MASP Circuits Review

Anoma

inference

20220602 – FINAL

Contents

1 Summary

Anoma solicited us to assess the correctness and security of new Groth16 circuits integrated in Anoma's extension of the Zcash code base, to implement the multi-asset shielded pool (MASP) logic. The code and documentation are available in the anoma/masp GitHub repository:

- The Convert circuit, in masp_proofs/src/circuit/convert.rs
- The Spend and Output circuits (adapted from Sapling), in masp_proofs/src/circuit/sapling.rs

We relied on the MASP specifications in docs/multi-asset-shielded-pool and the circuits documentation from the anoma/specs, in src/architecture/ledger/shielded-execution/masp.

We evaluated the circuits for correctness, correct usage of the circuit creation helpers, for logic flaws, for software security bugs, and other problems and inconsistencies. Some of the related risks are listed in section 0.10 of the MASP specification.

2 Findings

None of the findings below appears to be a security vulnerability, let alone an exploitable. We nonetheless report these as general quality improvements.

2.1 MASP-01: NoteCommit inconsistency with specs

In the original Zcash protocol specification, the NoteCommit is defined with regards to a fixed (3) number of arguments that are optionally converted to binary and concatenated in a predetermined order before being passed as input to the $\mathsf{WindowedPedersenCommit}()$ function. In both the Output and Spend circuits implementation, via the `expose_value_commitment()` function, the NoteCommit takes as input an additional parameter vb representing the value base (a `asset_generator` variable in the code), taken as a group element. This last input vb must first be converted to binary using $\mathsf{repr}_{\mathbb{J}}$ and becomes the first input to $\mathsf{WindowedPedersenCommit}()$. The function definition should emphasize that vb is converted.

To summarize, the specification defines:

$$\mathsf{cm}^{\mathsf{old}} = \mathsf{NoteCommit}^{\mathsf{Sapling}}_{\mathsf{rcm}^{\mathsf{old}}}(\mathsf{repr}_{\mathbb{J}}(\mathsf{g_d}), \mathsf{repr}_{\mathbb{J}}(\mathsf{pk_d}), \mathsf{v}^{\mathsf{old}}, \mathsf{repr}_{\mathbb{J}}(\mathsf{vb})).$$

However, the circuits define it as:

$$\mathsf{NoteCommit}^{\mathsf{Sapling}}_{\mathsf{rcm}}(\mathsf{g}\star_{\mathsf{d}}, \mathsf{pk}\star_{\mathsf{d}}, \mathsf{v}, \mathsf{vb}\star) := \mathsf{WindowedPedersenCommit}_{\mathsf{rcm}}\Big([1]^6 \| \mathsf{vb}\star \| \mathsf{I2LEBSP}_6 4(\mathsf{v}) \| \mathsf{g}\star_{\mathsf{d}} \| \mathsf{pk}\star_{\mathsf{d}}\Big).$$

Since the definition of $\mathsf{NoteCommit}^{\mathsf{Sapling}}_{\mathsf{rcm}}$ is modified from the original Zcash specification, the document could mention how the new input $\mathsf{vb}\star$ is now processed.

2.2 MASP-02: Output **circuit inconsistency with specs**

The function $\mathsf{PRF}^{\mathsf{vcgMASP}}$ should be defined to return a bit representation of the hash image to be consistent with the original specification. The output should then be denoted as $\mathsf{vb}\star := \mathsf{PRF}^{\mathsf{vcgMASP}}(t) \in \mathbb{B}^{[\ell_{\mathbb{J}}]}$, where the $\star$ symbol means that we are taking the binary representation of the group element vb.

Assuming the Output circuit takes as private input the group element vb, the specification would then have the following constraints:

$$\begin{aligned} \text{Value base computation:} \quad & \mathsf{vb}\star := \mathsf{PRF}^{\mathsf{vcgMASP}}(t) \\ \text{Value base integrity:} \quad & \mathsf{vb}\star = \mathsf{repr}_{\mathbb{J}}(\mathsf{vb}) \\ \text{Note commitment integrity:} \quad & \mathsf{cm}_u = \mathsf{Extract}_{\mathbb{J}^{(r)}}\left(\mathsf{NoteCommit}^{\mathsf{Sapling}}_{\mathsf{rcm}^{\mathsf{new}}}(\mathsf{g}\star_{\mathsf{d}}, \mathsf{pk}\star_{\mathsf{d}}, \mathsf{v}, \mathsf{vb}\star)\right) \end{aligned}$$

2.3 MASP-03: Output **circuit inconsistency with specs**

When computing the old value commitment $\mathsf{cv}^{\mathsf{old}}$, the variables $(\mathsf{v}^{\mathsf{new}}, \mathsf{rcm}^{\mathsf{new}})$ in the specification should actually be $(\mathsf{v}^{\mathsf{old}}, \mathsf{rcm}^{\mathsf{old}})$.

2.4 MASP-04: Allowed scalar overflows are undocumented

Some private inputs such as $\alpha, \mathsf{esk}, \mathsf{nsk} \in \{0...2^{\ell_{scalar}} - 1\}$ represent the bit-decomposition of an element in the scalar field of an elliptic curve with order $r_{\mathbb{J}}$. Unlike the variables $\mathsf{rcm}, \mathsf{rcv}$, the specification does not mention explicitly that the corresponding field elements do not need to constrained to the range $\{0, \ldots, r_{\mathbb{J}} - 1\}$. This would require more constraints in the circuit which are unnecessary since they are only used to compute a group scalar multiplication (which requires the bit-decomposition). If a congruent representation is given, the resulting group element would be the same, even if an overflow occurs. Witnessing this value serves only as a proof-of-knowledge of the secret. Moreover, since the Groth16 proofs are already randomized for zero-knowledge, it therefore does not make a difference if the prover uses a (possible) congruent value of $\alpha, \mathsf{esk}, \mathsf{nsk}$.

2.5 MASP-05: Specs typo

In section 0.12.2 of the specification, “The original Sapling Output circuit” should be “The original Sapling Spend circuit”.

3 Disclaimer

This security assessment report (“Report”) by Inference AG (“Inference”) is solely intended for [insert name of client] (“Client”) with respect to the Report’s purpose as agreed by the Client. The Report may not be relied upon by any other party than the Client and may only be distributed to a third party or published with the Client’s consent. If the Report is published or distributed by the Client or Inference (with the Client’s approval) then it is for information purposes only and Inference does not accept or assume any responsibility or liability for any other purpose or to any other party.

Security assessments of a software or technology cannot uncover all existing vulnerabilities. Even an assessment in which no weaknesses are found is not a guarantee of a secure system. Generally, code assessments enable the discovery of vulnerabilities that were overlooked during development and show areas where additional security measures are necessary. Within the Client’s defined time frame and engagement, Inference has performed an assessment in order to discover as many vulnerabilities of the technology or software analyzed as possible. The focus of the Report’s security assessment was limited to the general items and code parts defined by the Client. The assessment shall reduce risks for the Client but in no way claims any guarantee of security or functionality of the technology or software that Inference agreed to assess. As a result, the Report does not provide any warranty or guarantee regarding the defect-free or vulnerability-free nature of the technology or software analyzed.

In addition, the Report only addresses the issues of the system and software at the time the Report was produced. The Client should be aware that blockchain technology and cryptographic assets present a high level of ongoing risk. Given the fact that inherent limitations, errors or failures in any software development process and software product exist, it is possible that even major failures or malfunctions remain undetected by the Report. Inference did not assess the underlying third party infrastructure which adds further risks. Inference relied on the correct performance and execution of the included third party technology itself.