CODESPECT

Solana EDDSA Token Distributor

SECURITY ASSESSMENT REPORT

15 May, 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 EDDSA-token-distributor Solana programs of TokenTable. EDDSA-token-distributor is part of a larger suite of protocols designed to streamline token ownership registration and distribution.

This audit focuses on the EDDSA-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, decentralisation, and ensuring security through wallet address verification using Ed25519 and Secp256k1 signatures.

The audit was performed using:

a) Manual analysis of the codebase.

b) Dynamic analysis of programs, execution testing.

CODESPECT found four points of attention, one classified as `High` and three classified as `Informational`. All of the issues are summarised in Table 2.

Organisation 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
High	0	1	0
Informational	0	3	0
Total	**0**	**4**	**0**

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

5 Audit Summary

Audit Type	Security Review
Project Name	TokenTable
Type of Project	EDDSA Token Distributor
Duration of Engagement	5 Days
Duration of Fix Review Phase	1 Day
Draft Report	May 13, 2025
Final Report	May 15, 2025
Repository	tokentable-unlocker-solana
Commit (Audit)	87b79fe77b74d734fc5274da93300aa39444146c
Commit (Final)	54c916fe785c9249584836fed6bad594492c2a80
Documentation Assessment	Medium
Test Suite Assessment	High
Auditors	JecikPo, shaflow01

Table 3: Summary of the Audit

5.1 Scope - Audited Files

	File	LoC
0	eddsa-token-with-fees-distributor-solana/src/traits/mod.rs	0
1	eddsa-token-with-fees-distributor-solana/src/state/eddsa_distributor_data.rs	7
2	eddsa-token-with-fees-distributor-solana/src/state/airdrop.rs	13
3	eddsa-token-with-fees-distributor-solana/src/state/mod.rs	8
4	eddsa-token-with-fees-distributor-solana/src/state/config.rs	8
5	eddsa-token-with-fees-distributor-solana/src/state/misc.rs	5
6	eddsa-token-with-fees-distributor-solana/src/instructions/deploy.rs	26
7	eddsa-token-with-fees-distributor-solana/src/instructions/withdraw.rs	49
8	eddsa-token-with-fees-distributor-solana/src/instructions/transfer_program_admin.rs	29
9	eddsa-token-with-fees-distributor-solana/src/instructions/transfer_ownership.rs	20
10	eddsa-token-with-fees-distributor-solana/src/instructions/toggle_pause.rs	16
11	eddsa-token-with-fees-distributor-solana/src/instructions/version.rs	20
12	eddsa-token-with-fees-distributor-solana/src/instructions/utils.rs	319
13	eddsa-token-with-fees-distributor-solana/src/instructions/receive_program_admin.rs	24
14	eddsa-token-with-fees-distributor-solana/src/instructions/verify.rs	18
15	eddsa-token-with-fees-distributor-solana/src/instructions/mod.rs	34
16	eddsa-token-with-fees-distributor-solana/src/instructions/initialize.rs	66
17	eddsa-token-with-fees-distributor-solana/src/instructions/set_fee_collector.rs	50
18	eddsa-token-with-fees-distributor-solana/src/instructions/set_fee_token.rs	25
19	eddsa-token-with-fees-distributor-solana/src/instructions/set_default_fee_collector.rs	25
20	eddsa-token-with-fees-distributor-solana/src/instructions/claim.rs	111
21	eddsa-token-with-fees-distributor-solana/src/instructions/encode_hash_to_be_signed.rs	14
22	eddsa-token-with-fees-distributor-solana/src/instructions/set_base_params.rs	27
23	eddsa-token-with-fees-distributor-solana/src/instructions/deposit.rs	54
24	eddsa-token-with-fees-distributor-solana/src/errors.rs	30
25	eddsa-token-with-fees-distributor-solana/src/lib.rs	122
26	eddsa-token-with-fees-distributor-solana/src/models/user_claim_data.rs	13
27	eddsa-token-with-fees-distributor-solana/src/models/mod.rs	4
28	eddsa-token-with-fees-distributor-solana/src/models/claim.rs	11
29	eddsa-token-with-fees-distributor-solana/src/event.rs	18
30	eddsa-token-distributor-solana/src/traits/mod.rs	0
31	eddsa-token-distributor-solana/src/state/eddsa_distributor_data.rs	6
32	eddsa-token-distributor-solana/src/state/airdrop.rs	13
33	eddsa-token-distributor-solana/src/state/mod.rs	8
34	eddsa-token-distributor-solana/src/state/config.rs	8
35	eddsa-token-distributor-solana/src/state/misc.rs	5
36	eddsa-token-distributor-solana/src/instructions/deploy.rs	26
37	eddsa-token-distributor-solana/src/instructions/withdraw.rs	49
38	eddsa-token-distributor-solana/src/instructions/transfer_program_admin.rs	29
39	eddsa-token-distributor-solana/src/instructions/transfer_ownership.rs	20
40	eddsa-token-distributor-solana/src/instructions/toggle_pause.rs	16
41	eddsa-token-distributor-solana/src/instructions/version.rs	20
42	eddsa-token-distributor-solana/src/instructions/utils.rs	338
43	eddsa-token-distributor-solana/src/instructions/receive_program_admin.rs	24
44	eddsa-token-distributor-solana/src/instructions/verify.rs	18
45	eddsa-token-distributor-solana/src/instructions/mod.rs	34
46	eddsa-token-distributor-solana/src/instructions/initialize.rs	66
47	eddsa-token-distributor-solana/src/instructions/set_fee_collector.rs	50
48	eddsa-token-distributor-solana/src/instructions/set_fee_token.rs	25
49	eddsa-token-distributor-solana/src/instructions/set_default_fee_collector.rs	25
50	eddsa-token-distributor-solana/src/instructions/claim.rs	111
51	eddsa-token-distributor-solana/src/instructions/encode_hash_to_be_signed.rs	14
52	eddsa-token-distributor-solana/src/instructions/set_base_params.rs	27
53	eddsa-token-distributor-solana/src/instructions/deposit.rs	54
54	eddsa-token-distributor-solana/src/errors.rs	30
55	eddsa-token-distributor-solana/src/lib.rs	122
56	eddsa-token-distributor-solana/src/models/user_claim_data.rs	13
57	eddsa-token-distributor-solana/src/models/mod.rs	4
58	eddsa-token-distributor-solana/src/models/claim.rs	11
59	eddsa-token-distributor-solana/src/event.rs	18
	Total	**2350**

5.2 Findings Overview

	Finding	Severity	Update
1	Missing `fee_collector` check in the `Claim` instruction	High	Fixed
2	Changing the `projectToken` might not be functional.	Info	Fixed
3	The `fee_collector_storage` account constraint makes switching the `fee_collector` failed.	Info	Fixed
4	Unused code	Info	Fixed

6 System Overview

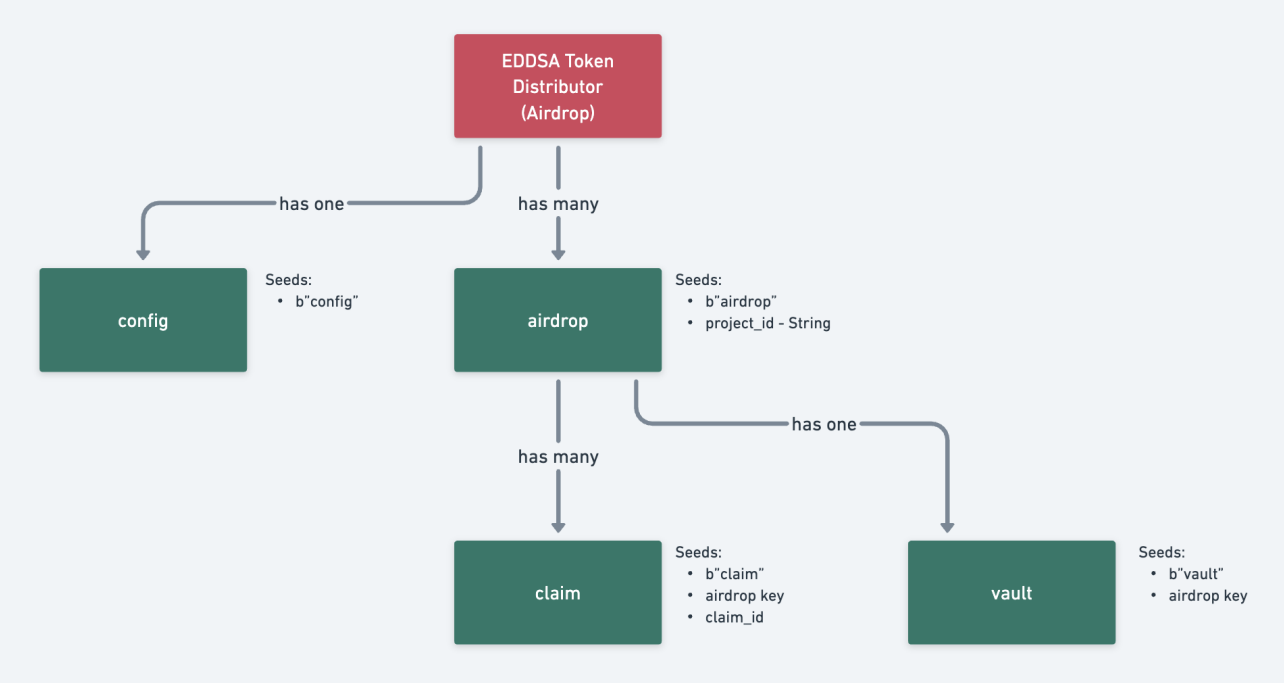
TokenTable has developed the EDDSA-token-distributor program, which works in conjunction with the fee-collector program to form an independent on-chain system. This system enables projects to conduct large-scale token airdrops and allows users to securely claim airdrop tokens by providing Ed25519 signatures.

6.1 EDDSA-token-distributor

In the EDDSA-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 parameters for the airdrop, such as the start and end times, as well as the Authorised Signer for signature verification.
3. Users can claim the airdrop by providing the correct signature of their airdrop claim after the claimable time is reached.

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



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

EDDSA-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.
5. `set_default_fee_collector` – sets the `fee_collector` for a given Project to the default fee collector program.

EDDSA-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.

EDDSA-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 authorized signer, airdrop start and end times, and url.
3. `toggle_pause` - Allows the Owner of the Project to pause and unpause the airdrop claims.
4. `transfer_ownership` – Owner transfers the `airdrop` ownership to a new account.
5. `withdraw` – Allows the Owner of the Project to withdraw deposited airdrop tokens.

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

1. `claim` - claims the airdrop tokens by providing the Ed25519 signature. The recipient account within the signature will obtain the airdropped tokens.
2. `deposit` – Deposits a specified amount of tokens into the Project.
3. `version` – Used to update the program version information.
4. `encode_hash_to_be_signed` – Precomputes the leaf hash based on the input data.
5. `verify` – Verifies the signature of the claim against the authorized signer account of the airdrop. Doesn't update any account state.

7 Issues

7.1 [High] Missing `fee_collector` check in the `Claim` instruction

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

Description: In the `Claim` instruction, `fee_collector` is not verified in the ctx, and no check is performed in the subsequent handler function either.

```
#[derive(Accounts)]
#[instruction(_project_id: String, _recipient: Pubkey, _claim_id: [u8; 32])]
pub struct Claim<'info> {
  //...
  #[account(mut)]
  pub fee_collector_vault: Option<UncheckedAccount<'info>>,
  /// CHECK: Checked in the function call.
  pub fee_collector: UncheckedAccount<'info>,
  //...
}
```

Impact: A malicious actor can pass in a malicious program to bypass fee payment.

Recommendation(s): It is recommended to verify that the `fee_collector` passed in the instruction matches the one recorded in the `airdrop` account

Status: Fixed

Update from TokenTable: Added fee collector account constraint in `f934c5c727a9bf5cf6bc3dbd362b957a9d4f90ea`.

7.2 [Info] Changing the `projectToken` might not be functional.

File(s): `set_base_params.rs` , `deposit.rs`

Description: In the `set_base_params` instruction, the airdrop owner is allowed to modify the token account used for the airdrop.

```
pub fn set_base_params(
  ctx: Context<SetBaseParams>,
  _project_id: String,
  token: Pubkey,
  start_time: u64,
  end_time: u64,
  authorized_signer: Pubkey
) -> Result<()> {
  //...
  ctx.accounts.airdrop.token = token;
```

However, switching may not function properly because the vault associated with the airdrop is singular and may have already been initialized as a TokenAccount for the previous mint. Moreover, the vault initialization lacks permission control, and the initializer is not required to hold the corresponding token

```
pub struct Deposit<'info> {
  //...
  pub airdrop: Account<'info, Airdrop>,
  #[account(
    init_if_needed,
    payer = authority,
    seeds = [b"vault".as_ref(), airdrop.key().as_ref()],
    bump,
    token::mint = token_mint,
    token::authority = airdrop,
    token::token_program = token_program
  )]
  pub vault: InterfaceAccount<'info, TokenAccount>,
```

Impact: Modifying the token associated with the airdrop account may not be feasible. For instance:

1. The project initializes the airdrop account, and the token is set to `mint1`;
2. Another user calls `deposit`, which initializes the vault as a `TokenAccount` for `mint1`;
3. The project then calls `set_base_params` to change the token to `mint2`;
4. When the project tries to deposit `mint2` tokens, it fails because the vault was already initialized as a `TokenAccount` for `mint1` by another user;

Recommendation(s): It is recommended to add permission control to the deposit instruction, allowing only the airdrop owner to call it.

Status: Fixed

Update from TokenTable: Removed the ability to update the project token using `set_base_params()` in all affected programs in `a30590bd697e7564447b3e0e48e64199f06fea7e`.

7.3 [Info] The `fee_collector_storage` account constraint makes switching the `fee_collector` failed.

File(s): `claim.rs`, `initialize.rs`, `set_fee_collector.rs`

Description: The EDDSA program allows different `fee_collector` programs to be configured for airdrop accounts. However, since the CPI to `fee_collector` program in the `ctx` uses an `Account<'info, fee_collector::models::FeeCollectorStorage>` type, it checks whether the `fee_collector_storage` account's owner matches `fee_collector::models::FeeCollectorStorage::owner`. This causes the instructions to fail if a different `fee_collector` program is configured, due to an owner mismatch on the `fee_collector_storage` account.

```
pub fee_collector_storage: Option<
    Box<Account<'info, fee_collector::models::FeeCollectorStorage>>
>,
```

Impact: The airdrop account cannot be updated or used with a different `fee_collector` program, which is not the expected behavior.

Recommendation(s): In the `ctx`, specify the type of `fee_collector_storage` as `UncheckedAccount<'info>` instead of `Account<'info, T>`. During the CPI call, the `fee_collector` program will verify whether the account is as expected.

Status: Fixed

Update from TokenTable: Switched to `UncheckedAccount<'info>` for all usages of `fee_collector_storage` in all programs in `c0d34b3e6128278a11bf9df06812ffb87df3d779`.

7.4 [Info] Unused code

File(s): `utils.rs`

Description: The `utils.rs` file containing signature verification capabilities of the program contains unused code:

```
pub fn merkle_verify(proof: Vec<[u8; 32]>, root: [u8; 32], leaf: [u8; 32]) -> bool {
  let mut computed_hash = leaf;
  for proof_element in proof.into_iter() {
    if computed_hash <= proof_element {
      computed_hash = keccak::hashv(&[&computed_hash, &proof_element]).0;
    } else {
      computed_hash = keccak::hashv(&[&proof_element, &computed_hash]).0;
    }
  }
  computed_hash == root
}
```

Which was likely copied unnecessarily from the Merkle Distributor program.

Another unreachable code section is related to the `verify_secp256k1_ix` which is not used anywhere in the code, there could be two options. As per design the Secp256k1 signatures are not supported by the protocol

Impact: Higher SOL fees for program deployment and upgrades

Recommendation(s): Remove the unnecessary code.

Status: Fixed

Update from TokenTable: Unused code removed in `f925d1f5aaf75371cc9fd99c5c34269362abba97` and `54c916fe785c9249584836fed6bad594492c2a80`.

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 (/.../019-TokenTable-Solana-ECDSA/code/programs/fee-collector)
   Compiling eddsa-token-with-fees-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/eddsa-token-with-fees-distributor-solana)
    Finished `release` profile [optimized] target(s) in 4.41s
   Compiling fee-collector v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/fee-collector)
   Compiling eddsa-token-with-fees-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/eddsa-token-with-fees-distributor-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 2.65s
     Running unittests src/lib.rs
      ↪  (/.../019-TokenTable-Solana-ECDSA/code/target/debug/deps/eddsa_token_with_fees_distributor_solana-f649e6a1274d962c)
   Compiling merkle-token-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/merkle-token-distributor-solana)
    Finished `release` profile [optimized] target(s) in 2.26s
   Compiling merkle-token-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/merkle-token-distributor-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.89s
     Running unittests src/lib.rs
      ↪  (/.../019-TokenTable-Solana-ECDSA/code/target/debug/deps/merkle_token_distributor_solana-4f0d534e4c965262)
   Compiling eddsa-token-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/eddsa-token-distributor-solana)
    Finished `release` profile [optimized] target(s) in 1.72s
   Compiling eddsa-token-distributor-solana v0.1.0
    ↪  (/.../019-TokenTable-Solana-ECDSA/code/programs/eddsa-token-distributor-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.88s
     Running unittests src/lib.rs
      ↪  (/.../019-TokenTable-Solana-ECDSA/code/target/debug/deps/eddsa_token_distributor_solana-f2fa01bf8b3ae121)
   Compiling fee-collector v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/fee-collector)
    Finished `release` profile [optimized] target(s) in 1.49s
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.30s
     Running unittests src/lib.rs
      ↪  (/.../019-TokenTable-Solana-ECDSA/code/target/debug/deps/fee_collector-fb9dae5dd11c4844)
   Compiling fee-collector v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/fee-collector)
   Compiling unlocker-v2-solana v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/unlocker-v2-solana)
    Finished `release` profile [optimized] target(s) in 3.88s
   Compiling fee-collector v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/fee-collector)
   Compiling unlocker-v2-solana v0.1.0 (/.../019-TokenTable-Solana-ECDSA/code/programs/unlocker-v2-solana)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 1.70s
     Running unittests src/lib.rs
      ↪  (/.../019-TokenTable-Solana-ECDSA/code/target/debug/deps/unlocker_v2_solana-4efcc2e69644e960)
```

9.2 Tests Output

EDDSA-token-distributor's test output:

```
> anchor test
  eddsa-token-distributor-solana
     Is deployed (714ms)
     Transfer program admin (3038ms)
     Is initialized! (554ms)
     Transfer ownership (1763ms)
     Set Base Params (fail - bad time)
     Set Base Params (fail - not owner) (525ms)
     Set Base Params (pass) (552ms)
     Set Fee Collector (fail - not deployer) (610ms)
     Set Fee Collector (succeed) (1023ms)
     Set Fee Token (fail - not deployer) (504ms)
     Set Fee Token (succeed) (1118ms)
     Toggle Pause (1041ms)
     encode_claim_id, encode_hash_to_be_signed, verify (InvalidSignature)
     encode_claim_id, encode_hash_to_be_signed, verify (pass) (473ms)
     Deposit SPL (496ms)
     Withdraw (fail - not owner) (1154ms)
     Withdraw (succeed) (1083ms)
     Claim (fail - not active) (499ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - outside claimable time range) (492ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - bad data/signature) (530ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (pass) (1002ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - double-claim attempt)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (pass, separate project ID) (3164ms)
     Fee Collector (lamports) (1463ms)
     Fee Collector (SPL) (3904ms)
    eddsa-token-with-fees-distributor-solana
     Is deployed (503ms)
     Transfer program admin (2024ms)
     Is initialized! (524ms)
     Transfer ownership (1475ms)
     Set Base Params (fail - bad time)
     Set Base Params (fail - not owner) (488ms)
     Set Base Params (pass) (507ms)
     Set Fee Collector (fail - not deployer) (573ms)
     Set Fee Collector (succeed) (1173ms)
     Set Fee Token (fail - not deployer) (503ms)
     Set Fee Token (succeed) (1087ms)
     Toggle Pause (1550ms)
     encode_claim_id, encode_hash_to_be_signed, verify (InvalidSignature)
     encode_claim_id, encode_hash_to_be_signed, verify (pass) (489ms)
     Deposit SPL (521ms)
     Withdraw (fail - not owner) (1070ms)
     Withdraw (succeed) (964ms)
     Claim (fail - not active) (652ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - outside claimable time range) (808ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - bad data/signature) (518ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (pass) (1029ms)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (fail - double-claim attempt)
     Encode Claim ID + Encode Hash to be Signed + Verify + Claim (pass, separate project ID) (3396ms)
     Fee Collector (lamports) (1591ms)
     Fee Collector (SPL) (4880ms)
```

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.