CODESPECT

Solana Merkle Airdrop

SECURITY ASSESSMENT REPORT

16 April, 2025

Prepared for

TOKENTABLE

Contents

1 About CODESPECT

CODESPECT is a specialized smart contract security firm dedicated to ensure the safety, reliability, and success of blockchain projects. Our services include comprehensive smart contract audits, secure design and architecture consultancy, and smart contract development across leading blockchain platforms such as Ethereum (Solidity), Starknet (Cairo), and Solana (Rust).

At CODESPECT, we are committed to build secure, resilient blockchain infrastructures. We provide strategic guidance and technical expertise, working closely with our partners from concept development through deployment. Our team consists of blockchain security experts and seasoned engineers who apply the latest auditing and security methodologies to help prevent exploits and vulnerabilities in your smart contracts.

Smart Contract Auditing: Security is at the core of everything we do at CODESPECT. Our auditors conduct thorough security assessments of smart contracts written in Solidity, Cairo, and Rust, ensuring that they function as intended without vulnerabilities. We specialize in providing tailored security solutions for projects on EVM-compatible chains and Starknet. Our audit process is highly collaborative, keeping clients involved every step of the way to ensure transparency and security. Our team is also dedicated to cutting-edge research, ensuring that we stay ahead of emerging threats.

Secure Design & Architecture Consultancy: At CODESPECT, we believe that secure development begins at the design phase. Our consultancy services offer deep insights into secure smart contract architecture and blockchain system design, helping you build robust, secure, and scalable decentralized applications. Whether you're working with Ethereum, Starknet, or other blockchain platforms, our team helps you navigate the complexity of blockchain development with confidence.

Tailored Cybersecurity Solutions: CODESPECT offers specialized cybersecurity solutions designed to minimize risks associated with traditional attack vectors, such as phishing, social engineering, and Web2 vulnerabilities. Our solutions are crafted to address the unique security needs of blockchain-based applications, reducing exposure to attacks and ensuring that all aspects of the system are fortified.

With a focus on the intersection of security and innovation, CODESPECT strives to be a trusted partner for blockchain projects at every stage of development and for each aspect of security.

2 Disclaimer

Limitations of this Audit: This report is based solely on the materials and documentation provided to CODESPECT for the specific purpose of conducting the security review outlined in the Summary of Audit and Files. The findings presented in this report may not be comprehensive and may not identify all possible vulnerabilities. CODESPECT provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, is entirely at your own risk.

Inherent Risks of Blockchain Technology: Blockchain technology is still evolving and is inherently subject to unknown risks and vulnerabilities. This review focuses exclusively on the smart contract code provided and does not cover the compiler layer, underlying programming language elements beyond the reviewed code, or any other potential security risks that may exist outside of the code itself.

Purpose and Reliance of this Report: This report should not be viewed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. Third parties should not rely on this report for any purpose, including making decisions related to investments or purchases.

Liability Disclaimer: To the maximum extent permitted by law, CODESPECT disclaims all liability for the contents of this report and any related services or products that arise from your use of it. This includes but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services: CODESPECT does not warrant, endorse, or assume responsibility for any third-party products or services mentioned in this report, including any open-source or third-party software, code, libraries, materials, or information that may be linked to, referenced by, or accessible through this report. CODESPECT is not responsible for monitoring any transactions between you and third-party providers. We strongly recommend conducting thorough due diligence and exercising caution when engaging with third-party products or services, just as you would for any other product or service transaction.

Further Recommendations: We advise clients to schedule a re-audit after any significant changes to the codebase to ensure ongoing security and reduce the risk of newly introduced vulnerabilities. Additionally, we recommend implementing a bug bounty program to incentivize external developers and security researchers to identify and disclose potential vulnerabilities safely and responsibly.

Disclaimer of Advice: FOR AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, AND ANY ASSOCIATED SERVICES OR MATERIALS SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Risk Classification

Severity Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Table 1: Risk Classification Matrix based on Likelihood and Impact

3.1 Impact

- **High** - Results in a substantial loss of assets (more than 10%) within the protocol or causes significant disruption to the majority of users.
- **Medium** - Losses affect less than 10% globally or impact only a portion of users, but are still considered unacceptable.
- **Low** - Losses may be inconvenient but are manageable, typically involving issues like griefing attacks that can be easily resolved or minor inefficiencies such as gas costs.

3.2 Likelihood

- **High** - Very likely to occur, either easy to exploit or difficult but highly incentivized.
- **Medium** - Likely only under certain conditions or moderately incentivized.
- **Low** - Unlikely unless specific conditions are met, or there is little-to-no incentive for exploitation.

3.3 Action Required for Severity Levels

- **Critical** - Must be addressed immediately if already deployed.
- **High** - Must be resolved before deployment (or urgently if already deployed).
- **Medium** - It is recommended to fix.
- **Low** - Can be fixed if desired but is not crucial.

In addition to High, Medium, and Low severity levels, CODESPECT utilizes two other categories for findings: **Informational** and **Best Practices**.

a) **Informational** findings do not pose a direct security risk but provide useful information the audit team wants to communicate formally.

b) **Best Practices** findings indicate that certain portions of the code deviate from established smart contract development standards.

4 Executive Summary

This document presents the security assessment conducted by CODESPECT for the Merkle-token-distributor Solana programs of TokenTable. Merkle-token-distributor is part of a larger suite of protocols designed to streamline token ownership registration and distribution.

This audit focuses on the Merkle-token-distributor Solana program, which allows project teams to conduct large-scale token airdrops. The solution offers unique advantages, such as handling massive token distributions, decentralization, and ensuring security through wallet address verification using Merkle proofs.

The audit was performed using:

a) Manual analysis of the codebase.

b) Dynamic analysis of programs, execution testing.

CODESPECT found 6 points of attention, one classified as `Low` and five classified as `Informational`. All of the issues are summarised in Table 2.

Organization of the document is as follows:

- **Section 5** summarizes the audit.
- **Section 6** describes the system overview.
- **Section 7** presents the issues.
- **Section 8** discusses the documentation provided by the client for this audit.
- **Section 9** presents the compilation and tests.

Issues found:

Severity	Unresolved	Fixed	Acknowledged
Low	0	1	0
Informational	0	5	0
Total	**0**	**6**	**0**

Table 2: Summary of Unresolved, Fixed, and Acknowledged Issues

5 Audit Summary

Audit Type	Security Review
Project Name	TokenTable
Type of Project	Merkle Airdrop Program
Duration of Engagement	5 Days
Duration of Fix Review Phase	2 Days
Draft Report	April 10, 2025
Final Report	April 16, 2025
Repository	tokentable-unlocker-solana
Commit (Audit)	67a39faff7b848ae05c5e3ab45e36b60efcc622e
Commit (Final)	8edb2ab7e2a63c37258b78f365bce2d43db3403f
Documentation Assessment	Medium
Test Suite Assessment	High
Auditors	JecikPo, shaflow01

Table 3: Summary of the Audit

5.1 Scope - Audited Files

	File	**LoC**
0	merkle-token-distributor-solana/src/traits/mod.rs	0
1	merkle-token-distributor-solana/src/state/merkle_airdrop.rs	22
2	merkle-token-distributor-solana/src/state/tokentable_merkle_distributor_data.rs	7
3	merkle-token-distributor-solana/src/state/mod.rs	6
4	merkle-token-distributor-solana/src/state/config.rs	7
5	merkle-token-distributor-solana/src/instructions/deploy.rs	21
6	merkle-token-distributor-solana/src/instructions/withdraw.rs	49
7	merkle-token-distributor-solana/src/instructions/transfer_program_admin.rs	26
8	merkle-token-distributor-solana/src/instructions/encode_leaf.rs	14
9	merkle-token-distributor-solana/src/instructions/transfer_ownership.rs	20
10	merkle-token-distributor-solana/src/instructions/toggle_pause.rs	16
11	merkle-token-distributor-solana/src/instructions/version.rs	20
12	merkle-token-distributor-solana/src/instructions/utils.rs	186
13	merkle-token-distributor-solana/src/instructions/receive_program_admin.rs	24
14	merkle-token-distributor-solana/src/instructions/set_claim_delegate.rs	23
15	merkle-token-distributor-solana/src/instructions/mod.rs	32
16	merkle-token-distributor-solana/src/instructions/initialize.rs	60
17	merkle-token-distributor-solana/src/instructions/set_fee_collector.rs	50
18	merkle-token-distributor-solana/src/instructions/set_fee_token.rs	22
19	merkle-token-distributor-solana/src/instructions/claim.rs	120
20	merkle-token-distributor-solana/src/instructions/set_base_params.rs	36
21	merkle-token-distributor-solana/src/instructions/deposit.rs	53
22	merkle-token-distributor-solana/src/errors.rs	32
23	merkle-token-distributor-solana/src/lib.rs	107
24	merkle-token-distributor-solana/src/models/leaf.rs	14
25	merkle-token-distributor-solana/src/models/mod.rs	2
26	merkle-token-distributor-solana/src/event.rs	18
	Total	**987**

5.2 Findings Overview

	Finding	Severity	Update
1	Fee configuration conflict	Low	Fixed
2	Allow the `fee_collector` to be set arbitrarily during the initialization of the airdrop account	Info	Fixed
3	Changing the `fee_collector` to a different program will cause instructions to fail	Info	Fixed
4	Lack of `Option` wrapper on `fee` account	Info	Fixed
5	Miscalculated `MerkleAirdrop` size	Info	Fixed
6	Redundant code	Info	Fixed

5.2.1 Findings Raised During Fix Review Phase

	Finding	Severity	Update
1	The `set_default_fee_collector` instruction cannot be executed	Medium	Fixed
2	Redundant check	Info	Fixed

6 System Overview

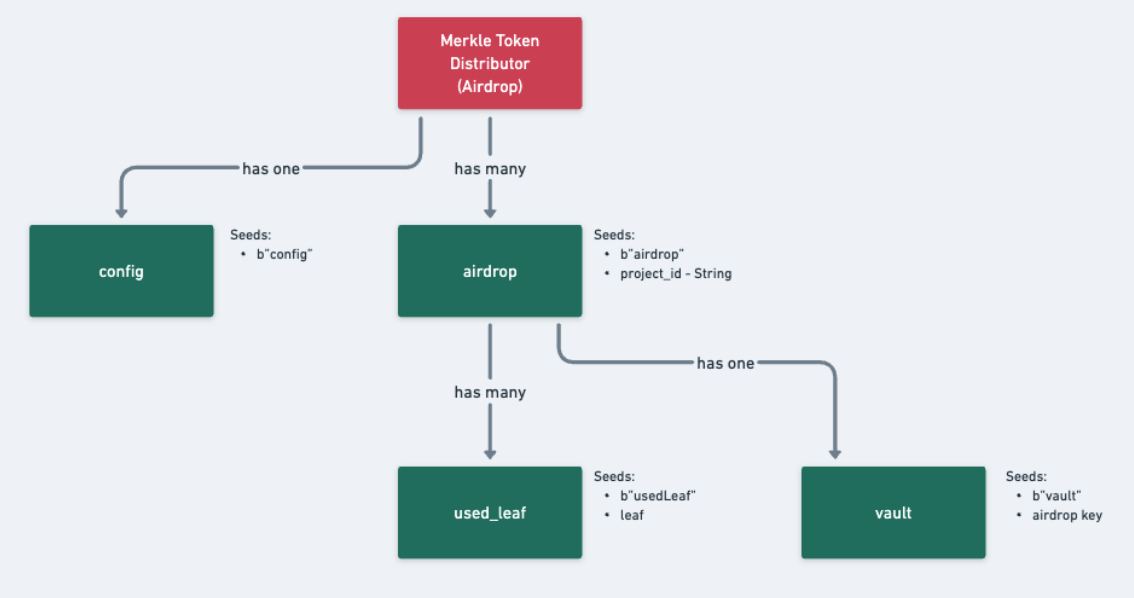
TokenTable has developed the Merkle-token-distributor program, which works in conjunction with the fee-collector program to form an independent on-chain system. This system enables projects to conduct large-scale token airdrops and allows users to securely claim airdrop tokens by providing Merkle proofs.

6.1 Merkle-token-distributor

In the Merkle-token-distributor program, the owner of the protocol creates an `airdrop` account a.k.a. Project (the protocol is not permissionless) that serves as the foundation for the project's airdrop activity. The following points describe a high-level view of a Project:

1. The project owner creates a Project (represented by a unique `airdrop` account). Permission is granted to the owner address of that Project.
2. The project owner sets the basic params for the airdrop, such as the start and end times, as well as the Merkle root containing the airdrop recipient information.
3. Users in the Merkle tree can claim the airdrop by providing a Merkle proof after the claimable time is reached.

The following picture presents the account structure of the Merkle-token-distributor program:



The section below outlines the program's instructions, categorized by the entities authorized to call them.

Merkle-token-distributor's instructions executable by the protocol owner:

1. `deploy` – sets the protocol admin account which is held within the `config`. Can be executed only once.
2. `set_fee_collector` – sets the `fee_collector` for a given Project.
3. `set_fee_token` – sets the `fee_token` for a given Project.
4. `transfer_program_admin` – transfers the protocol's ownership to a different account.

Merkle-token-distributor's instructions executable by the new Admin:

1. `receive_program_admin` - Allow the new Admin to accept the ownership of the protocol in the two-step ownership transfer.

Merkle-token-distributor's instructions executable by the Project owner:

1. `initialize` - creates an `airdrop` account to hold the individual airdrop information. Multiple airdrop accounts can be created using different project_ids.
2. `set_base_params` - Modify the basic airdrop parameters in the `airdrop` account, including the Merkle root, airdrop start and end times, and url.
3. `set_claiming_delegate` - A Project owner can delegate the claiming capability through this instruction. A delegatee can call the claim instruction to airdrop the token to the recipient and pay the fees on his behalf. A Project owner can also revoke the delegatee status.
4. `toggle_pause` - Allows the Owner of the Project to pause and unpause the airdrop claims.
5. `transfer_ownership` – Owner transfers the `airdrop` ownership to a new account.
6. `withdraw` – Allows the Owner of the Project to withdraw deposited airdrop tokens.

Merkle-token-distributor's instructions executable by a Airdrop Recipient or Delegatee:

1. `claim` - claims the airdrop tokens by providing the Merkle proof.

Merkle-token-distributor's instructions executable by anyone:

1. `deposit` – Deposits a specified amount of tokens into the Project.
2. `version` – Used to update the program version information.
3. `encode_leaf` – Precomputes the leaf hash based on the input data.

7 Issues

7.1 [Low] Fee configuration conflict

File(s): merkle-token-distributor-solana::initialize.rs, unlocker-v2-solana::initialize.rs

Description: If the airdrop account in the merkle-token-distributor-solana program and the unlocker account in the unlocker-v2-solana program share the same project_id, they will use the same fee configuration. Since both accounts can be created permissionlessly, there is no guarantee that they are created and controlled by the same owner. This may lead to a fee configuration conflict.

Impact: If unlock and airdrop accounts with the same project_id are controlled by different owners and each wants to use a different fee_token, there may be some complications in fee configuration.

Recommendation(s): It is recommended to distinguish between the fee configuration accounts for the two accounts.

Status: Fixed

Update from TokenTable:

Add distributor pubkey field as additional seeds parameter when deriving a fee account in 40ebaffac8ecbf186e0625568fb10de967340d6c.

7.2 [Info] Allow the fee_collector to be set arbitrarily during the initialization of the airdrop account

File(s): initialize.rs

Description: In the initialize instruction, if init_fee_account is set to false, then the check ctx.accounts.fee_collector.as_ref().unwrap().key() == fee_collector.key() is skipped. This means that it allows the airdrop.owner to initialize any fee_collector.

```
pub fn initialize(...) -> Result<()> {
  ctx.accounts.airdrop.owner = owner;
  ctx.accounts.airdrop.fee_collector = fee_collector;
  ctx.accounts.airdrop.project_token = project_token;

  if init_fee_account {
    // Before we init the fee account, ensure we are calling the expected fee_collector program from
    // the parameters and that all required accounts are provided.
    require!(
      ctx.accounts.fee_collector.is_some() &&
        ctx.accounts.fee.is_some() &&
        ctx.accounts.fee_collector_storage.is_some() &&
        ctx.accounts.fee_collector.as_ref().unwrap().key() == fee_collector.key(),
      TokenTableError::InvalidFeeCollector
    );
    //...
}
```

Impact: In the current system, allowing the fee_collector account to be set arbitrarily during initialization does not cause any loss, because the no-fee claim, as designed by the protocol, fails due to a constraint in the ctx. However, the airdrop.owner may have the motivation to initialize the fee_collector as pubkey::default during initialization. This would enable claims related to that account to be processed without any fees.

Recommendation(s): It is recommended not to allow the airdrop.owner to arbitrarily initialize the fee_collector.

Status: Fixed

Update from TokenTable:In aa48e8ab3c30b65f0e90a3be35cdb81a7f7f9461, fee_collector program account verification is handled manually. Anchor now expects an UncheckedAccount<>, and in all instructions where fee_collector can be set, we verify that the provided account matches the instruction parameter value and that the provided account is executable.

7.3 [Info] Changing the `fee_collector` to a different program will cause instructions to fail

File(s): `set_fee_collector.rs`

Description: The set_fee_collector instruction allows setting a different fee_collector program for Airdrop's fee processing capabilities, as per TokenTable's feedback from the previous audit:

The problem arises when the FeeCollector `program_id` is changed and certain instructions which take the `fee_collector` program account are called. The Anchor implementation under the hood will validate the program account against the `program_id` of the `FeeCollector` which was placed there at compile time:

```
pub fee_collector: Option<Program<'info, FeeCollector>>,
```

Impact: It will not be possible to update the `fee_collector` program account without also updating the entire merkle token distributor program.

Recommendation(s): Remove the `fee_collector` account from Anchor's context structs and handle it manually within the instruction code.

Status: Fixed

Update from TokenTable: In aa48e8ab3c30b65f0e90a3be35cdb81a7f7f9461, `fee_collector` program account verification is handled manually. Anchor now expects an `UncheckedAccount<>`, and in all instructions where `fee_collector` is used, we verify that the provided account matches the expected unlocker's/airdrop's `fee_collector` but skip the account executable check, since this would have already been checked when the account was set.

7.4 [Info] Lack of `Option` wrapper on `fee` account

File(s): `claim.rs`

Description: The `claim` instructions are invoked with a few accounts related to fee collection:

- `authority_fee_ata`;
- `fee_collector_storage`;
- `fee_collector_vault`;
- `fee_collector`;
- `fee_token_mint`;
- `fee`;
- `fee_token_program`;

The fee collection mechanism is optional, hence the design allows skipping them if they are unnecessary through the `Option` wrapper on the account type in the `instructions` contexts.

The `fee` account however is not:

```
/// CHECK: The account is checked in the FeeCollector, not here.
#[account(mut)]
pub fee: UncheckedAccount<'info>,
```

Impact: Expected difficulties in building the fee-less transactions as the `fee` account still needs to be provided to the instruction call.

Recommendation(s): Wrap the `fee` account type in `Option`.

Status: Fixed

Update from TokenTable: As of 78051afb53579a4e6558519000d6c35f510a5533, the fee collection mechanism is no longer optional. `fee` is a required account and the documented structure here is needed to support the updated fee collection mechanism.

7.5 [Info] Miscalculated MerkleAirdrop size

File(s): merkle_airdrop

Description: When creating the MerkleAirdrop account, use calculate_size to determine the allocated space.

```
pub fn calculate_size(uri_length: usize) -> usize {
  let mut size: usize = 0;
  size += 248; // Takes care of all non-vector items
  size += 4 + uri_length; // Add required data size of data vector

  size
}
```

The calculation seems to be implemented incorrectly as the total size of non-vector items should be 32 + 32 * 1 + 32 + 32 + 8 + 8 + 32 + 32 + 1 = 209 instead of 248.

Impact: This would result in unnecessary rent wastage.

Recommendation(s): Calculate account space using the correct size.

Status: Fixed

Update from TokenTable: Updated account size calculation from a base of 248 to 209 in 08d6356a40601e5e5b0cf8cb6dfac9102da23583.

7.6 [Info] Redundant code

File(s): utils.rs

Description: The protocol contains multiple pieces of redundant code.

1. Both if and require statements are used when checking the result of the merkle_verify call; however, a single require statement would suffice;

   ```
   pub fn _verify_and_claim<'info>(...) -> Result<u64> {
     //...
     if !merkle_verify(proof, root, leaf) {
       require!(false, TokenTableError::InvalidProof);
     }
   ```

Impact: Redundant code hinders readability and increases deployment costs.

Recommendation(s): It is recommended to optimize the redundant code.

Status: Fixed

Update from TokenTable:

merkle_verify() require statement simplified in 2025f68a4d699cc4997c133775f26f2768aba7e6.

Findings Raised During Fix Review Phase

[Medium] The `set_default_fee_collector` instruction cannot be executed

File(s): `set_fee_collector.rs`

Description: The `set_default_fee_collector` instruction is used to modify the `default_fee_collector`. Since it requires `config.admin` for permission validation, the `config` account should have already been initialized when calling the instruction. However, due to the incorrect assignment of the `init` attribute to the `config` account in the `ctx`, the `set_default_fee_collector` instruction fails to execute successfully.

```
#[derive(Accounts)]
#[instruction(_default_fee_collector: Pubkey)]
pub struct SetDefaultFeeCollector<'info> {
  #[account(
    init,
    seeds = [b"config".as_ref()],
    bump,
    payer = authority,
    space = 8 + Config::INIT_SPACE
  )]
  pub config: Account<'info, Config>,
  //...
}
```

Impact: The default_fee_collector cannot be successfully set

Recommendation(s): It is recommended to remove the init attribute from the config account in the ctx.

Status: Fixed

Update from TokenTable: Removed `init` attribute from the config account in the Anchor context in e2cf5fbc8802845c56d0e0ab48c874c0000ce015 and added `mut` attribute in 8edb2ab7e2a63c37258b78f365bce2d43db3403f.

[Info] Redundant check

File(s): **File(s)**: `claim.rs`,

Description: The `claim` instruction contain redundant checks for `fee_collector`. The `fee_collector` is checked in the `ctx` and then checked again in the execution logic.

```
  /// CHECK: Checked in the function call.
#[account(constraint = fee_collector.key() == airdrop.fee_collector.key())]
pub fee_collector: UncheckedAccount<'info>,

// ...
pub fn claim(...) -> Result<()> {
  // Fee collector
  require!(
    ctx.accounts.unlocker.fee_collector == ctx.accounts.fee_collector.key(),
    TokenTableError::InvalidFeeCollector
  );
```

Impact: Redundant checks increase the execution overhead of the transaction call.

Recommendation(s): It is recommended to remove the redundant checks.

Status: Fixed

Update from TokenTable: Redundant checks removed in 1aed8dad5fca73dd7e7b3d2a666c939a39a37be6.

8 Evaluation of Provided Documentation

The TokenTable team provided documentation in two forms:

- **Official Documentation Website:** The official documentation contains the protocol's design and implementation details, providing an overview of the protocol's purpose for both users and auditors. Unfortunately, the current state of the documentation website does not contain a version for Solana contracts.
- **Natspec Comments:** The code includes comments for key processes to help understand the logic. However, most functions lack comments, and expanding documentation coverage would enhance the overall comprehensibility of the code.

The documentation provided by TokenTable offered valuable insights into the protocol, significantly aiding CODESPECT's understanding. However, the public technical documentation could be further improved to better present the protocol's overall functionality and facilitate the understanding of each component.

Additionally, the TokenTable team was consistently available and responsive, promptly addressing all questions raised by CODESPECT during the evaluation process.

9 Test Suite Evaluation

9.1 Compilation Output

```
> anchor build
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling merkle-token-distributor-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/merkle-token-distributor-solana)
    Finished `release` profile [optimized] target(s) in 3.75s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling merkle-token-distributor-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/merkle-token-distributor-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 3.29s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
    Finished `release` profile [optimized] target(s) in 1.32s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.93s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling unlocker-v2-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/unlocker-v2-solana)
    Finished `release` profile [optimized] target(s) in 3.43s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling unlocker-v2-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/unlocker-v2-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 2.33s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling fungible-token-distributor-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fungible-token-distributor-solana)
    Finished `release` profile [optimized] target(s) in 2.79s
   Compiling fee-collector v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fee-collector)
   Compiling fungible-token-distributor-solana v0.1.0
    ↪  (/tmp/011-TokenTable-Solana-UnlockerV2-FollowUp-Merkle/code/unlocker/programs/fungible-token-distributor-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 1.71s
```

9.2 Tests Output

Merkle-token-distributor's test output:

```
merkle-token-distributor-solana
   Is deployed (501ms)
   Transfer program admin (1938ms)
   Is initialized! (500ms)
   Transfer ownership (1528ms)
   Initialize Merkle (encode_leaf verify no claim)
   Deposit SPL (501ms)
   Set Base Params (fail - bad time)
   Set Base Params (fail - not owner) (529ms)
   Set Base Params (succeed) (494ms)
   Set Fee Token (fail - not deployer) (517ms)
   Set Fee Token (succeed) (1025ms)
   Set Claim Delegate (fail - not owner) (530ms)
   Set Claim Delegate (succeed) (1564ms)
   Withdraw (fail - not owner) (534ms)
   Withdraw (succeed) (981ms)
   Claim (fail - not active) (547ms)
   Delegate Claim (fail - not active) (521ms)
   Delegate Claim (fail - not delegate) (1564ms)
   Encode Leaf + Verify + Claim (fail - outside claimable time range) (496ms)
   Encode Leaf + Verify + Claim (fail - bad proof) (506ms)
   Encode Leaf + Verify + Claim (succeed) (1544ms)
   Encode Leaf + Verify + Claim (fail - double-claim attempt)
   Encode Leaf + Verify + Claim (succeed, separate project ID) (3585ms)
   Encode Leaf + Verify + Delegate Claim (fail - outside claimable time range) (520ms)
   Encode Leaf + Verify + Delegate Claim (fail - bad proof) (499ms)
   Encode Leaf + Verify + Delegate Claim (succeed) (1012ms)
   Encode Leaf + Verify + Delegate Claim (fail - double-claim attempt)
   Encode Leaf + Verify + Delegate Claim (succeed, separate project ID) (3598ms)
   Fee Collector (lamports) (1520ms)
   Fee Collector (SPL) (4105ms)
```

9.3 Notes about Test suite

The TokenTable team delivered a comprehensive test suite, showcasing a well-structured approach to ensuring the protocol's correctness and resilience. Key observations of the test suite include:

- **Good functional coverage:** Despite protocol simplicity the test suite comprehensively covers all important functions of the protocol.
- **Good failure handling coverage:** Test suite design includes key failure scenarios - especially permissions testing and validation of the airdrop parameters behavior enforcement. This is a strong point from the protocol's security perspective.

Overall, the test suite reflects a mature development process and significantly enhances the reliability of the protocol.

CODESPECT also recommends explicitly adding tests involving both token programs Token and Token2022 to ensure flowless integration with both programs.