# Symbiotic KeyRegistry Audit Report

Prepared by Cyfrin

Version 2.0

**Lead Auditors**

Farouk

0kage

December 3, 2025

## Contents

## 1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

## 2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

## 3 Risk Classification

| | **Impact: High** | **Impact: Medium** | **Impact: Low** |
|---|---|---|---|
| **Likelihood: High** | Critical | High | Medium |
| **Likelihood: Medium** | High | Medium | Low |
| **Likelihood: Low** | Medium | Low | Low |

## 4 Protocol Summary

The audit focused on the integration of BLS12-381 signature support into the KeyRegistry. The scope covered the new contracts, logic paths, and verification routines introduced to enable BLS12-381 key registration, signature validation, and compatibility with the existing validator-set workflow. The review examined how the registry accepts, stores, and verifies BLS12-381 keys, with attention to domain separation, point validation, pairing-based checks, and consistency with the existing BN254 and ECDSA implementations already present in the system.

The purpose of this upgrade is to extend the KeyRegistry so it can support an additional signature scheme without altering the broader trust model or consensus flow.

## 5 Audit Scope

Cyfrin conducted an audit of the KeyRegistry upgrade for Symbiotic based on the code present in the repository commit hash 4bfd3b4. The following files construct the scope of the audit:

- src/libraries/keys/KeyBlsBls12381.sol
- src/libraries/sigs/SigBlsBls12381.sol
- src/modules/key-registry/KeyRegistry.sol
- src/modules/valset-driver/ValSetDriver.sol

## 6 Executive Summary

Over the course of 3 days, the Cyfrin team conducted an audit on the Symbiotic KeyRegistry smart contracts provided by Symbiotic. In this period, a total of 1 issues were found.

This audit reviewed the new BLS12-381 support added to the KeyRegistry. The upgrade is narrow by design and introduces no significant changes to the surrounding protocol logic.

The codebase was found to be well-structured, and the integration follows the expected verification patterns. One informational issue was identified regarding missing explicit zero-value checks on the G2 public key and G1 signature. While the pairing precompile already prevents malformed points from producing valid results, adding these checks improves clarity, aligns with standard BLS design expectations, and avoids future ambiguity for integrators.

No security-impacting issues were found in the scope of this audit.

**Summary**

| | |
|---|---|
| Project Name | Symbiotic KeyRegistry |
| Repository | relay-contracts |
| Commit | 4bfd3b425b35... |
| Audit Timeline | Nov 26st - Nov 28th |
| Methods | Manual Review |

**Issues Found**

| | |
|---|---|
| Critical Risk | 0 |
| High Risk | 0 |
| Medium Risk | 0 |
| Low Risk | 0 |
| Informational | 1 |
| Gas Optimizations | 0 |
| Total Issues | 1 |

**Summary of Findings**

| | |
|---|---|
| [I-1] Missing Explicit Zero-Value Checks for G2 Public Key and G1 Signature | Acknowledged |

# 7 Findings

## 7.1 Informational

### 7.1.1 Missing Explicit Zero-Value Checks for G2 Public Key and G1 Signature

**Description:** The BLS12381 verification logic explicitly rejects a zero G1 public key, but does not explicitly reject

1) a zero G2 public key, or
2) a zero G1 signature.

```
/**
 * @notice Verify a BLS signature.
 * @param keyG1 The G1 public key.
 * @param messageHash The message hash to verify.
 * @param signatureG1 The G1 signature.
 * @param keyG2 The G2 public key.
 * @return If the signature is valid.
 * @dev Burns the whole gas if pairing precompile fails.
 *      Returns false if the key is zero G1 point.
 */
function verify(
    BLS12381.G1Point memory keyG1,
    bytes32 messageHash,
    BLS12381.G1Point memory signatureG1,
    BLS12381.G2Point memory keyG2
) internal view returns (bool) {
    if (keyG1.x_a == 0 && keyG1.x_b == 0 && keyG1.y_a == 0 && keyG1.y_b == 0) {
        return false;
    }
    BLS12381.G1Point memory messageG1 = BLS12381.hashToG1(abi.encodePacked(messageHash));
    uint256 alpha = uint256(keccak256(abi.encode(signatureG1, keyG1, keyG2, messageG1))) %
    ↪  BLS12381.FR_MODULUS;

    return BLS12381.pairing(
        signatureG1.add(keyG1.scalarMul(alpha)),
        BLS12381.negGeneratorG2(),
        messageG1.add(BLS12381.generatorG1().scalarMul(alpha)),
        keyG2
    );
}
```

While these cases are not exploitable in practice due to the behavior of the BLS12-381 precompiles (which prevent malformed curve points from passing pairing checks), zero-valued keys and signatures are not valid inputs in the BLS specification and should be rejected for clarity and consistency.

Allowing zero G2 keys introduces a degenerate verification path where the second pairing term collapses to the identity. Zero signatures are similarly invalid from a protocol-design perspective and should be excluded explicitly to avoid ambiguity for integrators and future code maintainers.

**Recommended Mitigation:** Add explicit zero checks before invoking any curve or pairing operations, notably the G2 public key and G1 signature.

**Symbiotic:** Acknowledged.