
    ↪ get_bits_be());
```

3.66 CVF-66

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

Description The name is vague. '

Listing 66:

```
let mut sponge_output = rescue::rescue_hash(
```

3.67 CVF-67

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

Recommendation This should be an assertion on params rather than on 'sponge_output'.

Listing 67:

```
assert_eq!(sponge_output.len(), 1);

assert_eq!(account_leaf_hash.len(), 1);
```

3.68 CVF-68

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

Description Why is 'CircuitElement' created? Rescue is able to take field elements as inputs.

Listing 68:

```
let pair_info_hash_ce =
    CircuitElement::from_number(cs.namespace(|| "
    ↪ pair_info_hash_ce"), pair_info_hash)?;
```

3.69 CVF-69

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

Description This check seems to be redundant as nothing in the code depends on how many elements comprise the witness

Listing 69:

```
assert_eq!(account_data_packed_as_field_elements.len(), 5);
```

3.70 CVF-70

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

Recommendation Should be "==" or "<=" here, instead of ">=".

Listing 70:

```
assert!(index.len() >= audit_path.len());
```

3.71 CVF-71

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

Description It is not ensured, that audit_path.len() >= length_to_root.
Recommendation Consider adding such assert.

Listing 71:

```
let audit_path = &audit_path[0..length_to_root];
```

3.72 CVF-72

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

Description This is expensive.
Recommendation Consider calculating the sum and checking it is not zero.

Listing 72:

```
for (i, bool_x) in x.iter().enumerate() {
    result = Boolean::and(
        cs.namespace(|| format!("multi or iteration number: {}",
        ↪  i)),
        &result.not(),
        &bool_x.not(),
    )?
    .not();
}
```

3.73 CVF-73

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

Description This loop is very similar to what the 'continue_leftmost_subroot_to_root' function does.
Recommendation Consider using this function.

Listing 73:

```
// will hash top of the tree where RHS is always an empty tree
for i in processable_fees_tree_depth..params::balance_tree_depth
  ↪  () {
```

3.74 CVF-74

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

Description The variable 'tmp' is redundant, as its value is used only once.

Listing 74:

```
let tmp = sponge_output.pop().expect("must get a single element
  ↪  ");
node_hash = tmp;
```

3.75 CVF-75

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

Recommendation This check is expensive and should probably be computed on the higher level.

Listing 75:

```
let is_to_geq_from =
    CircuitElement::less_than_fixed(cs.namespace(|| "to geq from"), &to, &from)?.not();
```

3.76 CVF-76

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

Recommendation It would be more precise to multiply by the duration first, and then divide by 'total_lp'.

Listing 76:

```
                    bonus_amount_per_block * factor / total_lp

let increased_accumulate_bonus_per_lp = duration.mul(
    cs.namespace(|| "duration multiply bonus_amount_per_block"),
    &scaled_bonus_amount_per_block_per_lp_ce.get_number(),
)?;
```

3.77 CVF-77

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

Description The indentation of these lines is incorrect.
Recommendation Consider fixing it to make the code easier to read.

Listing 77:

```
5670      cs.namespace(|| "update accumulated_bonus"),
         &op_data.accumulate_bonus_per_lp_until_last_block,
         &cur.account
             .pair_info
             .bonus_info
             .accumulated_bonus_per_lp_until_last_block,
         &condition,
     )
     .unwrap();
```

3.78 CVF-78

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

Recommendation If this is a global parameter no need to pass it to every call.

Listing 78:

```
5695 bonus_scaling_factor: &CircuitElement<E>,
```

3.79 CVF-79

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

Recommendation Consider adding an assert to ensure this subtraction doesn't underflow.

Listing 79:

```
5743 .sub(
```

3.80 CVF-80

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

Recommendation A better name would be "update_balance_value_if" as the function updates the balance value only if a certain condition is met.

Listing 80:

```
fn update_balance_value<E, CS, EX>(
```

3.81 CVF-81

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

Recommendation Consider adding explicit assert statements for these preconditions.

Listing 81:

```
//  We have constrained the length of input parameter to prevent the multiplication from overflowing

// x*y and u*v + rem will not overflow, have been constrained before

// x*y and u*u + rem will not overflow, have been constrained before
```

3.82 CVF-82

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** primitives.rs

Description It is a bit odd that 0 = z/0 for any z.
Recommendation Maybe it should be only for z=0.
Client Comment This is for compatibility when the divisor is equal to zero.

Listing 82:

```
let is_v_zero_valid = Boolean::and(cs.namespace(|| "v==0 and u==0"), &is_v_zero, &is_u_zero)?;
```

3.83 CVF-83

- **Severity** Minor
- **Category** Documentation
- **Status** Fixed
- **Source** primitives.rs

Recommendation Should be ‘remainder‘.

Listing 83:

```
    let reminder = AllocatedNum::alloc(cs.namespace(|| "allocate reminder"), || {

let reminder = AllocatedNum::alloc(cs.namespace(|| "xy−uu"), || {
```

3.84 CVF-84

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

Description This constraint is true when u=v=0 and arbitrary x,y. Is it OK?

Listing 84:

```
pub fn is_mixed_division_correctly<E: Engine, CS: ConstraintSystem<E>>(
```

3.85 CVF-85

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

Description This function’s API is complicated.
Recommendation Consider implementing a div function that takes two arguments x and y and returning (x/y , x mod y).

Listing 85:

```
pub fn is_mixed_division_correctly<E: Engine, CS: ConstraintSystem<E>>(
```

3.86 CVF-86

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

Description The name ‘squirt‘ has obscene connotations.
Recommendation Consider renaming.

Listing 86:

```
pub fn is_squirt_correctly<E: Engine, CS: ConstraintSystem<E>>(
```

3.87 CVF-87

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

Description This function implements approach that could be suboptimal. It would be simpler to just ensure that the sum of the bits after the valid nits number is zero.

Listing 87:

```
pub fn is_bit_length_valid<E: Engine, CS: ConstraintSystem<E>>(
```

3.88 CVF-88

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

Recommendation Should be "<=".

Listing 88:

```
if ce.get_bits_le().len() < valid_bits_num {
```

3.89 CVF-89

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

Recommendation This function should be documented.

Listing 89:

```
pub fn no_unsigned_integer_overflow<E: Engine, CS:
    ↪ ConstraintSystem<E>>(
```

3.90 CVF-90

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

Description This code calculates 2^bit_length - 1 modulo field size, while it is guaranteed that the result will not overflow.
Recommendation Consider calculating the desired value as a big number and then converting to a field element.

Listing 90:

```
194 let two = E::Fr::from_str("2").unwrap();
    let power = E::Fr::from_str(&bit_length.to_string()).unwrap();
    let mut max_fr = two.pow(&power.into_repr());
    max_fr.sub_assign(&E::Fr::one());
```

3.91 CVF-91

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

Description Converting numbers to field element via strings is weird.
Recommendation Consider using more direct ways.

Listing 91:

```
194 let two = E::Fr::from_str("2").unwrap();
    let power = E::Fr::from_str(&bit_length.to_string()).unwrap();
```

3.92 CVF-92

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

Recommendation The words "zero" and "one" in names looks like the tokens are in order. Letters "A" and "B" would be better.

Listing 92:

```
43 pub amount_zero_desire: u128,
   pub amount_zero_min: u128,
   pub amount_one_desire: u128,
   pub amount_one_min: u128,

48 pub token_zero: u32,
   pub token_one: u32,
```

3.93 CVF-93

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

Recommendation Consider using arrays instead of series of named fields to make code simpler and easier to read.

Listing 93:

```
pub pair_before_a: OperationBranch<E>,
pub user_before_a: OperationBranch<E>,

pub pair_before_b: OperationBranch<E>,
pub user_before_b: OperationBranch<E>,

pub pair_before_c: OperationBranch<E>,
pub user_before_c: OperationBranch<E>,

pub pair_before_d: OperationBranch<E>,
pub user_before_d: OperationBranch<E>,

pub pair_before_e: OperationBranch<E>,
pub user_before_e: OperationBranch<E>,

pub pair_before_f: OperationBranch<E>,
pub user_before_f: OperationBranch<E>,

pub pair_after: OperationBranch<E>,
pub user_after: OperationBranch<E>,

pub before_a_root: Option<E::Fr>,
pub before_b_root: Option<E::Fr>,
pub before_c_root: Option<E::Fr>,
pub before_d_root: Option<E::Fr>,
pub before_e_root: Option<E::Fr>,
pub before_f_root: Option<E::Fr>,

pub after_root: Option<E::Fr>,
```

3.94 CVF-94

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 94:

```
pub user_before_a: OperationBranch<E>,

pub pair_before_b: OperationBranch<E>,

pub user_before_c: OperationBranch<E>,

pub pair_before_d: OperationBranch<E>,

pub user_before_e: OperationBranch<E>,

pub pair_before_f: OperationBranch<E>,
```

3.95 CVF-95

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** add_liquidity.rs

Description Overflow might be possible here as the ‘AllocatedOperationData‘ constructor requires that some of these values fit into 126 bits rather than 128 bits.

Listing 95:

```
let add_liquidity_data = AddLiquidityData {
```

3.96 CVF-96

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

Recommendation These checks should be done on the fields of 'add_liquidity' before constructing 'add_liquidity_data'.

Listing 96:

```
assert!(add_liquidity_data.amount_zero_desire <=
    ↪ MAX_AMOUNT_VALUE);
assert!(add_liquidity_data.amount_one_desire <= MAX_AMOUNT_VALUE
    ↪ );
assert!(add_liquidity_data.amount_zero_min <= MAX_AMOUNT_VALUE);
assert!(add_liquidity_data.amount_one_min <= MAX_AMOUNT_VALUE);
```

3.97 CVF-97

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

Recommendation The minimum amounts should be bounded from the top by the desire amounts rather than by the "MAX_AMOUNT_VALUE" constant.

Listing 97:

```
assert!(add_liquidity_data.amount_zero_min <= MAX_AMOUNT_VALUE);
assert!(add_liquidity_data.amount_one_min <= MAX_AMOUNT_VALUE);
```

3.98 CVF-98

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

Description Including the pair address into the public data is redundant, as it could be derived from the LP token ID.

Listing 98:

```
append_be_fixed_width(
    &mut pubdata_bits,
    &self.pair_before_a.address.unwrap(),
    ACCOUNT_ID_BIT_WIDTH,
);
```

3.99 CVF-99

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

Description Including the underlying token IDs into the public data is redundant, as they could be derived from the LP token ID.

Listing 99:

```
append_be_fixed_width(
    &mut pubdata_bits,
    &self
        .pair_before_a
        .witness
        .account_witness
        .pair_info
        .token_zero
        .unwrap(),
    TOKEN_BIT_WIDTH,
);

append_be_fixed_width(
    &mut pubdata_bits,
    &self
        .pair_before_a
        .witness
        .account_witness
        .pair_info
        .token_one
        .unwrap(),
    TOKEN_BIT_WIDTH,
);
```

3.100 CVF-100

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 100:

```
let operation_zero = Operation {
let operation_one = Operation {
let operation_two = Operation {
let operation_three = Operation {
let operation_four = Operation {
let operation_five = Operation {
```

3.101 CVF-101

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 101:

```
225        chunk: Some(Fr::from_str("0").unwrap()),

240        chunk: Some(Fr::from_str("1").unwrap()),

255        chunk: Some(Fr::from_str("2").unwrap()),

270        chunk: Some(Fr::from_str("3").unwrap()),

285        chunk: Some(Fr::from_str("4").unwrap()),

324 let min_liquidity_fe = Fr::from_str(&MIN_LIQUIDITY.to_string()).
      ↪ unwrap();

328 let account_address_user_fe = Fr::from_str(&add_liquidity.
      ↪ account_id.to_string()).unwrap();

330        Fr::from_str(&add_liquidity.account_pair_id.to_string()).
             ↪ unwrap();
    let token_zero_fe = Fr::from_str(&add_liquidity.token_zero.
      ↪ to_string()).unwrap();
    let token_one_fe = Fr::from_str(&add_liquidity.token_one.
      ↪ to_string()).unwrap();
    let token_lp_fe = Fr::from_str(&add_liquidity.lp_token.to_string
      ↪ ()).unwrap();
    let block_number = Fr::from_str(&add_liquidity.block_number.
      ↪ to_string()).unwrap();

337        Fr::from_str(&add_liquidity.amount_zero_desire.to_string()).
             ↪ unwrap();
    let amount_zero_min_fe = Fr::from_str(&add_liquidity.
      ↪ amount_zero_min.to_string()).unwrap();

340        Fr::from_str(&add_liquidity.amount_one_desire.to_string()).
             ↪ unwrap();
    let amount_one_min_fe = Fr::from_str(&add_liquidity.
      ↪ amount_one_min.to_string()).unwrap();
```

(... 381, 425, 504, 836)

3.102 CVF-102

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

Description This code fragment repeats several times with slight modifications.
Recommendation Consider extracting a utility function.

Listing 102:

```
let amount_zero_desire_bits = FloatConversions::to_float(
        add_liquidity.amount_zero_desire,
        AMOUNT_EXPONENT_BIT_WIDTH,
        AMOUNT_MANTISSA_BIT_WIDTH,
        10,
)
.unwrap();
let amount_zero_desire_encoded: Fr =
        le_bit_vector_into_field_element(&amount_zero_desire_bits);
```

3.103 CVF-103

- **Severity** Major
- **Category** Suboptimal
- **Status** Info
- **Source** add_liquidity.rs

Description This chunk does not use a,b,c,d variables and the available range checks for them, whereas they can save significantly many constraints.

Listing 103:

```
// 1. update pair's token_zero
```

3.104 CVF-104

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

Description The variable name "pair_audit_balance_before_a" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_zero_before_a".

Listing 104:

```
let (pair_audit_path_before_a, pair_audit_balance_before_a) =
    ↪ get_audits(
```

3.105 CVF-105

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

Description The variable name "user_audit_balance_before_a" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_zero_before_a".

Listing 105:

```
let (user_audit_path_before_a, user_audit_balance_before_a) =
```

3.106 CVF-106

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** add_liquidity.rs

Description Underflow is possible here.

Listing 106:

```
let lp = lp - MIN_LIQUIDITY;
```

3.107 CVF-107

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

Description The variables are redundant. Just return the expression values.

Listing 107:

```
let a0_fe = biguint_to_fe(a0);
let a1_fe = biguint_to_fe(a1);
let lp_fe = biguint_to_fe(lp);

(a0_fe, a1_fe, lp_fe)
```

3.108 CVF-108

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

Description The suffix '_b' here and in other variables is redundant as any balance is changed only once. It will be more readable to use just ‘before_update‘ and ‘after_update‘

Listing 108:

```
467 pair_account_witness_before_b,
```

3.109 CVF-109

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

Description Storing reserves alongside with balances would be redundant if direct token transfers to pair accounts would be forbidden.
Recommendation Consider forbidding such transfers and removing the reserve attributes from the account state.

Listing 109:

```
476 acc.pair_info.token_zero_reserve.add_assign(&amount_zero);

479 bal.balance_value.add_assign(&amount_zero);

533 acc.pair_info.token_one_reserve.add_assign(&amount_one);

536 bal.balance_value.add_assign(&amount_one);
```

3.110 CVF-110

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

Description The logic implemented here is different from what is implemented in the circuit. Here: new_balance = old_balance ± amount new_reserve = old_reserve ± amount In the circuit: new_balance = old_balance ± amount new_reserve = old_balance ± amount Note, that the circuit ignores the current reserve when calculating the new one.
Client Comment Current balance and current reserve are always the same.

Listing 110:

```
acc.pair_info.token_zero_reserve.add_assign(&amount_zero);

bal.balance_value.add_assign(&amount_zero);

acc.pair_info.token_one_reserve.add_assign(&amount_one);

bal.balance_value.add_assign(&amount_one);

acc.pair_info.token_lp_reserve.add_assign(&amount_lp);

bal.balance_value.add_assign(&amount_lp);
```

3.111 CVF-111

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

Description The variable name "pair_audit_balance_before_b" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_zero_before_b".

Listing 111:

```
let (pair_audit_path_before_b, pair_audit_balance_before_b) =
    ↪ get_audits(
```

3.112 CVF-112

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

Description The variable name "user_audit_balance_before_b" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_zero_before_b".

Listing 112:

```
491 let (user_audit_path_before_b, user_audit_balance_before_b) =
```

3.113 CVF-113

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

Description The variable name "pair_audit_balance_before_c" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_one_before_c".

Listing 113:

```
516 let (pair_audit_path_before_c, pair_audit_balance_before_c) =
```

3.114 CVF-114

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

Description The variable name "user_audit_balance_before_c" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_one_before_c".

Listing 114:

```
519 let (user_audit_path_before_c, user_audit_balance_before_c) =
```

3.115 CVF-115

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

Description The variable name "pair_audit_balance_before_d" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_one_before_d".

Listing 115:

```
542 let (pair_audit_path_before_d, pair_audit_balance_before_d) =
```

3.116 CVF-116

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

Description The variable name "user_audit_balance_before_d" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_one_before_d".

Listing 116:

```
545 let (user_audit_path_before_d, user_audit_balance_before_d) =
```

3.117 CVF-117

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

Description The variable name "pair_audit_balance_before_e" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_before_e".

Listing 117:

```
570 let (pair_audit_path_before_e, pair_audit_balance_before_e) =
```

3.118 CVF-118

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

Description The variable name "user_audit_balance_before_e" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_before_e".

Listing 118:

```
let (user_audit_path_before_e, user_audit_balance_before_e) =
```

3.119 CVF-119

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

Description We did not review these functions.

Listing 119:

```
let accumulate = calculate_updated_bonus_in_pair(acc,
    ↪ block_number);

    calculate_claimed_bonus(&bal, &accumulate_bonus_in_pair);
```

3.120 CVF-120

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

Description Storing LP token reserve alongside with balance would be redundant if direct token transfers to pair accounts would be forbidden.
Recommendation Consider forbidding such transfers and removing the LP reserve attribute from the account state.

Listing 120:

```
acc.pair_info.token_lp_reserve.add_assign(&amount_lp);
if is_initial_add_liquidity {
    acc.pair_info.token_lp_reserve.add_assign(&min_liquidity_fe);
}
```

```
bal.balance_value.add_assign(&amount_lp);
if is_initial_add_liquidity {
    bal.balance_value.add_assign(&min_liquidity_fe);
}
```

3.121 CVF-121

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** add_liquidity.rs

Description The logic implemented here is different from what is implemented in the circuit. Here: new_balance = old_balance ± amount new_reserve = old_reserve ± amount In the circuit: new_balance = old_reserve ± amount new_reserve = old_reserve ± amount Note, that the circuit ignores the current balance when calculating the new one.
Client Comment Current balance and current reserve are always the same.

Listing 121:

```
acc.pair_info.token_lp_reserve.add_assign(&amount_lp);
```

```
bal.balance_value.add_assign(&amount_lp);
```

3.122 CVF-122

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** add_liquidity.rs

Description This effectively makes the reserve tokens equivalent to MIN_LIQUIDITY forever stuck on the pair account as they are never returned back to the user. For tokens with small decimals this can be a serious loss.
Client Comment We will always add liquidity first after creating the pair.

Listing 122:

```
595 acc.pair_info.token_lp_reserve.add_assign(&min_liquidity_fe);
```

3.123 CVF-123

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

Description The 'token_lp' balance at 'pairId' should decrease rather than increase.

Listing 123:

```
599 bal.balance_value.add_assign(&amount_lp);
```

3.124 CVF-124

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

Description Balance update here is redundant as the LP total supply is already stored in a separate attribute.
Recommendation Consider removing the balance update logic from here.

Listing 124:

```
599 bal.balance_value.add_assign(&amount_lp);
600 if is_initial_add_liquidity {
        bal.balance_value.add_assign(&min_liquidity_fe);
    }
```

3.125 CVF-125

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

Description The variable name "pair_audit_balance_before_f" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_before_f".

Listing 125:

```
let (pair_audit_path_before_f, pair_audit_balance_before_f) =
```

3.126 CVF-126

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

Description The variable name "user_audit_balance_before_f" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_before_f".

Listing 126:

```
let (user_audit_path_before_f, user_audit_balance_before_f) =
```

3.127 CVF-127

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

Recommendation The variable names "pair_audit_balance_path_after" and "user_audit_balance_path_after" should not have the "path" word for consistency with other similar variables.

Listing 127:

```
let (pair_audit_path_after, pair_audit_balance_path_after) =

let (user_audit_path_after, user_audit_balance_path_after) =
```

3.128 CVF-128

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

Description The variable name "pair_audit_balance_path_after" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_after".

Listing 128:

```
650 let (pair_audit_path_after, pair_audit_balance_path_after) =
```

3.129 CVF-129

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

Description The variable name "user_audit_balance_path_after" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_after".

Listing 129:

```
653 let (user_audit_path_after, user_audit_balance_path_after) =
```

3.130 CVF-130

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

Description Including all this information into the public data is redundant, as it could be derived from the LP token ID.

Listing 130:

```
append_be_fixed_width(
    &mut pubdata_bits,
    &self.after.address.unwrap(),
    ACCOUNT_ID_BIT_WIDTH,
);

append_be_fixed_width(
    &mut pubdata_bits,
    &self
        .after
        .witness
        .account_witness
        .pair_info
        .token_zero
        .unwrap(),
    TOKEN_BIT_WIDTH,
);

append_be_fixed_width(
    &mut pubdata_bits,
    &self
        .after
        .witness
        .account_witness
        .pair_info
        .token_one
        .unwrap(),
    TOKEN_BIT_WIDTH,
);

append_be_fixed_width(
    &mut pubdata_bits,
    &self.after.witness.account_witness.address.unwrap(),
    ETH_ADDRESS_BIT_WIDTH,
);
```

3.131 CVF-131

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

Recommendation The value 7 should ba a named constant.

Listing 131:

```
commitment[7] = true;
```

3.132 CVF-132

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

Recommendation Value 256 should be a named constant.

Listing 132:

```
let signer_pub_key_packed = &[Some(false); 256];
```

3.133 CVF-133

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** create_pair.rs

Recommendation This code should be removed.

Listing 133:

```
log::debug!(
    "acc_path{} \n bal_path {} ",
    self.before.witness.account_path.len(),
    self.before.witness.balance_subtree_path.len()
);
```

3.134 CVF-134

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 134:

```
    chunk: Some(Fr::from_str("0").unwrap()),

    chunk: Some(Fr::from_str(&chunk.to_string()).unwrap()),

let token_zero = Fr::from_str(&create_pair.token_zero.to_string()).unwrap();
let token_one = Fr::from_str(&create_pair.token_one.to_string()).unwrap();
let token_lp = Fr::from_str(&create_pair.token_lp.to_string()).unwrap();
let account_address_fe = Fr::from_str(&create_pair.account_id.to_string()).unwrap();

    tx_type: Some(Fr::from_str("9").unwrap()),
```

3.135 CVF-135

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 135:

```
pub pair_intermediate: OperationBranch<E>,

pub user_before: OperationBranch<E>,
```

3.136 CVF-136

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 136:

```
let operation_zero = Operation {
let operation_one = Operation {
let operation_two = Operation {
```

3.137 CVF-137

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 137:

```
    chunk: Some(Fr::from_str("0").unwrap()),

    chunk: Some(Fr::from_str("1").unwrap()),

    chunk: Some(Fr::from_str("2").unwrap()),

    Fr::from_str(&mining_maintenance.account_pair_id.to_string()).unwrap();

    Fr::from_str(&mining_maintenance.user_account_id.to_string()).unwrap();
let bonus_id_fe = Fr::from_str(&mining_maintenance.bonus_id.to_string()).unwrap();

    Fr::from_str(&mining_maintenance.bonus_per_block.to_string()).unwrap();

    Fr::from_str(&mining_maintenance.start_block_number.to_string()).unwrap();

    Fr::from_str(&mining_maintenance.end_block_number.to_string()).unwrap();

    Fr::from_str(&mining_maintenance.cur_block_number.to_string()).unwrap();

        > Fr::from_str(&max_ordinary_token_id().to_string()).unwrap(),

        acc.nonce.add_assign(&Fr::from_str("1").unwrap());

    tx_type: Some(Fr::from_str("7").unwrap()),
```

3.138 CVF-138

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

Recommendation This must be a named constant.

Listing 138:

```
10,
```

3.139 CVF-139

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

Recommendation The AccountID of the Maintainer should be a named constant.

Listing 139:

```
&& mining_maintenance.user_account_id == 1 // assure only
  ↪ specific account can execute this TX.
```

3.140 CVF-140

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

Recommendation This check should be made a common function ‘checkIfAccountIsPair‘ if this is the purpose.

Listing 140:

```
assert!(
    pair_account_witness_before.pair_info.token_lp.unwrap()
        > Fr::from_str(&max_ordinary_token_id().to_string()).
           ↪ unwrap(),
    "incorrect pair_account",
);
```

3.141 CVF-141

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 141:

```
pub pair_intermediate: OperationBranch<E>,

pub user_before: OperationBranch<E>,
```

3.142 CVF-142

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

Recommendation This check should be done on the field of 'withdraw' before constructing 'withdraw_data'.

Listing 142:

```
assert!(withdraw_data.amount < MAX_AMOUNT_VALUE);
```

3.143 CVF-143

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 143:

```
let operation_zero = Operation {

let operation_one = Operation {

let rest_operations = (2..WithdrawOp::CHUNKS).map(|chunk| Operation {
```

3.144 CVF-144

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 144:

```
145        chunk: Some(Fr::from_str("0").unwrap()),

159        chunk: Some(Fr::from_str("1").unwrap()),

174        chunk: Some(Fr::from_str(&chunk.to_string()).unwrap()),

197       &Fr::from_str("3").unwrap(), //Corresponding tx_type

251 let account_address_fe = Fr::from_str(&withdraw.account_address.to_string()).unwrap();
    let token_fe = Fr::from_str(&withdraw.token.to_string()).unwrap();
    let amount_as_field_element = Fr::from_str(&withdraw.amount.to_string()).unwrap();
    let pair_account_fe = Fr::from_str(&withdraw.pair_account.to_string()).unwrap();

258 let fee_as_field_element = Fr::from_str(&withdraw.fee.to_string()).unwrap();

270 let block_number = Fr::from_str(&withdraw.block_number.to_string()).unwrap();

321             acc.nonce.add_assign(&Fr::from_str("1").unwrap());

435        tx_type: Some(Fr::from_str("3").unwrap()),
```

3.145 CVF-145

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

Description This comment is confusing.
Recommendation Consider removing it.

Listing 145:

```
191 //redundant code
```

3.146 CVF-146

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 146:

```
// External deps

            chunk: Some(Fr::from_str("1").unwrap()),

            chunk: Some(Fr::from_str(&chunk.to_string()).unwrap
              ↪ ()),

        let account_address_fe = Fr::from_str(&deposit.
          ↪ account_address.to_string()).unwrap();
        let pair_account_fe = Fr::from_str(&deposit.pair_account
          ↪ .to_string()).unwrap();
        let token_fe = Fr::from_str(&deposit.token.to_string()).
          ↪ unwrap();
        let amount_as_field_element = Fr::from_str(&deposit.
          ↪ amount.to_string()).unwrap();

        let block_number = Fr::from_str(&deposit.block_number.
          ↪ to_string()).unwrap();

            tx_type: Some(Fr::from_str("1").unwrap()),
```

3.147 CVF-147

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 147:

```
pub pair_intermediate: OperationBranch<E>,

pub user_before: OperationBranch<E>,
```

3.148 CVF-148

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

Recommendation The value 7 should be a named constant.

Listing 148:

```
commitment[7] = true;
```

3.149 CVF-149

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

Recommendation The value 256 should be a named constant.

Listing 149:

```
let signer_pub_key_packed = &[Some(false); 256]; //doesn't
  ↪ matter for deposit
```

3.150 CVF-150

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** deposit.rs

Recommendation This code should be removed.

Listing 150:

```
log::debug!(
    "pair acc_path{} \n pair bal_path {} ",
    self.pair_before.witness.account_path.len(),
    self.pair_before.witness.balance_subtree_path.len()
);
log::debug!(
    "user acc_path{} \n user bal_path {} ",
    self.user_before.witness.account_path.len(),
    self.user_before.witness.balance_subtree_path.len()
);
```

3.151 CVF-151

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 151:

```
let operation_zero = Operation {

let operation_one = Operation {

let rest_operations = (2..DepositOp::CHUNKS).map(|chunk|
    ↪ Operation {
```

3.152 CVF-152

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

Recommendation This subroutine must be a separate function called by others.

Listing 152:

```
let is_token_lp = { deposit.token > (max_ordinary_token_id() as
    ↪ u32) };
```

3.153 CVF-153

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

Description We did not review these functions.

Listing 153:

```
    check_pair_circuit_account_not_overflow(&tree, deposit.
        ↪ pair_account, vec![]);

check_circuit_account_not_overflow(&tree, deposit.
    ↪ account_address, vec![deposit.token]);
```

3.154 CVF-154

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 154:

```
pub pair_intermediate: OperationBranch<E>,

pub user_before: OperationBranch<E>,
```

3.155 CVF-155

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 155:

```
            .map(|amount| Fr::from_str(&amount.0.to_string()).unwrap())

        chunk: Some(Fr::from_str("0").unwrap()),

        chunk: Some(Fr::from_str("1").unwrap()),

        chunk: Some(Fr::from_str(&chunk.to_string()).unwrap()),

let account_address_fe = Fr::from_str(&full_exit.account_address.to_string()).unwrap();
let pair_account_fe = Fr::from_str(&full_exit.pair_account.to_string()).unwrap();
let token_fe = Fr::from_str(&full_exit.token.to_string()).unwrap();

let block_number = Fr::from_str(&full_exit.block_number.to_string()).unwrap();

        tx_type: Some(Fr::from_str("5").unwrap()),
```

3.156 CVF-156

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

Recommendation Value 256 should be a predefined constant.

Listing 156:

```
r_packed: vec![Some(false); 256],
s: vec![Some(false); 256],

signer_pub_key_packed: vec![Some(false); 256],

signer_pub_key_packed: vec![Some(false); 256],

signer_pub_key_packed: vec![Some(false); 256],
```

3.157 CVF-157

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 157:

```
let operation_zero = Operation {

let operation_one = Operation {

let rest_operations = (2..FullExitOp::CHUNKS).map(|chunk|
    ↪ Operation {
```

3.158 CVF-158

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

Recommendation 6 paths is probably too many as we update at most two accounts in this operation.

Listing 158:

```
FullExitWitness {
```

3.159 CVF-159

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

Recommendation The named constant for 'full_exit' should be used here.

Listing 159:

```
tx_type: Some(Fr::from_str("5").unwrap()),
```

3.160 CVF-160

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

Recommendation In case the LP token start ID would be a power of two, this check could be done easier by just checking that some of top bits are non-zero.

Listing 160:

```
let is_lp_token_first_condition = CircuitElement::
    ↪ less_than_fixed(
      cs.namespace(|| "token_id >= lp_token_id_start"),
      &token_id,
      &lp_token_id_start,
)?
.not();
```

3.161 CVF-161

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

Recommendation In case the LP token end ID would be a power of two, this check could be done easier by just checking that all the top bits are zero.

Listing 161:

```
let is_lp_token_second_condition = CircuitElement::
    ↪ less_than_fixed(
      cs.namespace(|| "token_id < lp_token_id_end"),
      &token_id,
      &lp_token_id_end,
)?;
```

3.162 CVF-162

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

Recommendation As there is no chunking here, consider using a single variable to store the LPtokenId instead of two.

Listing 162:

```
|| "from_branch.token == pair_account_lp_branch.token_lp",
```

3.163 CVF-163

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

Description This parameter is redundant as it can be derived from 'pair_account_id'.

Listing 163:

```
lp_token_id: TokenId,
```

3.164 CVF-164

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

Description Converting numbers to field elements via strings is weird.
Recommendation Consider using more direct ways.

Listing 164:

```
324 let account_id_fe = Fr::from_str(&account_id.to_string()).unwrap
    ↪ ();

334 let pair_account_id_fe = Fr::from_str(&pair_account_id.to_string
    ↪ ()).unwrap();

340 let token_lp_fe = Fr::from_str(&lp_token_id.to_string()).unwrap
    ↪ ();
```

3.165 CVF-165

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

Description The words 'reserves' and 'supply' are confused many times in the code.
Recommendation Consider unifying naming.

Listing 165:

```
let pair_token_lp_supply =
    fe_to_biguint(&pair_account_witness.pair_info.
      ↪ token_lp_reserve.unwrap());
```

3.166 CVF-166

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

Description These part of pubdata are redundant, as they could be derived from the pair address.

Listing 166:

```
&pair_account_witness.pair_info.token_zero.unwrap(),

&pair_account_witness.pair_info.token_one.unwrap(),
```

3.167 CVF-167

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

Recommendation Tests should be moved to a separate file.

Listing 167:

```
#[cfg(test)]
```

3.168 CVF-168

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

Recommendation If the number of ordinary tokens is a power of two, it would be enough to just check that the top bits are all zeros.

Listing 168:

```
let is_ordinary_token = CircuitElement::less_than_fixed(
    cs.namespace(|| "is ordinary token"),
   &branch.token,
   &number_of_ordinary_tokens,
)?;
```

3.169 CVF-169

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

Description Converting number to field element through string is weird.
Recommendation Consider using more direct ways.

Listing 169:

```
let account_address_fe = Fr::from_str(&account_id.to_string()).
    ↪ unwrap();

let token_id_fe = Fr::from_str(&token_id.to_string()).unwrap();
```

3.170 CVF-170

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

Description The term "account address" is used for both, real Ethereum -style account address and Lp account ID.
Recommendation Consider using different terms.

Listing 170:

```
let account_address_fe = Fr::from_str(&account_id.to_string()).
    ↪ unwrap();
let account_address = account_tree
```

3.171 CVF-171

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

Recommendation This should be a named constant.

Listing 171:

```
192 assert_eq!(pubdata_commitment.len(), 592);
```

3.172 CVF-172

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

Recommendation The CAPACITY constant should be used here.

Listing 172:

```
202 hash_result[0] &= 0x1f; // temporary solution, this nullifies
    ↪ top bits to be encoded into field element correctly 253
    ↪ bits
```

3.173 CVF-173

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

Recommendation This should be put to a separate test file.

Listing 173:

```
225 #[cfg(test)]
```

3.174 CVF-174

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

Recommendation Consider using arrays instead of series of named fields to make code simpler and easier to read.

Listing 174:

```
pub pair_before_a: OperationBranch<E>,
pub pair_before_b: OperationBranch<E>,
pub pair_before_c: OperationBranch<E>,
pub pair_before_d: OperationBranch<E>,
pub pair_after: OperationBranch<E>,
pub user_before_a: OperationBranch<E>,
pub user_before_b: OperationBranch<E>,
pub user_before_c: OperationBranch<E>,
pub user_before_d: OperationBranch<E>,
pub user_after: OperationBranch<E>,

pub before_root: Option<E::Fr>,
pub before_b_root: Option<E::Fr>,
pub before_c_root: Option<E::Fr>,
pub before_d_root: Option<E::Fr>,
pub after_root: Option<E::Fr>,
```

3.175 CVF-175

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 175:

```
pub pair_before_b: OperationBranch<E>,

pub pair_before_d: OperationBranch<E>,

pub user_before_a: OperationBranch<E>,

pub user_before_c: OperationBranch<E>,
```

3.176 CVF-176

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

Recommendation This check should be done on the field of "op" before constructing "data".

Listing 176:

```
assert!(data.amount_in < (1u128 << 117) - 1);
```

3.177 CVF-177

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

Recommendation Including the pair address into public data is redundant as it could be derived from the token zero and token one IDs.

Listing 177:

```
append_be_fixed_width(
    &mut pubdata_bits,
    &self.pair_before_a.address.unwrap(),
    ACCOUNT_ID_BIT_WIDTH,
);
```

3.178 CVF-178

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

Description This parameter is probably redundant as it can be obtained from the pair account.

Listing 178:

```
&self.args.token_one.unwrap(),
```

3.179 CVF-179

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 179:

```
let operation_zero = Operation {
let operation_one = Operation {
let operation_two = Operation {
let operation_three = Operation {
```

3.180 CVF-180

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** swap.rs

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 180:

```
    chunk: Some(Fr::from_str("0").unwrap()),

    chunk: Some(Fr::from_str("1").unwrap()),

    chunk: Some(Fr::from_str("2").unwrap()),

    chunk: Some(Fr::from_str("3").unwrap()),

let user_account_address_fe = Fr::from_str(&data.user_account_address.to_string()).unwrap();
let pair_account_address_fe = Fr::from_str(&data.pair_account_address.to_string()).unwrap();
let token_in_fe = Fr::from_str(&data.token_in.to_string()).unwrap();
let token_out_fe = Fr::from_str(&data.token_out.to_string()).unwrap();

let fee_as_field_element = Fr::from_str(&data.fee.to_string()).unwrap();

let max_fee = Fr::from_str(&max_fee.to_string()).unwrap();

        acc.nonce.add_assign(&Fr::from_str("1").unwrap());

    tx_type: Some(Fr::from_str("12").unwrap()),
```

3.181 CVF-181

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

Description This code fragment repeats several times with slight modifications.
Recommendation Consider extracting a utility function.

Listing 181:

```
let amount_in_bits = FloatConversions::to_float(
    data.amount_in,
    AMOUNT_EXPONENT_BIT_WIDTH,
    AMOUNT_MANTISSA_BIT_WIDTH,
    10,
)
.unwrap();

let amount_in_encoded: Fr = le_bit_vector_into_field_element(&
  ↪ amount_in_bits);
```

3.182 CVF-182

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

Recommendation This should be a named constant.

Listing 182:

```
let max_fee = Fr::from_str(&max_fee.to_string()).unwrap();
```

3.183 CVF-183

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

Recommendation The balance path variable names should have the corresponding token name inside.

Listing 183:

```
let (pair_audit_path_before, pair_audit_balance_path_before) =

let (user_audit_path_before, user_audit_balance_path_before) =

let (pair_audit_path_before_b, pair_audit_balance_path_before_b) =

let (user_audit_path_before_b, user_audit_balance_path_before_b) =

let (pair_audit_path_before_c, pair_audit_balance_path_before_c) =

let (user_audit_path_before_c, user_audit_balance_path_before_c) =

let (pair_audit_path_before_d, pair_audit_balance_path_before_d) =

let (user_audit_path_before_d, user_audit_balance_path_before_d) =

let (pair_audit_path_after, pair_audit_balance_path_after) =

let (user_audit_path_after, user_audit_balance_path_after) =
```

3.184 CVF-184

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

Recommendation The numbers should be named constants.

Listing 184:

```
let amount_in_without_fee = amount_in.clone() * 997usize;

    / (amount_in_without_fee + reserve_in * 1000usize);
```

3.185 CVF-185

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

Recommendation Updating the reserve amounts twice wouldn't be necessary if direct token transfers to pair accounts would be forbidden.

Listing 185:

```
    acc.pair_info.token_one_reserve.sub_assign(&amount_out_fe);

    acc.pair_info.token_zero_reserve.sub_assign(&amount_out_fe);

bal.balance_value.sub_assign(&amount_out_fe);

    acc.pair_info
        .token_zero_reserve
        .add_assign(&amount_in_pool_fe);

    acc.pair_info
        .token_one_reserve
        .add_assign(&amount_in_pool_fe);

bal.balance_value.add_assign(&amount_in_pool_fe);
```

3.186 CVF-186

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

Recommendation The numerical constants should be turned to named constants.

Listing 186:

```
let amount_in_pool = amount_in.clone() * 9995usize / 10000usize;
```

3.187 CVF-187

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

Description This requirement makes the fee input parameter redundant.
Recommendation Consider refactoring.

Listing 187:

```
assert_eq!(fee_as_field_element, val_fee_fe, "incorrect fee");
```

3.188 CVF-188

- **Severity** Major
- **Category** Flaw
- **Status** Fixed
- **Source** swap.rs

Description This should be '>='.

Listing 188:

```
bal.balance_value > amount_in_fe,
```

3.189 CVF-189

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

Description If some witness elements are redundant, they just do not have to be generated, or their equality should be demonstrated at the place where they are generated.

Listing 189:

```
account_witness: user_account_witness_before_b.clone(),

balance_witness: user_token_in_balance_before_d.clone(),
```

3.190 CVF-190

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

Description The words "zero" and "one" in names looks like the tokens are in order.
Recommendation Letters "A" and "B" would be better.

Listing 190:

```
pub amount_zero_min: u128,
pub amount_one_min: u128,

pub token_zero_id: u32,
pub token_one_id: u32,
```

3.191 CVF-191

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

Recommendation Consider using arrays instead of series of named fields to make code simpler and easier to read.

Listing 191:

```
pub pair_before_a: OperationBranch<E>,
pub pair_before_b: OperationBranch<E>,
pub pair_before_c: OperationBranch<E>,
pub pair_before_d: OperationBranch<E>,
pub pair_before_e: OperationBranch<E>,
pub pair_before_f: OperationBranch<E>,
pub pair_after: OperationBranch<E>,
pub user_before_a: OperationBranch<E>,
pub user_before_b: OperationBranch<E>,
pub user_before_c: OperationBranch<E>,
pub user_before_d: OperationBranch<E>,
pub user_before_e: OperationBranch<E>,
pub user_before_f: OperationBranch<E>,
pub user_after: OperationBranch<E>,

pub before_root: Option<E::Fr>,
pub before_b_root: Option<E::Fr>,
pub before_c_root: Option<E::Fr>,
pub before_d_root: Option<E::Fr>,
pub before_e_root: Option<E::Fr>,
pub before_f_root: Option<E::Fr>,
pub after_root: Option<E::Fr>,
```

3.192 CVF-192

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 192:

```
pub pair_before_b: OperationBranch<E>,

pub pair_before_d: OperationBranch<E>,

pub pair_before_f: OperationBranch<E>,

pub user_before_a: OperationBranch<E>,

pub user_before_c: OperationBranch<E>,

pub user_before_e: OperationBranch<E>,
```

3.193 CVF-193

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

Recommendation This check should be done on the field of "op" before constructing "data".

Listing 193:

```
assert!(data.amount_lp < MAX_AMOUNT_VALUE);
```

3.194 CVF-194

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 194:

```
166 let operation_zero = Operation {
181 let operation_one = Operation {
196 let operation_two = Operation {
211 let operation_three = Operation {
226 let operation_four = Operation {
241 let operation_five = Operation {
```

3.195 CVF-195

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 195:

```
    chunk: Some(Fr::from_str("0").unwrap()),

    chunk: Some(Fr::from_str("1").unwrap()),

    chunk: Some(Fr::from_str("2").unwrap()),

    chunk: Some(Fr::from_str("3").unwrap()),

    chunk: Some(Fr::from_str("4").unwrap()),

    chunk: Some(Fr::from_str("5").unwrap()),

let user_account_address_fe = Fr::from_str(&data.user_account_address.to_string()).unwrap();
let pair_account_address_fe = Fr::from_str(&data.pair_account_address.to_string()).unwrap();
let token_zero_fe = Fr::from_str(&data.token_zero_id.to_string()).unwrap();
let token_one_fe = Fr::from_str(&data.token_one_id.to_string()).unwrap();
let token_lp_fe = Fr::from_str(&data.token_lp_id.to_string()).unwrap();
let block_number = Fr::from_str(&data.block_number.to_string()).unwrap();

let amount_lp_fe = Fr::from_str(&data.amount_lp.to_string()).unwrap();

let fee_as_field_element = Fr::from_str(&data.fee.to_string()).unwrap();

        acc.nonce.add_assign(&Fr::from_str("1").unwrap());

    tx_type: Some(Fr::from_str("11").unwrap()),
```

3.196 CVF-196

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

Recommendation This function can be a common macro for readability.

Listing 196:

```
let amount_zero_min_bits = FloatConversions::to_float(

let amount_one_min_bits = FloatConversions::to_float(

let amount_lp_bits = FloatConversions::to_float(
```

3.197 CVF-197

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

Description This code fragment repeats several times with slight modifications.
Recommendation Consider extracting a utility function.

Listing 197:

```
let amount_zero_min_bits = FloatConversions::to_float(
    data.amount_zero_min,
    AMOUNT_EXPONENT_BIT_WIDTH,
    AMOUNT_MANTISSA_BIT_WIDTH,
    10,
)

let amount_zero_min_encoded: Fr =
  ↪ le_bit_vector_into_field_element(&amount_zero_min_bits);
```

3.198 CVF-198

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

Recommendation This should be a named constant.

Listing 198:

```
10,

10,

10,
```

3.199 CVF-199

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

Recommendation The balance path variable names should contain the name of the token they correspond to.

Listing 199:

```
let (pair_audit_path_before, pair_audit_balance_path_before) =
let (user_audit_path_before, user_audit_balance_path_before) =
let (pair_audit_path_before_b, pair_audit_balance_path_before_b) =
let (user_audit_path_before_b, user_audit_balance_path_before_b) =
let (pair_audit_path_before_c, pair_audit_balance_path_before_c) =
let (user_audit_path_before_c, user_audit_balance_path_before_c) =
let (pair_audit_path_before_d, pair_audit_balance_path_before_d) =
let (user_audit_path_before_d, user_audit_balance_path_before_d) =
let (pair_audit_path_before_e, pair_audit_balance_path_before_e) =
let (user_audit_path_before_e, user_audit_balance_path_before_e) =
let (pair_audit_path_before_f, pair_audit_balance_path_before_f) =
let (user_audit_path_before_f, user_audit_balance_path_before_f) =
```

3.200 CVF-200

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

Recommendation It would help readability and increase consistency with the main circuit code if we also check for underflows here.

Listing 200:

```
acc.pair_info.token_zero_reserve.sub_assign(&amount_zero_fe);

bal.balance_value.sub_assign(&amount_zero_fe);
```

3.201 CVF-201

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

Recommendation Storing and updating balances in two places would be redundant in case direct token transfers to pair accounts would be forbidden.

Listing 201:

```
acc.pair_info.token_zero_reserve.sub_assign(&amount_zero_fe);

bal.balance_value.sub_assign(&amount_zero_fe);

acc.pair_info.token_one_reserve.sub_assign(&amount_one_fe);

bal.balance_value.sub_assign(&amount_one_fe);

acc.pair_info.token_lp_reserve.sub_assign(&amount_lp_fe);

bal.balance_value.sub_assign(&amount_lp_fe);
```

3.202 CVF-202

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** remove_liquidity.rs

Description This logic is different here and in the circuit. Here: new_reserve = old_reserve - amount In the circuit: new_reserve = old_balance - amount
Client Comment Current balance and current reserve are always the same.

Listing 202:

```
acc.pair_info.token_zero_reserve.sub_assign(&amount_zero_fe);

bal.balance_value.sub_assign(&amount_zero_fe);

acc.pair_info.token_one_reserve.sub_assign(&amount_one_fe);

bal.balance_value.sub_assign(&amount_one_fe);

acc.pair_info.token_lp_reserve.sub_assign(&amount_lp_fe);

bal.balance_value.sub_assign(&amount_lp_fe);
```

3.203 CVF-203

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

Description Reusing paths from other operations just because they are the same actually makes the code less readable.
Recommendation Consider using all the generated paths for readability as only a few of them are reused.

Listing 203:

```
account_witness: user_account_witness_before_b.clone(),

balance_witness: user_token_zero_balance_before_b.clone(),

balance_witness: user_token_one_balance_before_d.clone(),

balance_witness: user_lp_balance_before_f.clone(),
```

3.204 CVF-204

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

Recommendation Consider using arrays instead of series of named fields to make code simpler and easier to read.

Listing 204:

```
pub pair_before_a: OperationBranch<E>,
pub pair_before_b: OperationBranch<E>,
pub pair_before_c: OperationBranch<E>,
pub pair_after: OperationBranch<E>,
pub user_before_a: OperationBranch<E>,
pub user_before_b: OperationBranch<E>,
pub user_before_c: OperationBranch<E>,
pub user_after: OperationBranch<E>,

pub before_root: Option<E::Fr>,
pub before_b_root: Option<E::Fr>,
pub before_c_root: Option<E::Fr>,
pub after_root: Option<E::Fr>,
```

3.205 CVF-205

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

Description Including witnesses for non-current branches into chunks is redundant and error-prone, as such witnesses are not verified by the circuit against the tree.
Recommendation Consider removing witnesses for non-current branches from chunks.

Listing 205:

```
pub pair_before_b: OperationBranch<E>,

pub user_before_a: OperationBranch<E>,

pub user_before_c: OperationBranch<E>,
```

3.206 CVF-206

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

Recommendation Consider using an array instead of a series of variables. This would make the code simpler and easier to read.

Listing 206:

```
let operation_zero = Operation {

let operation_one = Operation {

let operation_two = Operation {
```

3.207 CVF-207

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

Description Converting numbers to field elements through strings is weird.
Recommendation Consider using more direct ways.

Listing 207:

```
    chunk: Some(Fr::from_str("0").unwrap()),

    chunk: Some(Fr::from_str("1").unwrap()),

    chunk: Some(Fr::from_str("2").unwrap()),

let user_account_address_fe = Fr::from_str(&data.user_account_address.to_string()).unwrap();
let pair_account_address_fe = Fr::from_str(&data.pair_account_address.to_string()).unwrap();
let token_lp_fe = Fr::from_str(&data.token_lp_id.to_string()).unwrap();
let token_bonus_fe = Fr::from_str(&data.token_bonus_id.to_string()).unwrap();
let block_number = Fr::from_str(&data.block_number.to_string()).unwrap();

let fee_as_field_element = Fr::from_str(&data.fee.to_string()).unwrap();

        acc.nonce.add_assign(&Fr::from_str("1").unwrap());

    tx_type: Some(Fr::from_str("8").unwrap()),
```

3.208 CVF-208

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

Description The variable name "pair_audit_balance_path_before" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_path_before".

Listing 208:

```
let (pair_audit_path_before, pair_audit_balance_path_before) =
```

3.209 CVF-209

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

Description The variable name "user_audit_balance_path_before" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_path_before".

Listing 209:

```
let (user_audit_path_before, user_audit_balance_path_before) =
```

3.210 CVF-210

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

Description We did not review this function.

Listing 210:

```
let accumulate = calculate_updated_bonus_in_pair(acc,
    ↪ block_number);
```

3.211 CVF-211

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

Description It is unclear what should be set in args.
Recommendation Consider clarifying.

Listing 211:

```
// This should be set in args
```

3.212 CVF-212

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

Description The variable name "pair_audit_balance_path_before_b" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_path_before_b".

Listing 212:

```
let (pair_audit_path_before_b, pair_audit_balance_path_before_b)
    ↪ =
```

3.213 CVF-213

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

Description The variable name "user_audit_balance_path_before_b" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_path_before_b".

Listing 213:

```
let (user_audit_path_before_b, user_audit_balance_path_before_b)
    ↪ =
```

3.214 CVF-214

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

Description The variable name "pair_audit_balance_path_before_c" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_path_before_c".

Listing 214:

```
let (pair_audit_path_before_c, pair_audit_balance_path_before_c)
    ↪ =
```

3.215 CVF-215

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

Description The variable name "user_audit_balance_path_before_c" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_path_before_c".

Listing 215:

```
let (user_audit_path_before_c, user_audit_balance_path_before_c)
    =
```

3.216 CVF-216

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

Description The variable name "pair_audit_balance_path_after" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "pair_audit_balance_lp_path_after".

Listing 216:

```
let (pair_audit_path_after, pair_audit_balance_path_after) =
```

3.217 CVF-217

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

Description The variable name "user_audit_balance_path_after" is ambiguous as it is unclear what token the balance is for.
Recommendation Consider renaming to "user_audit_balance_lp_path_after".

Listing 217:

```
let (user_audit_path_after, user_audit_balance_path_after) =
```

3.218 CVF-218

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

Description Using different names for the same field is confusing.

Listing 218:

```
let account_address = account_address.pad(franklin_constants::
    ↪ ACCOUNT_ID_BIT_WIDTH);

    account_id: account_address,
```

3.219 CVF-219

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

Description This approach doesn't scale.
Recommendation Consider an array of flags with some parameterizable length.

Listing 219:

```
pub is_chunk_second: Boolean,
pub is_chunk_third: Boolean,
pub is_chunk_fourth: Boolean,
pub is_chunk_fifth: Boolean,
```

3.220 CVF-220

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

Recommendation Consider encapsulating a pair of corresponding packed and unpacked amounts into a struct with constraints that bind the two amounts together.

Listing 220:

```
pub amount_packed: CircuitElement<E>,

pub amount_unpacked: CircuitElement<E>,
```

3.221 CVF-221

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

Recommendation This should be 'AMOUNT_PACKED_BIT_WIDTH'.

Listing 221:

```
152 franklin_constants::AMOUNT_EXPONENT_BIT_WIDTH
      + franklin_constants::AMOUNT_MANTISSA_BIT_WIDTH,
```

3.222 CVF-222

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

Recommendation This should be a single constant.

Listing 222:

```
158 franklin_constants::FEE_EXPONENT_BIT_WIDTH + franklin_constants
     ↪ ::FEE_MANTISSA_BIT_WIDTH,
```

3.223 CVF-223

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

Recommendation The exponent base should be a named constant.

Listing 223:

```
389 10,

397 10,
```

3.224 CVF-224

- **Severity** Major
- **Category** Flaw
- **Status** Fixed
- **Source** allocated_structures.rs

Recommendation These arrays should be padded to BALANCE_BIT_WIDTH elements.

Listing 224:

```
470 let amount_zero = CircuitElement::from_fe_with_known_length(

476 let amount_one = CircuitElement::from_fe_with_known_length(
```

3.225 CVF-225

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

Recommendation If direct token transfers to pair account would be forbidden, these fields would be redundant, as the token zero, token one, and LP token balances of the pair account could be used instead.

Listing 225:

```
 81 pub token_zero_reserve: Option<E::Fr>,
    pub token_one_reserve: Option<E::Fr>,
    pub token_lp_reserve: Option<E::Fr>,

109 pub token_zero_reserve: CircuitElement<E>,
110 pub token_one_reserve: CircuitElement<E>,
    pub token_lp_reserve: CircuitElement<E>,
```

3.226 CVF-226

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

Recommendation This function should accept "circuit_account.pair_info" as the argument, rather than just "circuit_account".

Listing 226:

```
 89 pub fn from_circuit_account(circuit_account: &CircuitAccount<E>)
      ↪  -> Self {
```

3.227 CVF-227

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

Recommendation This function should accept "circuit_account.pair_info.bonus_info" as the argument, instead of just "circuit_account".

Listing 227:

```
pub fn from_circuit_account(circuit_account: &CircuitAccount<E>)
    ↪ -> Self {
```