Least Authority
PRIVACY MATTERS

MPC Circuit

Security Audit Report

Worldcoin

Final Audit Report: 7 March 2025

Table of Contents

Overview

Background

Tools for Humanity has requested that Least Authority perform a second security audit of Worldcoin's MPC Circuit within the Mobile IrisCode Self-Custody Upgrade project, which allows users to self-host biometric data on their personal device while supporting high-integrity authentication for the World ID service.

Project Dates

- **January 13, 2025 - January 24, 2025:** Initial Code Review *(Completed)*
- **January 28, 2025:** Delivery of Initial Audit Report *(Completed)*
- **February 5, 2025:** Delivery of Updated Initial Audit Report *(Completed)*
- **March 7, 2025:** Verification Review *(Completed)*
- **March 7, 2025:** Delivery of Final Audit Report *(Completed)*

Review Team

- George Gkitsas, Security / Cryptography Researcher and Engineer
- Jasper Hepp, Security Researcher and Engineer
- Burak Atasoy, Project Manager
- Jessy Bissal, Technical Editor

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and a review of the MPC Circuits within the Mobile IrisCode Self-Custody Upgrade project followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repositories are considered in scope for the review:

- modulus-labs-remainder:
    - https://github.com/Modulus-Labs/Remainder/tree/mpc-circuits/remainder_prover/src/worldcoin_mpc
    - https://github.com/Modulus-Labs/Remainder/tree/mpc-circuits/remainder_hyrax/src/hyrax_worldcoin_mpc

Specifically, we examined the Git revision for our initial review:

- `2e081b8053943ac19d8ccc91a859155870cae5d8`

For the verification, we examined the Git revision:

- `177173e6aa5bbff0668c9f1e8c04b8329ca49736`

For the review, this repository was cloned for use during the audit and for reference in this report:

- modulus-labs-remainder:
  https://github.com/LeastAuthority/modulus-labs-remainder

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zkSNARKs without trusted setup" (referred to as the Hyrax paper). *IACR Cryptology ePrint Archive*, 2017, [WTS+17]
- "Scaling Intelligence: Verifiable Decision Forest Inference with Remainder." *Modulus Labs*, 2024. [Modulus Labs 24]
- Project Overview: https://future-seaplane-bec.notion.site/Mobile-IrisCode-Client-side-ZKML-for-Private-Biometric-Authentication-efbc6ba76206444fbea91e2f9ffdab23#a2a3c1e026da4564bd40c1d06cfdf089
- Shamir Secret Sharing generation circuit: https://future-seaplane-bec.notion.site/Audit-Companion-Shamir-Secret-Share-Generation-Prover-Verifier-13768687d27e80979588f4e0747d86a0
- MPC Circuit Builders (ext): https://www.notion.so/MPC-Circuit-Builders-ext-13768687d27e8023baabc4f3dd27d792
- Websites:
  - https://worldcoin.org
  - https://www.toolsforhumanity.com

In addition, this audit report references the following documents:

- J. Surin and S. Cohney, "A Gentle Tutorial for Lattice-Based Cryptanalysis." *IACR Cryptology ePrint Archive*, 2023, [SC23]
- Crate zeroize: https://docs.rs/zeroize/latest/zeroize/index.html#
- Pedersen Commitments: https://www.zkdocs.com/docs/zkdocs/commitments/pedersen
- Galois Ring MPC Inner Products, Technical Analysis (DRAFT v0.1).pdf (*shared with Least Authority via Slack on 19 November 2024*)

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation and alignment with specifications;
- Denial of Service (DoS) and other security exploits that would impact the intended use or disrupt the execution;
- Inappropriate permissions and excess authority;
- Opportunities for targeted manipulation or malicious influence on the AI system;
- Data privacy, including data leaking, unexpected data inference, voids, and information integrity;
- Effective guardrails and prevention of prompt engineering attacks for extraction or injections;
- Unplanned favoring of majority data, ignoring outliers, and potential subsequent bias;
- Any applicable use of cryptography, including key management and encryption protocols;
- Challenges to transparency and governance measures; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

This project is part of the Worldcoin solution for private iris-based identification. Worldcoin aims to improve this solution by ensuring that no sensitive data is revealed to its servers. Instead, privacy-sensitive computations will be performed directly on the user's phone. During user registration, the system performs a uniqueness check based on the user's iris code. This comparison check is done through a multi-party computation (MPC) to avoid having all iris-relevant information stored in one place. Additionally, zero-knowledge proofs are produced by the user to prove that the user has not tampered with the data when generating the MPC shares. This project implements the generation and verification of these proofs through the Hyrax protocol ([WTS+17], [Modulus Labs 24])–a zk-SNARK-based proof system that was implemented by the same team and was also assessed by Least Authority in a previous audit, which we delivered on January 2, 2025.

System Design

The project consists of a circuit (called MPC circuit), a prover, and a verifier (tested in the file `src/bin/worldcoin_mpc.rs`). The complete setup also contains the circuit (called V3 circuit) that was not in scope for this review, as our team had reviewed it previously. The two circuits together prove the correct generation of the iris code from an iris image (V3 circuit) and the correct computation of secret shares that encode the iris code as a secret (MPC circuit). Each proof with a secret share is shared with one of the three Worldcoin servers. The Worldcoin servers use the secret shares to verify the uniqueness of the sign-up attempt. The user acts as a prover, and the Worldcoin servers as verifiers.

MPC Circuit

The MPC Circuit encodes the iris code and mask into a linear polynomial with the user-generated, random slope `m`. From the linear polynomial, three shares are produced by evaluating the polynomial on publicly known evaluation points. Two of the three points are sufficient to reproduce the encoded secret.

We identified one general issue that led to two critical findings (Issue A and Issue B). In theory, the computation of shares takes place in the Galois ring GR4 equal to `(\mathbb{Z}/2^{16}\mathbb{Z})[x]/(x^4-x-1)`. In practice, the circuit inputs are elements from the finite field `Fr` and not from GR4–that is, the circuit is reducing `modulus r` and not `2^16`. This discrepancy leads to over- and underflow issues. In particular, we found that this leads to an incorrect calculation of the secret shares because the masked iris code takes negative values that are reduced in `Fr` and not in `Z_{2^16}` (Issue A).

A similar problem arises for the slope `m`. The value is implicitly assumed to be in GR4; however, it is controlled by the prover and not checked by the verifier. Hence, the slope `m` can take any value in `Fr`, which can lead to overflows and incorrect calculations of the secret shares (Issue B). This issue allows a malicious attacker to sign up multiple times, breaking the uniqueness check of the setup.

We did not find any further issues with respect to the implementation of the circuit logic.

Verifier

In the current setup, only one verifier exists. Note that the goal is to have three verifying parties (Worldcoin servers). We reviewed the verifier under the assumption that there are three different proofs and verifiers. Since we found several issues in the verifier, we recommend rewriting the verifier and decoupling the verifier and prover (Suggestion 3) to facilitate a deeper and more comprehensive analysis.

The verifier trusts the prover to provide the correct configuration file (including a secure hash function) (Issue C) and the correct Pedersen commitment generators (Issue D). Furthermore, the verifier should truncate the secret share obtained via the proof in the MPC circuit (Issue G).

For each secret share, the verifier obtains two proofs, one for each circuit. However, the verifier fails to enforce these two proofs to use the same commitments for the iris code and the masked code (Issue F). In addition, the verifier (or the three verifying parties) need to ensure that the commitments on the slopes are the same across the three different proofs (Issue E).

Statistical Attacks

We investigated possible statistical attack vectors for information leakage through the secret sharing mechanisms and assessed them to mitigate risks.

We identified concerns stemming from the choice of evaluation points, the encoding matrix, and the domain size of the iris codes. The zeros in the evaluation points eliminate some factors within their multiplication calculation with slopes, effectively rendering part of the slope values irrelevant. With regards to the encoding matrix, the multiplication by even constants (within the ring) introduces heavy biases, which could assist in distinguishing attacks. The zeros present in the encoding matrix also reduce the calculation’s diffusion factor. Additionally, the small iris code domain further increases the distance of the secrets’ distribution from the uniform distribution.

We did not identify ways in which these choices could lead to a direct attack path. However, this was a preliminary statistical analysis, and we recommend performing a comprehensive statistical analysis, which includes accurate calculations of the effectiveness of both attacks and countermeasure choices.

Guided Brute-Forcing

Additionally, we assessed guided brute-forcing based on the fact that secrets can take only a limited amount of values out of their domain. We explored different options for heuristics, assuming knowledge of the success of the final threshold check and knowledge of when slopes are zero. We did not identify any attacks within this context.

Lattice-Based Attacks

We examined the efficacy of Lattice-based attacks since the construction involves sets of linear equations of secrets with the slope randomness acting as the additive noise. We did not identify any issues when constructing the Learning With Errors (LWE) problem in a straightforward manner. This is because LWE requires the noise to be narrow, but the slope randomness is drawn from the entire Galois ring. An alternative approach would be to treat the iris data as noise and try to recover slope values, which are sufficient for recovering the secrets. In this case, the advantage is that the noise has a much smaller domain since the iris codes can only take values of zero and one. However, this problem requires a set of equations for the same slope and different secrets. We did not identify a feasible way of obtaining such a set. The above is a preliminary effort, and we encourage further examination of the potential impact of Lattice-based attacks.

Dependencies

Our team did not identify any security concerns resulting from the unsafe use of dependencies.

Code Quality

We performed a manual review of the repositories in scope and found the code to be clean, well-organized, and in adherence to development best practices.

Tests

The repositories in scope include sufficient test coverage. Our team additionally noted that all the tests pass. Nevertheless, we recommend adding a test that checks the computation of the circuit due to Issue A.

Documentation and Code Comments

The project documentation provided by the Worldcoin team was sufficient in describing the intended functionality of the system. Additionally, the codebase is well-commented, which was helpful in understanding the intended functionality of most of the components. However, we noted a few minor deviations that we recommend resolving (Suggestion 1).

Scope

The scope of this audit included the MPC circuit, the accompanying files in the folder `remainder_prover/src/worldcoin_mpc`, the test functions implemented in the folder `remainder_hyrax/src/hyrax_worldcoin_mpc`, as well as the file `remainder_hyrax/src/bin/worldcoin_mpc.rs`. We additionally reviewed the verifier and prover functions, but only to the extent that they included the code calls to the Hyrax prover and verifier, as we had already comprehensively reviewed the Hyrax verifier and prover in a previous audit.

Additionally, our team noted that, as with the previous audit, several critical verifier-related functionalities remain incomplete, which made it difficult to review the implementation in its current state. We recommend performing a comprehensive, follow-up security audit once the verifier is implemented in its final version (independently from the prover) and integrated into the Worldcoin system.

Specific Issues & Suggestions

We list the issues and suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

ISSUE / SUGGESTION	STATUS
Issue A: [Circuit] Incorrect Computation of Shares Within Circuit Due To Incorrect Handling of Negative Values for Masked Iris Code	Resolved
Issue B: [Circuit] Missing Range Check on Slope Allows Multiple Sign-Ups	Resolved
Issue C: [Verifier] Verifier Trusts Prover To Provide Correct Configuration	Resolved
Issue D: [Verifier] Verifier Trusts Prover To Provide Correct Pedersen Commitment Generators	Resolved
Issue E: [Verifier] Verifier Does Not Check the Slope Commitments Across Proofs	Resolved
Issue F: [Verifier] Verifier Does Not Check the Iris Code and Mask Commitment Between Proofs for the V3 Circuit and MPC Circuit	Resolved
Issue G: [Verifier] Verifier Does Not Truncate the Shares	Resolved

Issue H: [Prover] Leakage Through Residual Data	Resolved
Issue I: [Prover] Slope Randomness Is Not Adequate	Resolved
Suggestion 1: [All Components] Improve Documentation and Code Comments	Resolved
Suggestion 2: [Prover] Avoid Multiple Secret Sharing Executions of an Iris With Different Randomness	Unresolved
Suggestion 3: [Verifier] Implement the Verifier Independently From the Prover	Resolved
Suggestion 4: [All Components] Reduce Susceptibility to Statistical Attacks	Unresolved

Issue A: [Circuit] Incorrect Computation of Shares Within Circuit Due To Incorrect Handling of Negative Values for Masked Iris Code

Location

`src/worldcoin_mpc/circuits.rs`

Synopsis

The masked iris code takes values in `{-1, 0, +1}`. Negative values are not handled properly, which leads to an incorrect computation of the secret shares.

Impact

Critical. The circuit produces incorrect shares if the masked iris code takes the value minus one.

Preconditions

None. This is true for every user since the probability of having a negative value for the masked iris code is high.

Feasibility

Straightforward.

Technical Details

The core of the problem is that GR4 elements are represented as elements of the form `[a0,a1,a2,a3]` with `ai` in `Fr` within the circuit. A correct representation would enforce `ai` in `Z_2^16`.

In what follows, we describe the computational steps in the circuit:

1. When the iris code and mask are combined, this results in the masked iris code u in `{0,+1,-1}`. Each `-1` is taken as `r-1>0` in the circuit since the operation occurs over `Fr`.
2. If u times E is taken as the matrix multiplication (again over `Fr`) to map u to GR4, each entry of the result gets reduced `mod r`.
3. For the slope times evaluation point `(m*alpha_j)`, the code implements the GR4 wiring and the result is an element of the form `[a0,a1,a2,a3]` with each ai in `Fr`.
4. After this step, the addition `b + (m*alpha_j)` is performed in `Fr`.
5. The computation `q*2^16 + s - sb` is then performed. Note that, again, `add` and `sub` occur in `Fr`, and the result is reduced `mod r`.

6. In the last step, the final value is checked to verify if it is equal to `0`, in `Fr`.

Next, consider the aforementioned setup with values as an example. In this example, the representation for GR4 elements is altered. First, the representation is as noted above, and of the form `[a0,a1,a2,a3]` with `ai` in `Fr`. Second, the representation is, as polynomials, reduced by the quotient polynomial `x^4-x-1`. For the setup, consider the prime r for the field `Fr`. Note that `r mod 2^16` is zero. The values u and `m` can be selected in their simplest form, and the computational step `u*E` can be ignored since it only yields more complicated negative values but does not change the message of the example.

Example setup: u `= [-1,0,0,0] (in Fr^4), m = 1 (in GR4), alpha1 = 1 (in GR4), alpha2 = x (in GR4), alpha3 = 1+x (in GR4)`.

With this data, the circuit is transforming the masked iris code into u `= [r-1,0,0,0]`. Assume b `= u*E = (r-1, 0, 0, 0)` (that is, E is the identity matrix). Then for each share, the following computation can be performed while varying the evaluation point:

```
1.  sb1 = b+m*alpha1 = (r-1+1,0,0,0)    = (0,0,0,0)
2.  sb2 = b+m*alpha2                    = (r-1,1,0,0)
3.  sb3 = b+m*alpha3 = (r-1+1,1,0,0)    = (0,1,0,0)
```

The final shares and quotients are then:

```
1.  s1=0, q1=0
2.  s2=x, q2=n since sb2 mod 2^16=0+x and with n such that (r-1) div 2^16=n*q
3.  s3=x, q3=0
```

This results in the points p1=`(1,0)`, p2=`(x,x)`, p3=`(1+x,x)`.

When the slope m is now recalculated from pairs of two points with the common formula `(y1-y2)/(x1-x2)` in GR4, for the pair combining points 2 and 3, this results in `(x-x)/(x-(1+x)) = 0`, and not 1. This is because `r-1` was taken as the first component in u, instead of having it reduced by `2^16` and taking `2^16-1` as the value. If one performs the calculation with this value, the recalculation of the slope returns 1 as expected.

Remediation

We recommend modifying the masked iris code to reduce it properly to `modulus 2^16`. Note that the reduction needs to be constrained in the circuit. In addition, we recommend adding tests that check the correctness of the circuit with respect to the data output.

Status

The Worldcoin team has correctly mapped the masked iris code values to the Galois field by adding `2^16` to the masked code calculation, which maps `0` to `0`, 1 to 1, and `-1` to `2^16-1`.

Verification

Resolved.

Issue B: [Circuit] Missing Range Check on Slope Allows Multiple Sign-Ups

Location

`remainder_prover/src/worldcoin_mpc/circuits.rs`

`src/worldcoin_mpc/components.rs`

Synopsis

For large values of the slope m, the computation within the circuit overflows the finite field Fr. This leads to incorrect values for the secret shares and allows multiple sign-ups from an attacker with a high success rate.

Impact

Critical. An attacker can likely sign up multiple times.

Preconditions

None.

Feasibility

Straightforward.

Technical Details

The computation in the circuit occurs in the finite field Fr; however, the computation of the share within the circuit should occur in GR4–that is, reduced `modulo 2^16` in each component. This leads to the following problem, as illustrated in this example:

Select `m` such that `sb = b+m*alpha_j = r`. As a result:

1. The circuit code reduces `sb` in `Fr`. Consequently, `sb = r = 0`. This results in the values `q=0` and `s=0`, which will be sent to the verifier.
2. In theory, a reduction should not occur in `Fr`. Since `r` is prime (and not 2) there exist `x,y!=0` such that `r = x*2^16 + y`. Therefore, `sb = r = x*2^16 + y`. Due to this, the shared values are `q=y` and `s=x` with `x,y != 0`.

Both cases produce different outcomes for the shares. The "correct" outcome is the second case because that is the point that lies on the line (constructed in GR4), but the code is producing a different share. The incorrect shares produced in the first case lead to an incorrect linear polynomial and, hence, to incorrect secrets reconstructed by two parties. Note that only `sb >= r` is needed so that this example produces the difference between the two cases.

This can be used to sign up multiple times. The attacker can pick different slopes that lead to an overflow for `sb` in `Fr`. For each slope, the attacker produces a different set of secret shares, and, therefore, the uniqueness check can be bypassed.

Remediation

We recommend adding a range check on the slope `m` via a lookup table.

Status

The Worldcoin team has introduced range checks for the slopes using multiplicities-based lookups.

Verification

Resolved.

Issue C: [Verifier] Verifier Trusts Prover To Provide Correct Configuration

Location

`src/config/mod.rs#L238`

Synopsis

The verifier's configuration (including the choice of hash function) is derived directly from data in the proof itself via `new_from_prover_config`. As a result, an attacker acting as the prover can force the verifier to use insecure parameters—specifically, a non-cryptographically secure hash function—compromising the soundness of the proof system.

Impact

Critical. A malicious prover can cause the verifier to adopt a weak hash function, and, hence, the security guarantees of the system are undermined. This could allow an attacker to forge or manipulate proofs. If exploited, it compromises the core cryptographic assumptions and undermines the trust in the validity of proofs.

Preconditions

None.

Feasibility

Straightforward.

Technical Details

The function `new_from_prover_config` constructs a `VerifierConfig` object from data that originates in the prover's proof. This process includes selecting the hash function. The current code permits the `DefaultRustHash` type, which is not cryptographically secure. Although there is a check comparing the newly created verifier config to the prover config, it does not mitigate the issue because the verifier config is already produced from the untrusted proof data.

Remediation

We recommend rewriting the verifier code to prevent it from blindly deriving security-critical parameters from the proof.

Status

The Worldcoin team currently enforces the configuration by pinning it to the same constant value for the prover and verifier.

Verification

Resolved.

Issue D: [Verifier] Verifier Trusts Prover To Provide Correct Pedersen Commitment Generators

Location

`src/bin/worldcoin_mpc.rs#L215`

Synopsis

The verifier currently relies on the prover to supply the Pedersen commitment generators via the proof. If the prover can supply generators such that a discrete logarithm relation is known or easy to compute, the binding property of the commitment scheme will be broken.

Impact

Critical. Knowledge of the discrete logarithm between the generators allows the attacker to break the binding property. This directly undermines a core security guarantee of the protocol.

Preconditions

None. The verifier is not checking the correctness of the Pedersen generators.

Feasibility

If a malicious prover can choose or manipulate generator parameters, the attack is straightforward—in this case, it requires simply embedding a known or easily computable discrete log relationship into the generators.

Technical Details

A known discrete log relationship between generators for a Pedersen Commitment can easily be exploited. For more details, see here.

Remediation

We recommend modifying the verifier logic to generate the generators via a public string that the prover and verifier agreed upon before the execution of the protocol.

Status

The Worldcoin team had previously identified this issue and remediated it prior to the audit by using the same Pedersen generator for both the prover and verifier. The implemented solution was enforced via the deserialization switch that defaults the proofs' generators to the same implementation (see here and here).

Verification

Resolved.

Issue E: [Verifier] Verifier Does Not Check the Slope Commitments Across Proofs

Location

`src/worldcoin_mpc/circuits.rs`

Synopsis

The circuit for verifying share generation does not ensure that the same slope m is used across all shares. As a result, each individual proof can be valid for different slopes `m1`, `m2`, and `m3` for the three different parties. The shares produced from these inconsistent slopes will not reconstruct the original secret, thereby breaking the secret sharing scheme. Consequently, an attacker can sign up multiple times.

Impact

Critical. An attacker who exploits this vulnerability can register multiple times by producing conflicting shares for the three parties with different slopes. This effectively allows the attacker to use an invalid secret for a sign-up that does not necessarily match any existing secret in the system, hence circumventing the uniqueness check.

Preconditions

None.

Feasibility

Straightforward. The verifier will not reject these differing slopes because there is neither a constraint nor a check tying them together.

Technical Details

In the secret sharing scheme, each party's share is generated using a linear function parameterized by the same slope `m`, the secret `s`, and an evaluation point for each party. The prover is generating the slope `m` non-deterministically. The verification currently checks the correctness of each individual share but does not check that the same slope is used for all shares.

Consequently, a malicious prover can produce share `s1` using slope `m1` and share `s2` using slope `m2` with `m2≠m1`. Each share passes individual verification because, locally, it is derived consistently from its own slope. However, since `m1≠m2`, the shares `s1` and `s2` will not combine to yield the intended original secret, yielding instead a different value as the secret. With this approach, the attacker can produce different secrets by varying the slope and can therefore attempt to sign up several times with the same iris image.

Remediation

We recommend updating the verifier logic to make it check that the commitments for the slope in each of the three proofs are consistent.

Status

The Worldcoin team has added checks to verify the slope commitments in each verifier instance (here, here, and here).

Verification

Resolved.

Issue F: [Verifier] Verifier Does Not Check the Iris Code and Mask Commitment Between Proofs for the V3 Circuit and MPC Circuit

Location

`src/bin/worldcoin_mpc.rs`

Synopsis

There is currently no verifier-side check that ensures the commitment of the iris code and mask in the V3 circuit is the same commitment used in the corresponding MPC circuit. While the prover correctly carries this commitment between circuits, the verifier never enforces that they match. As a result, the system cannot detect if inconsistent or malicious commitments are submitted for the iris code and mask in the MPC circuit.

Impact

Critical. An attacker could provide a valid proof for the V3 circuit while presenting a different iris code or mask in the MPC circuit, without being detected. This would result in the bypassing of the uniqueness check and, hence, allow an attacker to sign up multiple times.

Preconditions

None.

Feasibility

Straightforward.

Technical Details

The V3 circuit and the MPC circuit each take as input the commitments to the iris code and mask. In the current design, the prover is using the same commitments across both proofs. However, the verifier side does not assert that these commitments match.

An attacker can therefore produce a correct proof for the first circuit based on their own iris code. For the second proof, they could use a random iris code. The verifier accepts both proofs. The attacker can repeat this process multiple times and thus circumvent the actual goal of the second circuit (that is, sharing the iris code with the Worldcoin server), which then enforces the uniqueness check.

Remediation

We recommend updating the verifier logic to explicitly check that the commitments used in the MPC circuit match those generated in the V3 circuit.

Status

The Worldcoin team has added the following checks to verify the iris and mask commitments in each verifier instance:

- verifier 0 iris code commitment check
- verifier 0 mask code commitment check
- verifier 1 iris code commitment check
- verifier 1 mask code commitment check
- verifier 2 iris code commitment check
- verifier 2 mask code commitment check

Verification

Resolved.

Issue G: [Verifier] Verifier Does Not Truncate the Shares

Location

`src/bin/worldcoin_mpc.rs`

Synopsis

The verifier does not truncate the padded data to the shares.

Impact

Low. If the verifier does not truncate the received share properly, an attacker could add arbitrary data. This would lead to undefined behavior. In particular, it could be used to sign up multiple times if the server is using the untruncated share for the uniqueness check.

Preconditions

The server would have to use the untruncated share for the uniqueness check. This scenario seems unlikely; however, our team could not verify it since the relevant code segment was out of scope for this review.

Feasibility

This exploit is straightforward if the preconditions are met.

Technical Details

The iris code is of length `3200` field elements and is padded with zeros to match the next power of two, `4096`. The padding is not enforced within the circuits. In principle, this is not an issue since the relevant values (that is, the first `3200` elements) are not mixed with the padded values. However, the verifier should truncate the received secret share to the length of `3200`, discarding the padded values to avoid any attacks from the prover. In addition, although there is no known attack, it is recommended to also check that the discarded values are actually zero.

Remediation

We recommend checking that the unpadded lengths of the secret shares align with expectations, and truncating them to the expected lengths.

Status

The Worldcoin team has introduced truncation and length checks for the secret shares.

Verification

Resolved.

Issue H: [Prover] Leakage Through Residual Data

Location

`src/bin/worldcoin_mpc.rs#L247-L248`

`src/bin/worldcoin_mpc.rs#L233-L234`

`src/worldcoin_mpc/data.rs#L132-L133`

`src/worldcoin_mpc/data.rs#L104-L105`

Synopsis

Sensitive data could persist after the end of the application's execution. An attacker could be able to access them, leading to their disclosure.

Impact

High. This issue could potentially result in the full disclosure of the iris codes or the iris image data.

Preconditions

The attacker would have to be logically local to the target device.

Feasibility

The feasibility depends on the application-level design decisions and threat model. Due to this reason, our team did not have enough information to accurately assess feasibility.

Technical Details

Residual data can leak information about sensitive values, such as the iris image, the slope values, the secret shares, and the iris codes and masks. There are multiple ways for attackers to obtain residual data, such as by allocating memory recently freed by the target application, causing core dumps, or via swap memory.

Remediation
We recommend utilizing memory zeroization at the end of the lifecycle of sensitive values in order to reduce the leakage surface.

Status
The Worldcoin team has introduced memory zeroization for all sensitive values.

Verification
Resolved.

Issue I: [Prover] Slope Randomness Is Not Adequate

Location
`src/bin/worldcoin_mpc.rs#L247`

Synopsis
The slope values are not sampled from a randomness source; instead, a random value is reused. Not having fresh randomness could lead to biases.

Impact
This issue could allow statistical attacks to recover slope values, which would result in the partial recovery of the iris code.

Preconditions
None.

Feasibility
The attacker would need to analyze and construct statistical attacks that target the introduced correlation.

Technical Details
The security of secret sharing relies on the assumption that slopes are independently random. Reusing random values that are used elsewhere can interfere with independence, as it can introduce unintended correlation. This can be utilized by statistical attacks to gain additional information that was not intended to be disclosed.

In its current setting, the prover code is taking the commitment and MLE from the iris image and assigning them to the slope commitment and MLE. A code comment includes a TODO noting that this should be replaced.

Remediation
We recommend using fresh, independent random values for the slopes.

Status
The Worldcoin team has introduced a cryptographically secure pseudorandom number generator, seeded by system randomness. Their solution addresses the need for secure randomness for the slopes.

Verification

Resolved.

Suggestions

Suggestion 1: [All Components] Improve Documentation and Code Comments

Location

`Documentation`

Synopsis

During our review, our team identified some minor errors in the documentation and code comments. Below, we list all the instances we found:

- There is a lack of consistency regarding the usage of the constant `MPC_NUM_IRIS_4_CHUNKS` in the code and within the documentation. The code uses `4096`, while the documentation specifies `3200`.
- In the documentation, the quotients are labeled as private inputs, while the code takes them as public inputs.
- We found an incorrect code comment here on the verifier check for the auxiliary data.
- The documentation (MPC Circuit Builders (ext)) defines the domain for slopes as the set of integers, while the correct domain is the Galois ring.

Mitigation

We recommend addressing the items listed above.

Status

The Worldcoin team has enhanced both the quantity and clarity of code comments.

Verification

Resolved.

Suggestion 2: [Prover] Avoid Multiple Secret Sharing Executions of an Iris With Different Randomness

Location

`src/bin/worldcoin_mpc.rs#L247`

Synopsis

Resharing the same secrets using different randomness does not present a significant risk by itself since this is equivalent to a one-time pad. However, allowing it can amplify other attacks. For example, an attacker able to force a reshare, by, for example, reporting an error, could combine this capability with the concerns reported in Suggestion 4. This can provide the attacker with additional samples, increasing the feasibility and accuracy of statistical attacks.

Mitigation

During integration, we recommend ensuring that shares are constructed once for all control flow cases. This could be achieved by securely storing the shares the first time they are generated and reusing them for any subsequent secret sharing invocations.

Alternatively, we recommend examining if the randomness for slopes can be safely generated in a replicable manner per iris/user.

Status

The Worldcoin team stated that addressing this suggestion requires implementing changes that are beyond the scope of this project and that an ongoing design discussion is underway. As a result, the suggestion remains unresolved at the time of verification.

Verification

Unresolved.

Suggestion 3: [Verifier] Implement the Verifier Independently From the Prover

Location

`src/bin/worldcoin_mpc.rs`

Synopsis

In the `test` function in scope, the prover and verifier are closely intertwined. Three CLI commands (`Circuit`, `Prove`, and `Verify`) are implemented. This allows one user to test the functionality. However, the verifier and prover are not properly separated. This leads to several issues (Issues C-G) because the verifier is trusting the prover on data beyond the core proof struct. These issues could have been avoided or detected more easily if the verifier were implemented separately from the prover.

Mitigation

We recommend decoupling the prover and verifier. In addition, we recommend updating the verifier logic so that it generates its own circuit.

We also recommend adding a check in the verifier function to verify the correctness of the circuit structure. Currently, the verifier is loading the circuit structure locally, as generated by the user via the CLI command `Circuit`. It is not clear if the verifier is generating the circuit structure on its own or if it also trusts another entity, such as the prover, to provide the correct circuit. If the verifier is trusting the prover to provide the correct circuit, a further check by the verifier on the correctness is necessary.

Status

The Worldcoin team has decoupled the implementation of the proving and verifying logic. The project currently produces different binaries for each functionality. Furthermore, the verifiers are configured to read the circuit from a locally generated file that is independently generated and validated. The circuit generation logic is also decoupled in a standalone binary.

Verification

Resolved.

Suggestion 4: [All Components] Reduce Susceptibility to Statistical Attacks

Location

src/worldcoin_mpc/parameters.rs#L49-L52

src/worldcoin_mpc/parameters.rs#L99

Synopsis

The combination of a small domain for secret values, the choice of evaluation points, and the choice of the encoding matrix introduces risks for statistical attacks.

This analysis is not conclusive. A detailed analysis that includes enumerating different scenarios and calculating the success probabilities is required to conclude on root causes and effective countermeasures. Below, we outline some proactive measures that could help reduce the efficacy of such attacks.

One issue is that the chosen evaluation points for all servers have zeros on their third and fourth elements, which results in the elimination of several factors in the multiplication equations between the slope and the evaluation points. Some of the resulting equations yield a single slope value as a mask. Since the number of possible values of one slope is `65536`, it is practical for the attacker to run statistical attacks for each potential value.

Another issue is the small domains for the secret elements. The iris codes have the domain `{-1,0,1}`. Each element of the vector that is the result of the multiplication of iris codes with the encoding matrix has a very small amount of possible values related to the ring size. This results in highly specific distributions for each element. The elements of the secret are derived by adding uniformly random slope values to the elements of the multiplication result. A malicious MPC server could leverage the knowledge of the specific distributions within its secret share to identify the slope values and thus determine iris code values. One approach to mitigating this is to investigate mappings from the small to the bigger domain that aid in statistical indistinguishability. Another strategy is to increase the diffusion factor by involving more iris code values in the calculation of each secret element (e.g., encoding matrix without zeros, secrets depending on more than four iris codes, etc.).

A third issue is the choice of the encoding matrix.   The matrix is bijective, which maintains the uniform distribution of its multiplicand vector; however, this is not true element-wise. The usage of even and multiples of four as multipliers produces clustering and alters their distribution. This fact can be leveraged as a distinguishing factor for a statistical attack. Finding a set of values that follow the altered distribution (given a guess for the slope value) could help determine the actual slope value. As an illustration, the computation of the second element of a secret from the iris codes is:

$$s_1 = c_1 + 25194c_2 + 51956c_3$$

which is biased. For the server addressed by index zero, the corresponding element of the share is

$$sh_{0,1} = m_1 + s_1$$

where m_1 is uniformly distributed.

The distribution of s_1 is known and applies to all `3200` shares per iris. For a guess m of the slope value, the distribution of $sh_{0,1}$ is the distribution of s_1 shifted by m. Within this sample, an attacker can identify the most promising subset that meets the shifted distribution, and for that, it can assume $m_1 = m$.

The construction of the encoding matrix was out of scope and, thus, we are uncertain whether the matrix can be changed; however, if feasible, we recommend investigating alternatives.

Mitigation

We recommend carefully choosing evaluation points such that none of their elements are always zero across all points. We expect that this will increase the diffusion factor of the multiplication calculation, which will increase the complexity of the statistical analysis.

Additionally, we recommend investigating ways to increase the input domain size to aid in approaching the uniform distribution in the calculations that involve iris codes.

We also recommend, if possible, investigating alternative encoding matrices that do not include even numbers to avoid distribution clustering.

Another recommendation is to increase the diffusion factor in the calculation of the secret values from the iris codes. This requires increasing the number of iris codes involved in the calculation of each secret element. It could be achieved by a combination of changes. Some preliminary ideas include increasing the iris chunk size, calculating over overlapping chunks to introduce neighborhood noise, and decreasing the zeros in the encoding matrix.

As a general comment, increasing the size of the Galois ring can also be beneficial.

Currently, our team cannot provide concrete recommendations, as we were unable to conduct a comprehensive statistical analysis. The above recommendations are shared only as precautionary measures. Thus, our final recommendation is to conduct a thorough statistical analysis of the design to uncover root causes, accurately determine the efficacy of such attacks, and benchmark the effectiveness of different countermeasures.

Status

The Worldcoin team has not addressed this suggestion due to prior design decisions and performance considerations. As a result, it remains unresolved at the time of verification.

Verification

Unresolved.

Appendix

Appendix A: Rationale Behind Statistical Attacks

The calculation of the multiplication of the slopes with the evaluation points is:

$$sc_{j,0} = \alpha_{j,0}m_0^{(i)} + \alpha_{j,1}m_3^{(i)} + \alpha_{j,2}m_2^{(i)} + \alpha_{j,3}m_1^{(i)}$$

$$sc_{j,1} = \alpha_{j,0}m_1^{(i)} + \alpha_{j,1}m_0^{(i)} + \alpha_{j,1}m_3^{(i)} + \alpha_{j,2}m_2^{(i)} + \alpha_{j,3}m_1^{(i)} + \alpha_{j,2}m_3^{(i)} + \alpha_{j,3}m_2^{(i)}$$

$$sc_{j,2} = \alpha_{j,0}m_2^{(i)} + \alpha_{j,1}m_1^{(i)} + \alpha_{j,2}m_0^{(i)} + \alpha_{j,2}m_3^{(i)} + \alpha_{j,3}m_2^{(i)} + \alpha_{j,3}m_3^{(i)}$$

$$sc_{j,3} = \alpha_{j,0}m_3^{(i)} + \alpha_{j,1}m_2^{(i)} + \alpha_{j,2}m_1^{(i)} + \alpha_{j,3}m_0^{(i)} + \alpha_{j,3}m_3^{(i)}$$

with $\alpha_{j,k}$ the elements of the evaluation point j and $m_l^{(i)}$ the elements of the slope values. The chosen evaluation points are $a_0 = [1, 0, 0, 0]$ for Server 0, $a_1 = [0, 1, 0, 0]$ for Server 1 and $a_2 = [1, 1, 0, 0]$ for Server 2.

Applying the chosen evaluation points to the above calculation, and choosing only one sharing (a fixed value for i) for simplicity, results in:

Server 0 share: $sh_0 = \left[m_0 + s_0, m_1 + s_1, m_2 + s_2, m_3 + s_3\right]$

Server 1 share: $sh_1 = \left[m_3 + s_0, m_0 + m_3 + s_1, m_1 + s_2, m_1 + s_3\right]$

Server 2 share: $sh_2 = \left[m_0 + m_3 + s_0, m_1 + m_0 + m_3 + s_1, m_2 + m_1 + s_2, m_3 + m_1 + s_3\right]$

We observe that (with very high probability):

$sh_1[0] = sh_2[0] \Leftrightarrow m_0 = 0$

$(sh_0[2] = sh_2[2] \ \vee \ sh_0[3] = sh_2[3]) \ \Leftrightarrow m_1 = 0$

$sh_2[2] = sh_3[2] \Leftrightarrow m_2 = 0$

$sh_2[3] = sh_3[3] \Leftrightarrow m_3 = 0$

Thus, if one can determine the truth of the above equalities between shares, one can deduce that the corresponding slope is zero. The elements of a secret are related to the iris code values through the encoding calculation. Its expanded form is:

$s_0 = c_0 + 58082c_1 + 60579c_2 + 17325c_3$

$s_1 = c_1 + 25194c_2 + 51956c_3$

$s_2 = c_2 + 57011c_3$

$s_3 = c_3$

The probability of n slopes having the zero value is $1/65536^{n}$. We only account for n=1 since any other case yields a very low probability.

Combining all the above, we conclude that:

- Server 0 can determine the value c_3 if it can deduce that $sh_2[3] = sh_3[3]$
- Server 1 can determine the values of c_2, c_3 if it can deduce one of: $sh_0[2] = sh_2[2]$ or $sh_0[3] = sh_2[3]$
- Server 2 needs knowledge of multiple slope values being zero to determine an iris code.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, zero-knowledge protocols, and consensus protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.