iSwap Solana Bridge

AUDIT REPORT

Version 1.0.0

Serial No. 20220215000420 20

Presented by Fairyproof

Feb 15, 2022

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the iSwap Solana Bridge project.

Audit Start Time:

Jan 26, 2022

Audit End Time:

Feb 14, 2022

Audited Source Files:

The source files audited include all the files as follows:

```
programs/
└── i_bridge_solana_program
    ├── Cargo.toml
    ├── Xargo.toml
    └── src
        └── lib.rs

2 directories, 3 files
```

The goal of this audit is to review iSwap's implementation for its cross-chain bridge function on Solana, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the iSwap team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from off-chain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— The iSwap Team's Consent/Acknowledgement:

The audited materials of the project including but not limited to the documents, home site, source code, etc are all developed, deployed, managed, and maintained outside Mainland CHINA.

The members of the team, the foundation, and all the organizations that participate in the audited project are not Mainland Chinese residents.

The audited project doesn’t provide services or products for Mainland Chinese residents.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Fairyproof to make sure we understand the size, scope, and functionality of the project's source code.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Fairyproof describe.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run the test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the source code to improve maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

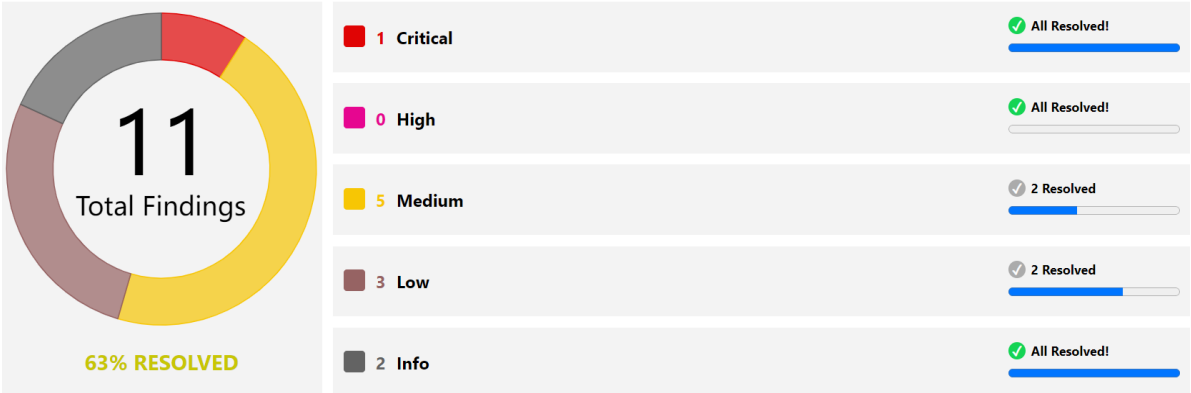
For this audit, we used the following source of truth about how the cross-chain bridge system should work:

https://www.iswap.com

This was considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the iSwap team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Audit Result
2022021500042020	Fairyproof Security Team	2022.01.26 - 2022.02.14	Medium



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, 1 risk of critical-severity, 5 risks of medium-severity, 3 risks of low-severity and 2 risks of informational-severity were discovered. The iSwap team fixed 1 risk of critical-severity, 2 risks of medium-severity, 2 risks of low-severity and 2 risks of informational-severity, and confirmed 3 risks of medium-severity and 1 risk of low-severity.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to iSwap Solana Bridge

This is a cross-chain bridge deployed on Solana. The iSwap team will build asset pools on multiple chains. When a user deposits a crypto asset into this DApp he/she can get that asset from a pool on another supported chain. When a user deposits a crypto asset from another chain to Solana, this DApp will send that crypto asset to this user.

Note: when a user transfers a crypto asset from Solana to another chain, he/she needs to use the parameters and interfaces from this DApp otherwise he/she may be punished.

Note: only the code that takes crypto assets and sends crypto assets on Solana was covered in this audit. Other functions or features were not covered by this audit.

04. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Missing ownership checks
- Missing signer checks
- Signed invocation of unverified programs
- Solana account confusions
- Redeployment with cross-instance confusion

- Missing freeze authority checks
- Insuicient SPL account verification
- Missing rent exemption assertion
- Casting truncation
- Arithmetic over- or underflows
- Numerical precision errors
- Denial of service attacks
- Ruling out economic attacks
- Instructions account checks
- Rug pull mechanisms or hidden backdoors checks
- Outdated dependencies
- None returned during any failure
- Unreferenced accounts checks
- Serialize and deserialize checks
- Design logic checks
- Account address checks
- Unaudited external dependencies
- Program upgrade checks
- Account mutable checks
- Code optimization checks

05. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

06. List of functions audited

The Fairyproof security team analyzed the major functions during the audit and the result is as follows:

Function Name	Rent-Exemption Check	Account Check	Signer Check	Program_id check
initialize	Yes	3/3	N/A	Yes
change_governor	N/A	2/2	Yes	Yes
add_support_token	Yes	8/8	Yes	Yes
cross_chain_token	N/A	8/8	Yes	Yes
cross_chain_token_confirm	Yes	12/12	Yes	Yes
refund_token	N/A	8/8	Yes	Yes
withdrawal	N/A	7/7	Yes	Yes
withdrawal_punish	N/A	7/7	Yes	Yes
set_support_token_min_amount	N/A	4/4	Yes	Yes

07. List of function accounts audited

The Fairyproof security team analyzed the major function accounts during the audit and the result is as follows:

- initialize

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	Init	Yes	PDA,seed="vault-meta"
1	user	Signer	Yes	
2	system_program	N/A	Yes	System

- change_governor

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seed="vault-meta"
1	governor	Signer	Yes	

- add_support_token

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	governor	Writable/Signer	Yes	
2	mint	N/A	Yes	
3	vault_token_account	Init	Yes	PDA,seeds=["token-seed",mint]
4	vault_authority	N/A	Yes	PDA,seeds=["vault-authority"]
5	system_program	N/A	Yes	System
6	rent	N/A	Yes	System
7	token_program	N/A	Yes	Program
8	vault_meta_account	Init	Yes	PDA,seeds=["vault-config-meta",mint]

- set_support_token_min_amount

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	governor	Writable/Signer	Yes	
2	mint	N/A	Yes	
3	vault_meta_account	Writable	Yes	PDA,seeds=["vault-config-meta",mint]

- cross_chain_token

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	mint	N/A	Yes	
2	user	Signer	Yes	
3	user_token_account	Writable	Yes	
4	vault_token_account	Writable	Yes	PDA,seeds=["token-seed",mint]
5	treasury_token_account	Writable	Yes	
6	token_program	N/A	Yes	Program
7	vault_meta_account	N/A	Yes	PDA,seeds=["vault-config-meta",mint]

- cross_chain_token_confirm

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	mint	N/A	Yes	
2	relayer	Writable/Signer	Yes	
3	receiver	N/A	Yes	
4	receiver_token_account	Writable	Yes	
5	vault_token_account	Writable	Yes	PDA,seeds=["token-seed",mint]
6	vault_authority	N/A	Yes	PDA,seeds=["vault-authority"]
7	treasury_token_account	Writable	Yes	
8	rent	N/A	Yes	System
9	token_program	N/A	Yes	Program
10	associated_program	N/A	Yes	Program
11	system_program	N/A	Yes	System

- refund_token

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	mint	N/A	Yes	
2	relayer	Signer	Yes	
3	receiver	N/A	Yes	
4	receiver_token_account	Writable	Yes	
5	vault_token_account	Writable	Yes	PDA,seeds=["token-seed",mint]
6	vault_authority	N/A	Yes	PDA,seeds=["vault-authority"]
7	token_program	N/A	Yes	Program

- withdrawal

Index	Account Name	Writable/Signer/Init	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	mint	N/A	Yes	
2	custodian	Signer	Yes	
3	custodian_token_account	Writable	Yes	
4	vault_token_account	Writable	Yes	PDA,seeds=["token-seed",mint]
5	vault_authority	N/A	Yes	PDA,seeds=["vault-authority"]
6	token_program	N/A	Yes	Program

- withdrawal_punish

Index	Account Name	Writable/Signer/Init/System	Checked	Other
0	meta_account	N/A	Yes	PDA,seeds=["vault-meta"]
1	mint	N/A	Yes	
2	risk_control	Signer	Yes	
3	risk_control_token_account	Writable	Yes	
4	vault_token_account	Writable	Yes	PDA,seeds=["token-seed",mint]
5	vault_authority	N/A	Yes	PDA,seeds=["vault-authority"]
6	token_program	N/A	Yes	Program

08. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Using Inappropriate Account to Keep Global Variables	Account address checks	Critical	✓Fixed
FP-2	Incorrect Divisor and Dividend	Design logic checks	Medium	✓Fixed
FP-3	Unable to Create Multiple Vaults	Design logic checks	Medium	✓Fixed
FP-4	Multiple Accounts Could Withdraw Assets Without Limitation	Design logic checks	Medium	Confirmed
FP-5	Use of Unaudited Framework	Unaudited external dependencies	Medium	Confirmed
FP-6	Contract Upgradeable	Program upgradability checks	Medium	Confirmed
FP-7	Missing Constrains for Input Parameters	Denial of service attacks	Low	Confirmed
FP-8	Arithmetic Operation Overflow	Arithmetic over- or underflows	Low	✓Fixed
FP-9	Some Non-writable Accounts Were `mut`	Account mutable checks	Low	✓Fixed
FP-10	Vault Creation Could be Simplified	Code optimization checks	Informational	✓Fixed
FP-11	Missing Verification for Accounts	Instructions account checks	Informational	✓Fixed

09. Issue descriptions

[FP-1] Using Inappropriate Account to Keep Global Variables Critical ✓Fixed

Issue/Risk: Account address checks

Description:

The `initialize` function used a randomly generated account as a global account to keep global variables. This account was not necessarily unique and couldn't be verified. Users could create another global account and pass verification to steal assets.

Recommendation:

Consider using a PDA account to keep global variables.

Status:

It has been fixed by the iSwap team.

[FP-2] Incorrect Divisor and Dividend

Medium

✓Fixed

Issue/Risk: Design logic checks

Description:

In the `cross_chain_token` function, the `DENOMINATOR` and `CROSS_CHAIN_FEE_RATIO` should be reversed.

Recommendation:

Consider changing the code as follows:

```
if cross_chain_fee > (amount * DENOMINATOR / CROSS_CHAIN_FEE_RATIO) {
    return Err(ErrorCode::ExceedTheCrossChainFeeLimit.into());
}
```

Status:

It has been fixed by the iSwap team.

[FP-3] Unable to Create Multiple Vaults

Medium

✓Fixed

Issue/Risk: Design logic checks

Description:

The `add_support_token` function used a PDA account for tokens therefore this account could only have one token vault. This didn't match the expected design.

Recommendation:

Consider using a PDA seed and a token address as a seed to support multiple token vaults and using the PDA seed and the token address together for verification.

Here is a code sample:

```
#[account(
    init,
    seeds = [VAULT_PDA_SEED.as_ref(), mint.key().as_ref()],
    bump = vault_token_account_bump,
    payer = governor,
    token::mint = mint,
    token::authority = vault_authority,
)]
pub vault_token_account: Account<'info, TokenAccount>,
```

Status:

It has been fixed by the iSwap team.

[FP-4] Multiple Accounts Could Withdraw Assets Without Limitation Medium Confirmed

Issue/Risk: Rug pull mechanisms checks

Description:

In its implementation, `relayer`, `custodian` and `risk_control` all could withdraw assets without any limitation. If their private keys were leaked the assets would be exposed to huge risks.

Recommendation:

Consider carefully managing these accounts and transferring the private keys of `custodian` and `risk_control` to multi-sig wallets.

Status:

The initial assets were all provided by the iSwap team, therefore even if the assets were to be exploited users would not suffer any loss.

[FP-5] Use of Unaudited Framework Medium Confirmed

Issue/Risk: Unaudited external dependencies

Description:

The implementation used `Anchor` developed by `serum` . However `Anchor` was not audited and it even had the following statement:

> - **This code is unaudited. Use at your own risk.**

Therefore we marked this as a medium-severity risk.

Recommendation:

Consider watching the latest updates about `Anchor` especially any bug fix and doing contract upgrade accordingly.

Status:

`Anchor` has been developed and maintained by the Solana team. The iSwap team will keep watching its updates and bug fixes. Whenever there are any bug fixes, the iSwap team will upgrade the contracts accordingly.

[FP-6] Contract Upgradeable

Medium Confirmed

Issue/Risk: Program upgradability checks

Description:

In Solana, a contract can be upgraded unless its upgradeability is turned off when it is deployed or upgraded. Therefore users need to trust the iSwap team's design and operation. And the iSwap team needs to keep the private keys safe and secure.

Recommendation:

Consider keeping the private keys safe and secure.

Status:

In order for the ease of future development and bug fixes and for the handling of emergent cases, the contracts are all upgradeable.

[FP-7] Missing Constrains for Input Parameters

Low Confirmed

Issue/Risk: Denial of service attacks

Description:

In the `cross_chain_token` function, parameters of `order_id`, `cross_chain_fee`, `gas_fee` and `channel` could be input by users without constraints. Therefore users could exploit or attack the application by inputting inappropriate parameters such as an existing `order_id`.

After communicating with the iSwap team, if these parameters are inappropriate the users would be considered as malicious actors and would be punished. However, users could still attack the application without paying much cost.

Recommendation:

Consider adding constraints for these parameters to prevent users from attacking the application.

Status:

The iSwap team added constraints.

[FP-8] Arithmetic Operation Overflow

Low ✓Fixed

Issue/Risk: Arithmetic over- or underflows

Description:

In the `cross_chain_token` function, the code `let balance_after = balance_before + remaining_amount.unwrap();` may have overflow. Since this was used in events we marked it as a low-severity risk.

Recommendation:

Consider using `checked_add` instead of `+` .

Status:

It has been fixed by the iSwap team.

[FP-9] Some Non-writable Accounts Were `mut`

Low ✓Fixed

Issue/Risk: Account mutable checks

Description:

In Solana, an account can be `mut` which means that the account is mutable. However in some functions, `meta_account` and `receiver` were incorrectly defined as `mut`. This would lead to unexpected issues.

Recommendation:

Consider removing these incorrect `mut` keywords.

Status:

It has been fixed by the iSwap team.

[FP-10] Vault Creation Could be Simplified

Informational ✓Fixed

Issue/Risk: Code optimization checks

Description:

In the `add_support_token` function, it created a token account for `governor` and then transferred it to PDA. This could be simplified by using PDA as its seed on account creation.

Recommendation:

Consider simplifying the code as follows:

```
#[account(seeds = [VAULT_AUTH_PDA_SEED.as_ref()], bump)]
pub auth_pad: AccountInfo<'info>,

#[account(
    init,
    seeds = [VAULT_PDA_SEED.as_ref(), mint.key().as_ref()],
    bump = vault_token_account_bump,
    payer = governor,
    token::mint = mint,
    token::authority = auth_pad,
)]
pub vault_token_account: Account<'info, TokenAccount>,
```

Status:

It has been fixed by the iSwap team

[FP-11] Missing Verification for Accounts

Informational ✓Fixed

Issue/Risk: Code optimization checks

Description:

Some functions missed verification for their `owner` and `mint`.

Recommendation:

Consider adding verification as follows:

```
 #[account(
    mut,
    constraint = user_token_account.amount >= amount
    @ ErrorCode::UserNotEnoughFund,
    constraint = user_token_account.mint == mint.key(),
    constraint = user_token_account.owner == user.key()
)]
pub user_token_account: Account<'info, TokenAccount>,
```

Status:

It has been fixed by the iSwap team.

10. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- Transferring Admin Rights to Multi-sig Wallets

11. Appendix

Calculated SHA-256 Value for Audited File

```
programs/i_bridge_solana_program/src/lib.rs:
0xa27001fa8f10c48818c40a09db3e561ba00dbaecc661769f87bebb0777ab4841
```

Tools and Utilities

The following tools were used during this portion of the test. A link for more information about the tool is provided as well. Tools used during the code review and assessment • Rust – cargo tools • IDE modules for Rust and analysis of source code • Cargo audit which uses https://rustsec.org/advisories/ to find vulnerabilities cargo.

The resut of the `cargo upgrades` command.

```
iBridge-solana-program: /iswap/programs/i_bridge_solana_program/Cargo.toml
        anchor-lang matches 0.19.0;     latest is 0.21.0
        anchor-spl matches 0.19.0;      latest is 0.21.0
```

The result of the `cargo audit` command.

```
    Fetching advisory database from `https://github.com/RustSec/advisory-db.git`
      Loaded 397 security advisories (from /Users/likai/.cargo/advisory-db)
    Updating crates.io index
    Scanning Cargo.lock for vulnerabilities (139 crate dependencies)
Crate:         zeroize
Version:       1.5.0
Warning:       yanked
Dependency tree:
zeroize 1.5.0
└── curve25519-dalek 3.2.0
    └── solana-program 1.9.5
        ├── spl-token 3.2.0
        |   ├── spl-associated-token-account 1.0.3
        |   |   ├── iBridge-solana-program 0.1.0
        |   |   └── anchor-spl 0.19.0
```

```
        |   |       └── iBridge-solana-program 0.1.0
        |   ├── iBridge-solana-program 0.1.0
        |   └── anchor-spl 0.19.0
        ├── spl-associated-token-account 1.0.3
        ├── anchor-spl 0.19.0
        └── anchor-lang 0.19.0
            ├── iBridge-solana-program 0.1.0
            └── anchor-spl 0.19.0

warning: 1 allowed warning found
```

Unit Testing

File used for unit testing: `tests/i_bridge_solana_program.js`

```
const anchor = require('@project-serum/anchor');
const assert = require("assert");
const { Token, TOKEN_PROGRAM_ID, ASSOCIATED_TOKEN_PROGRAM_ID} =
require('@solana/spl-token');
const PublicKey = anchor.web3.PublicKey

describe('Iswap Solana Bridge Test', () => {
  const provider = anchor.Provider.local();
  // Configure the client to use the local cluster.
  anchor.setProvider(provider);
  const program = anchor.workspace.IBridgeSolanaProgram;
  // tokens and pda
  let USDT,ATS;
  let META_PDA;
  let AUTH_PDA;
  // all users
  const user = anchor.web3.Keypair.generate();
  const governor = anchor.web3.Keypair.generate();
  const treasury = anchor.web3.Keypair.generate();
  const custodian = anchor.web3.Keypair.generate();
  const risk_control = anchor.web3.Keypair.generate();
  const relayer = anchor.web3.Keypair.generate();
  // token_account
  let user_usdt,user_ats;
  let USDT_PDA,ATS_PDA;
  let treasury_usdt,treasury_ats;
  let USDT_CONFIG_PDA,ATS_CONFIG_PDA;

  it("Init users,tokens and auth_pda", async () => {
    // air drop
    const user_drop = await
provider.connection.requestAirdrop(user.publicKey,anchor.web3.LAMPORTS_PER_SOL);
    const treasury_drop = await
provider.connection.requestAirdrop(treasury.publicKey,anchor.web3.LAMPORTS_PER_S
OL);
    await provider.connection.confirmTransaction(user_drop);
    await provider.connection.confirmTransaction(treasury_drop);

    // create mint
    ATS = await Token.createMint(
```

```
      provider.connection,
      provider.wallet.payer,
      provider.wallet.publicKey,
      null,
      9,
      TOKEN_PROGRAM_ID,
    );
    USDT = await Token.createMint(
      provider.connection,
      provider.wallet.payer,
      provider.wallet.publicKey,
      null,
      9,
      TOKEN_PROGRAM_ID,
    );

    // create token account
    user_ats = await
Token.getAssociatedTokenAddress(ASSOCIATED_TOKEN_PROGRAM_ID,TOKEN_PROGRAM_ID,ATS
.publicKey,user.publicKey)
    user_usdt = await USDT.createAssociatedTokenAccount(user.publicKey);
    treasury_usdt = await USDT.createAssociatedTokenAccount(treasury.publicKey);
    treasury_ats = await ATS.createAssociatedTokenAccount(treasury.publicKey);

    // mint usdt to user
    await USDT.mintTo(user_usdt,provider.wallet.publicKey,[],10000);
    let info = await USDT.getAccountInfo(user_usdt);
    assert.ok(info.amount.toNumber() === 10000);

    //cal auth_pda
    const [auth_pda,] = await PublicKey.findProgramAddress(
      [Buffer.from(anchor.utils.bytes.utf8.encode("vault-authority")),],
      program.programId
    );
    AUTH_PDA = auth_pda;
  });

  it('Init meta data test', async () => {
    // cal meta_pda
    const [meta_pda,meta_nonce] = await PublicKey.findProgramAddress(
      [Buffer.from(anchor.utils.bytes.utf8.encode("vault-meta")),],
      program.programId
    )
    META_PDA = meta_pda;
    await program.rpc.initialize(
      meta_nonce,
      governor.publicKey,
      treasury.publicKey,
      custodian.publicKey,
      risk_control.publicKey,
      relayer.publicKey,
      {
        accounts: {
          metaAccount:meta_pda,
          user:provider.wallet.publicKey,
          systemProgram:anchor.web3.SystemProgram.programId
        }
      }
```

```
    );

    const meta_account = await program.account.metaAccount.fetch(meta_pda);
    assert.ok(meta_account.governor.toBase58() == governor.publicKey.toBase58())
    assert.ok(meta_account.treasury.toBase58() == treasury.publicKey.toBase58())
    assert.ok(meta_account.custodian.toBase58() ==
custodian.publicKey.toBase58())
    assert.ok(meta_account.riskControl.toBase58() ==
risk_control.publicKey.toBase58())
    assert.ok(meta_account.relayer.toBase58() == relayer.publicKey.toBase58())

    // init twice failed
    await assert.rejects(
      program.rpc.initialize(
        meta_nonce,
        governor.publicKey,
        treasury.publicKey,
        custodian.publicKey,
        risk_control.publicKey,
        relayer.publicKey,
        {
          accounts: {
            metaAccount:meta_pda,
            user:provider.wallet.publicKey,
            systemProgram:anchor.web3.SystemProgram.programId
          }
        }
      ),
      (e) => {
        assert.ok(e.logs[3].includes("already in use"));
        return true;
      }
    );
  });

  it("Change meta data test", async () => {
    // change governor
    const new_governor = anchor.web3.Keypair.generate();
    await program.rpc.changeGovernor(new_governor.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:governor.publicKey,
      },
      signers:[governor]
    });
    let meta_account = await program.account.metaAccount.fetch(META_PDA);
    assert.ok(meta_account.governor.toBase58() ==
new_governor.publicKey.toBase58())
    // change with previous governor will be failed
    await assert.rejects(
      program.rpc.changeGovernor(new_governor.publicKey,{
        accounts: {
          metaAccount:META_PDA,
          governor:governor.publicKey,
        },
        signers:[governor]
      }),
      (e) => {
```

```
        return true;
      }
    );
    // change others
    await program.rpc.changeTreasury(treasury.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:new_governor.publicKey,
      },
      signers:[new_governor]
    });
    await program.rpc.changeCustodian(custodian.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:new_governor.publicKey,
      },
      signers:[new_governor]
    });
    await program.rpc.changeRiskControl(risk_control.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:new_governor.publicKey,
      },
      signers:[new_governor]
    });
    await program.rpc.changeRelayer(relayer.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:new_governor.publicKey,
      },
      signers:[new_governor]
    });

    // change governor back
    await program.rpc.changeGovernor(governor.publicKey,{
      accounts: {
        metaAccount:META_PDA,
        governor:new_governor.publicKey,
      },
      signers:[new_governor]
    });
    meta_account = await program.account.metaAccount.fetch(META_PDA);
    assert.ok(meta_account.governor.toBase58() == governor.publicKey.toBase58())
  });


  it("AddSupportToken test", async () => {
    // add usdt
    const [usdt_pda,usdt_nonce] = await PublicKey.findProgramAddress(
      [Buffer.from(anchor.utils.bytes.utf8.encode("token-
seed")),USDT.publicKey.toBuffer()],
      program.programId
    );
    USDT_PDA = usdt_pda;

    const [usdt_config,usdt_config_nonce] = await PublicKey.findProgramAddress(
      [Buffer.from(anchor.utils.bytes.utf8.encode("vault-config-
meta")),USDT.publicKey.toBuffer()],
```

FAIRYPROOF

https://medium.com/@FairyproofT

https://twitter.com/FairyproofT

https://www.linkedin.com/company/fairyproof-tech

https://t.me/Fairyproof_tech

Reddit: https://www.reddit.com/user/FairyproofTech