Report

v. 2.0

Customer

Nil foundation

Circuit audit

Blockchain Verifier

29th January 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	25.01.24	A. Zveryanskaya	Initial Draft
0.2	25.01.24	A. Zveryanskaya	Minor revision
1.0	26.01.24	A. Zveryanskaya	Release
1.1	29.01.24	A. Zveryanskaya	CVF-16, 19, 25, 26, 28, 29, 38, 65, 75 marked as fixed due to the new implementation
2.0	29.01.24	A. Zveryanskaya	Release

2 Introduction

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

=nil; Foundation's in-EVM Mina Protocol state verification enables every state transition, every transaction made within Mina state to be proved on Ethereum directly, in a completely trustless way.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

algebra/curves/pasta/		
endo_scalar.hpp	unified_addition.hpp	variable_ base_scalar_mul_15_wires.hpp
algebra/fields/		
exponentiation.hpp	field_operations.hpp	
hashes/poseidon/		
poseidon_15_wires.hpp		
systems/snark/kimchi/detail/		
limbs.hpp		

4 Methodology

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

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

We classify issues by the following severity levels:

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

5 Our findings

We found 8 critical, 7 major, and a few less important issues. All identified Critical and Major issues have been fixed.

Issues	Info	Fixed
Severity: Critical	0	8
Major	0	7
Moderate	6	3
Minor	35	15

Fixed 33 out of 74 issues

6 Critical Issues

CVF-3. FIXED

- **Category** Flaw
- **Source** field_operations.hpp

Description There is no enable_selector call for this index.

Client Comment *Fixed during the interfaces update.*

```
124 bp.add_gate(first_selector_index, {constraint_1});
219 bp.add_gate(first_selector_index, {constraint_1});
316 bp.add_gate(first_selector_index, {constraint_1, constraint_2});
413 bp.add_gate(first_selector_index, {constraint_1});
621 bp.add_gate(first_selector_index, {constraint_1});
```

CVF-4. FIXED

- **Category** Flaw
- **Source** field_operations.hpp

Description These functions are never called.

Client Comment *Fixed during the interfaces update.*

```
128 generate_copy_constraints(blueprint<ArithmetizationType> &bp,
223 generate_copy_constraints(blueprint<ArithmetizationType> &bp,
320 generate_copy_constraints(blueprint<ArithmetizationType> &bp,
417 generate_copy_constraints(blueprint<ArithmetizationType> &bp,
625 generate_copy_constraints(blueprint<ArithmetizationType> &bp,
```

CVF-8. FIXED

- **Category** Flaw
- **Source** field_operations.hpp

Description Two constraints are not enough for this component: it allows arbitrary value for W3 (and thus arbitarary result value) in case W1 is zero.

Recommendation Correct constraints are: W1*W3=W4 W4(W4-1)=0 (W3-W1)(W4-1)=0 W0*W3=W2

618
```
auto constraint_1 = bp.add_constraint(var(W0, 0) * var(W3, 0) - var(
    ↪ W2, 0));
auto constraint_2 = bp.add_constraint(var(W1, 0) * var(W3, 0) * var(
    ↪ W1, 0) - var(W1, 0));
```

CVF-10. FIXED

- **Category** Flaw
- **Source** field_operations.hpp

Description Second constraint is not used here.

621
```
bp.add_gate(first_selector_index, {constraint_1});
```

CVF-17. FIXED

- **Category** Flaw
- **Source** variable_base_scalar_mul_15_wires.hpp

Description It should be 1 or 3 instead of 5. The current formula doesn't add constraints for row 102.

229
```
start_row_index + rows_amount - 5, 2);
```

CVF-24. FIXED

- **Category** Overflow/Underflow
- **Source** variable_base_scalar_mul_15_wires.hpp

Description This may overflow in the last rows.

Recommendation Consider adding a range check.

```
318 auto constraint_16 =
        bp.add_constraint(var(W5, 0) - (32 * (var(W4, 0)) + 16 * var(W2,
            ↪  +1) + 8 * var(W3, +1) +
320                                     4 * var(W4, +1) + 2 * var(W5,
                                           ↪ +1) + var(W6, +1)));
```

CVF-30. FIXED

- **Category** Flaw
- **Source** variable_base_scalar_mul_15_wires.hpp

Description The Halo2 reference suggests that the most significant bits of scalar should be processed with complete addition whereas it seems not to be the case here.

```
425     bp.add_gate(selector_index_2,
                    {bit_check_1,   bit_check_2,   bit_check_3,
                       ↪ bit_check_4,   bit_check_5,
                     constraint_1,  constraint_2,  constraint_3,
                       ↪ constraint_4,  constraint_5,
                     constraint_6,  constraint_7,  constraint_8,
                       ↪ constraint_9,  constraint_10,
                     constraint_11, constraint_12, constraint_13,
                       ↪ constraint_14, constraint_15,
430                  constraint_16, constraint_17, constraint_18,
                       ↪ constraint_19, constraint_20,
                     constraint_21, constraint_22});
}
```

CVF-69. FIXED

- **Category** Overflow/Underflow
- **Source** limbs.hpp

Description This constraint allows limbs that exceed 64 bits.

Recommendation Consider adding range checks.

Client Comment *Here we use an additional range_component to check that all values in the array are less than 64 bits.*

```
260 auto constraint_1 = bp.add_constraint(var(W1, 0) + var(W2, 0) *
       ↪ scalar.pow(64) +
                                          var(W3, 0) * scalar.pow(128) +
                                             ↪ var(W4, 0) * scalar.pow
                                             ↪ (192) - var(W0, 0));
```

7 Major Issues

CVF-7. FIXED

- **Category** Flaw
- **Source** field_operations.hpp

Description Should be ‘witness(W3)‘.

```
608 assignment.witness(3)[j] = assignment.var_value(params.y) == 0 ? 0 :
       ↪  assignment.var_value(params.y).inversed();
```

CVF-19. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description These constraints are different from what is described in the technical reference, however constraints in the code seem more reasonable.

Client Comment *The technical reference was updated.*

```
253 auto constraint_1 = bp.add_constraint((var(W2, 0) - var(W0, 0)) *
       ↪ var(W7, +1) -
                                          (var(W3, 0) - (2 * var(W2, +1)
                                             ↪  - 1) * var(W1, 0)));
    auto constraint_2 = bp.add_constraint((var(W7, 0) - var(W0, 0)) *
       ↪ var(W8, +1) -
                                          (var(W8, 0) - (2 * var(W3, +1)
                                             ↪  - 1) * var(W1, 0)));
    auto constraint_3 = bp.add_constraint((var(W9, 0) - var(W0, 0)) *
       ↪ var(W9, +1) -
                                          (var(W10, 0) - (2 * var(W4,
                                             ↪ +1) - 1) * var(W1, 0)));
    auto constraint_4 = bp.add_constraint((var(W11, 0) - var(W0, 0)) *
       ↪ var(W10, +1) -
260                                       (var(W12, 0) - (2 * var(W5,
                                             ↪ +1) - 1) * var(W1, 0)));
    auto constraint_5 = bp.add_constraint((var(W13, 0) - var(W0, 0)) *
       ↪ var(W11, +1) -
                                          (var(W14, 0) - (2 * var(W6,
                                             ↪ +1) - 1) * var(W1, 0)));
```

CVF-28. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description In the technical specification it is “var(W0, +1)” while in the code it is ”var(W0, 0)”.

Recommendation Consider checking which one is correct.

Client Comment *The technical reference was updated.*

```
412 auto constraint_20 = bp.add_constraint((var(W8, +1)*var(W2, +1)*var(
        ↪ W0, 0)) +
```

CVF-29. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description In the technical specification it is “var(W1, +1)” while in the code it is ”var(W1, 0)”.

Recommendation Consider checking which one is correct.

Client Comment *The technical reference was updated.*

```
418 auto constraint_21 = bp.add_constraint((var(W8, +1)*var(W2, +1)*var(
        ↪ W1, 0)) +
```

CVF-65. FIXED

- **Category** Unclear behavior
- **Source** endo_scalar.hpp

Description The reference document deals with different rows here.

Recommendation Consider explaining the difference or fix it.

Client Comment *The technical reference was updated.*

```
    var(W4, 0) - (256 * var(W2, 0) + 128 * c_f(var(W7, 0)) + 64 *
       ↪ c_f(var(W8, 0)) +
                  32 * c_f(var(W9, 0)) + 16 * c_f(var(W10, 0)) + 8 *
                     ↪  c_f(var(W11, 0)) +
                  4 * c_f(var(W12, 0)) + 2 * c_f(var(W13, 0)) + c_f(
                     ↪ var(W14, 0))));
auto constraint_10 = bp.add_constraint(
    var(W5, 0) - (256 * var(W3, 0) + 128 * d_f(var(W7, 0)) + 64 *
       ↪ d_f(var(W8, 0)) +
                  32 * d_f(var(W9, 0)) + 16 * d_f(var(W10, 0)) + 8 *
                     ↪  d_f(var(W11, 0)) +
                  4 * d_f(var(W12, 0)) + 2 * d_f(var(W13, 0)) + d_f(
                     ↪ var(W14, 0))));
auto constraint_11 = bp.add_constraint(
    var(W1, 0) - ((1 << 16) * var(W0, 0) + (1 << 14) * var(W7, 0) +
       ↪ (1 << 12) * var(W8, 0) +
                  (1 << 10) * var(W9, 0) + (1 << 8) * var(W10, 0) +
                     ↪ (1 << 6) * var(W11, 0) +
                  (1 << 4) * var(W12, 0) + (1 << 2) * var(W13, 0) +
                     ↪ var(W14, 0)));
```

CVF-68. FIXED

- **Category** Flaw
- **Source** limbs.hpp

Description Range check is missing for input elements.

Recommendation Consider implementing it or declaring explicitly that it is not asserted.

```
141 auto constraint_1 = bp.add_constraint(var(W0, 0) + var(W1, 0) *
       ↪ scalar.pow(64) - var(W2, 0));
```

```
260 auto constraint_1 = bp.add_constraint(var(W1, 0) + var(W2, 0) *
       ↪ scalar.pow(64) +
                                         var(W3, 0) * scalar.pow(128) +
                                            ↪ var(W4, 0) * scalar.pow
                                            ↪ (192) - var(W0, 0));
```

CVF-75. FIXED

- **Category** Unclear behavior
- **Source** unified_addition.hpp

Description The constraint differs from the technical reference.

Recommendation Consider fixing either of them.

Client Comment *The technical reference was updated.*

```
213 bp.add_constraint((var(W2, 0) - var(W0, 0)) *
                      ((var(W2, 0) - var(W0, 0)) * var(W10, 0) - (var(W3
                         ↪ , 0) - var(W1, 0))));
```

8 Moderate Issues

CVF-45. INFO

- **Category** Documentation
- **Source** poseidon_15_wires.hpp

Description This value is different from the reference doc (there it is 5).

Recommendation Consider documenting.

```
707 constexpr static const std::size_t sbox_alpha = 7;
```

CVF-52. INFO

- **Category** Unclear behavior
- **Source** poseidon_15_wires.hpp

Description By technical reference and Mina code the 5 consecutive Poseidon states are laid as 0, 4,1,2,3.

Recommendation Consider documenting and explaining the difference.

```
787 assignment.witness(W3)[i] = next_state[0];
    assignment.witness(W4)[i] = next_state[1];
    assignment.witness(W5)[i] = next_state[2];
```

```
838 auto constraint_1 = bp.add_constraint(
        var(W3, 0) -
840     (var(W0, 0).pow(sbox_alpha) * mds[0][0] + var(W1, 0).pow(
           ↪ sbox_alpha) * mds[0][1] +
         var(W2, 0).pow(sbox_alpha) * mds[0][2] + round_constant[z][0]))
            ↪ ;
```

CVF-15. FIXED

- **Category** Flaw
- **Source** variable_base_scalar_mul_15_wires.hpp

Description In case "n_next" is 0×224698fc0994a8dd8c46eb2100000000, 0×224698fc0994a8dd8c46eb2100000001, or 0×2000000000000000000000000000003369e57a0e5efd4c526a60b180000001, the "inversed" function will be called on a zero value and probably fail, while the "if" statements suggest that these values are valid, yet special.

```
174 (n_next - 0x2000000000000000000000000000003369e57a0e5efd4c526a60b
    180000001_cppui255)*(n_next - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255));
```

```
180 if (n_next == 0x224698fc0994a8dd8c46eb2100000000_cppui255) {
```

```
188         if (n_next == 0x224698fc0994a8dd8c46eb2100000001_cppui255) {
```

CVF-16. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Recommendation According to the technical reference, "x" and "y" should be assigned to "W9" and "W10", while "W0" and "W1" are for "P[5].X" and "P[5].Y".

Client Comment *The technical reference was updated.*

```
204 assignment.witness(W0)[start_row_index + rows_amount - 1] = x;
    assignment.witness(W1)[start_row_index + rows_amount - 1] = y;
```

CVF-39. INFO

- **Category** Procedural
- **Source** exponentiation.hpp

Description This assumes that the witnesses indexes follow each other, which is not guaranteed.

Recommendation Consider not relying on this fact.

Client Comment *We do not consider it as an issue.*

```
172     std::size_t column_idx = W14 - j * (bits_per_intermediate_result
           ↪ )-bit_column;
```

```
186 assignment.witness(intermediate_start + j)[row] = acc1;
```

```
212 std::size_t column_idx = W14 - j * (bits_per_intermediate_result)-
       ↪ bit_column;
```

CVF-48. INFO

- **Category** Unclear behavior
- **Source** poseidon_15_wires.hpp

Description This function actually does nothing. Probably it should do something.

Client Comment *We do not consider it as an issue.*

```
763 generate_copy_constraints(bp, assignment, params,
       ↪ component_start_row);
```

CVF-50. INFO

- **Category** Suboptimal
- **Source** poseidon_15_wires.hpp

Description The pow function is computed three times with the same argument.

Recommendation Consider refactoring to compute it only once.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

783
```
next_state[j] = state[0].pow(sbox_alpha) * mds[j][0] +
                state[1].pow(sbox_alpha) * mds[j][1] +
                state[2].pow(sbox_alpha) * mds[j][2] +
                   ↪ round_constant[(i - row) * 5][j];
```

840
```
(var(W0, 0).pow(sbox_alpha) * mds[0][0] + var(W1, 0).pow(sbox_alpha)
   ↪  * mds[0][1] +
```

844
```
(var(W0, 0).pow(sbox_alpha) * mds[1][0] + var(W1, 0).pow(sbox_alpha)
   ↪  * mds[1][1] +
```

848
```
(var(W0, 0).pow(sbox_alpha) * mds[2][0] + var(W1, 0).pow(sbox_alpha)
   ↪  * mds[2][1] +
```

CVF-62. INFO

- **Category** Procedural
- **Source** endo_scalar.hpp

Description This assumes that W7, ..., W14 indexes follow each other, i.e. that W8 = W7 + 1, W9 = W8 + 1 etc, which may not be the case.

Recommendation Consider properly translating a "j" value into a wire index without relying on assumptions that could be wrong.

Client Comment *We do not consider it as an issue.*

201
```
assignment.witness(W7 + j)[row] = crumb_value;
```

CVF-70. FIXED

- **Category** Overflow/Underflow
- **Source** limbs.hpp

Description Even when all limbs fit into 64 bits, overflow is possible here making the result non-deterministic in come cases.

Recommendation Consider implementing overflow protection.

```
260 auto constraint_1 = bp.add_constraint(var(W1, 0) + var(W2, 0) *
       ↪ scalar.pow(64) +
                                          var(W3, 0) * scalar.pow(128) +
                                             ↪ var(W4, 0) * scalar.pow
                                             ↪ (192) - var(W0, 0));
```

9 Minor Issues

CVF-1. INFO

- **Category** Procedural
- **Source** field_operations.hpp

Description We didn't review these files.

Client Comment *The files are not related to the circuit description.*

```
36 #include <nil/crypto3/zk/snark/arithmetization/plonk/
      ↪ constraint_system.hpp>
```

```
38 #include <nil/crypto3/zk/blueprint/plonk.hpp>
   #include <nil/crypto3/zk/assignment/plonk.hpp>
40 #include <nil/crypto3/zk/algorithms/generate_circuit.hpp>
```

CVF-2. INFO

- **Category** Suboptimal
- **Source** field_operations.hpp

Description This constructor is redundant, as it is superseded by the other constructor.

Recommendation Consider removing this constructor.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

95
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

188
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

284
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

383
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

463
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

585
```
result_type(const params_type &params, std::size_t start_row_index)
    ↪ {
```

CVF-5. INFO

- **Category** Readability
- **Source** field_operations.hpp

Description The argument ordering is different for these two calls, which is confusing.

Recommendation Consider making the argument ordering the same or explaining why it is different.

138
```
bp.add_copy_constraint({params.x, component_x});
bp.add_copy_constraint({component_y, params.y});
```

CVF-6. INFO

- **Category** Bad naming
- **Source** field_operations.hpp

Description The field name is too generic.

Recommendation Consider renaming to “y” or “c”.

```
458 typename BlueprintFieldType::value_type constant;
```

CVF-11. INFO

- **Category** Procedural
- **Source** variable_base_scalar_mul_15_wires.hpp

Description We didn’t review this file.

Client Comment *The files are not related to the circuit description.*

```
31 #include <nil/marshalling/algorithms/pack.hpp>
```

```
33 #include <nil/crypto3/zk/snark/arithmetization/plonk/
   ↪ constraint_system.hpp>
```

```
35 #include <nil/crypto3/zk/blueprint/plonk.hpp>
#include <nil/crypto3/zk/assignment/plonk.hpp>
#include <nil/crypto3/zk/algorithms/generate_circuit.hpp>
```

CVF-12. INFO

- **Category** Suboptimal
- **Source** variable_base_scalar_mul_15_wires.hpp

Description The same row index is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
89 X = var(W0, start_row_index + rows_amount - 1, false, var::
      ↪ column_type::witness);
90 Y = var(W1, start_row_index + rows_amount - 1, false, var::
      ↪ column_type::witness);
```

CVF-13. FIXED

- **Category** Procedural
- **Source** variable_base_scalar_mul_15_wires.hpp

Description This code relies on the fact that addition consumes exactly 2 rows.

Recommendation Consider not relying on this and using “mul_rows_amount” instead of “rows_amount - 3”.

```
125 for (std::size_t i = j; i < j + rows_amount - 3; i = i + 2) {
```

CVF-14. INFO

- **Category** Suboptimal
- **Source** variable_base_scalar_mul_15_wires.hpp

Description This assignment is performed on every loop iteration.

Recommendation Consider performing once before the loop.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
141 Q.X = T.X;
```

CVF-18. INFO

- **Category** Readability
- **Source** variable_base_scalar_mul_15_wires.hpp

Recommendation The step value 2 should be a named constant.

```
229 start_row_index + rows_amount - 5, 2);
```

CVF-20. INFO

- **Category** Readability
- **Source** variable_base_scalar_mul_15_wires.hpp

Recommendation Should be “+1” rather than just “1” for consistency.

```
265 (2 * var(W3, 0) - var(W7, 1) * (2 * var(W2, 0) - var(W7, 1).pow(2) +
       ↪  var(W0, 0))) *
        (2 * var(W3, 0) - var(W7, 1) * (2 * var(W2, 0) - var(W7, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W2, 0) - var(W7, 1).pow(2) + var(W0, 0)) *
     (2 * var(W2, 0) - var(W7, 1).pow(2) + var(W0, 0)) *
     (var(W7, 0) - var(W0, 0) + var(W7, 1).pow(2))));

271 (2 * var(W8, 0) - var(W8, 1) * (2 * var(W7, 0) - var(W8, 1).pow(2) +
       ↪  var(W0, 0))) *
        (2 * var(W8, 0) - var(W8, 1) * (2 * var(W7, 0) - var(W8, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W7, 0) - var(W8, 1).pow(2) + var(W0, 0)) *
     (2 * var(W7, 0) - var(W8, 1).pow(2) + var(W0, 0)) *
     (var(W9, 0) - var(W0, 0) + var(W8, 1).pow(2))));

277 (2 * var(W10, 0) - var(W9, 1) * (2 * var(W9, 0) - var(W9, 1).pow(2)
       ↪ + var(W0, 0))) *
        (2 * var(W10, 0) - var(W9, 1) * (2 * var(W9, 0) - var(W9, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W9, 0) - var(W9, 1).pow(2) + var(W0, 0)) *
280  (2 * var(W9, 0) - var(W9, 1).pow(2) + var(W0, 0)) *
     (var(W11, 0) - var(W0, 0) + var(W9, 1).pow(2))));

295  (var(W0, 1) - var(W0, 0) + var(W11, +1).pow(2))));
```

CVF-21. INFO

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description Here a complex expression is multiplied by itself.

Recommendation Consider using ".pow(2)" instead.

```
265 (2 * var(W3, 0) - var(W7, 1) * (2 * var(W2, 0) - var(W7, 1).pow(2) +
       ↪  var(W0, 0))) *
        (2 * var(W3, 0) - var(W7, 1) * (2 * var(W2, 0) - var(W7, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W2, 0) - var(W7, 1).pow(2) + var(W0, 0)) *
     (2 * var(W2, 0) - var(W7, 1).pow(2) + var(W0, 0)) *
```

```
271 (2 * var(W8, 0) - var(W8, 1) * (2 * var(W7, 0) - var(W8, 1).pow(2) +
       ↪  var(W0, 0))) *
        (2 * var(W8, 0) - var(W8, 1) * (2 * var(W7, 0) - var(W8, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W7, 0) - var(W8, 1).pow(2) + var(W0, 0)) *
     (2 * var(W7, 0) - var(W8, 1).pow(2) + var(W0, 0)) *
```

```
277 (2 * var(W10, 0) - var(W9, 1) * (2 * var(W9, 0) - var(W9, 1).pow(2)
       ↪ + var(W0, 0))) *
        (2 * var(W10, 0) - var(W9, 1) * (2 * var(W9, 0) - var(W9, 1).pow
           ↪ (2) + var(W0, 0))) -
    ((2 * var(W9, 0) - var(W9, 1).pow(2) + var(W0, 0)) *
280  (2 * var(W9, 0) - var(W9, 1).pow(2) + var(W0, 0)) *
```

```
283 (2 * var(W12, 0) - var(W10, +1) * (2 * var(W11, 0) - var(W10, +1).
       ↪ pow(2) + var(W0, 0))) *
        (2 * var(W12, 0) -
         var(W10, +1) * (2 * var(W11, 0) - var(W10, +1).pow(2) + var(W0,
            ↪  0))) -
    ((2 * var(W11, 0) - var(W10, +1).pow(2) + var(W0, 0)) *
     (2 * var(W11, 0) - var(W10, +1).pow(2) + var(W0, 0)) *
```

```
290 (2 * var(W14, 0) - var(W11, +1) * (2 * var(W13, 0) - var(W11, +1).
       ↪ pow(2) + var(W0, 0))) *
        (2 * var(W14, 0) -
         var(W11, +1) * (2 * var(W13, 0) - var(W11, +1).pow(2) + var(W0,
            ↪  0))) -
    ((2 * var(W13, 0) - var(W11, +1).pow(2) + var(W0, 0)) *
     (2 * var(W13, 0) - var(W11, +1).pow(2) + var(W0, 0)) *
```

CVF-23. INFO

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description This constraint is different from what is described in the technical reference, however the constraint in the code looks more reasonable.

```
309 auto constraint_14 = bp.add_constraint(
310     (var(W14, 0) + var(W12, 0)) * (2 * var(W11, 0) - var(W10, +1).
        ↪ pow(2) + var(W0, 0)) -
    ((var(W11, 0) - var(W13, 0)) *
     (2 * var(W12, 0) - var(W10, +1) * (2 * var(W11, 0) - var(W10,
        ↪ +1).pow(2) + var(W0, 0))));
```

CVF-25. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description These constraints are different from what is described in the technical reference, however, the constraints in the code looks more reasonable.

Client Comment *The technical reference was updated.*

```
335 constraint_1 = bp.add_constraint((var(W2, -1) - var(W0, -1)) * var(
       ↪ W7, 0) -
                                        (var(W3, -1) - (2 * var(W2, 0)
                                           ↪  - 1) * var(W1, -1)));
    constraint_2 = bp.add_constraint((var(W7, -1) - var(W0, -1)) * var(
       ↪ W8, 0) -
                                        (var(W8, -1) - (2 * var(W3, 0)
                                           ↪  - 1) * var(W1, -1)));
    constraint_3 = bp.add_constraint((var(W9, -1) - var(W0, -1)) * var(
       ↪ W9, 0) -
340                                     (var(W10, -1) - (2 * var(W4,
                                           ↪ 0) - 1) * var(W1, -1)));
    constraint_4 = bp.add_constraint((var(W11, -1) - var(W0, -1)) * var(
       ↪ W10, 0) -
                                        (var(W12, -1) - (2 * var(W5,
                                           ↪ 0) - 1) * var(W1, -1)));
    constraint_5 = bp.add_constraint((var(W13, -1) - var(W0, -1)) * var(
       ↪ W11, 0) -
                                        (var(W14, -1) - (2 * var(W6,
                                           ↪ 0) - 1) * var(W1, -1)));
```

CVF-26. FIXED

- **Category** Unclear behavior
- **Source** variable_base_scalar_mul_15_wires.hpp

Description This constraint is different from that is described in the technical reference, however the constraint in the code seems more reasonable.

Client Comment *The technical reference was updated.*

392
```
constraint_14 = bp.add_constraint(
    ((var(W14, -1) + var(W12, -1)) * (2 * var(W11, -1) - var(W10, 0)
       ↪ .pow(2) + var(W0, -1)) -
    ((var(W11, -1) - var(W13, -1)) *
     (2 * var(W12, -1) - var(W10, 0) * (2 * var(W11, -1) - var(W10,
        ↪ 0).pow(2) + var(W0, -1)))))*
     var(W8, +1)*var(W2, +1));
```

CVF-27. FIXED

- **Category** Bad datatype
- **Source** variable_base_scalar_mul_15_wires.hpp

Recommendation These constants must be named.

```
408 auto constraint_18 = bp.add_constraint(((var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000000_cppui255)
    *var(W3, +1) - 1) * (var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000000_cppui255));
410 auto constraint_19 = bp.add_constraint(((var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255)
    *var(W4, +1) - 1) * (var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255));
```

```
413 ((var(W5, +1) - 0x224698fc0994a8dd8c46eb2100000000_cppui255)
    *var(W3, +1) - (var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255)
    *var(W4, +1))* ((var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000000_cppui255)
    *var(W3, +1) - (var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255)
```

```
419 ((var(W5, +1) - 0x224698fc0994a8dd8c46eb2100000000_cppui255)
420 *var(W3, +1) - (var(W5, +1) - 0
       ↪ x224698fc0994a8dd8c46eb2100000001_cppui255)
```

```
422 auto constraint_22 = bp.add_constraint(var(W8, +1) - ((var(W5, +1) -
       ↪  0x224698fc0994a8dd8c46eb2100000000_cppui255)
    *(var(W5, +1) - 0
       ↪ x200000000000000000000000000003369e57a0e5efd4c526a60b
    180000001_cppui255)*
    (var(W5, +1) - 0x224698fc0994a8dd8c46eb2100000001_cppui255)));
```

CVF-31. INFO

- **Category** Procedural
- **Source** exponentiation.hpp

Description We didn't review these files.

Client Comment *The files are not related to the circuit description*

```
34 #include <nil/marshalling/algorithms/pack.hpp>

36 #include <nil/crypto3/zk/snark/arithmetization/plonk/
      ↪ constraint_system.hpp>

38 #include <nil/crypto3/zk/blueprint/plonk.hpp>
   #include <nil/crypto3/zk/assignment/plonk.hpp>
40 #include <nil/crypto3/zk/component.hpp>
```

CVF-32. FIXED

- **Category** Documentation
- **Source** exponentiation.hpp

Description This comment does not match the code where bits are in pairs.

Recommendation Consider changing the comment.

```
54 // W10 | W11 | W12 | W13  | W14  | base   | n = [b0...b8] | base^[
      ↪ b0b1b2] | W1^8 + base^[b3b4b5] | ... |
   // - | - | b_8 | b_7 | b_6 | b_5 | b_4 | b_3 | b_2 | b_1 |
   // ...    | ...           | ...           | ...                  |
      ↪ ... | - | - | ... | ... | ...
   // base   | n = scalar    | res           | ...                  |
      ↪ ... | - | - | ...
```

CVF-33. FIXED

- **Category** Suboptimal
- **Source** exponentiation.hpp

Recommendation These two values could be calculated simpler like this: main_rows = (ExponentSize + bits_per_row - 1) / bits_per_row; padded_exponent_size = main_rows * bits_per_row;

```
84 constexpr static const std::size_t padded_exponent_size =
```

```
87 constexpr static const std::size_t main_rows = (ExponentSize %
      ↪ bits_per_row == 0) ?
```

CVF-34. FIXED

- **Category** Suboptimal
- **Source** exponentiation.hpp

Recommendation This could be calculated as: padded_exponent_size / bits_per_row;

```
87 constexpr static const std::size_t main_rows = (ExponentSize %
      ↪ bits_per_row == 0) ?
                                                  (ExponentSize /
                                                     ↪ bits_per_row) :
                                                  (ExponentSize /
                                                     ↪ bits_per_row) +
                                                     ↪ 1;
```

CVF-35. INFO

- **Category** Suboptimal
- **Source** exponentiation.hpp

Description This constructor seems redundant as it is superseded by the other constructor.

Recommendation Consider removing this constructor.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue*

```
103 result_type(const params_type &params, std::size_t
       ↪ component_start_row) {
```

CVF-36. INFO

- **Category** Unclear behavior
- **Source** exponentiation.hpp

Description This silently drops all extra bits.

Recommendation Consider explicitly asserting the dropped bits to be zero.

```
153 std::copy(bits_all.end() - padded_exponent_size, bits_all.end(),
       ↪ bits.begin());
```

CVF-37. INFO

- **Category** Procedural
- **Source** exponentiation.hpp

Description This implicitly assumes that “padded_exponent_size” doesn’t exceed 255.

Recommendation Consider asserting this fact explicitly.

```
153 std::copy(bits_all.end() - padded_exponent_size, bits_all.end(),
       ↪ bits.begin());
```

CVF-38. FIXED

- **Category** Documentation
- **Source** exponentiation.hpp

Recommendation “woth” → “with”

Client Comment *The technical reference was updated.*

```
159 // we use first empty row to unify first row gate woth others
```

CVF-40. INFO

- **Category** Bad naming
- **Source** exponentiation.hpp

Description This assignment is confusing and it is overwritten in the next line.

Recommendation Consider assigning the final value.

```
200 typename ArithmetizationType::field_type::value_type exponent_shift
       ↪ = 2;
```

CVF-41. INFO

- **Category** Suboptimal
- **Source** exponentiation.hpp

Description The “bit_res” value is not used as is, but only after adding (1 - var(column_idx, 0) to it.

Recommendation Consider removing the “bit_res” variable and using an expression instead.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
216 snark::plonk_constraint<BlueprintFieldType> bit_res = var(W0, 0) *
       ↪ var(column_idx, 0);
```

```
224                                     (bit_res + (1 - var(column_idx, 0))
                                           ↪ );
```

CVF-42. INFO

- **Category** Procedural
- **Source** poseidon_15_wires.hpp

Description We didn't review these files.

Client Comment *The files are not related to the circuit description*

```
30 #include <nil/crypto3/detail/literals.hpp>
   #include <nil/crypto3/algebra/matrix/matrix.hpp>
```

```
33 #include <nil/crypto3/zk/blueprint/plonk.hpp>
   #include <nil/crypto3/zk/assignment/plonk.hpp>
```

```
36 #include <nil/crypto3/algebra/fields/pallas/base_field.hpp>
   #include <nil/crypto3/algebra/fields/vesta/base_field.hpp>
```

CVF-43. INFO

- **Category** Documentation
- **Source** poseidon_15_wires.hpp

Description These round constants are not public.

Recommendation Consider referencing how they were derived.

```
72 constexpr static const std::array<std::array<typename FieldType::
   ↪ value_type, state_size>, rounds_amount> round_constant =
```

CVF-44. INFO

- **Category** Documentation
- **Source** poseidon_15_wires.hpp

Description This MDS matrix is not public.

Recommendation Consider referencing.

```
54  {{0x1a9bd250757e29ef4959b9bef59b4e60e20a56307d6491e7b7ea1fac679c7903_
    cppui253,

    0x384aa09faf3a48737e2d64f6a030aa242e6d5d455ae4a13696b48a7320c506cd_
    cppui253,

60  0x3d2b7b0209bc3080064d5ce4a7a03653f8346506bfa6d076061217be9e6cfed5_
    cppui253
    }},
    {{
    0x9ee57c70bc351220b107983afcfabbea79868a4a8a5913e24b7aaf3b4bf3a42_
    cppui253,

    0x20989996bc29a96d17684d3ad4c859813115267f35225d7e1e9a5b5436a2458f_
    cppui253,

70  0x14e39adb2e171ae232116419ee7f26d9191edde8a5632298347cdb74c3b2e69d_
    cppui253
    }},
```

CVF-46. FIXED

- **Category** Readability
- **Source** poseidon_15_wires.hpp

Recommendation This value should be derived from “rounds_amount” and “rounds_per_row”.

```
717 constexpr static const std::size_t rows_amount = 12;
```

CVF-47. INFO

- **Category** Suboptimal
- **Source** poseidon_15_wires.hpp

Description The variable “z” is not used inside the loop.

Recommendation Consider removing this variable and iterating over “i” values instead like this: for (std:size_t i = 0; i < gates_amount; i++).

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
758 for (std::size_t z = 0; z < rounds_amount; z += rounds_per_row) {
```

CVF-49. FIXED

- **Category** Unclear behavior
- **Source** poseidon_15_wires.hpp

Description This code works only for state_size=3.

Recommendation Consider asserting this explicitly or make it a loop.

```
782 for (int j = 0; j < state_size; j++) {
        next_state[j] = state[0].pow(sbox_alpha) * mds[j][0] +
                        state[1].pow(sbox_alpha) * mds[j][1] +
                        state[2].pow(sbox_alpha) * mds[j][2] +
                           ↪ round_constant[(i - row) * 5][j];
    }
```

CVF-51. INFO

- **Category** Suboptimal
- **Source** poseidon_15_wires.hpp

Description The coded fragment starting with this lime repeats “rounds_per_row” times with slight modifications.

Recommendation Consider using a loop and/or extracting a function to make the code smaller.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
782 for (int j = 0; j < state_size; j++) {
```

```
791 for (int j = 0; j < state_size; j++) {
```

```
800 for (int j = 0; j < state_size; j++) {
```

```
809 for (int j = 0; j < state_size; j++) {
```

```
818 for (int j = 0; j < state_size; j++) {
```

CVF-53. INFO

- **Category** Procedural
- **Source** endo_scalar.hpp

Description We didn’t review these files.

Client Comment *The files are not related to the circuit description.*

```
32 #include <nil/marshalling/algorithms/pack.hpp>
```

```
34 #include <nil/crypto3/algebra/curves/vesta.hpp>
   #include <nil/crypto3/algebra/curves/pallas.hpp>
```

```
37 #include <nil/crypto3/zk/snark/arithmetization/plonk/
      ↪ constraint_system.hpp>
```

```
39 #include <nil/crypto3/zk/blueprint/plonk.hpp>
40 #include <nil/crypto3/zk/assignment/plonk.hpp>
   #include <nil/crypto3/zk/algorithms/generate_circuit.hpp>
```

CVF-54. INFO

- **Category** Documentation
- **Source** endo_scalar.hpp

Description It is unclear what this component does.

Recommendation Consider explaining in a comment.

Client Comment *We do not consider it as an issue.*

```
48 template<typename ArithmetizationType,
           typename CurveType,
50         std::size_t ScalarSize,
           std::size_t... WireIndexes>
class endo_scalar;
```

CVF-55. INFO

- **Category** Suboptimal
- **Source** endo_scalar.hpp

Recommendation This initialization is redundant, as the field is reinitialized in the constructor.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
136 var output = var(0, 0, false);
```

CVF-56. FIXED

- **Category** Bad datatype
- **Source** endo_scalar.hpp

Recommendation These variables should be turned into constants.

```
172 std::size_t crumbs_per_row = 8;
std::size_t bits_per_crumb = 2;
std::size_t bits_per_row =
```

CVF-57. FIXED

- **Category** Procedural
- **Source** endo_scalar.hpp

Description This implicitly assumes that ScalarSize <= 255.

Recommendation Consider explicitly asserting this fact.

```
183 std::array<bool, 255> bits_msb_all = nil::marshalling::pack<nil::
       ↪ marshalling::option::big_endian>(integral_scalar, status);
    std::copy(bits_msb_all.end() - ScalarSize, bits_msb_all.end(),
       ↪ bits_msb.begin());
```

CVF-58. FIXED

- **Category** Procedural
- **Source** endo_scalar.hpp

Description Here “ScalarSize” lowest bits are copied from “bits_msb_all” into “bits_msb”, while other bits are ignored.

Recommendation Consider explicitly asserting that ignored bits are zero.

```
184 std::copy(bits_msb_all.end() - ScalarSize, bits_msb_all.end(),
       ↪ bits_msb.begin());
```

CVF-59. INFO

- **Category** Readability
- **Source** endo_scalar.hpp

Recommendation Consider using “ScalarSize” instead of “bits_msb.size()”.

```
190 for (std::size_t chunk_start = 0; chunk_start < bits_msb.size();
       ↪ chunk_start += bits_per_row) {
```

CVF-60. FIXED

- **Category** Procedural
- **Source** endo_scalar.hpp

Description This assumes that “ScalarSize” is a factor of “bits_per_row”.

Recommendation Consider asserting this fact explicitly.

```
190 for (std::size_t chunk_start = 0; chunk_start < bits_msb.size();
       ↪ chunk_start += bits_per_row) {
```

CVF-61. INFO

- **Category** Unclear behavior
- **Source** endo_scalar.hpp

Description This code is valid only when “bits_per_cramb” is 2.

Recommendation Consider replacing with a loop.

Client Comment *We do not consider it as an issue.*

```
197 typename BlueprintFieldType::value_type b0 = static_cast<int>(
       ↪ bits_msb[crumb + 1]);
    typename BlueprintFieldType::value_type b1 = static_cast<int>(
       ↪ bits_msb[crumb + 0]);
```

CVF-63. INFO

- **Category** Suboptimal
- **Source** endo_scalar.hpp

Description The values 11/6, -5/2, 2/3, 29/6 and -7/2 are calculated on every function invocation.

Recommendation Consider precomputing them.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
240 return (F(11) * F(6).inversed()) * x + (-F(5) * F(2).inversed()) * x
      ↪  * x +
          (F(2) * F(3).inversed()) * x * x * x;

245 return -F::one() + (F(29) * F(6).inversed()) * x + (-F(7) * F(2).
      ↪ inversed()) * x * x +
          (F(2) * F(3).inversed()) * x * x * x;
```

CVF-64. INFO

- **Category** Suboptimal
- **Source** endo_scalar.hpp

Description These constraints are very similar.

Recommendation Consider extracting a utility function to avoid code duplication.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
250 bp.add_constraint(var(W7, 0) * (var(W7, 0) - 1) * (var(W7, 0) - 2) *
    ↪  (var(W7, 0) - 3));

252 bp.add_constraint(var(W8, 0) * (var(W8, 0) - 1) * (var(W8, 0) - 2) *
    ↪  (var(W8, 0) - 3));

254 bp.add_constraint(var(W9, 0) * (var(W9, 0) - 1) * (var(W9, 0) - 2) *
    ↪  (var(W9, 0) - 3));

256 bp.add_constraint(var(W10, 0) * (var(W10, 0) - 1) * (var(W10, 0) -
    ↪ 2) * (var(W10, 0) - 3));

258 bp.add_constraint(var(W11, 0) * (var(W11, 0) - 1) * (var(W11, 0) -
    ↪ 2) * (var(W11, 0) - 3));

260 bp.add_constraint(var(W12, 0) * (var(W12, 0) - 1) * (var(W12, 0) -
    ↪ 2) * (var(W12, 0) - 3));

262 bp.add_constraint(var(W13, 0) * (var(W13, 0) - 1) * (var(W13, 0) -
    ↪ 2) * (var(W13, 0) - 3));

264 bp.add_constraint(var(W14, 0) * (var(W14, 0) - 1) * (var(W14, 0) -
    ↪ 2) * (var(W14, 0) - 3));
```

CVF-66. INFO

- **Category** Procedural
- **Source** limbs.hpp

Description We didn't review these files.

```
35 #include <nil/marshalling/algorithms/pack.hpp>
```

```
37 #include <nil/crypto3/zk/snark/arithmetization/plonk/
   ↪ constraint_system.hpp>
```

```
39 #include <nil/crypto3/zk/blueprint/plonk.hpp>
40 #include <nil/crypto3/zk/assignment/plonk.hpp>
   #include <nil/crypto3/zk/component.hpp>
```

CVF-67. INFO

- **Category** Suboptimal
- **Source** limbs.hpp

Recommendation The "scalar" value should be 2^64 rather than 2.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
127 typename BlueprintFieldType::value_type scalar = 2;
    scalar = scalar.pow(64) * second_limb + first_limb;
```

CVF-71. INFO

- **Category** Procedural
- **Source** unified_addition.hpp

Description We didn't review this file.

Client Comment *The files are not related to the circuit description.*

```
33 #include <nil/crypto3/zk/snark/arithmetization/plonk/
   ↪ constraint_system.hpp>
```

```
35 #include <nil/crypto3/zk/blueprint/plonk.hpp>
   #include <nil/crypto3/zk/assignment/plonk.hpp>
   #include <nil/crypto3/zk/algorithms/generate_circuit.hpp>
```

CVF-72. INFO

- **Category** Suboptimal
- **Source** unified_addition.hpp

Description This constant is not used.

Recommendation Consider removing it.

Client Comment *Constraint is used by algorithm in generate_circuit.hpp.*

```
83 constexpr static const std::size_t selector_seed = 0x0f01;
```

CVF-73. INFO

- **Category** Suboptimal
- **Source** unified_addition.hpp

Description The expression “P + Q” is already calculated and its value is stored in the variable “R“.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
167 assignment.witness(W4)[j] = (P + Q).X;
    assignment.witness(W5)[j] = (P + Q).Y;
```

CVF-74. INFO

- **Category** Suboptimal
- **Source** unified_addition.hpp

Description The expression “Q.X - P.X” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *The influence of the suboptimal implementation on the overall preformance is so minor that we do not consider it as an issue.*

```
185 assignment.witness(W10)[j] = (Q.Y - P.Y) / (Q.X - P.X);
```

```
189 assignment.witness(W8)[j] = (Q.X - P.X).inversed();
```

CVF-76. FIXED

- **Category** Procedural
- **Source** unified_addition.hpp

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

Client Comment *It's a bug and copy constraints should be here.*

```
256                                 const std::size_t start_row_index) {
}
```

ABDK
Consulting

About us

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

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

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting